Pattern Recognition 65 (2017) 108–118

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Point correspondence by a new third order graph matching algorithm

Xu Yang[a], Hong Qiao[a,b,c], Zhi-Yong Liu[a,b,c,*]

[a] State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing 100190, China
[b] Center for Excellence in Brain Science and Intelligence Technology, Chinese Academy of Sciences, Shanghai 200031, China
[c] University of Chinese Academy of Sciences, Beijing 100049, China

ARTICLE INFO

Keywords:
Graph matching
Point correspondence
High order constraints
Adjacency tensor

ABSTRACT

The correspondence between point sets is a fundamental problem in pattern recognition, which is often formulated and solved by graph matching. In this paper, we propose to solve the correspondence problem by a new third order graph matching algorithm. Compared with some previous hyper-graph matching algorithms, the proposed one achieves considerable memory reduction and is applicable to both undirected and directed graphs. Specifically, the correspondence is formulated by the matching between adjacency tensors encoding the third order structural information of each graph, which is then transformed to be a tractable matrix form. Two types of gradient based optimization methods, the graduated nonconvexity and concavity procedure (GNCCP) and graduated assignment (GA) algorithm, are generalized to solve the problem. Comparative experiments with state-of-the-art algorithms on both synthetic and real data witness the effectiveness of the proposed method.

1. Introduction

Graph matching, aiming to find the optimal correspondence between vertices of two graphs, is a fundamental problem in theoretical computer science, and it also plays an important role in artificial intelligence. Particularly, Graph matching can be directly applied to point correspondence which further lays the foundations for many pattern recognition applications such as object detection [1], classification [17] and visual tracking [31]. The correspondence between two point sets can be well defined by graph matching, by representing the points with graph vertices and encoding the point relations with graph edges. Note that although we mainly focus on the point correspondence problem in this paper, graph matching has more general implications. For instance, it was shown that the graph matching algorithm can be generalized to solve the maximum a posteriori (MAP) of Markov Random Field [20,6].

The graph matching methods can be categorized according to the constraint order. The first order constraints care only the differences between vertices without considering the edges. This problem is in nature a bipartite graph matching problem, which can be formulated as a linear programming problem and efficiently solved by for example the Hungarian algorithm [16] or the Volgenant-Jonker algorithm [13] in polynomial time. However, the first order method may suffer from the point descriptor ambiguities such as repeated textures [26]. The second order constraints can overcome this drawback, which preserve the geometric relations between points by minimizing the edge dissimilarity. The basic idea is, taking the distance constraint for example, if two points in the first set are close to each other, their corresponding points in the second set should also be kept close. The problem is often formulated as the minimization/maximization of a quadratic term. Since it is an NP-hard combinatorial optimization problem, many algorithms have been proposed to approximately solve it. However, the second order constraints still suffer from geometric transformations such as scale changes and rotations.

Recently, several researchers [36,8,4,22,18,26] have proposed to tackle the above problem using high order constraints. Most of the existing high order methods typically extend the spectral decomposition algorithm [19] to seek the rank-one-approximation of a large *affinity tensor* associated with the association hyper-graph of the input hyper-graphs. These methods have a solid theoretical foundation based on the tensor decomposition techniques [15] and exhibit superior matching performance. However, one main obstacle of these methods is the huge memory expense caused by the large affinity tensor, with the storage complexity as high as $O(N^6)$ in the worst case [4]. Besides, the affinity tensor is required to be super-symmetric, which implies that the methods [8,4,18,26,22] are only applicable to undirected hyper-graphs. On the other hand, the third order constraints, which target at minimizing the discrepancy between the corresponding triplets, may be enough to deal with some typical limitations encountered by the second order constraints, because diverse descriptors could be defined over the triplets, and they are often inherently invariant to geometric transformations such as affine transformation

* Corresponding author.
E-mail addresses: xu.yang@ia.ac.cn (X. Yang), hong.qiao@ia.ac.cn (H. Qiao), zhiyong.liu@ia.ac.cn (Z.-Y. Liu).

http://dx.doi.org/10.1016/j.patcog.2016.12.006
Received 17 June 2015; Received in revised form 9 December 2016; Accepted 10 December 2016
Available online 13 December 2016
0031-3203/

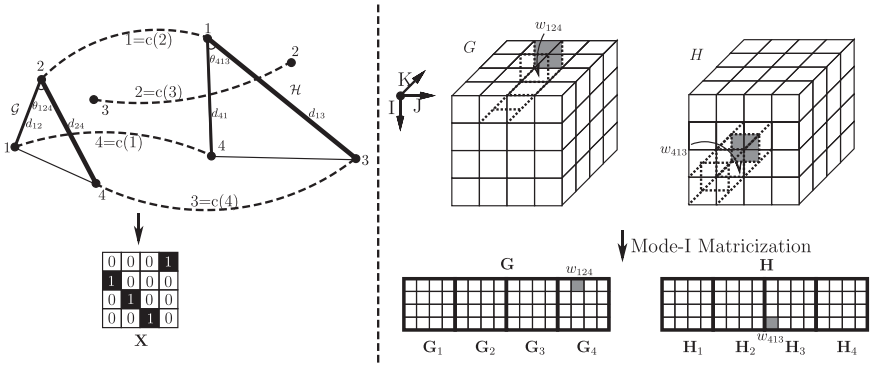

Fig. 1. Adjacency tensor matching. (1) Triplet descriptors are often inherently robust to geometric transformations, such as the directed descriptors θ_{124} and θ_{413}, or $\frac{d_{12}}{d_{24}}$ and $\frac{d_{41}}{d_{13}}$. (2) The correspondence is represent by a permutation matrix $\mathbf{X}$, and each assignment is denoted by $i_H = c(i_G)$ such as $4 = c(1)$. (3) The adjacency tensor G (H) is the array of hyper-edge weights, i.e. triplet descriptors. If permutating H with $\mathbf{X}$, the weights in G and H would be correspondingly located. For example, H_{413} and G_{124} would locate in the same position. (4) G (H) is flattened to $\mathbf{G}$ ($\mathbf{H}$) by mode-I matricization, with each submatrix denoted as $\mathbf{G}_i$ ($\mathbf{H}_i$).

and reflection transformation, as illustrated in Fig. 1.

In this paper, aiming at the memory expense reduction, we propose a new third order graph matching method based on *adjacency tensor* instead of affinity tensor. Consequently, the proposed method enjoys an $O(N^3)$ storage complexity in the worst case, and furthermore, it can be applied to both undirected hyper-graphs and directed hyper-graphs. The major novelty of this paper is formulating the third order graph matching by an adjacency tensor matching problem and further transforming it to be a tractable matrix form. Meanwhile, different from the previous spectral decomposition, the problem is approximately solved by the gradient based optimization methods.

Next after a brief survey on related works, the algorithm is proposed in Section 2 and discussed in Section 3. The experimental results are given in Section 4. Finally, Section 5 concludes the paper.

1.1. Related works

Establishing correspondence between point sets has long been a fundamental problem in pattern recognition. Since earlier works [28] exploring the first order similarity may fail in the presence of local appearance ambiguities, most recent methods try to efficiently incorporate the structural information by exploring the pairwise consistency, and several more recent methods consider high order constraints for the complementary structural information beyond second order relations.

Since graph matching using pairwise constraints is an NP-hard problem, some approximations are necessary for efficiency reasons [3]. Some approximate algorithms are based on bipartite graph matching [27,29], which typically approximates pairwise graph matching by the linear assignment problem with efficient solutions. An important group of approximate methods are known to be error-correcting or error-tolerant by assigning a cost (e.g. graph edit cost) to each type of error [2]. Another important group of approximate methods are based on spectral decomposition. Umeyama's algorithm [33] is regarded as the first spectral method. Leordeanu and Hebert [19] reformulated the objective function by affinity matrix and aimed for its optimal rank-one approximation by spectral decomposition. Some other algorithms [7,21,5] further extended [19] with considerable improvements. By gradually pushing the relaxed continuous result to be a discrete one, the GA algorithm [11] was integrated by some other relaxation algorithms [9,32] to get the matching result. A recently proposed group of algorithms are based on the convex and concave relaxation procedure pioneered by [35,25], which is a type of deterministic annealing method. Zhou and De La Torre [37] generalized it to the factorized affinity matrix. The recently proposed GNCCP [24] further simplified it by an implicit construction of the convex and concave relaxations.

In spite of a novel research direction, graph matching using high order constraints has gained interest of many researchers. Zass and Shashua [36] proposed the first hyper-graph matching based on matrix Kronecker product and interpreted the graph matching problem in a probabilistic way. Duchenne et al. [8] extended Leordeanu and Hebert's spectral method [19] to compute the rank-one approximation of the affinity tensor. Chertok and Keller [4] independently proposed a high order spectral algorithm similar to [8] which works on the unfolded affinity tensor. Leordeanu et al. [22] extend their integer projected fixed point (IPFP) [21] algorithm to hyper-graphs in a semi-supervised learning framework. Lee et al. [18] incorporated the matching constraints in each rank-one approximation iteration by extending their second order random walk method. Park et al. [26] also focused on the rank-one approximation of the affinity tensor and proposed an effective way to reduce the redundancy in it. The main obstacle of the previous high order algorithms is the huge storage expense, as also pointed out in [26]. In fact, though the third order constraints could well deal with the drawbacks of the second order methods, no previous algorithms are dedicated to it to get a specific storage saving. Besides, most of these algorithms [8,4,22,18,26] are limited to the undirected graph.

2. Adjacency tensor matching

When using the third order constraints, a graph $\mathcal{G}$ is also called hyper-graph defined by a finite vertex set $V = 1, 2, \ldots, N$ and a hyper-edge set $E = V \times V \times V$. Each edge ijk, $i, j, k \in V$ is a triplet, and is assigned with a weight w_{ijk}. The graph is either directed or *undirected*, where 'directed' means the triplets *ijk*, *ikj*, *jik*, *jki*, *kij* and *kji* are described respectively, while 'undirected' means they are considered as the same triplet. Though the undirected graph could well express many matching problems, the directed graph generally owns more distinctive ability [38]. Hereafter, by the term *graph* we mean such a weighted hyper-graph. As a generalization of the concept of adjacency matrix, *adjacency tensor* $G \in \mathbb{R}^{N\times N\times N}$, which is the multidimensional array of weights, is used to represent a graph $\mathcal{G}$, as shown in Fig. 1.

Given two graphs $\mathcal{G}$ and $\mathcal{H}$, the adjacency tensor based matching

objective is formulated as

$$\min F_0(\mathcal{G},\mathcal{H},\mathbf{X}) = \| G - H \times_I \mathbf{X} \times_J \mathbf{X} \times_K \mathbf{X} \|^2, \text{ s. t. } \mathbf{X}\in\mathcal{D}, \mathcal{D} := \left\{ \mathbf{X} | \sum_i^N \mathbf{X}_{ij} = 1, \sum_j^N \mathbf{X}_{ij} = 1, \mathbf{X}_{ij}\in\{0,1\}, \forall\, i,j \right\} \tag{1}$$

$\mathcal{D}$ is the set of permutation matrices, where correspondence between vertices under the one-to-one assumption is equivalent to finding a permutation matrix $\mathbf{X}$. $\mathbf{X}_{ij}=1$ means assigning vertex i in $\mathcal{G}$ to vertex j in $\mathcal{H}$, as shown in Fig. 1. $\|\cdot\|$ denotes the tensor norm. $\times_I$, $\times_J$ and $\times_K$ denote the tensor multiplications. Please see Appendix A for a brief introduction about the tensor notations and operations. In Eq. (1), first H is permutated by $\mathbf{X}$ from mode-I, J, K respectively, and the objective then measures the dissimilarity between G and permutated H with respect to $\mathbf{X}$. The objective is abbreviated as $F_0(\mathbf{X})$ hereafter. Compared with the previous methods, the storage saving here is mainly attributed to representing each graph by an adjacency tensor, without needing to build the large size association hyper-graph represented by the affinity tensor. In fact, the adjacency tensor bears close relations to the affinity tensor, which will be discussed in Section 3.1.

On the other hand, though gaining a considerable storage saving, the tensor decomposition techniques [15] cannot be directly used to solve Eq. (1). Here we propose to adopt the gradient based optimization methods to solve the problem. To make it easy to manipulate and get the derivative of the objective, by using the matricization or unfolding technique (See Appendix A for details) we transform the tensors in Eq. (1) into matrices as follows:

$$F_1(\mathbf{X}) = \| \mathbf{G} - \mathbf{X}\mathbf{H}(\mathbf{I}\otimes\mathbf{X}^T)(\mathbf{X}^T\otimes\mathbf{I})\|_F^2, \mathbf{X}\in\mathcal{D}, \tag{2}$$

where $\|\cdot\|_F$ is the Frobenius matrix norm defined as $\|\mathbf{G}\|_F = \sqrt{\sum_i\sum_j \mathbf{G}_{ij}^2} = \sqrt{\mathrm{tr}(\mathbf{G}^T\mathbf{G})}$, $\mathbf{G}$ and $\mathbf{H}$ are mode-I matricization forms of G and H, the sign $\otimes$ denotes the matrix Kronecker product, and the pre-multiplied $\mathbf{X}$, post-multiplied $(\mathbf{I}\otimes\mathbf{X}^T)$ and $(\mathbf{X}^T\otimes\mathbf{I})$ correspond to the three tensor multiplications $\times_I$, $\times_J$ and $\times_K$ respectively. Taking advantage of some properties of the Kronecker product and permutation matrix, Eq. (2) can be simplified as

$$\begin{aligned} F_2(\mathbf{X}) &= \| \mathbf{G} - \mathbf{X}\mathbf{H}(\mathbf{I}\otimes\mathbf{X}^T)(\mathbf{X}^T\otimes\mathbf{I})\|_F^2 = \| \mathbf{G} - \mathbf{X}\mathbf{H}(\mathbf{X}^T\otimes\mathbf{X}^T)\|_F^2 \\ &= \mathrm{tr}[(\mathbf{G} - \mathbf{X}\mathbf{H}(\mathbf{X}^T\otimes\mathbf{X}^T))(\mathbf{G} - \mathbf{X}\mathbf{H}(\mathbf{X}^T\otimes\mathbf{X}^T))^T] \\ &= \mathrm{tr}[\mathbf{G}\mathbf{G}^T + \mathbf{X}\mathbf{H}(\mathbf{X}^T\otimes\mathbf{X}^T)(\mathbf{X}\otimes\mathbf{X})\mathbf{H}^T\mathbf{X}^T - \mathbf{X}\mathbf{H}(\mathbf{X}^T\otimes\mathbf{X}^T)\mathbf{G}^T \\ &\quad - \mathbf{G}(\mathbf{X}\otimes\mathbf{X})\mathbf{H}^T\mathbf{X}^T] = \mathrm{tr}(\mathbf{G}\mathbf{G}^T) + \mathrm{tr}(\mathbf{X}\mathbf{H}\mathbf{H}^T\mathbf{X}^T) \\ &\quad - 2\mathrm{tr}(\mathbf{G}(\mathbf{X}\otimes\mathbf{X})\mathbf{H}^T\mathbf{X}^T), \mathbf{X}\in\mathcal{D}. \end{aligned} \tag{3}$$

Similar results can be obtained by mode-J or mode-K matricizations. By the above transformations, $F_2(\mathbf{X})$ is actually a relaxation of $F_0(\mathbf{X})$, which means that $F_2(\mathbf{X})$ and $F_0(\mathbf{X})$ are equivalent in the discrete domain $\mathcal{D}$, but generally inequivalent in the continuous domain. Optimizing $F_2(\mathbf{X})$ may slightly degrade the performance compared with directly optimizing $F_0(\mathbf{X})$, but with a tractable matrix form.

Then to calculate $\nabla F_2(\mathbf{X})$, the gradient ∇T of $T(\mathbf{X}) = \mathrm{tr}(\mathbf{G}(\mathbf{X}\otimes\mathbf{X})\mathbf{H}^T\mathbf{X}^T)$ in the third term of Eq. (3) is needed. Though it can be obtained by formula deviation, it is better to expand $\mathbf{X}\otimes\mathbf{X}$ as a sum for efficiency reasons as

$$T(\mathbf{X}) = \sum_i^N\sum_i^N (\mathbf{X}_{ij}\mathrm{tr}(\mathbf{G}_i\mathbf{X}\mathbf{H}_j^T\mathbf{X}^T)) \tag{4}$$

where $\mathbf{G}_i$ is the ith $N\times N$ submatrix in $\mathbf{G}$ shown in Fig. 1. $\mathbf{H}_j$ has similar definition. Then the gradient is

$$\nabla F_2(\mathbf{X}) = 2\mathbf{X}\mathbf{H}\mathbf{H}^T - 2\sum_i^N\sum_j^N \mathbf{S}^{ij}\mathrm{tr}(\mathbf{G}_i\mathbf{X}\mathbf{H}_j^T\mathbf{X}^T) - 2\sum_i^N\sum_j^N (\mathbf{G}_i^T\mathbf{X}\mathbf{H}_j + \mathbf{G}_i\mathbf{X}\mathbf{H}_j^T) \tag{5}$$

where $\mathbf{S}^{ij}\in\mathbb{R}^{N\times N}$ is the matrix with full of '**0**'s except one '**1**' locating at the ith row and jth column.

Next the GNCCP [24] and GA [11], two typical gradient based optimization frameworks over the set of permutation matrices, are generalized to minimize $F_2(\mathbf{X})$. Similar to the spectral decomposition algorithms, the GNCCP and GA also belong to the group of continuous methods, which typically involve relaxing the discrete problems to be continuous and then projecting the continuous solutions back to discrete domains.

Algorithm 1. Adjacency tensor matching Algorithm 1 (ATM1)

Input: Two matricized tensors **G**and **H**
  Initialize $\mathbf{X} = \mathbf{1}_{NN}/N$, $\zeta = 1$
  repeat
    repeat (Begin Frank-Wolfe iterations:)
    Compute the search direction $\mathrm{d} = \mathbf{Y} - \mathbf{X}$
    Compute the step size α
    Update $\mathbf{X} = \mathbf{X} + \alpha\mathrm{d}$
    until **X** converges or maximal iteration number is reached
    $\zeta = \zeta - d\zeta$
  until $\zeta < -1$ or $\mathbf{X}\in\mathcal{D}$
Output: A permutation matrix **X**

Specifically, the GNCCP has its root in the path following algorithm [35,25,37] which first relaxes the discrete domain to its convex hull and then gradually projects the continuous solution to the discrete domain following a path, by optimizing a series of linear combination of the convex relaxation and concave relaxation of the original objective. The GNCCP realizes the linear combination in a implicit way without needing to explicitly construct the convex or concave relaxations, which are task dependent and often difficult to construct. For the adjacency tensor matching problem, it takes the following form:

$$\begin{aligned} \min F_\zeta(\mathbf{X}) &= \begin{cases} (1-\zeta)F_2(\mathbf{X}) + \zeta\mathrm{tr}\mathbf{X}^T\mathbf{X}\mathrm{if}1 \ge \zeta \ge 0, \\ (1+\zeta)F_2(\mathbf{X}) + \zeta\mathrm{tr}\mathbf{X}^T\mathbf{X}\mathrm{if}0 > \zeta \ge -1, \end{cases} \text{ s. t. } \mathbf{X}\in C, C \\ &:= \{\mathbf{X} | \sum_i^N \mathbf{X}_{ij} = 1, \sum_j^N \mathbf{X}_{ij} = 1, \mathbf{X}_{ij}\ge 0, \forall\, i,j,\} \end{aligned} \tag{6}$$

where C, known as Birkhoff polytope, is the convex hull of $\mathcal{D}$. As ζ decreases from 1 to 0 (graduated nonconvexity) and to -1 (graduated concavity), the minimum point is finally pushed into $\mathcal{D}$ which is the extreme point set of C. The GNCCP based optimization is summarized in Algorithm 1. For each specific ζ, $F_\zeta(\mathbf{X})$ is minimized by the simple yet effective Frank-Wolfe (FW) algorithm [10,12], also known as conditional gradient method. The gradient $\nabla F_\zeta(\mathbf{X})$ is given as follows:

$$\nabla F_\zeta(\mathbf{X}) = \begin{cases} (1-\zeta)\nabla F_2(\mathbf{X}) + 2\zeta\mathbf{X}\mathrm{if}1 \ge \zeta \ge 0, \\ (1+\zeta)\nabla F_2(\mathbf{X}) + 2\zeta\mathbf{X}\mathrm{if}0 > \zeta \ge -1, \end{cases} \mathbf{X}\in C. \tag{7}$$

Please see Appendix B for some implementation details.

The GA [11,32] is also a continuous method with an annealing parameter to gradually discretize the softassign matrix. Based on GA, the adjacency tensor matching is summarized in Algorithm 2. The key step in each iteration is a softassign procedure incorporating the mapping constraints, which is realized by Sinkhorn's bistochastic normalization [30] (repeated row and column normalizations). The gradient $\nabla F_2(\mathbf{X})$ is given in Eq. (5).

Algorithm 2. Adjacency tensor matching Algorithm 2 (ATM2)

Input: Two matricized tensors **G**and **H**
  Initialize $\mathbf{X} = \mathbf{1}_{NN}/N$, $\beta = \beta_0$
  repeat
    repeat
      Compute the gradient $\nabla F_2(\mathbf{X})$

Update $\mathbf{X} = \exp(\beta \nabla F_2(\mathbf{X}))$
repeat (Begin bistochastic normalization:)
Normalize across rows: $\mathbf{X}'_{ij} = \frac{\mathbf{X}_{ij}}{\sum_j \mathbf{X}_{ij}}$
Normalize across rows: $\mathbf{X}'_{ij} = \frac{\mathbf{X}_{ij}}{\sum_i \mathbf{X}_{ij}}$
until $\mathbf{X}'$ Converges or maximal iteration number is reached
until $\mathbf{X}$ converges or maximal iteration number is reached
$\beta = \rho\beta$, $\rho > 1$
until $\beta > \beta_{max}$ or $\mathbf{X} \in \mathcal{D}$
Output: A permutation matrix $\mathbf{X}$

3. Discussions and extensions

3.1. Relations and differences between adjacency tensor and affinity tensor

In this subsection we intend to discuss the relations and differences between affinity tensor $\mathbf{A}$ based objective and ours. The former objective usually takes the following form

$$\max F_3(\mathbf{x}) = \mathbf{A} \otimes_1 \mathbf{x} \otimes_J \mathbf{x} \otimes_K \mathbf{x}, \tag{8}$$

where $\mathbf{A} \in \mathbb{R}^{NN \times NN \times NN}$ and $\mathbf{x} \in \{0, 1\}^{NN}$ is the row-wise vectorized replicas of $\mathbf{X}$[1]
$\mathbf{A}$ is usually constructed by projecting a dissimilarity tensor $\mathbf{D}$ into the interval [0, 1] by for example $\mathbf{A} = \exp(-\mathbf{D})$ where

$$\mathbf{D}_{i'j'k'} = \mathbf{D}_{(i_G-1)N+i_H,(j_G-1)N+j_H,(k_G-1)N+k_H} = \mathbf{d}(G_{i_G j_G k_G}, T_{i_H j_H k_H}). \tag{9}$$

Without considering the nonlinear projection effect of $\exp(\cdot)$, Eq. (8) is equivalent to

$$\min F_4(\mathbf{x}) = \mathbf{D} \otimes_I \mathbf{x} \otimes_J \mathbf{x} \otimes_K \mathbf{x}. \tag{10}$$

We show below that Eq. (10) is equivalent to Eq. (1) in case

$$\mathbf{d}(G_{i_G j_G k_G}, T_{i_H j_H k_H}) = (G_{i_G j_G k_G} - T_{i_H j_H k_H})^2, \tag{11}$$

which is a frequently used dissimilarity measure. With the assignment notation $i_H = c(i_G)$ shown in Fig. 1, it can be easily checked by expanding both formulas as

$$\begin{aligned} \mathbf{D} \otimes_I \mathbf{x} \otimes_J \mathbf{x} \otimes_K \mathbf{x} &= \sum_i^N \sum_j^N \sum_k^N \mathbf{d}(G_{ijk}, H_{c(i)c(j)c(k)}) \\ &= \sum_i^N \sum_j^N \sum_k^N (G_{ijk} - H_{c(i)c(j)c(k)})^2, \end{aligned} \tag{12}$$

and

$$\begin{aligned} \| G - H \times_I \mathbf{X} \times_J \mathbf{X} \times_K \mathbf{X} \|^2 &= \| G - \{H_{c(i)c(j)c(k)}\}^{N \times N \times N} \|^2 \\ &= \sum_i^N \sum_j^N \sum_k^N (G_{ijk} - H_{c(i)c(j)c(k)})^2. \end{aligned} \tag{13}$$

Therefore, Eq. (1) could be treated as a special case of Eq. (10).

On the other hand, the affinity tensor pre-computes the affinities of all triplet combinations between graphs and stores them, while the adjacency tensor only needs to store the triplet descriptors. In general, the affinity tensor is more flexible on the dissimilarity measure, while the adjacency tensor enjoys a much smaller storage load, on which a detailed discussion is given in next subsection.

[1] Following the definition of $\otimes_M$ in Appendix A, it should be $F_3(\mathbf{x}) = \mathbf{A} \otimes_I \mathbf{x}^T \otimes_J \mathbf{x}^T \otimes_K \mathbf{x}^T$. However, given $\mathbf{x}$ is a vector, the transpose symbol is usually omitted [15].

3.2. Storage complexity analysis

The proposed adjacency tensor matching algorithm involves an $O(N^3)$ storage complexity in the worst case, when the graph is constructed in a complete manner, that is any three different vertices are connected by a triplet. In this case, the affinity tensor based algorithms would theoretically involve a storage complexity as high as $O(N^6)$ [4].

Fortunately, in practical tasks the affinity tensor could be very sparse by elaborately designing the graph structure [22,26] or directly applying sparsification techniques to it [4]. Given the triplet numbers P and Q respectively in $\mathcal{G}$ and $\mathcal{H}$, the affinity tensor based algorithms would at least involve a $\max(O(PQ), O(N^2))$ storage complexity, while that of the proposed adjacency tensor based algorithm would be $\max(O(P), O(Q), O(N^2))$, usually much smaller than the previous one in real application.[2] In the two complexity measures, $O(PQ)$ is related to sparse affinity tensor, $O(P)$ and $O(Q)$ are related the sparse adjacency tensors, and $O(N^2)$ is related to the matrix $\mathbf{X} \in C$ (or vector $\mathbf{x}$ in Eq. (8)) and other intermediate matrices.

3.3. Multigraph situation, the first and second order constraints

It is straightforward to extend the proposed method to multigraph, in which each hyper-edge is assigned with more than one types of triplet descriptors. It is realized by the convex combination of the objectives for all types of descriptors as

$$F_5(\mathbf{X}) = \sum_i \lambda_i \| G^i - H^i \times_I \mathbf{X} \times_J \mathbf{X} \times_K \mathbf{X} \|^2, \quad \sum_i \lambda_i = 1. \tag{14}$$

and the gradient is modified accordingly.

It is also convenient to include the first and second order constraints in the proposed objective as

$$\begin{aligned} F_6(\mathbf{X}) = \mu_1 F_2(\mathbf{X}) &+ \mu_2 \mathrm{tr}((A_G \mathbf{X} - \mathbf{X} A_H)^T (A_G \mathbf{X} - \mathbf{X} A_H)) \\ &+ (1 - \mu_1 - \mu_2) \mathrm{tr}(\mathbf{C}^T \mathbf{X}) \end{aligned} \tag{15}$$

$A_G, A_H \in \mathbb{R}^{N \times N}$ are the adjacency matrices storing pairwise descriptors for graphs $\mathcal{G}$ and $\mathcal{H}$ respectively [25,35], and C in the unary term is a cost matrix with c_{ij} denoting the difference between appearance descriptors of i in $\mathcal{G}$ and j in $\mathcal{H}$.

The proposed optimization algorithms can be directly applied to $F_6(\mathbf{X})$ where the gradient is given as

$$\begin{aligned} \nabla F_6(\mathbf{X}) = \mu_1 \nabla F_2(\mathbf{X}) &+ 2\mu_2 (A_G^2 \mathbf{X} - A_G^T \mathbf{X} A_H^T - A_G \mathbf{X} A_H + \mathbf{X} A_H^2) \\ &+ (1 - \mu_1 - \mu_2) \mathbf{C}. \end{aligned} \tag{16}$$

4. Simulations

In this section, we evaluate the proposed algorithms by comparing them with some state-of-the-art algorithms in both synthetic point matching and real image matching, and also validate their applicability on the directed graph. Though incorporating the first and second order constraints may improve matching performance, they are not included in the following experiments because we want to focus on the comparison of high order algorithms.

The algorithms included for comparison are: probabilistic hyper-graph matching (PHM) [36], tensor matching (TM) [8], efficient high-order matching (EHM) [4], integer projected fixed point method for third-order graph matching (IPFP-3D) [22], reweighted random walks for hyper-graph matching (RRWHM) [18]. The algorithms PHM, TM,

[2] Please see Appendix C for the comparison between $\max(O(P), O(Q), O(N^2))$ and $\max(O(PQ), O(N^2))$. The key point of the comparison is that to guarantee every vertex being associated with at least one triplet, the triplet number P and Q must increase super-linear, or at least linearly, with respect to N.

RRWM, and IPFP-3D are implemented by public codes,[3] and EHM is implemented by ourselves. Some default parameters for the proposed algorithms ATM1 and ATM2 are $d\zeta = 0.001$, $\beta = 0.5$, $\rho = 1.075$ and $\beta_{max} = 20$.

Both *matching error* and *matching accuracy* are used as comparison criterions, where *matching error* refers to the objective value calculated by Eq. (1) or Eq. (10). In the *matching error* comparison, the affinity tensor and adjacency tensor are constructed in an equivalent way based on the same triplet dissimilarity measure Eq. (11). In the *matching accuracy* comparison, to avoid any underestimate of the affinity tensor based algorithms, the affinity tensor is also constructed following the way in each paper [36,8,4,18,22], which typically involves the exponential scaling operator $\exp(\cdot)$ in computing the triplet dissimilarity. Accordingly, the related algorithms are denoted by appending "-E", e.g. PHM-E.

4.1. Synthetic point matching

The synthetic points are generated following a similar way as in [5,8]. The first point set of size N is randomly and uniformly sampled in 2D-plane, and then it is permutated by a randomly generated permutation matrix and disturbed by Gaussian noise $\mathcal{N}(0, \sigma^2)$ to get the second point set. The undirected triplet descriptor is defined by

$$G^1_{ijk} = \frac{1}{6}\left(\frac{d_{ij}+d_{ik}}{d_{jk}} + \frac{d_{ij}+d_{jk}}{d_{ik}} + \frac{d_{ik}+d_{jk}}{d_{ij}}\right), \tag{17}$$

where d_{ij} denotes the distance between vertices i and j. The graph structure is either full connection or sparse connection where the triplet density for the latter one is fixed as 0.1.

The performance is compared with respect to σ and N. In the full connection situation, σ is increased from 0 to 0.1 by a step size 0.01, and N is increased from 10 to 40 by a step size 3. However, for the affinity tensor based algorithms our RAM capacity (2 GB) could at most afford the matching of 16 points, so the comparison is further carried out as N is increased from 10 to 16 by a step size 1. In the sparse connection situation, σ is increased from 0 to 0.2 by a step size 0.02. The noise also disturbs the sparse graph structure following a similar way as in [35], by randomly adding and removing $\frac{1}{2}\sigma\#Tri$ triplets to each sparse graph, where $\#Tri$ denotes the number of triplets. N is still increased from 10 to 40 by a step size 3, while for the affinity tensor based algorithms N is increased to at most 34.

The results are depicted in Fig. 2. It can be observed that when using the same triplet dissimilarity Eq. (11), the proposed method ATM1 outperforms the affinity tensor based algorithms from both the matching accuracy and matching error aspects. By introducing the nonlinear scaling operator $exp(\cdot)$, significant matching accuracy improvements for all the affinity tensor based algorithms are obtained, which may be attributed to the better discriminative ability of such a nonlinear transformation than using only the least-square measure Eq. (11). In this case, ATM1 and IPFP-3D-E achieve comparable performance, and respectively outperform ATM2 and other affinity tensor based algorithms. Their superiorities may be due to the soft projection strategy in their optimization processes.

The running time is compared in Fig. 3, in which the growth rates indicate the computational complexity. we can observe that the running times of both ATM1 and ATM2 grow slower than the other five competitors. Specifically, in the full connection situation, the growth rates are generally 3.8 ± 0.5 for ATM1 and ATM2, and 6.1 ± 0.5 for PHM, TM, EHM, IPFP-3D and RRWHM, while in the sparse connection situation, the growth rates are respectively 3.5 ± 0.5 and 5.7 ± 0.5.

[3] Codes for PHM and RRWHM are available at cv.snu.ac.kr/research/~RRWHM/. Codes for TM are available at www.cs.cmu.edu/olivierd/. Codes for IPFP-3D are kindly provided by Prof. Leordeanu (leordeanu@gmail.com), the first author of [22].

4.2. Real image matching

The real image matching experiment is first carried out on the *House* sequence.[4] The dataset contains 111 frames sampled from a 3D-rotating *House* video clip, and a larger frame number separation implies in general a more difficult matching task. Each frame is manually labeled with 30 points as in [37]. Besides Eq. (17), a second undirected triplet descriptor is defined as follows:

$$G^2_{ijk} = \frac{1}{3}(sin(\angle i) + sin(\angle j) + sin(\angle k)). \tag{18}$$

Either type of descriptors are normalized by a global constant to make them range from 0 to 1, and the descriptor weights are set to be equal, i.e. $\lambda_1 = \lambda_2 = 0.5$. The graph structure is constructed by Delaunay triangulation as in [22,37]. In Fig. 4, the matching accuracy and matching error are respectively compared with respect to frame number separation, which witnesses some algorithms, including ATM1, achieve perfect matching performance even when the frame number separation is 100. Two typical *House* matching samples are also provided in Fig. 4.

The comparisons are also carried out on a benchmark matching dataset of real-world images [21], which contains 20 pairs of *Motorbike* images and 30 pairs of *Car* images taken from Pascal 2007 Challenge. These images are labeled with $15 \sim 52$ ground truth assignments. The descriptor and graph structure settings are the same with *House* matching. The average matching accuracy and matching error are summarized in Table 1, which witness the state-of-the-art performance of ATM1. Besides, the inconsistency between matching accuracy and matching error is observed. For instance, in *Motorbike* matching TM achieves lower matching error than EHM, while its matching accuracy is unexpectedly lower than the latter one. Besides the influence of massive local optimum points, another possible reason for the inconsistency is that the manually labeled ground truth assignments do not always correspond to the lowest matching error. Some matching samples obtained by the proposed algorithms are shown in Fig. 5.

4.3. Directed graph matching

It has been observed that the directed graph owns more invariance against geometric transformations in the second order graph matching [34,38]. We next show that a similar conclusion could be drawn for the third order graph matching. Since TM, EHM, IPFP-3D and RRWHM are inapplicable to the directed graph, only PHM, ATM1 and ATM2 are applied to the directed graph. A handwritten Chinese character dataset is used, which contains four characters with each one consisting of 10 samples [23], as illustrated in Fig. 6. For each character, we manually label 28, 23, 28, 23 ground truth points respectively and build the graph structure roughly according to the character skeletons. $G^{1'}_{ijk} = \frac{d_{ij}}{d_{jk}}$ and $G^{2'}_{ijk} = \sin(\angle j)$ are utilized as the directed descriptors, and the undirected descriptors are the same with the above experiment. The descriptor weights are $\lambda_1 = \lambda_2 = 0.5$. All the $4 \times C^2_{10}$ possible matching pairs are tested on, and the results are given in Table 2 and 3, with some typical matching samples shown in Fig. 7. Despite the same algorithms, ATM1 and ATM2 in general both achieve considerable performance improvements by the directed graph, compared with the undirected graph.

5. Conclusions

A novel graph matching method using the third order constraints is proposed to tackle point correspondence problem. Compared with previous hyper-graph matching algorithms, it achieves considerable memory expense reduction and is applicable to both undirected and

[4] Available at http://vasc.ri.cmu.edu//idb/html/motion/house/index.html.

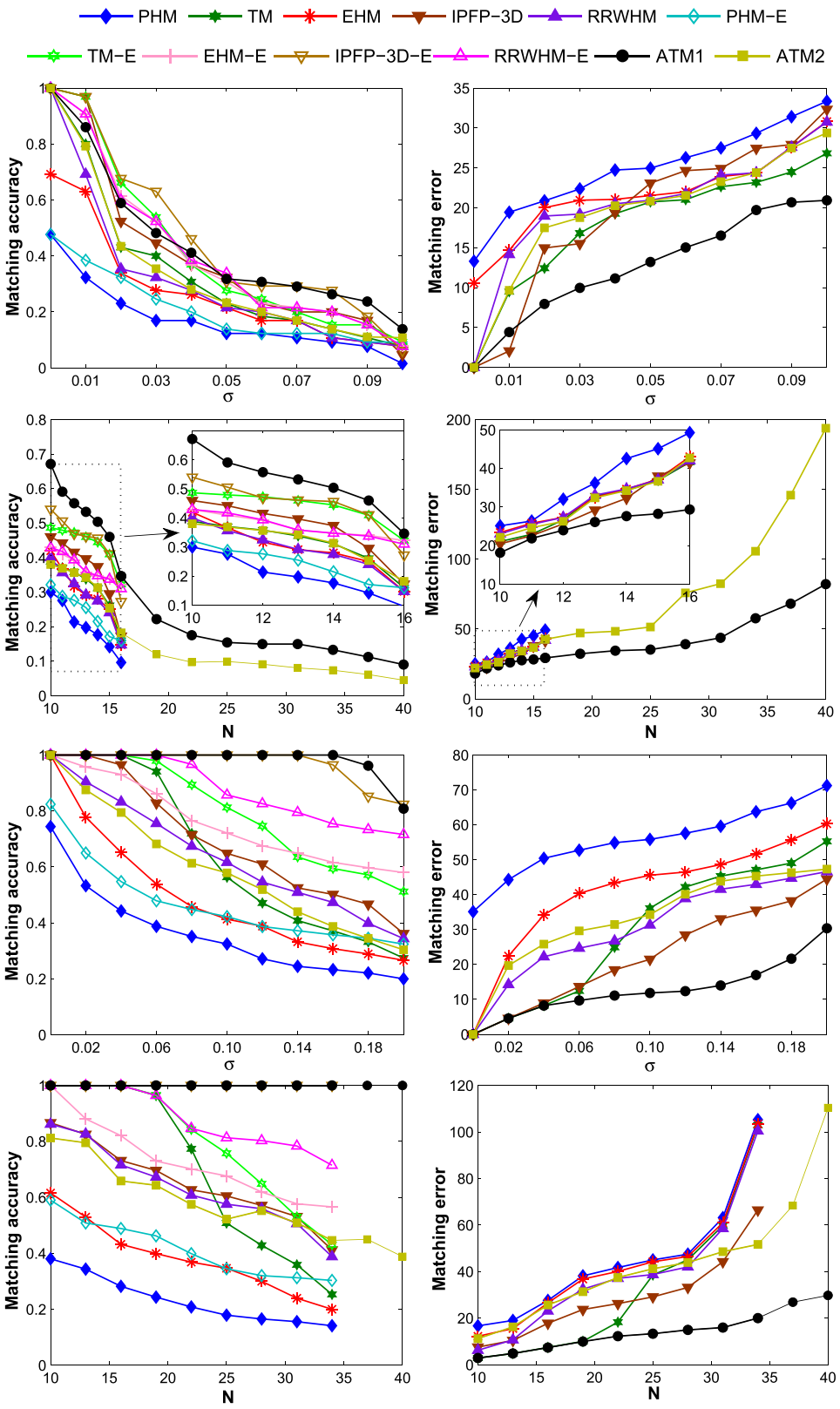

Fig. 2. Synthetic point matching results. The matching accuracy and matching error are compared with respect to noise level and problem scale. The upper two rows and the lower two rows respectively correspond to full and sparse connection situations.

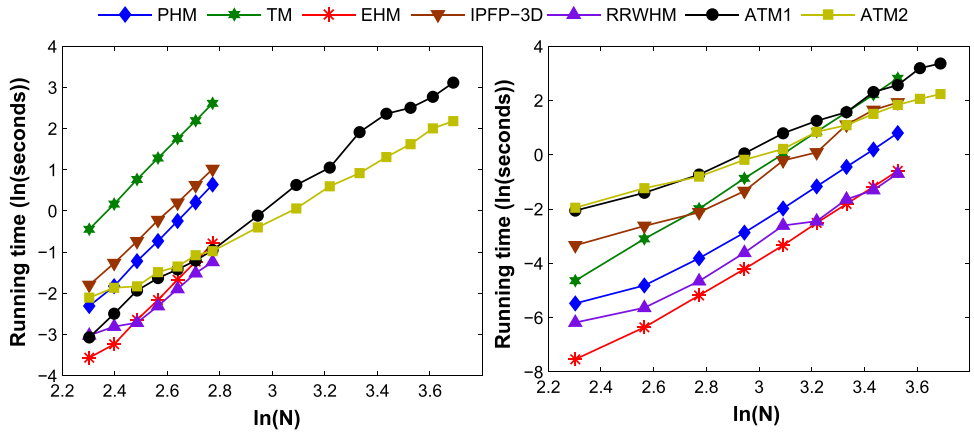

Fig. 3. Running time comparison results. The left plot and right plot respectively correspond to full and sparse connection situations.

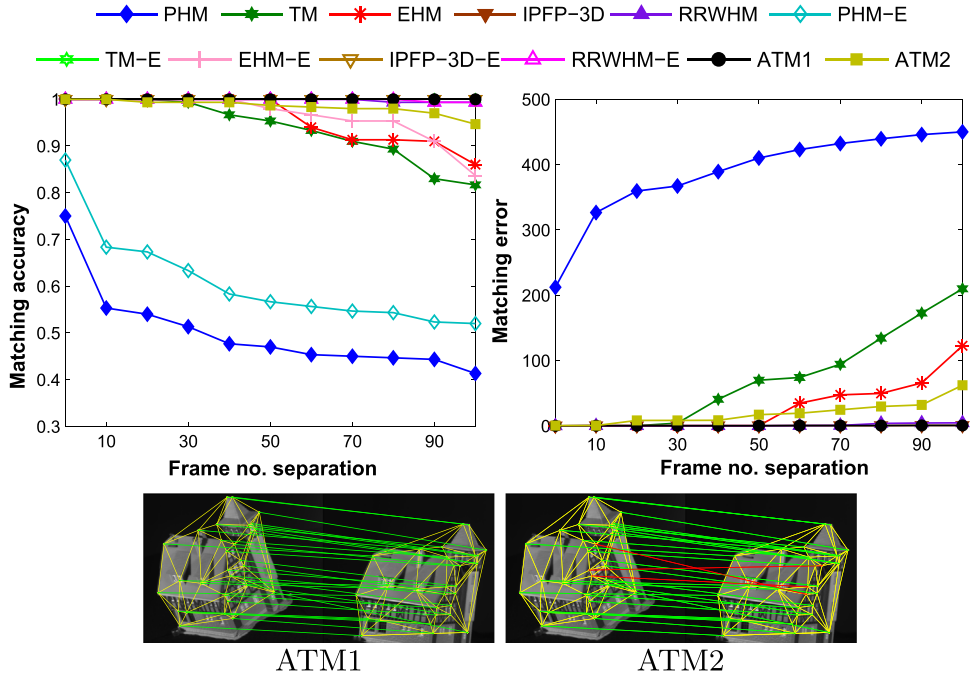

Fig. 4. *House* matching results. In the upper row, the matching accuracy and matching error are respectively compared with respect frame number separation. Typical *House* matching samples are in the lower row, where green/red lines denote correct/wrong assignments, and yellow lines denote graph structures. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

Table 1
Average matching accuracy and matching error in *Motorbike* and *Car* matching.

Motorbike		PHM	TM	EHM	IPFP-3D	RRWHM	ATM1
	$\mathcal{A}$	0.439	0.799	0.812	0.954	0.956	**1.000**
	$\mathcal{E}$	436.32	200.06	183.63	11.03	26.06	**0.97**
		PHM-E	TM-E	EHM-E	IPFP-3D-E	RRWHM-E	ATM2
	$\mathcal{A}$	0.508	0.964	0.899	**1.000**	0.979	0.965
	$\mathcal{E}$	–	–	–	–	–	16.83
Car		PHM	TM	EHM	IPFP-3D	RRWHM	ATM1
	$\mathcal{A}$	0.411	0.732	0.716	0.959	0.960	**1.000**
	$\mathcal{E}$	548.64	292.50	307.74	17.39	22.37	**1.31**
		PHM-E	TM-E	EHM-E	IPFP-3D-E	RRWHM-E	ATM2
	$\mathcal{A}$	0.424	0.950	0.894	**1.000**	**1.000**	0.973
	$\mathcal{E}$	–	–	–	–	–	18.51

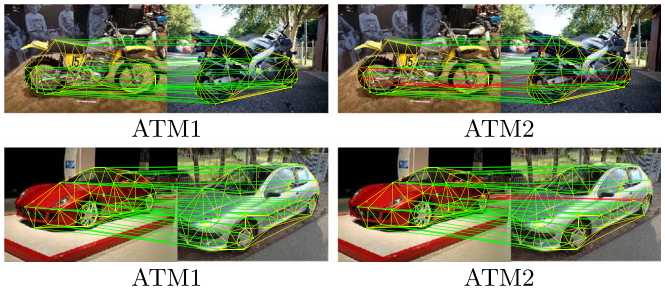

Fig. 5. Typical *Motorbike* matching and *Car* matching samples. Green/red lines denote correct/wrong assignments, and yellow lines denote graph structures. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

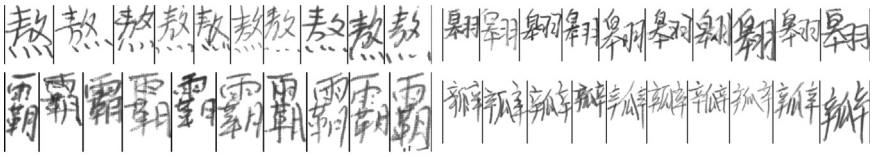

Fig. 6. The handwritten Chinese character dataset.

Table 2
Average matching accuracy and matching error in handwritten Chinese character matching: on undirected graph.

C1		PHM	TM	EHM	IPFP-3D	RRWHM	ATM1
	$\mathcal{A}$	0.445	0.735	0.736	0.835	0.725	**0.845**
	$\mathcal{E}$	323.44	77.74	71.07	16.13	64.15	**15.85**
		PHM-E	TM-E	EHM-E	IPFP-3D-E	RRWHM-E	ATM2
	$\mathcal{A}$	0.451	0.837	0.767	0.841	0.805	0.713
	$\mathcal{E}$	–	–	–	–	–	81.74
C2		PHM	TM	EHM	IPFP-3D	RRWHM	ATM1
	$\mathcal{A}$	0.460	0.762	0.746	0.852	0.732	**0.893**
	$\mathcal{E}$	239.34	64.17	104.94	18.83	64.91	**7.99**
		PHM-E	TM-E	EHM-E	IPFP-3D-E	RRWHM-E	ATM2
	$\mathcal{A}$	0.473	0.889	0.775	0.883	0.851	0.714
	$\mathcal{E}$	–	–	–	–	–	68.69
C3		PHM	TM	EHM	IPFP-3D	RRWHM	ATM1
	$\mathcal{A}$	0.531	0.876	0.834	0.960	0.841	0.949
	$\mathcal{E}$	460.12	153.62	157.27	**21.09**	158.32	25.31
		PHM-E	TM-E	EHM-E	IPFP-3D-E	RRWHM-E	ATM2
	$\mathcal{A}$	0.564	0.954	0.879	**0.961**	0.864	0.852
	$\mathcal{E}$	–	–	–	–	–	158.24
C4		PHM	TM	EHM	IPFP-3D	RRWHM	ATM1
	$\mathcal{A}$	0.621	0.788	0.803	0.988	0.877	**0.999**
	$\mathcal{E}$	252.33	141.84	138.63	2.47	32.15	**1.61**
		PHM-E	TM-E	EHM-E	IPFP-3D-E	RRWHM-E	ATM2
	$\mathcal{A}$	0.640	0.992	0.889	0.996	0.946	0.900
	$\mathcal{E}$	–	–	–	–	–	21.52

Table 3
Average matching accuracy and matching error in handwritten Chinese character matching: on directed graph.

		PHM	PHM-E	ATM1	ATM2
C1	$\mathcal{A}$	0.519	0.544	**0.959**	0.929
	$\mathcal{E}$	174.89	–	**9.90**	10.85
C2	$\mathcal{A}$	0.657	0.745	**0.915**	0.909
	$\mathcal{E}$	121.80	–	**4.63**	5.69
C3	$\mathcal{A}$	0.663	0.682	**0.992**	0.907
	$\mathcal{E}$	199.05	–	**4.81**	18.98
C4	$\mathcal{A}$	0.814	0.821	**0.998**	0.956
	$\mathcal{E}$	88.23	–	**1.42**	1.96

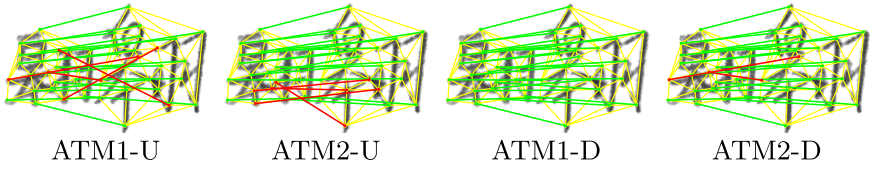

Fig. 7. Typical handwritten Chinese character matching results. Green/red lines denote correct/wrong assignments, and yellow lines denote graph structures. Appending "-U"/"-D" means undirected/directed graph matching. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

directed graphs. Different from the affinity tensor based algorithms, the problem is formulated by matching two adjacency tensors and solved by two gradient based optimization algorithms. The effectiveness of the proposed scheme is verified by experiments on both synthetic and real data.

Acknowledgement

The authors would like to thank the associate editor and anonymous reviewers whose comments have greatly improved the manuscript. This work is supported partly by the National Key Research and

Development Plan of China (Grant 2016YFC0300801), partly by the National Natural Science Foundation (NSFC) of China (Grants 61503383, 61633009, 61375005, U1613213, 61303174, 61210009, and 61305137), and partly by the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant XDB02080003).

Appendix A. Tensor notations and operations used in the paper

We adopt similar notations as in [14,15]. A tensor is an M-dimensional array or formally defined as the tensor product of M vector spaces. The number of tensor modes (dimensions) is called order. The first and second order tensors are called vector and matrix. In this paper we focus on the third order tensor abbreviated as *tensor* hereafter, of which an example is given in Fig. 1 in the main manuscript.

Notations: given mode (dimension)-I, J, K, an item in a tensor $T \in \mathbb{R}^{I\times J\times K}$ is denoted by T_{ijk}. A colon is used to indicate all the items in one mode. For example, $T_{:jk}$ is a vector and $T_{::k}$ is a matrix. The mode-M, $M = \{I, J, K\}$ fiber is a generalized concept of row (column) in matrix, which is defined by fixing all the mode coordinates but mode-M. For example, $T_{:jk}$ denotes a mode-I fiber.

Operations: *Matricization* is the process that reshapes a tensor into a matrix. We focus on the mode-M matricization which rearranges the mode-M fibers to be the columns of a matrix. Instead of formal definition, we give the following example to illustrate the concept. Let a tensor $T \in \mathbb{R}^{2\times 2\times 2}$ be $T_{::1} = \left[\begin{smallmatrix}1 & 3\\ 2 & 4\end{smallmatrix}\right]$, $T_{::2} = \left[\begin{smallmatrix}5 & 7\\ 6 & 8\end{smallmatrix}\right]$, and then the mode-$I$ matricization of T is $\mathbf{M}_I = \left[\begin{smallmatrix}1 & 3 & 5 & 7\\ 2 & 4 & 6 & 8\end{smallmatrix}\right]$. Similarly, $\mathbf{M}_J = \left[\begin{smallmatrix}1 & 2 & 5 & 6\\ 3 & 4 & 7 & 8\end{smallmatrix}\right]$, $\mathbf{M}_K = \left[\begin{smallmatrix}1 & 2 & 3 & 4\\ 5 & 6 & 7 & 8\end{smallmatrix}\right]$. Tensor norm $\|\cdot\|$ is defined by $\| T \| = \sqrt{\sum_i \sum_j \sum_k T_{ijk}^2}$. Tensor multiplication, or say the mode-M product of a tensor T and a matrix U, is denoted by $T \otimes_M U$ and defined by pre-multiplying each mode-M fiber with U. It can be explained with the help of the tensor matricization as

$$B = A \times_M U \quad \Leftrightarrow \quad \mathbf{B}_M = U\mathbf{A}_M \tag{A.1}$$

where $\mathbf{A}_M$ and $\mathbf{B}_M$ are the mode-M matricization forms of tensors A and B respectively.

Appendix B. Some implementation details

In each iteration, once known the gradient ∇F_ζ, the main operation is to find the optimal descent direction $\mathrm{d} = \mathbf{Y} - \mathbf{X}$ with $\mathbf{Y}$:=argmin$\langle \mathbf{Y}, \nabla F_\zeta(\mathbf{X})\rangle$, $\mathbf{Y} \in C$ where $\langle\cdot\rangle$ denotes matrix Frobenius inner product. The linear programming problem can be efficiently solved by the Hungarian algorithm [16]. The step size α can be determined by inexact line search, e.g. backtracking method, or directly by the iteration serial number l, e.g. $\alpha = \frac{2}{l+2}$. The termination criterion can be constructed as **if** $\langle \mathbf{Y} - \mathbf{X}, -\nabla F_\zeta(\mathbf{X})\rangle < \varepsilon |F_\zeta(\mathbf{X}) - \langle \mathbf{Y} - \mathbf{X}, -\nabla F_\zeta(\mathbf{X})\rangle|$, **or** the maximal iteration number L is reached, **then terminate**, where ε is a small number.

Appendix C. Comparison between $\max(O(P), O(Q), O(N^2))$ and $\max(O(PQ), O(N^2))$

On sparse hyper-graphs, the proposed algorithm involves a $S_{adj} = \max(O(P), O(Q), O(N^2))$ storage complexity[5] given the vertex number N and triplet numbers P and Q of two hyper-graphs. It is still smaller, at least not greater than the storage complexity $S_{aff} = \max(O(PQ), O(N^2))$ of affinity tensor based algorithms, i.e. $S_{adj} \le S_{aff}$.

The comparison between S_{adj} and S_{aff} mainly depends on the increase rate of triplet number $P(N)$ ($Q(N)$) w.r.t. vertex number N. Supposing $P(N) = O(Q(N))$, if $P(N)$ increases sub-linearly w.r.t. N, i.e. $P(N) < O(N)$, there is $S_{adj} = O(N^2) = S_{aff}$. Once $P(N) \ge O(N)$, the term $O(PQ)$ takes effect and there is $S_{adj} \le S_{aff} = O(PQ)$. On the other hand, by generalizing the concept of degree to *hyper-degree* (third order for convenience), i.e., the number of triplets incident to a vertex, a hyper-degree sum formula can be straightforward generalized as follows:

$$3P(N) = \sum_i^N d^H(i), \tag{C.1}$$

where $d^H(i)$ denotes the hyper-degree of vertex i. To guarantee every vertex being associated with at least a triplet, there must be $d^H(i) \ge 1$, $\forall\, i$, and thus

$$3P(N) = \sum_i^N d^H(i) \ge N, \tag{C.2}$$

which implies

$$P(N) \ge O(N), \tag{C.3}$$

and thus we have $S_{adj} \le S_{aff}$. For instance, if $P(N) = O(N^2)$, there is $S_{adj} = O(N^2) < S_{aff} = O(N^4)$.

The key point of the above discussion is that to guarantee every vertex being associated with at least one triplet, $P(N)$ must increase super-linear, or at least linearly, w.r.t. N. Since triplets are often generated from second order edges, below we also try to make an investigation on triplet formation from random undirected second order graphs, and obtain a similar point from the statistic sense. Specifically, it begins with the degree sum formula:

$$2|E| = \sum_i^N d(i), \tag{C.4}$$

where $d(i)$ denotes the degree of vertex i and $|E|$ denotes the edge number. By denoting the average degree as $D(N)$, there is

[5] As commonly used, $F(N) = O(G(N))$ is used to represent $F(N) \in O(G(N))$ for convenience. Below a similar usage $F(N) \le O(G(N))$ implies a smaller or equal complexity of $F(N)$ than $G(N)$.

$$2|E| = D(N)N. \tag{C.5}$$

Based on the random graph assumption, the probability for an arbitrary triplet ABC (A, B, and C are different vertices) $Pr(ABC)$ is

$$Pr(ABC) = Pr(AB, BC, CA) = Pr(AB)Pr(BC)Pr(CA) = Pr^3(AB), \tag{C.6}$$

where the edge probability $Pr(AB)$ is given by

$$Pr(AB) = \frac{|E|}{C_N^2} = \frac{D(N)}{N-1} \tag{C.7}$$

The combination ${C_N}^2$ refers to the number of all possible edges.[6] Thus $P(N)$ (with a slight abuse of notation w.r.t. its estimate $\widehat{P}$) is

$$P(N) = C_N^3 Pr(ABC) = \frac{N(N-1)(N-2)}{6(N-1)^3} D^3(N) = O(D^3(N)). \tag{C.8}$$

where ${C_N}^3$ refers to the number of all possible triplets. On the other hand, since for two edges incident to a vertex, the probability of them being in a triplet is the edge probability $Pr(AB)$, to guarantee a vertex being associated with at least a triplet, from the statistical sense there must be

$$C_{D(N)}^2 Pr(AB) \geq 1. \tag{C.9}$$

where $C_{D(N)}^2$ refers to the number of all possible pairs of edges incident to the vertex. Then there is

$$D^2(N)(D(N)-1) \geq 2(N-1), \tag{C.10}$$

which implies

$$D(N) \geq O(N^{\frac{1}{3}}) \tag{C.11}$$

and then based on Eq. (C.8) it finally deduces Eq. (C.3), i.e.

$$P(N) \geq O(N). \tag{C.12}$$

When $D(N) = O(N^{\frac{1}{3}})$, there is $P(N) = O(N)$ and $S_{adj} = O(N^2) = S_{aff}$. However, once $D(N)$ becomes larger, $O(PQ)$ would be much greater. For instance, when $D(N) = O(N^{\frac{2}{3}})$, there is $P(N) = O(N^2)$ and $S_{adj} = O(P) = O(N^2) < S_{aff} = O(PQ) = O(N^4)$.

References

[1] A.C. Berg, T.L. Berg, J. Malik, Shape matching and object recognition using low distortion correspondences, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005, pp. 26–33.

[2] H. Bunke, Error correcting graph matching: on the influence of the underlying cost function, IEEE Trans. Pattern Anal. Mach. Intell. 21 (9) (1999) 917–922.

[3] H. Bunke, G. Allermann, Inexact graph matching for structural pattern recognition, Pattern Recognit. Lett. 1 (4) (1983) 245–253.

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

[5] M. Cho, J. Lee, K.M. Lee, Reweighted random walks for graph matching, in: Proceedings of the European Conference on Computer Vision, 2010, pp. 492–505.

[6] T. Cour, J.B. Shi, Solving markov random fields with spectral relaxation, in: Proceedings of the International Conference on Artificial Intelligence and Statistics, 2007, pp. 75–82.

[7] T. Cour, P. Srinivasan, J.B. Shi, Balanced graph matching, in: Proceedings of the Advances in Neural Information Processing Systems, 2006, pp. 313–320.

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

[9] A. Egozi, Y. Keller, H. Guterman, A probabilistic approach to spectral graph matching, IEEE Trans. Pattern Anal. Mach. Intell. 35 (1) (2013) 18–27.

[10] M. Frank, P. Wolfe, An algorithm for quadratic programming, Nav. Res. Logist. Q. 3 (1–2) (1956) 95–110.

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

[12] M. Jaggi, Revisiting Frank-Wolfe: projection-free sparse convex optimization, in: Proceedings of the International Conference on Machine Learning, 2013, pp. 427–435.

[13] R. Jonker, A. Volgenant, A shortest augmenting path algorithm for dense and sparse linear assignment problems, Computing 38 (4) (1987) 325–340.

[14] H.A.L. Kiers, Towards a standardized notation and terminology in multiway analysis, J. Chemom. 14 (3) (2000) 105–122.

[15] T.G. Kolda, B.W. Bader, Tensor decompositions and applications, SIAM Rev. 51 (3) (2009) 455–500.

[16] H.W. Kuhn, The hungarian method for the assignment problem, Nav. Res. Logist. Q. 2 (1–2) (1955) 83–97.

[17] S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: spatial pyramid matching for recognizing natural scene categories, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, volume 2, 2006, pp. 2169–2178.

[18] J. Lee, M. Cho, K.M. Lee, Hyper-graph matching via reweighted random walks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2011, pp. 1633–1640.

[19] M. Leordeanu, M. Hebert, A spectral technique for correspondence problems using pairwise constraints, in: Proceedings of the IEEE International Conference on Computer Vision, 2005, pp. 1482–1489.

[20] M. Leordeanu, M. Hebert, Efficient map approximation for dense energy functions, in: Proceedings of the International Conference on Machine learning, 2006, pp. 545–552.

[21] M. Leordeanu, R. Sukthankar, M. Hebert, Unsupervised learning for graph matching, Int. J. Comput. Vis. 96 (1) (2012) 28–45.

[22] M. Leordeanu, A. Zanfir, C. Sminchisescu, Semi-supervised learning and optimization for hypergraph matching, in: Proceedings of the IEEE International Conference on Computer Vision, 2011, pp. 2274–2281.

[23] C.L. Liu, F. Yin, D.H. Wang, Q.F. Wang, CASIA online and offline chinese handwriting databases, in: Proceedings of the International Conference on Document Analysis and Recognition, 2011, pp. 37–41.

[24] Z.Y. Liu, H. Qiao, GNCCP – graduated nonconvexity and concavity procedure, IEEE Trans. Pattern Anal. Mach. Intell. 36 (6) (2014) 1258–1267.

[25] Z.Y. Liu, H. Qiao, L. Xu, An extended path following algorithm for graph-matching problem, IEEE Trans. Pattern Anal. Mach. Intell. 34 (7) (2012) 1451–1456.

[26] S. Park, S.K. Park, M. Hebert, Fast and scalable approximate spectral matching for higher-order graph matching, IEEE Trans. Pattern Anal. Mach. Intell. 36 (3) (2014) 479–492.

[27] K. Riesen, H. Bunke, Approximate graph edit distance computation by means of bipartite graph matching, Image Vis. Comput. 27 (7) (2009) 950–959.

[28] G.L. Scott, H.C. Longuet-Higgins, An algorithm for associating the features of two images, Proc. R. Soc. Lond. Ser. B: Biol. Sci. 244 (1309) (1991) 21–26.

[29] F. Serratosa, Fast computation of bipartite graph matching, Pattern Recognit. Lett. 45 (2014) 244–250.

[30] R. Sinkhorn, A relationship between arbitrary positive matrices and doubly stochastic matrices, Ann. Math. Stat. (1964) 876–879.

[31] J. Sullivan, S. Carlsson, Recognizing and tracking human action, in: Proceedings of the European Conference on Computer Vision, 2002, pp. 629–644.

[32] Y. Tian, J. Yan, H. Zhang, Y. Zhang, X. Yang, H. Zha, On the convergence of graph matching: graduated assignment revisited, in: Proceedings of the European Conference on Computer Vision, 2012, pp. 821–835.

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

[34] X. Yang, H. Qiao, Z.Y. Liu, Feature correspondence based on directed structural model matching, Image Vis. Comput. 33 (2015) 57–67.

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

[6] The combination ${C_N}^2$ is also denoted by $(N)_2$ or $\binom{N}{2}$.

[36] R. Zass, A. Shashua, Probabilistic graph and hypergraph matching, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2008, pp. 1–8.
[37] F. Zhou, F. De la Torre, Factorized graph matching, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2012, pp. 127–134.
[38] F. Zhou, F. De la Torre, Deformable graph matching, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 2922–2929.

Xu Yang is an assistant professor at the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include computer vision, pattern recognition and robotics.

Hong Qiao is a professor at the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China. Her research interests include robotics, machine learning, and computer vision.

Zhi-Yong Liu is a professor at the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, Beijing, China. His research interests include machine learning, pattern recognition, computer vision, and bioinformatics.