Adaptive Discrete Hypergraph Matching

Junchi Yan, *Member, IEEE*, Changsheng Li, Yin Li, and Guitao Cao

***Abstract*—This paper addresses the problem of hypergraph matching using higher-order affinity information. We propose a solver that iteratively updates the solution in the discrete domain by linear assignment approximation. The proposed method is guaranteed to converge to a stationary discrete solution and avoids the annealing procedure and *ad-hoc* post binarization step that are required in several previous methods. Specifically, we start with a simple iterative discrete gradient assignment solver. This solver can be trapped in an *m*-circle sequence under moderate conditions, where *m* is the order of the graph matching problem. We then devise an adaptive relaxation mechanism to jump out this degenerating case and show that the resulting new path will converge to a fixed solution in the discrete domain. The proposed method is tested on both synthetic and real-world benchmarks. The experimental results corroborate the efficacy of our method.**

***Index Terms*—Computer vision, optimal matching, pattern recognition.**

I. Introduction

CORRESPONDENCE is a fundamental problem in computer vision and pattern recognition. It has a wide spectrum of applications including action recognition [1], contour matching [2], shape reconstruction [3], scene understanding [4], and visual tracking and action recognition [5]. The problem becomes even more intriguing when graph structure is involved, which can convey rich geometrical information of the data as shown in recent state-of-the-arts for data analytics and machine learning [6]–[9]. For instance, via incorporating graph regularization, the non-negative low-rank matrix factorization method proposed in [7] achieves state-of-the-art performance on image clustering. By encoding structural information via a weighted graph, graph matching (GM) provides a powerful tool to formulate these correspondence problems. We refer to [10]–[12] as recent surveys of GM and its applications. Beyond the linear assignment problem [13] incorporating unary node-to-node affinity such that it can be solved in polynomial time, GM explores edge-to-edge affinity and it is NP-hard.

Higher-order GM, i.e., the matching of hypergraphs have recently received considerable interests [14]–[16]. For instance, third-order models using the sampled triplets among nodes are shown to be successful in computer vision applications [15], [16] for their tradeoff between efficiency and expressiveness against noise.

In line with many GM methods (see [15]–[18]) mentioned in this paper, we assume that the graph is constructed beforehand by a certain means and the affinity between graphs is given. In fact, how to set up an affinity function between graphs is an important open problem. It is also often dependent on specific applications which is out of the scope of this paper. In particular, we refer the readers for state-of-the-arts [19], [20] for graph learning. Nie *et al.* [19] proposed a principled and parameter-free method for learning the data graph with exact k connected components, where k is a predefined parameter. Cho *et al.* [20] presented a unified framework for learning the structure and attributes from graphs with labeled correspondence to each other.

A. Motivation for Discrete Methods

This paper is devoted to a *discrete* method with guaranteed *convergence* for general higher-order GM. This is in contrast to the fact that many second-order and hypergraph matching methods relax the problem domain from discrete to continuous domain, and adopt *soft* techniques such as deterministic annealing [18], soft-max [21], and iterative bistochastic normalization [16] to relax assignment matrix domain to the continuous convex hull—a doubly stochastic matrix or being of unit length [22].

We argue that relaxing m-order GM into the continuous domain might not be the best option in terms of efficiency and theoretical soundness. First, the annealing procedure or iterative normalization can be time-consuming and more perhaps importantly, it also brings about more parameters. Unfortunately there is no principled method for tuning these parameters. While the Hungarian method [23] and its alternative [24] are simple and often efficient in practice. Second, these methods cannot ensure a convergence to a fixed discrete solution, thus *ad-hoc* post rounding is needed to binarize the solution. However, so far the effect of this rounding step is still theoretically unclear and it can incur arbitrary accuracy loss, which is further discussed in [25].

Manuscript received May 11, 2016; revised November 12, 2016; accepted January 11, 2017. Date of publication February 17, 2017; date of current version January 15, 2018. This work was supported in part by the National Natural Science Foundation of China under Grant 61602176, in part by the NSFC-Zhejiang Joint Fund for the Integration of Industrialization and Informatization under Grant U1609220, and in part by the China Post-Doctoral Science Foundation under Project 2016M590337. This paper was recommended by Associate Editor X. Li. *(Corresponding author: Guitao Cao.)*

J. Yan is with the Shanghai Key Laboratory of Trustworthy Computing, and the School of Computer Science and Software Engineering, East China Normal University, Shanghai 200062, China, and also with IBM Research—China, Shanghai 201203, China (e-mail: jcyan@sei.ecnu.edu.cn).

C. Li is with Alibaba Group, Beijing 100101, China (e-mail: changsheng.lcs@alibaba-inc.com).

Y. Li is with the School of Interactive Computing, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: yli440@gatech.edu).

G. Cao is with the School of Computer Science and Software Engineering, East China Normal University, Shanghai 200062, China (e-mail: gtcao@sei.ecnu.edu.cn).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCYB.2017.2655538

B. Outline of Our Method

We propose a novel discrete solver that iteratively solves the higher-order assignment problem by an approximate first-order linear assignment. This linear approximate framework is inspired by a few GM methods for both second-order [18], [22], [26] and higher-order hypergraph matching [15], [16]. At each iteration, we employ the gradient assignment method (e.g., the Hungarian method) for discrete optimization as used in [25] and [27]. Such a discrete method removes the heuristic post-rounding step, and it also dispenses annealing loops and associated parameters which otherwise are required in continuous methods (see [18], [26] for time saving).

To our knowledge, it is by far one of the few discrete GM methods. When compared with state-of-the-art continuous methods, our method is comparable in terms of accuracy yet can be faster with appropriate settings. We also compare with one of representative work, i.e., the (quasi) discrete solver: integer projected fixed point (IPFP) [25] and its hypergraph generalization (HIPFP) in Section III-D. Our method mainly involves only one main parameter, i.e., the weighting parameter λ for the adaptive relaxation term.

The challenge within our framework is to design an effective discrete optimization mechanism. We propose an *adaptive discrete gradient assignment* method, with three key highlights.

1) No heuristic reweighting is required in the updating step of each iteration as used by [16], [25], and [26] nor annealing or iterative bistochastic normalization. This leads to fewer parameters and saving of computational overhead.
2) Integer solutions are returned that avoids *ad-hoc* rounding step.
3) Convergence to a fixed discrete point is theoretically guaranteed.

C. Contribution

Preliminary results of this paper appear in [28] for hypergraph matching, which is connected to its second-order embodiment in part of [27]. We consider our contributions for novelty and practical use as follows.

1) We propose a novel *full discrete algorithm* for (hyper) GM, in contrast to the prevalent continuous methods [18], [22], [26] and a few quasi-discrete methods [17], [25].[1] IPFP [25] may still generate continuous solutions since it uses Hungarian algorithm interleaved with projection onto the assignment constraints. The factorized GM model [17] induces discrete solution by annealing the weights of convex and concave relaxations of the original objective. Though these methods mitigate the artifact of continuous relaxation, they still phase through the continuous space over optimization.
2) We theoretically prove the convergence of our method. Specifically we design an *adaptive relaxation mechanism*, which ensures our discrete method converging to a stationary solution under moderate conditions. This is achieved by jumping out of the local cycling situation.
3) Combing the efficient discrete methodology and its adaptive relaxation mechanism in an on-demand fashion, our method empirically shows a competitive tradeoff between accuracy and efficiency on benchmark datasets.

[1]We find few other discrete second-order GM methods: Markov chain Monte Carlo-based model [29] and its sequential Monte Carlo sampling-based extension [30] for speedup. But the iterative sampling procedure render them less efficient. Another state-of-the-art is [31] where tabu search is involved.

II. Related Work

This section shows the perspective that a line of hypergraph matching methods are rooted from the second-order methods. This perspective motivates the design of our approach.

The second-order GM can be formulated as the quadratic assignment problem (QAP) in Lawler's form—we use bold lower (upper) case for vector (matrix/tensor) in this paper

$$\max_{\mathbf{x}\in C} \mathbf{x}^T\mathbf{K}\mathbf{x}.$$

For hypergraph matching, a concrete formula for the mostly addressed third-order matching problem can be written as

$$\max_{\mathbf{x}\in C} \sum_{i,j,k} H_{i,j,k} x_i x_j x_k$$

where $\mathbf{x}$ is the vectorized assignment matrix and $\mathbf{x}^T$ is its transposition; C denotes the assignment constraints depending on applications, $\mathbf{K}$ ($\mathbf{H}$) denotes the affinity matrix (tensor). Formal definition and exposition will be presented later.

We sketch the idea for the iterative linear approximation applied to the second-order and hypergraph matching problems, i.e., fix $\mathbf{x}^{t-1}$ at the previous iteration and solve $\mathbf{x}$ by

$$\max_{\mathbf{x}\in C} \mathbf{x}^T\mathbf{K}\mathbf{x}_{t-1}, \max_{\mathbf{x}\in C} \sum_k \sum_{i,j} H_{i,j,k} x_{i(t-1)} x_{j(t-1)} x_k.$$

The technical details will be further covered in Section III. We observe for various GM methods under the above linear approximate framework, the main difference lies in how the solution is updated at each iteration as summarized in Table I.[2] We also distill the common meta-algorithms by these methods in Table II. The Sinkhorn bistochasticization (used in softassign [21]) and power iteration are in continuous domain. The Hungarian method works in the discrete domain. We discuss in more details about the relevant work as follows.

A. Second-Order GM by Approximate Linear Assignment

GM has been mainly addressed via edge-to-edge second-order affinities, and most of them adopt an approximate linear assignment framework [18], [22], [25], [26], [35].

Gold and Rangarajan [18] proposed the GAGM to solve a series of linear approximations on a general cost function

[2]Acronyms for algorithm names are used in this paper. Reweighted random walk GM (RRWM) [26]; spectral matching (SM) [22]; IPFP [25]; and graduated assignment GM (GAGM) [18]. We dub their hypergraph variants as tensor spectral matching (TSM) [15], reweighted random walks hyper-graph matching (RRWHM) [16], HIPFP, and HGAGM [32]. Moreover, as will be shown in the sequel, the proposed two algorithms are termed as: 1) hyper discrete gradient assignment (HDGA) and 2) hyper ADGA (HADGA).

TABLE I
METHODS UNDER THE LINEAR APPROXIMATE FRAMEWORK; $\mathbf{x}_{\text{pm}}$—RETURN OF POWER METHOD [33] WHOSE UNIT COST IS $O(n^3)$ OR LESS, WHERE n IS THE NUMBER OF NODES; $\mathbf{x}_{\text{sa}}$—RETURN OF SOFTASSIGN [21] WHOSE UNIT COST IS $O(n^2)$ PER ITERATION; $\mathbf{x}_{\text{hg}}$—RETURN OF THE HUNGARIAN METHOD WHOSE UNIT COST IS $O(n^3)$; AND $\mathbf{x}_{\text{pre}}$—PREVIOUS ITERATIVE SOLUTION. HIPFP IS IMPLEMENTED IN [28] FOR EVALUATION. THE TIME COMPLEXITY HERE REFERS TO THE OVERHEAD FOR SOLVING THE ASSIGNMENT SUBPROBLEM AT EACH ITERATION EXCLUDING THE DOMINANT OVERHEAD (I.E., FOR THE COMPUTATION IN THE BRACKET OF LINE 3 FOR BOTH ALGORITHMS 1 AND 2) AS THE SIZE OF THE INVOLVED $\mathbf{H}$ CAN BE $(n^2)^m$ WHEN NO SPARSIFICATION IS CONSIDERED. BY SUCH AN EXCLUSION, THE COMPLEXITY IS THE SAME FOR BOTH SECOND-ORDER AND HYPERGRAPH MATCHING METHODS BECAUSE THEY BOTH ADOPT LINEAR ASSIGNMENT AT EACH ITERATION, THOUGH THE LATTER WILL INVOLVE MORE COST FOR COMPUTING THE INPUT MATRIX. THE COMPLEXITY FOR SM THAT EMPLOYS EIGEN DECOMPOSITION CAN BE REDUCED TO $O(2^w)$ FOR $2 < w < 2.376$ [34] THUS THE COMPLEXITY FOR COMPUTING $\mathbf{x}_{\text{pm}}$ CAN BE LESS THAN $O(n^3)$

second-order method	hyper generalization	updating mechanism	unit cost
GAGM [18](1996)	HGAGM [32](2011)	$\mathbf{x}_{sa}$ by raw tensor	$O(n^2)$
SM [22](2006)	TSM [15](2011)	$\mathbf{x}_{pm}$ by raw tensor	$O(n^3)$
RRWM [26](2010)	RRWHM [16](2011)	$\eta\mathbf{x}_{pm}$+(1-η)$\mathbf{x}_{sa}$(raw)	$O(n^3)$
IPFP [25](2009)	HIPFP [28](2015)	$\eta\mathbf{x}_{hg}$+(1-η)$\mathbf{x}_{pre}$(raw)	$O(n^3)$
DGA(Alg.3)	HDGA(Alg.1)	$\mathbf{x}_{hg}$ by raw tensor	$O(n^3)$
ADGA(Alg.4)	HADGA(Alg.2)	$\mathbf{x}_{hg}$ by relaxed tensor	$O(n^3)$

TABLE II
META-ALGORITHMS: m, n IS THE GRAPH EDGE AND NODE NUMBER

method	used by	cost
Sinkhorn bistochasticization	(H)GAGM, RRW(H)M for $\mathbf{x}_{sa}$	$O(n^2)$
Power iteration	(T)SM, RRW(H)M for $\mathbf{x}_{pm}$	$O(n^3)$
Hungarian method	(H)IPFP for $\mathbf{x}_{hg}$	$O(n^3)$

to encode the unary and second-order affinity. As a continuous relaxation method, deterministic annealing is used to relax the solution domain to its convex hull, i.e., doubly stochastic matrix. By constructing a discrete time Lyapunov functions, the authors prove when the affinity matrix is positive definite, GAGM will converge to a binary solution [36]. GAGM is found more accurate and robust than heuristic tree search [37] or relaxation label [38]. It is found still rather competitive in recent evaluations [27].

GAGM's linear assignment-based approximate framework is widely followed by a series of studies. Leordeanu and Hebert [22] formally introduced the form of affinity matrix and devise an efficient approximation using a spectral relaxation. Cour *et al.* [35] developed a more general scheme that incorporates affine constraints in the spectral relaxation. Cho *et al.* [26] generated the solution by simulating random walks with reweighting jumps enforcing the matching constraints. Different from these continuous methods, Leordeanu *et al.* [25] proposed an integer projection algorithm to optimize the objective function in quasi-discrete domain by reweighting the discrete solution from a Hungarian method and a projection.

There is another popular thread of suboptimal methods based on graph edit distance [39]–[41]: bipartite (BP) GM [42] and its speeded variant—fast BP (FBP) [43]–[46]. These methods also have a mechanism to approximate higher-order problem by a node-to-node cost matrix that encodes local clique structure.

B. Higher-Order GM by Approximate Linear Assignment

Hypergraph matching is relatively less studied in the literature. The approximate linear assignment framework is often maintained when the second-order methods [18], [22], [25], [26] are extended to hypergraph variants [15], [16], [32], [47].

In the seminal work, Duchenne *et al.* [15] used tensor to encode the higher-order affinity to generalize second-order GM to higher-order. They also extended the SM method [22] via a multidimensional power method [33]. Reweighted random walks is devised for second-order [26], and subsequently for higher-order [16]. Chang and Kimia [32] generalized GAGM [18] to higher-order as termed HGAGM in this paper. The probabilistic method [14] assumes the higher-order correspondences are conditional independence. This hurts its applicability and performance [16].

C. Other Specialized Methods

References [48] and [49] are specialized hypergraph matching solvers. The former is tailored to finding 3-D and 2-D projection correspondence. The latter assumes the affinity tensor is highly redundant as it requires the existence of many tuples of feature points whose corresponding angles can be rather close to each other. While their method is inapplicable if the tensor is sparse and generated by Delaunay triangulation, a common protocol to save space and time.

D. Open Issues

Most of existing continuous methods in general do not mathematically have a rigorous convergence guarantee, though a few methods [18], [25] can enjoy convergence under particular conditions [36], e.g., the affinity matrix is positive definite for the second-order case as discussed in [27] and [36]. In fact, given an arbitrary affinity matrix or tensor, convergence in general does not hold. Moreover, even if such an iterative strategy reaches convergence to a fixed point, the solution may not necessarily lie in the discrete domain and satisfy the matching constraints. Therefore it involves *ad-hoc* post-rounding whose influence on the accuracy is unclear. For instance, in the reweighted random walk methods [16], [26], the distribution has to be binarized by a certain means.

III. MAIN METHOD AND THEORY

In line with the approximate linear assignment framework for second-order [18], [22], [26], [27] and hypergraph matching [15], [16], [32], we show how to explore the capability of a full discrete method, especially improving its cost-effectiveness. Our key technique is an adaptive objective relaxation mechanism over the course of optimization in the discrete domain directly. We prove its convergence under moderate conditions via a particular *dimension-spread* method (see the Appendix).

We will first introduce the general m-order formulation for GM under tensor representation. Then we present our algorithms with the adaptive mechanism and the behind theoretical analysis. To make our method more accessible, we also present

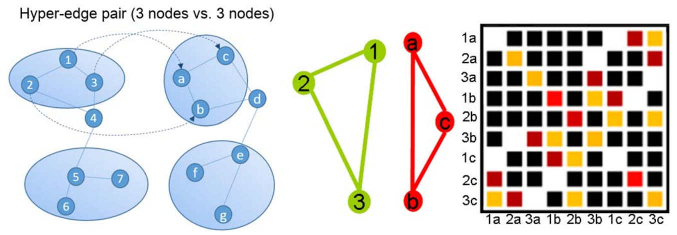

Hyper-edge pair (3 nodes vs. 3 nodes)
1a 2a 3a 1b 2b 3b 1c 2c 3c

Fig. 1. Illustration for the hyper-edge associated with three nodes (left); and the second-order affinity matrix, given two graphs to match, where darker color of the cells denote lower affinity value in the matrix (right).

an embodiment on the second-order case and compare with another quasi-discrete solver IPFP. Finally, we give discussion on its nonadaptive baselines.

A. General Formulation for m-Order Graph Matching

We describe the formulation and algorithms under the general setting of m-order GM. In fact, they can be readily applied to the widely studied second-order case.

Given two hypergraphs $\mathcal{G}_1$ and $\mathcal{G}_2$ with n_1 and n_2 nodes, respectively, there exist affinity measurements between each subset of m nodes (a hyperedge) in one graph and m nodes in the other, which together form a set of pairs of hyperedges between two graphs. Given an assignment matrix for one-to-one node correspondence, by adding up all corresponding hyperedge pairs determined by the assignment matrix, one can obtain an overall affinity objective score. The goal of hypergraph matching is to establish node mapping of two hypergraphs such that the objective is maximized.

Formally, let tuple $(i_1, i_2, \ldots, i_m)$ denote m-order hyperedge correspondence where each element in the tuple $\{i_k\}_{k=1}^m$ indicates a node-to-node matching $(i_k^1 \leftrightarrow i_k^2)$ for the fact that node i_k^1 $(i_k^1 = 1, \ldots, n_1)$ in $\mathcal{G}_1$ matches node i_k^2 $(i_k^2 = 1, \ldots, n_2)$ in $\mathcal{G}_2$. More concretely, i_k is calculated by $i_k = (i_k^1 - 1)n_2 + i_k^2$ thus $1 \le i_k \le n_1 n_2$. We allow $(i_k^1 \leftrightarrow i_k^2) = (i_s^1 \leftrightarrow i_s^2)$, i.e., $i_k = i_s$ for $k \neq s$, to encode lower-order ($< m$) hyperedges via a unified representation. In line with [15] and [16], we use an m-order super-symmetric tensor $\mathbf{H}$ to encode the hyperedge to hyperedge affinity score. Element $H_{i_1,i_2,\ldots,i_m}$ is the affinity value between two hyperedges with order m determined by the correspondence tuple $(i_1, i_2, \ldots, i_m)$.

Let $\mathbf{x} \in \{0,1\}^{n_1 n_2 \times 1}$ denote the vectorized form of the node correspondence matrix $\mathbf{X} \in \{0,1\}^{n_1 \times n_2}$ by its rows for one-to-one matching. For its element $x_{i_k} \in \{0,1\}$ whose position in $\mathbf{x}$ is indexed by i_k $(i_k = 1, 2, \ldots, n_1 + 1, \ldots, n_1 n_2)$, $x_{i_k} = 1$ if there is a node-to-node correspondence $(i_k^1 \leftrightarrow i_k^2)$. By assuming $n_1 \le n_2$ without loss of generality, the hypergraph matching problem in general can be formulated as the following constrained combinatorial optimization problem widely adopted by state-of-the-arts [15], [16]:

$$\mathbf{x}^* = \arg\max\left(\Sigma_{i_1,i_2,\ldots,i_m} H_{i_1,i_2,\ldots,i_m} x_{i_1} x_{i_2} \ldots x_{i_m}\right)$$
$$\text{s.t.} \quad \mathbf{X}\mathbf{1}_{n_2} = \mathbf{1}_{n_1}, \mathbf{X}^T\mathbf{1}_{n_1} \preceq \mathbf{1}_{n_2}, \mathbf{x} \in \{0,1\}^{n_1 n_2 \times 1}. \quad (1)$$

The symbol "=" and "$\preceq$" in (1) denote the comparison is made by the value of the elements in the vectors.[3]

[3]This notational convention also applies to other formulas in this paper if not otherwise explicitly specified.

The constraints refer to two-way constraints for the one-to-one node matching. Note one can further add dummy nodes in $\mathcal{G}_1$ to make the number of nodes in two graphs become equal. This is a common technique from linear programming and is widely adopted by [17], [21], and [50], such that is supposed can handle superfluous nodes in a statistically robust manner (see [21, Sec. 2.3, last paragraph]). For discussion convenience, we assume $n_1 = n_2 = n$ if not otherwise explicitly specified.

Furthermore, (1) can be rewritten in a more compact objective function based on tensor marginalization as adopted by existing hypergraph matching works [14], [15], [51]

$$\mathbf{x}^* = \arg\max(\mathbf{H} \otimes_1 \mathbf{x} \otimes_2 \mathbf{x} \ldots \otimes_m \mathbf{x})$$
$$\text{s.t.} \quad \mathbf{X}\mathbf{1}_{n_2} = \mathbf{1}_{n_1}, \mathbf{X}^T\mathbf{1}_{n_1} \preceq \mathbf{1}_{n_2}, \mathbf{x} \in \{0,1\}^{n_1 n_2 \times 1}. \quad (2)$$

B. Short Note on the Tensor Formulation

A tensor is an n-dimensional generalization of a 2-D matrix, and can be viewed as an n-dimensional hyper-rectangle. It can be multiplied by a vector in various ways. In line with the convention [14], [15], [51], we use the following notation for $\otimes_k$ [52].

For $\mathbf{B} = \mathbf{A} \otimes_k \mathbf{V}$, the element $B_{i_1,\ldots,i_{k-1},i_{k+1},\ldots,i_n}$ in $\mathbf{B}$ is by definition computed as follows:

$$B_{i_1,\ldots,i_{k-1},i_{k+1},\ldots,i_n} = \sum_{i_k} A_{i_1,\ldots,i_k,\ldots,i_n} V_{i_k}$$

where $\mathbf{V}$ is a vector and $\mathbf{A}$, $\mathbf{B}$ is an m and $m-1$ order tensor, respectively. The index k in $\otimes_k$ indicates that we multiply on the kth-dimension. Readers are referred to [15, Sec. 3.1] for more details on tensor multiplication.

To make it more clear that the objective in (1) is indeed a scalar, one can note the tensor $\mathbf{H}$ is multiplied for m times by $\mathbf{x}$. This will reduce the dimension of the result from m to 0, which corresponds to a scalar. In particular when $m = 2$ for second-order GM, we have $\mathbf{H} \otimes_1 \mathbf{x} \otimes_2 \mathbf{x} = \mathbf{x}^T\mathbf{H}\mathbf{x}$ when $\mathbf{H}$ is a 2-D matrix. This leads to the QAP formulation widely used in [22], [25], [26], and [35].

To further concretize (2) for $m = 3$, we show the scores in the above two expressions in (1) and (2) are equivalent. In fact this also holds for any m

$$\begin{aligned}\mathbf{H} \otimes_1 \mathbf{x} \otimes_2 \mathbf{x} \otimes_3 \mathbf{x} &= \left(\left(\sum_i H_{i,j,k} x_i\right)_{j,k} \otimes_2 \mathbf{x}\right) \otimes_3 \mathbf{x} \\ &= \sum_j \left(\sum_i H_{i,j,k} x_i\right) x_j \otimes_3 \mathbf{x} \\ &= \sum_{i,j,k} H_{i,j,k} x_i x_j x_k.\end{aligned}$$

This tensor-based formulation is also widely adopted by [15] and [16] where the super-symmetric tensor $\mathbf{H}$ is invariant under any permutation of the order in the tuple. For the third-order case, all $\{H_{i,j,k}\}$ are equal to each other for different i, j, k permutations. One aims to find the solution that maximizes the affinity score in the binary assignment matrix space. We describe our main results under m-order formulation. The experiments focus on the second and third-order cases, in line with hypergraph matching protocols by [14]–[16], [51], and [53] as the third-order information can be easily derived from geometric measurements such as the interior angles for a sampled triplet. This has been a popular protocol for evaluating state-of-the-arts [15], [16], [28].

C. Proposed m-Order Convergent Algorithms

We begin with a baseline method HDGA by using a series of iterative linear assignments to approximate the original objective and fixing the recent m-1 variables $\{\mathbf{x}_j\}_{j=k+2-m}^{k}$ in (2). It adopts the Hungarian method as denoted by $H_d(\cdot)$ to obtain the *global-optimal discrete* solution regarding the linearized objective per iteration: $\mathbf{x}_{k+1} = H_d(\mathbf{H} \otimes_1 \mathbf{x}_{k+2-m} \dots \otimes_{m-1} \mathbf{x}_k)$. This baseline method is formally described in Algorithm 1.

Note that the input to the Hungarian method $\mathbf{K} \triangleq \mathbf{H} \otimes_1 \mathbf{x}_{k+2-m} \dots \otimes_{m-1} \mathbf{x}_k$ in fact is a vector after m-1 times of tensor multiplication operations are performed on the m-order tensor $\mathbf{H}$. As a result, $\mathbf{K} \otimes_m \mathbf{x}_{k+1}$ based on (2) can be equally rewritten as $\mathbf{K}\mathbf{x}_{k+1}$ in the matrix form. It is reduced to a linear assignment problem that can be solved in polynomial time $O(n^3)$ by the Hungarian method. In this sense, it is clear that our method follows the linear approximate framework. More specifically, we refer the score at each iteration: $\mathbf{H} \otimes_1 \mathbf{x}_{k+1-m} \otimes_2 \mathbf{x}_{k+2-m} \dots \otimes_m \mathbf{x}_k$ as the *approximate objective score* which involves the variables $\mathbf{x}_{k+1-m}, \dots, \mathbf{x}_{k-1}$ over recent iterations and the latest $\mathbf{x}_k$ for solving the linear assignment problem.

HDGA generates a score-ascending solution sequence with respect to the approximate objective score. It is worth noting that, the sequence generated by the second-order GM solver IPFP [25] for second-order matching is score-ascending with respect to the original objective. Nevertheless, the generated solution is a reweighted combination between the return of the Hungarian method and the solution of the previous iteration, thus in general the result is nondiscrete except for the special cases, e.g., the quadratic function is convex. Thus we dub it a quasi-discrete solver.

Enjoying the merits of being a pure discrete solver, one key issue of HDGA is the solution sequence can deviate from the original objective (2) when the iteration becomes divergent. Thus it is appealing to design a convergent algorithm which is able to find a convergent solution [and hopefully better one with respect to the objective in (2)] in discrete domain for the approximate objective.[4] Theorem 1 shows under the moderate condition as stated in Assumption 1, HDGA will converge to an m-point cycling sequence. This finding pinpoints the defect of HDGA.

Assumption 1: Given the vectorized assignment matrix $\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \dots, \mathbf{x}_{i_{m-1}}$ that are generated over the solution sequence of Algorithm 1, for any $\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_{m-1}}, \mathbf{x}_j, \mathbf{x}_k, j \neq k$, we assume $\mathbf{H} \otimes_1 (\mathbf{x}_j - \mathbf{x}_k) \otimes_2 \mathbf{x}_{i_1} \dots \otimes_m \mathbf{x}_{i_{m-1}} \neq 0$ and the tensor $\mathbf{H}$ is super-symmetric.

[4]In fact, when the sequence converges, the approximate objective becomes the raw objective: $\mathbf{H} \otimes_1 \mathbf{x}_{k+1} \otimes_2 \mathbf{x}_{k+2} \dots \otimes_m \mathbf{x}_{k+m}$ given all $\{\mathbf{x}_i\}_{i=k+1}^{k+m}$ equal $\mathbf{x}$. This also gives the motivation for seeking a convergent sequence.

Algorithm 1 HDGA
```
1: Input: $\mathbf{x}_0 = [\frac{1}{n^2}, \dots, \frac{1}{n^2}]^T$, $\mathbf{H}$, $k_{max}$
2: for k=1:$k_{max}$ do
3:     $\mathbf{x}_{k+1} = H_d(\mathbf{H} \otimes_1 \mathbf{x}_{k+2-m} \otimes_2 \dots \otimes_{m-1} \mathbf{x}_k)$
4:     if converge then
5:         return $\mathbf{x}^* = \mathbf{x}_{k+1}$;
6:     end if
7: end for
```

Algorithm 2 HADGA
```
1: Input: $\mathbf{x}_0 = [\frac{1}{n^2}, \dots, \frac{1}{n^2}]^T$, $\mathbf{x}^* = \mathbf{x}_0$, $\mathbf{H}$, $\lambda$, $k_{max}$
2: for k=1:$k_{max}$ do
3:     $\mathbf{x}_{k+1} = H_d(\mathbf{H} \otimes_1 \mathbf{x}_{k+2-m} \otimes_2 \cdots \otimes_{m-1} \mathbf{x}_k)$
4:     if converge then
5:         return: $\mathbf{x}^* = \mathbf{x}_{k+1}$
6:     else if Fall into the $m$-cycling loop sequence then
7:         $\mathbf{H} = \mathbf{H} + \lambda\mathbf{H}^{(2)} + \lambda\mathbf{H}^{(3)} + \dots + \lambda\mathbf{H}^{(m)}$
8:     end if
9: end for
```

Theorem 1: Algorithm 1 will converge to an m-point circle: $(\mathbf{x}_{k-m}, \mathbf{x}_{k+1-m}, \dots, \mathbf{x}_{k-1}) \rightarrow (\mathbf{x}_{k+1-m}, \mathbf{x}_{k+2-m}, \dots, \mathbf{x}_k) \rightarrow \dots \rightarrow (\mathbf{x}_{k-m}, \mathbf{x}_{k+1-m}, \dots, \mathbf{x}_{k-1}) \rightarrow \cdots$ if Assumption 1 holds.

Proof: Given a super-symmetric affinity tensor $\mathbf{H}$, HDGA is score-ascending with respect to the *approximate* objective as it employs the gradient assignment approach, e.g., Hungarian method iteratively. Since the score function is bounded due to the feasible domain, i.e., assignment matrix space is enumerable and limited, there exists a certain k, such that after k rounds of iterations, the value of the *approximate* objective will stop ascending: $\mathbf{H} \otimes_1 \mathbf{x}_{k-m} \otimes_2 \mathbf{x}_{k+1-m} \dots \otimes_m \mathbf{x}_{k-1} = \mathbf{H} \otimes_1 \mathbf{x}_{k+1-m} \otimes_2 \mathbf{x}_{k+2-m} \dots \otimes_m \mathbf{x}_k$. Meanwhile, Assumption 1 ensures the equal score value leads to the unique solution such that $\mathbf{x}_k = \mathbf{x}_{k-m}$, thus henceforth we are in the m-circle sequence $(\mathbf{x}_{k+1-m}, \mathbf{x}_{k+2-m}, \dots, \mathbf{x}_k)$. Remember the tensor $\mathbf{H}$ is super-symmetric thus the score value is invariant to permutations of $\{\mathbf{x}_i\}_{i=k-m}^{k-1}$ in the above score function. ■

To achieve strong convergence for this discrete algorithm, one can monitor the iteration sequence of Algorithm 1 and once the m-circle track is detected, an adaptive disturbance mechanism shall be launched to guide the iteration sequence to jump out the current trap and move to a fixed point. Theorem 2 shows by modifying the affinity tensor in a particular way without changing the optima of the raw problem, the iteration will converge to a fixed discrete point. To make this paper self-contained, two definitions for Theorem 2 are introduced as follows.

Definition 1: For an m-tuple $(j_1, j_2, \dots, j_m)$, where $\{j_k\}_{k=1}^m$ is a positive integer, its multiplicity is defined as the largest number of duplicate elements in the tuple.

Definition 2: A q-multiplicity unit tensor is defined by

$$\mathbf{H}^{(q)} = \begin{cases} \mathbf{H}^{(q)}_{i_1, i_2, \dots, i_m} = 1 (i_1, i_2, \dots, i_m)\text{'s multiplicity is } q \\ \mathbf{H}^{(q)}_{i_1, i_2, \dots, i_m} = 0, \text{otherwise.} \end{cases}$$

Algorithm 3 DGA

```
1: for k=1:$k_{max}$ do
2:     $\mathbf{x}_{k+1} = H_d(\mathbf{K}\mathbf{x}_k)$
3:     if converge then
4:         return: $\mathbf{x}^* = \mathbf{x}_{k+1}$
5:     end if
6: end for
```

Theorem 2: If Assumption 1 holds, one can introduce $\lambda_2, \lambda_3, \ldots, \lambda_m$ to modify the affinity tensor by $\mathbf{H} = \mathbf{H} + \lambda_2\mathbf{H}^{(2)} + \lambda_3\mathbf{H}^{(3)} + \ldots + \lambda_m\mathbf{H}^{(m)}$ subject to HADGA will converge to a fixed point in the assignment matrix space without affecting the optima of the raw problem.

We leave the proof details for Theorem 2 in the Appendix. This theorem stimulates the *adaptively relaxed* gradient assignment algorithm as described in Algorithm 2. We use the term *adaptively relaxed* because it gradually modifies the objective function that implicitly penalizes the deviation between two iterations.

Compared with Algorithm 1, its *adaptively relaxed* counterpart Algorithm 2 obtains strong convergence at the cost of more iterations by adaptively changing the original tensor. Note that the global optimal solution regarding the new objective function under the modified affinity tensor will be the same as the raw problem. This is because the added terms $\mathbf{H}^{(k)}$ as defined in Definition 2 have no discrimination in the assignment matrix space and the resulting modification to the raw objective leads the equal score perturbation given any assignment matrix.

D. Retrospection on the Second-Order Embodiment

The above general formulation and mechanism for hypergraph matching incorporate the second-order problem, and the convergence proof in the Appendix is general for any m. However, consider the fact that second-order matching and its matrix form affinity is conventionally dominant and more familiar to the community, we think it is useful to embody the proposed approach in the second-order context, to better position this paper in the literature and make it more accessible.

First we write out the dominant QAP formulation used in second-order GM methods [17], [22], [25]–[27]

$$\mathbf{x}^* = \arg\max\left(\mathbf{x}^T\mathbf{K}\mathbf{x}\right)$$
$$\text{s.t.} \quad \mathbf{X}\mathbf{1}_{n_2} = \mathbf{1}_{n_1}, \mathbf{X}^T\mathbf{1}_{n_1} \preceq \mathbf{1}_{n_2}, \mathbf{x} \in \{0,1\}^{n_1 n_2 \times 1}$$

the affinity matrix $\mathbf{K} \in \mathcal{R}^{n_1 n_2 \times n_1 n_2}$ encodes the second-order affinity over two graphs. Fig. 1 illustrates one setting of the affinity matrix that is widely adopted in related works.

Now we rewrite Algorithms 1 and 2 into two second-order versions as depicted in Algorithms 3 and 4. To link Algorithms 2 and 4, one notable fact is that the added term $\mathbf{I}$, i.e., the identity matrix in adaptive discrete gradient assignment (ADGA) and the term $\mathbf{H}^{(m)}$ in HADGA for m-order matching both only encode the unary node-to-node affinity.

We provide convergence analysis to these two second-order embodiments, especially the second one. Though our proof

Algorithm 4 ADGA

```
1: for k=1:$k_{max}$ do
2:     $\mathbf{x}_{k+1} = H_d(\mathbf{K}\mathbf{x}_k)$
3:     if converge then
4:         return: $\mathbf{x}^* = \mathbf{x}_{k+1}$
5:     else if Fall into the two-circle loop sequence then
6:         $\mathbf{K} = \mathbf{K} + \lambda\mathbf{I}$
7:     end if
8: end for
```

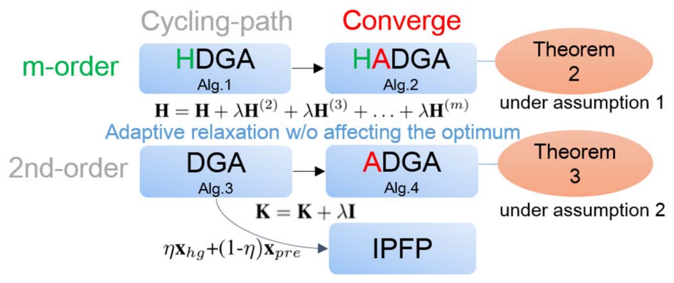

Cycling-path
Converge
m-order
HDGA
Alg.1
HADGA
Alg.2
Theorem 2
under assumption 1
$\mathbf{H} = \mathbf{H} + \lambda\mathbf{H}^{(2)} + \lambda\mathbf{H}^{(3)} + \ldots + \lambda\mathbf{H}^{(m)}$
Adaptive relaxation w/o affecting the optimum
2nd-order
DGA
Alg.3
ADGA
Alg.4
Theorem 3
under assumption 2
$\mathbf{K} = \mathbf{K} + \lambda\mathbf{I}$
$\eta\mathbf{x}_{hg}$+(1-η)$\mathbf{x}_{pre}$
IPFP

Fig. 2. Overview of the relation of the proposed methods and theory.

in the Appendix for hypergraph is applicable here, while we present a more concise proof to the second-order case.

We write out Assumption 2 as a second-order version to Assumption 1 where the tensor $\mathbf{H}$ is replaced by matrix $\mathbf{K}$.

Assumption 2: Assume the following inequality holds:

$$\forall i; \quad \forall j, k, j \neq k; \left(\mathbf{x}_j - \mathbf{x}_k\right)^T\mathbf{K}\mathbf{x}_i \neq 0 \tag{3}$$

where $\mathbf{x}_i, \mathbf{x}_j$, and $\mathbf{x}_k$ are vectorized assignment matrix and $\mathbf{K}$ is the affinity matrix defining the second-order matching problem.

In fact, the second-order matching method DGA in Algorithm 3 will in general trap into a two-circle sequence under Assumption 2. This is because DGA is also score-ascending with respect to the approximate score as the same with the hypergraph case. Thus the proof is identical with HDGA and now $m = 2$.

On the other hand, by Algorithm 4, one can add a weighted identity matrix $\lambda\mathbf{I}$ to the second-order GM objective $\mathbf{x}^T\mathbf{K}\mathbf{x}$ when the looping sequence is formed, without affecting the global optimality of the original problem.

Theorem 3: If Assumption 2 holds, ADGA will converge to a fixed point in the assignment matrix space without affecting the optima of the raw problem.

Proof: The two-circle loop sequence $(\mathbf{x}_k, \mathbf{x}_{k+1}, \mathbf{x}_k, \ldots)$ reaches the following two relations:

$$\mathbf{x}_{k+1}^T\mathbf{K}\mathbf{x}_k = H_d(\mathbf{K}\mathbf{x}_k)^T\mathbf{K}\mathbf{x}_k \geq \mathbf{x}_k^T\mathbf{K}\mathbf{x}_k,$$
$$\mathbf{x}_k^T\mathbf{K}\mathbf{x}_{k+1} = H_d(\mathbf{K}\mathbf{x}_{k+1})^T\mathbf{K}\mathbf{x}_{k+1} \geq \mathbf{x}_{k+1}^T\mathbf{K}\mathbf{x}_{k+1}.$$

A symmetric positive definite matrix $\mathbf{K}$ can be decomposed by $\mathbf{K} = \mathbf{K}_d\mathbf{K}_d^T$. Adding the above two inequalities leads to

$$<\mathbf{K}_d^T\mathbf{x}_{k+1}, \mathbf{K}_d\mathbf{x}_k> \geq \frac{<\mathbf{K}_d^T\mathbf{x}_{k+1}, \mathbf{K}_d^T\mathbf{x}_{k+1}>}{2} + \frac{<\mathbf{K}_d^T\mathbf{x}_k, \mathbf{K}_d^T\mathbf{x}_k>}{2}$$

where $<\cdot,\cdot>$ refers to the inner product between two vectors. Since the score stops increasing in the two-circle looping sequence, the above relation in fact becomes an equation thus

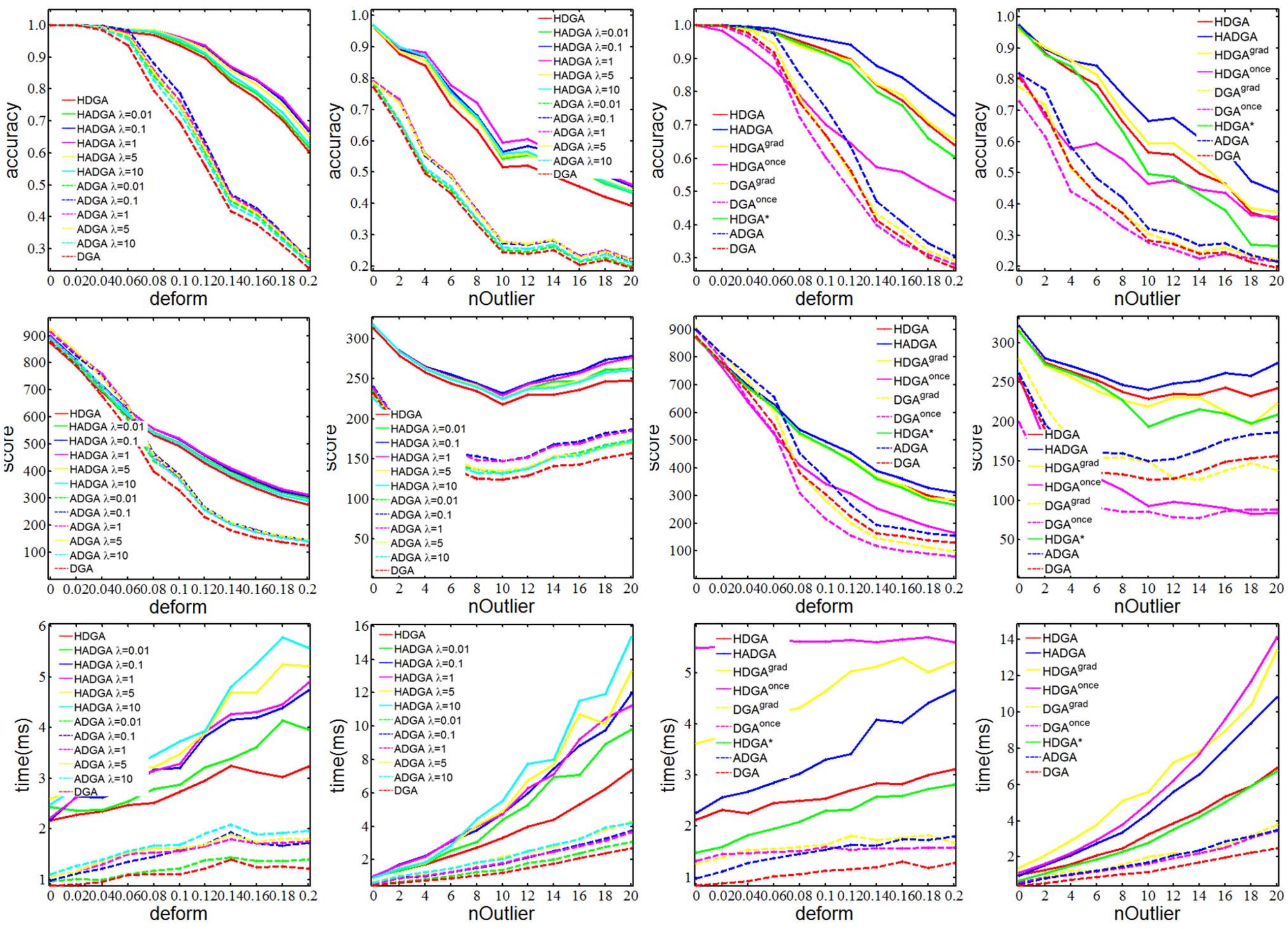


Fig. 3. Left two columns: sensitivity test of λ on random synthetic deformation test ($n_i = 30$, $n_o = 0$) and outlier test ($n_i = 20$, $\varepsilon = 0.1$). Right two columns: comparison with two nonadaptive alternatives and a variant of HDGA. Score in the plot always refers to the raw objective score.

$\mathbf{K}_d^T\mathbf{x}_k = \mathbf{K}_d^T\mathbf{x}_{k+1}$. This leads to $\mathbf{x}_{k+1}^T\mathbf{K}\mathbf{x}_{k+1} = \mathbf{x}_{k+1}^T\mathbf{K}\mathbf{x}_k = \mathbf{x}_k^T\mathbf{K}\mathbf{x}_k$. Then by Assumption 2, we finish the proof. ■

E. Algorithmic Wrap-Up and Distinction to IPFP

The highlights of the main method (H)ADGA are as follows.

1) Solve the problem in discrete domain that dispenses the slow annealing procedure often used in continuous methods.
2) The convergence in discrete domain is fulfilled by adaptively modifying the affinity tensor which avoids the *m*-circle case without affecting the global optimum of the raw problem. This also dispenses the *ad-hoc* post binarization step that otherwise arbitrarily hurts the final accuracy.
3) The second-order and higher-order formulations are both under the approximate linear assignment framework. This perspective also helps understand the position of our methods and peer approaches as depicted in Table I. Fig. 2 illustrates the relation between our methods and the behind theory by two perspectives: a) higher-order versus second-order and b) nonadaptive versus adaptive method. However, as a common limitation also encountered by existing (quasi) convergent algorithms [17], [18], [25], there lacks a theoretical error bound for our convergent algorithms regarding with the global optimality.

1) Relation to the Quasi-Discrete Solver IPFP: Our methods and IPFP [25] are the few with a philosophy that a well designed procedure with local optimality properties in the original domain, can have a greater impact on the final solution than the global optimality in the relaxed continuous domain.

We discuss their relation under the second-order case without loss of generality: under the same first-order approximate framework, our methods and IPFP both use the Hungarian method to solve the linear assignment problem. However, the baseline DGA only using Hungarian method suffers the convergence issue. IPFP adopts the idea for reweighting the Hungarian method solution with the one in previous iteration to address this problem: $\mathbf{x}_t = \eta\mathbf{x}_{\text{hg}} + (1-\eta)\mathbf{x}_{t-1}$ where η is determined to maximize the affinity score $\mathbf{x}_t^T\mathbf{K}\mathbf{x}_t$.

Our method ADGA adopts another strategy: adaptively relax the affinity matrix by $\mathbf{K} = \mathbf{K} + \lambda\mathbf{I}$ when a degenerating case is encountered. Such different mechanisms lead to different behaviors.

1) IPFP may still generate continuous solution while ADGA always returns discrete solution directly.

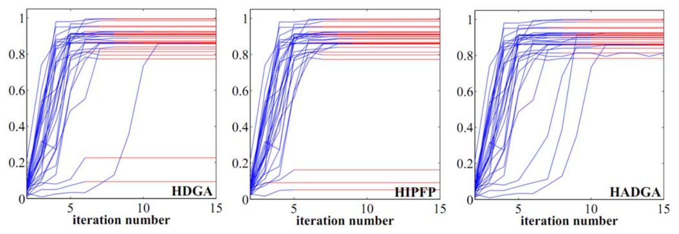


Fig. 4. Tendency curves of normalized score of the original objective with respect to the number of iterations for three methods: 30 tests on CMU *Hotel* spaced by 50 frames. Red denotes for iterations when stoping criterion is satisfied.

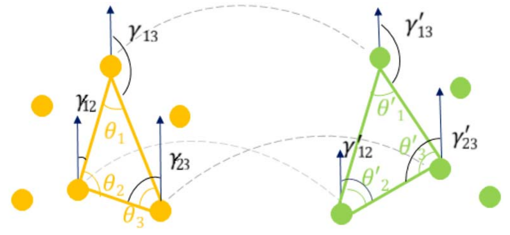


Fig. 5. Example of affinity computation: candidate correspondences form two triangles. Third-order term is calculated by comparing corresponding three interior angles θ_1, θ_2, θ_3 and θ'_1, θ'_2, θ'_3. For second-order term, one considers the angles between the edge length and the vertical direction in two graphs: γ_1, γ_2, γ_3 and γ'_1, γ'_2, γ'_3 and compute their difference.

2) ADGA can converge under moderate conditions (see Assumption 2), while IPFP's convergence is conditioned on the affinity matrix being semidefinite positive.

To empirically compare the performance, we show in Fig. 4 the objective score tendency curves for HDGA, HADGA, and HIPFP along with iterations. Thanks to the adaptive relaxation mechanism, HADGA can jump out of the degenerating case often associated with a lower score where HDGA fails, and switch to a new convergent path with a higher score.

F. Other Discussion

For implementation, first we discuss the technique for determining the timing for adding relaxation terms in the objective. Second, we perform a sensitivity test with the main parameter λ in our algorithm—HADGA and its second order variant ADGA. Moreover, we compare our *adaptive* method HADGA (ADGA) with two nonadaptive alternatives.

1) Cycling Path Detection: One basic idea is comparing solutions among the consecutive iterations which brings about additional cost. More importantly, there exists a subtle case that breaks Assumption 1 when the global optimal solution of Hungarian method is not unique. Consider an extreme case when all elements in the tensor are equal, then any permutation matrix is a best solution. These cases all lead to the score with respect to the relaxed objective function stop climbing. Based on this observation, we propose that instead of separately examining the cycling behavior of the solution path or verifying the uniqueness of the Hungarian method, a unified way is to monitor the score value over iterations. Once the objective score stops climbing and meanwhile does not converge to a fixed point, further relaxation can be launched.

2) Sensitivity Test for λ: We perform sensitivity test for both second-order and hypergraph matching. The results are displayed in the first two columns of Fig. 3. One can find neither too small ($\lambda = 0.01$ in green) nor too large ($\lambda = 10$ in light blue) is optimal for our method since a large λ will make the relaxed objective less discriminative while a small one cannot have enough impact on the behavior of HDGA/DGA. The performance is relatively stable for a wide range of λ at least from 0.1 (in blue) to 1 (in pink). Throughout the experiments in this paper, we set $\lambda = 2 * \text{mean}(\mathbf{H})$ for all tests whose values are mostly within $[0.1, 0.3]$. Moreover, to verify the effect of the addition by λ, in all our tests we only count for the tests when the cycling path is detected and excludes those convergent cases.

3) Nonadaptive Alternatives: To highlight the merit of the *adaptive* relaxation mechanism, we also implement two nonadaptive alternatives by the following.

1) Gradually adding relaxation terms by a constant coefficient at each iteration.
2) Adding the relaxation once for all before iteration starts.

In the right two columns of Fig. 3 we evaluate these two strategies where they are termed by HDGA$^{\text{grad}}$ and HDGA$^{\text{once}}$, where the superscript stands for graduated adding the relaxation term and once for all before iteration, respectively. Here for HDGA$^{\text{grad}}$, the increment at each iteration is $\lambda = \text{mean}(\mathbf{H})/4$; and for HDGA$^{\text{once}}$ the added $\lambda = 20 * \text{mean}(\mathbf{H})$. As one can observe from the figure, nonadaptive methods can hardly improve the performance and adding a big relaxation carelessly hurt the matching accuracy and increase the time overhead significantly partly due to the affinity tensor $\mathbf{H}$ grows more and more dense. As an adaptive method that adds the relaxation in an on-demand manner, HADGA achieves better cost-effectiveness compared with two baselines for both second-order and hypergraph matching. In addition, we test a third alternative which we term HDGA* by replacing the updating step in Algorithm 4 with $\mathbf{x}_{k+1} = H_d(\mathbf{H} \otimes_1 \mathbf{x}_k \otimes_2 \cdots \otimes_{m-1} \mathbf{x}_k)$. Note for the second-order solver DGA, there is no difference since only the solution in last iteration is kept given $m = 2$. The resultant method does not hold fixed point nor m-circle convergence property such that the proposed adaptive relaxation mechanism cannot be used. In our tests it performs similar or even worse than HDGA and we find it often converges even faster than HDGA but to a poor fixed point. To make this paper more focused, these methods are not plotted in the rest of this paper.

IV. Experiments and Discussion

In line with [14]–[16] and [32], our experiments focus on the m-order ($m \leq 3$) problem. How to set up an m-order ($m > 3$) affinity is an open problem and out of the scope of this paper.

A. Affinity Setting and Evaluation Measurements

1) Graph Construction and Tensor Sampling: In line with [17] and [26] and other relevant hypergraph matching works [15], [16], standard Delaunay triangulation is performed to generate the graph structure and affinity tensor.

2) Third-Order Term: The triplet correspondence term $H_{i,j,k}$ relates to the sine values of three angles of each triplet

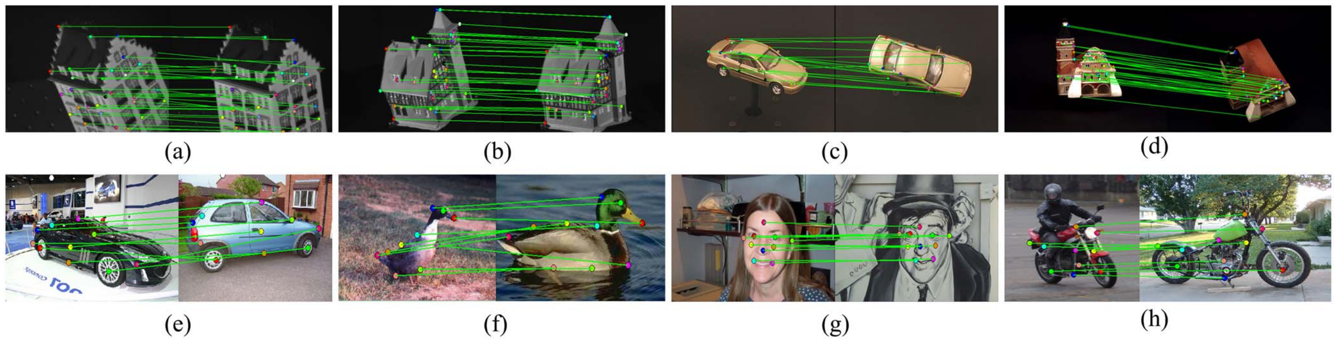


Fig. 6. Real image data tested in this paper: CMU sequence, POSE sequence, and Willow-ObjectClass. (a) CMU *Hotel*. (b) CMU *House*. (c) Pose *VolvoC70*. (d) Pose *Houseblack*. (e) Willow-ObjectClass *Car*. (f) Willow-ObjectClass *Duck*. (g) Willow-ObjectClass *Face*. (h) Willow-ObjectClass *Motorbike*.

(i_1, j_1, k_1) versus (i_2, j_2, k_2) from two graphs. Fig. 5 illustrates the setting for the affinity metrics based on sampled triplets

$$H_{i,j,k} = \exp\left(\sum_{w=i,j,k} \frac{\left|\sin\theta_w - \sin\theta'_w\right|}{\sigma_h}\right) \tag{4}$$

where i denotes the node matching of $i_1 \leftrightarrow i_2$ for node i_1 and node i_2 from two graphs, and so for j, k.

3) Second-Order Term: Since the affinity tensor bears permutation invariance, we have $H_{i,j,j} = H_{i,i,j} = H_{i,j,i} = H_{j,i,j} = H_{j,j,i} = H_{j,i,i}$ for the matching pair i, j. They are measured by the following score including both edge length and angle similarity weighted by the parameter $\beta \in [0, 1]$. The angle γ for each edge is computed as the absolute angle between the edge and the vertical line as used in [15] and [17] (see Fig. 5). This setting is in line with [17], [20], and [54]

$$\beta\exp\left(\frac{\left|l_{ij} - l_{ij}\right|^2}{\sigma_p}\right) + (1-\beta)\exp\left(\frac{\left|\gamma_{ij} - \gamma'_{ij}\right|}{\sigma_p}\right). \tag{5}$$

4) Unary Term: In line with [15] and [16], no unary affinity is used since the local feature descriptor is often neither distinctive on real images nor unavailable on synthetic point sets.

The affinity sensitivity parameter σ_p for second-order affinity and σ_h for hypergraph are both set to 0.1 throughout all tests. The synthetic experimental results are acquired by the average of 100 trials. For real image tests, the number of tests is determined by the trials for traversing the whole image collection. Standard deviation is omitted on the figures.

5) Evaluation Measurements: The performance measurements include: 1) accuracy as computed by $(\mathbf{X}^{\text{alg}}\mathbf{X}^{\text{tru}}/\mathbf{1}_{n_1\times n_2}\mathbf{X}^{\text{tru}})$, where $\mathbf{X}^{\text{alg}}$ and $\mathbf{X}^{\text{tru}}$ are the results of a matching algorithm and the ground truth; 2) affinity score between two graphs as defined in (2); and 3) the computational time overhead in milliseconds. We calculate the accuracy for common inliers and ignore the matching results over outliers in line with [17] and [55]. Moreover, we exclude the cases if they are convergent such that no addition on the affinity tensor $\mathbf{H}$ is needed. We only count for the cycling cases in all our tests. This policy magnifies the performance advantage for the adaptive mechanism.

B. Testing Datasets

Three categories of datasets are tested.

1) Synthetic random point sets.
2) Image sequences.
3) Objects in the image with different appearances but in the same category.

For the first two categories, we set the edge length similarity weight $\beta = 1$ in line with existing work on these two datasets, and set $\beta = 0.7$ for third category since symmetry exists such as *Face* and *Wine Bottle* (see Fig. 6) in the object dataset as also discussed in [54]. Moreover, in the real image tests for dataset 2) and 3), we perform linear transformation on the coordinates $\mathbf{c}$ of the points: $\mathbf{c} = (\mathbf{c} - \text{mean}(\mathbf{c}))/\text{std}(\mathbf{c})$ where mean and std is the mean and standard deviation from the raw coordinate. Fig. 6 depicts several examples on real image datasets.

1) Synthetic Random Point Set: The random point set matching is a standard test bed [22], [56]. First, n_i inliers $\{\mathbf{p}_k\}_{k=1}^{n_i}$ are randomly generated on the 2-D plane via Gaussian distribution $N(0, 1)$ as the reference point set. Then each point is copied with Gaussian noise $N(0, \varepsilon)$ to generate N random point set by further adding n_o outliers via $N(0, 1)$.

2) CMU-POSE Sequence: The first two sequences used in [17], [20], [26], and [57]–[60] are from CMU *House* (30 landmarks, 101 frames) and CMU *Hotel* (30 landmarks, 111 frames) sequences (http://vasc.ri.cmu.edu//idb/html/motion/). The other two sequences are 225 frames of *VolvoC70* and *Houseblack* with 19 and 30 landmarks, covering a wide range of viewing angles from the POSE benchmark [61].

3) WILLOW-ObjectClass (ObjectClass for Short): This annotated dataset released in [20] is constructed using images from Caltech-256 and PASCAL VOC2007, with different number of images in each category: 109 *Face*, 50 *Duck*, 66 *Wine Bottle*, 40 *Motorbike*, and 40 *Car*. For each image, ten feature points are manually labeled on the object, and n_o random outliers detected by an scale-invariant feature transformation detector are added similar to [20] and [54].

C. Comparing Methods and Setting

1) Compared Methods: As our approach is general to handle both classic second-order GM and emerging hypergraph matching cases. Thus we focus on three important GM methods and their hyper variants as introduced earlier in this

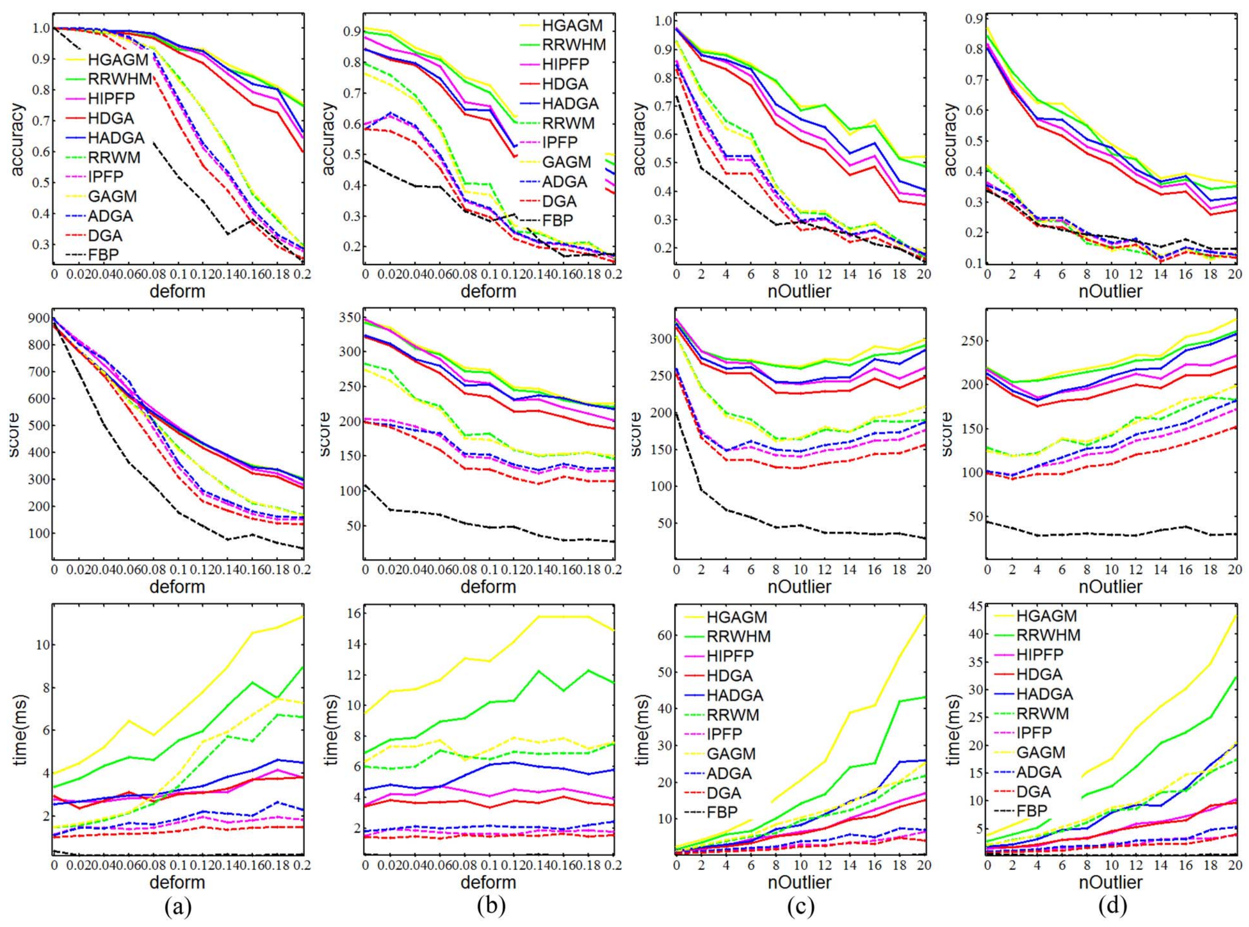


Fig. 7. GM on synthetic data by 100 trials. Left to right—(a) deformation test for $n_i = 30$, $n_o = 0$, (b) deformation test for $n_i = 20$, $n_o = 10$, (c) outlier test for $n_i = 20$, $\varepsilon = 0.1$, and (d) outlier test for $n_i = 20$, $\varepsilon = 0.2$. Best viewed in color.

paper: 1) RRWM [26]; 2) GAGM [18]; 3) IPFP [25]; and 4) the hypergraph variants RRWHM [16], HGAGM [32], and HIPFP [28]. The implementations are mostly from the authors' released source code. Due to space limitation, less competitive methods [14], [15], [22] under the linear assignment framework as tested in [16] are not evaluated. We also do not include [47] as it slightly outperforms RRWHM at the expense of more run time than RRWHM as shown in the evaluation of [47]. Thus RRWHM is our major competitor instead of [47].

We also evaluate the graph edit distance [39]–[41] based FBP GM [43] because: 1) in spirit with our method, this suboptimal method is designed for efficiency by a node-to-node cost matrix where structural information is approximately encoded and 2) recently, it becomes one of the most used practical algorithms. Serratosa [44] further proposed the speeded variant—square FB by replacing the Hungarian method with the Jonker–Volgenant alternative [24] coupled with a well-designed cost matrix. In our experiment, we adopt the Jonker–Volgenant method as the liner assignment solver for its efficiency [44].

2) Method Setting: For RRWM [26] and RRWHM [16]: we set $\lambda = 0.2$ for the jump reweighting parameter, and set the annealing parameter $\beta = 30$, the same as in the original paper. For GAGM [18] and HGAGM [32]: we fix $\beta_0 = 30$ rather than gradually increasing it for deterministic annealing to save time overhead. Thus the only difference between GAGM and RRWM is the update strategy as shown in Table I. We set the iteration stopping threshold $c = 0.01$ for these continuous methods. Or the iteration continues until it reaches the maximum iteration $k_{\max} = 50$. For Sinkhorn bistochasticization and other inner iterations of the continuous methods, we set the stopping threshold to $c = 0.01$ to reduce run time without sacrificing accuracy. Readers are referred to [18, Sec. 2.4] for discussion on setting a loose stop criterion for cost-efficiency. For (H)IPFP and our discrete solvers HADGA, HDGA and their second-order versions, we only need to set the maximum iteration count $k_{\max} = 50$ which helps achieve significant speedup, in addition with the code level optimization by using C++, compared with the test in [28] with excessive and unnecessary iterations. Moreover, to reduce the computational overhead caused by the addition on $\mathbf{H}$ that renders the tensor more dense, we break the super-symmetry and only using a third ratio by H_{iij} without H_{iji}, H_{jii}. We empirically find this policy does not significantly hurt the performance while saves both space and time overhead.

For FBP [43] and its variants, one key preprocessing step before running the linear assignment solver is building the node-to-node cost matrix. Very recently Serratosa and Cortés [46] evaluated eight variants of local structure to build the cost matrix C. In our experiments, we choose the Levenshtein distance based one to obtain C_{ed} due to

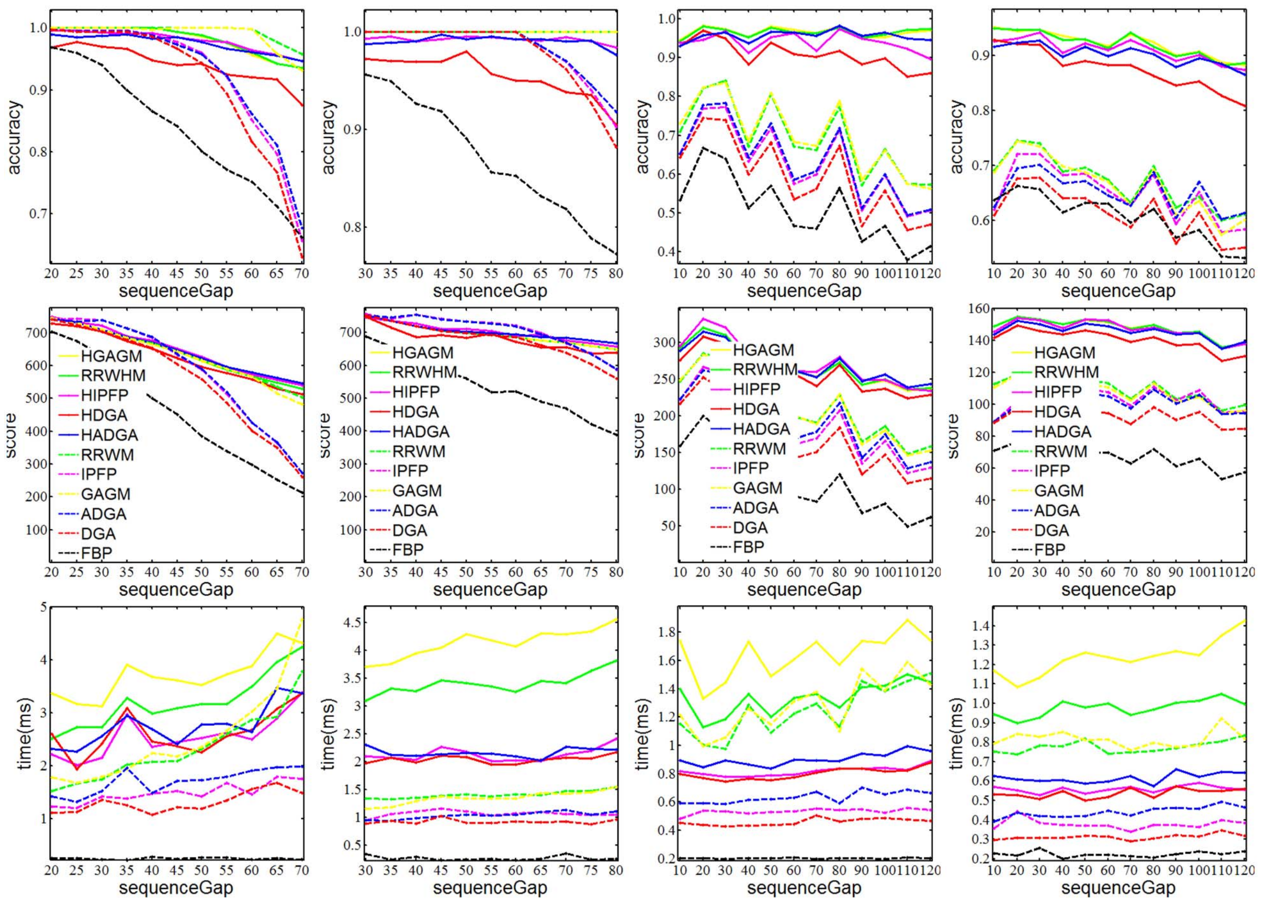


Fig. 8. GM on image sequence by varying gaps. Left to right: *Hotel*, *House* of CMU, *VolvoC70*, and *Houseblack* of POSE. Best viewed in color.

its leading performance as shown in [46]. To further improve the performance, we add the global information by the method used in [54]. It connects the global affinity matrix $\mathbf{K}$ used in the QAP formulation to the node-to-node cost used in FBP—denoted as C_{qa}. Specifically, we employ the Hungarian method to compute the assignment cost for the rest of nodes, by fixing the mapping $i \to a$. The input to the Hungarian method is the row/column of $\mathbf{K}$ (by inverting the value so as to change it from affinity to cost) that can be reshapen into the cost matrix with respect to $i \to a$, i.e., vec2mat(max($\mathbf{K}$) $-$ $\mathbf{K}_{ia,:}$) (see Fig. 1). Note this adaption may sacrifice the performance of FBP which is originally designed for effective exploring the individual graph attribute/adjancey information by the graph edit cost for deletion/insertion/substitution for nodes/edges [39]. Finally, we use the weighted cost matrix $C = (1/2)C_{\text{qa}} + (1/2)C_{\text{ed}}$ as the input to the linear assignment solver—Jonker–Volgenant [24]. Moreover, we reimplement this solver in C other than MATLAB as we find the latter runs significantly slower.

D. Results and Discussion

We test peer methods for second-order and hypergraph matching, respectively. For hypergraph matching, we include the third-order and second-order affinity terms as defined in (4) and (5), respectively. For second-order GM, the difference is that it only encodes the second-order terms.

The results of synthetic data tests are plotted in Fig. 7. It covers both deformation test and outlier test by varying the deformation parameter σ and number of outliers, respectively. Fig. 8 shows the performance by varying the number of gap frames over CMU and POSE sequences and so for the image collections Willow-ObjectClass in Fig. 9. All the experiments cover both second-order and hypergraph matching curves. We make serval observations and present our analysis as follows.

1) Hypergraph matching methods achieve better matching accuracy compared to their second-order counterparts at the cost of more overhead especially when the noises increase. For the first-order solver FBP, it runs significantly faster than other methods since one-shot linear assignment is performed given the input node-to-node cost matrix. However, only using the approximate cost matrix hurts its matching accuracy though in some tests it performs competitively against second-order methods. It is worth noting that the overhead for computing C_{qa} is not included in the run time plots for FBP. In fact computing C_{qa} needs $O(n^2)$ times of Hungarian method.
2) Discrete methods HADGA (ADGA), HDGA (DGA) and the peer method HIPFP (IPFP), all show cost-effectiveness in contrast to the continuous methods RRHWM (RRWM), HGAGM (GAGM). This shows the cost-effectiveness of discrete solvers. Remember that we have relaxed the stopping criterion for the continuous

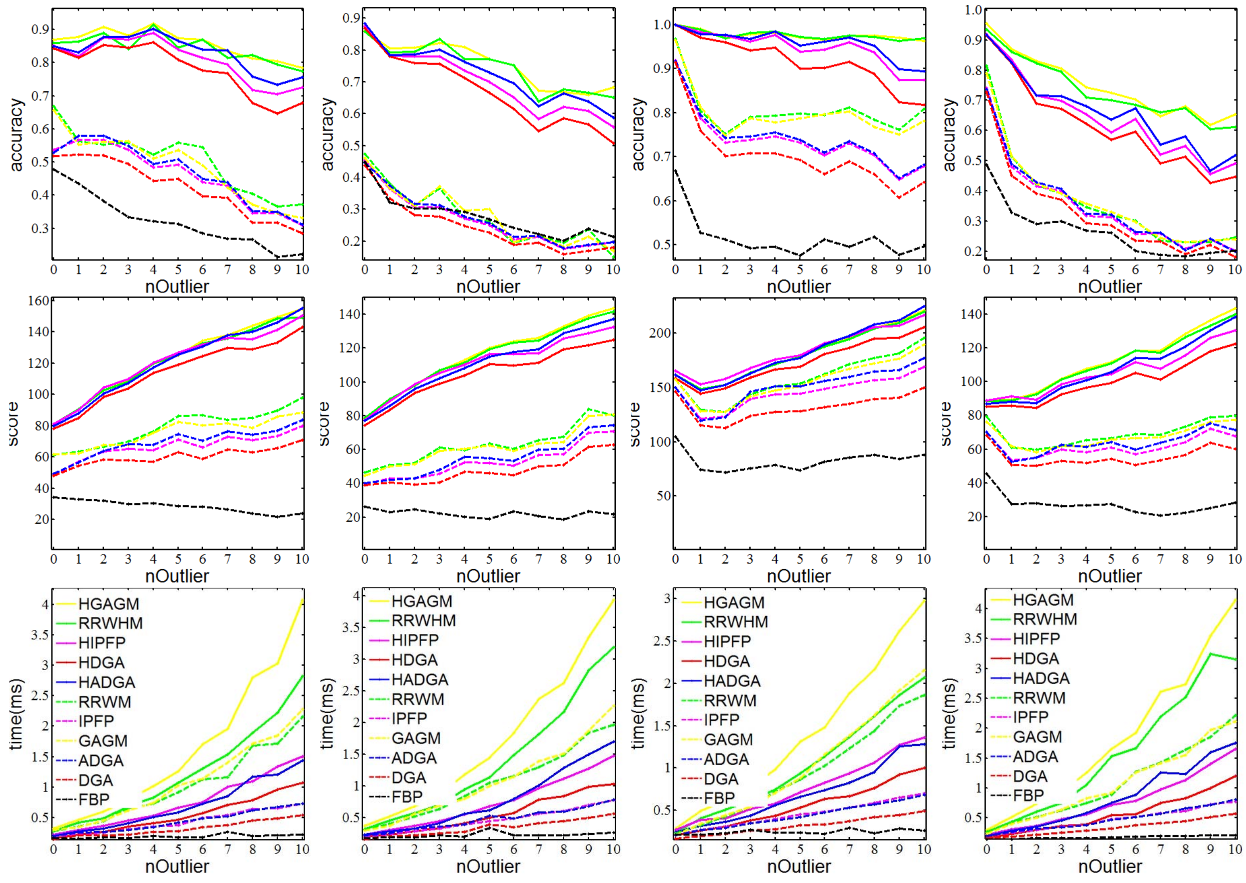


Fig. 9. GM on Willow-ObjectClass by varying the number of outliers. Left to right: *Car*, *Duck*, *Face*, and *Wine Bottle*. Best viewed in color.

methods much more than the setting of the original paper in order to speed up these methods, to an extent that almost approaching the saturation of their cost-effectiveness.

3) The matching score is in general consistent with the matching accuracy, though the gap between affinity score is often smaller. We think this is because modeling an informative affinity function is also challenging and already triggers emerging researches on learning-based affinity modeling [20], [53], [58]. Our parametric setting is in line with [16]–[18], [22], [25], and [26], which is a simplified and practical approximation to the semantic affinity. However, for the outlier tests, when the number of outliers increases the matching score often grows since more node correspondences are involved in the objective whatever they are inliers or outliers, which has no mechanism to suppress or reject outliers. The matching score for the FBP gets even smaller in outlier tests. This is perhaps because the node-to-node cost matrix implicitly leads to a different score function against the objective in (1).

4) There are some additional interesting observations. First, for CMU sequence—left two columns of Fig. 8, the accuracy for second-order matching sometimes is higher than hypergraph matching when the sequence frame gap is small. This is perhaps because the second-order affinity is enough informative given small deformation on that two less-challenging sequences. Second, in some tests, the continuous methods outperform discrete methods notably for matching accuracy when only second-order affinity information is used—refer to the third columns in Figs. 8 and 9 for the dotted curves. The gap is narrowed when higher-order information is added in hypergraph case (see the solid curves). We think the advantage of accuracy for the continuous methods in the second-order case is compensated in the hypergraph case as more distinctive information is added and the advantage of a complex algorithm is reduced. Third, for the POSE sequence in the right two columns of Fig. 8, the second-order matching accuracy shows a wavy shape over frame gaps. This is perhaps because there are two viewing angles azimuth and zenith under the spherical coordinate. Each image is viewed by these two angles which form a 2-D grid. We perform a zigzag traverse to form each sequence and this results in fluctuating performance.

5) Finally, we mention the behavior of HADGA, HDGA and their second-order versions. The baseline HDGA (in red) is the fastest one among all tests except for the one-shot first-order method FBP. The adaptive solver HADGA increases its accuracy with more time overhead but notably less than the continuous methods HGAGM, RRWHM. The quasi-discrete method HIPFP also runs faster than continuous methods and achieves similar

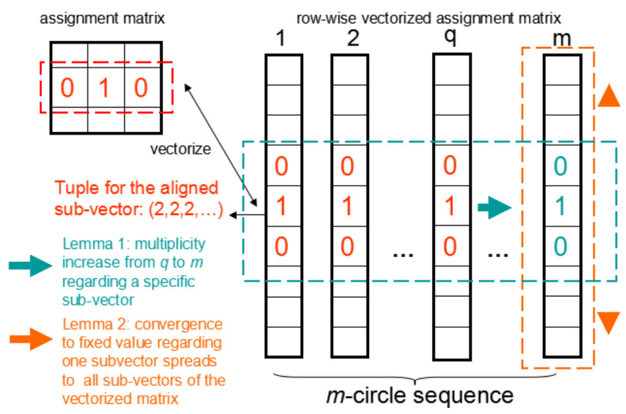

assignment matrix
row-wise vectorized assignment matrix
1
2
q
m
vectorize
Tuple for the aligned sub-vector: (2,2,2,...)
Lemma 1: multiplicity increase from q to m regarding a specific sub-vector
Lemma 2: convergence to fixed value regarding one subvector spreads to all sub-vectors of the vectorized matrix
m-circle sequence

Fig. 10. *Dimension-spread* idea illustration for Lemmas 1 and 2.

accuracy to HADGA in many cases. HADGA outperforms HIPFP on the challenging Willow-ObjectClass image collections though it runs moderately slower.

V. Conclusion

We propose a novel adaptive relaxation mechanism to advance the linear assignment-based framework for solving the general GM problem. The theoretical analysis shows the proposed algorithm will guarantee to converge to a fixed point under moderate conditions, and it always generates discrete solution via repeating the gradient assignment method, e.g., the Hungarian method. As a result, neither annealing nor subloop iterative normalization is needed during the linearized iterations. Extensive experimental results empirically suggest the cost-effectiveness of our discrete approach compared with the more widely used continuous methods, especially for the higher-order case. One interesting but also persistent effort is to study its global optimality property in theory, which is still an open question not well covered by existing literature on GM.

Appendix

Proof of Theorem 2: Now we present Lemmas 1 and 2 and prove their establishment. These two lemmas directly establish Theorem 2, which is the main result of this paper.

Let $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ denote the m-circle sequence that HDGA (Algorithm 1) converges to, where each $\mathbf{x}_s \in \{0,1\}^{n^2}$ by its definition is the vectorized permutation matrix of n unequal base vectors: $\mathbf{x}_s = [\mathbf{e}_{s_1}^T, \ldots, \mathbf{e}_{s_n}^T]^T$. The elements in $\{\mathbf{e}_s\}_{s=1}^n \in \{0,1\}^n$ are all zero except at the sth position being 1. Given a fixed d, the subparts $\{\mathbf{x}_s^d \triangleq \mathbf{x}_s(nd-n+1:nd)\}_{d=1}^n \in \{0,1\}^n$ in the sequence $\{\mathbf{x}_s\}_{s=1}^m$ also form a sequence $(\mathbf{x}_1^d, \mathbf{x}_2^d, \ldots, \mathbf{x}_m^d)$, and the total number of these sequences is n. One such sequence is illustrated by the green rectangle in Fig. 10 as the starting point for later proof.

The technical details are left in the constructive proof of Lemmas 1 and 2. We first conceptually describe our key idea termed as *dimension-spread* as illustrated in Fig. 10. Given n above defined sequences $\{(\mathbf{x}_1^d, \mathbf{x}_2^d, \ldots, \mathbf{x}_m^d)\}_{d=1}^n$, we can rewrite them to a compact scalar form $(j_1^d, j_2^d, \ldots, j_m^d)$ where each integer $j^d \in [1,n]$ represents 1s position in $\mathbf{x}^d$. Assume the largest multiplicity by Definition 2 over all $\{(j_1^d, j_2^d, \ldots, j_m^d)\}_{d=1}^n$ is $q_k = \max\{q_d\}_{d=1}^n \in [1,m]$ where q_d is the multiplicity of $(j_1^d, j_2^d, \ldots, j_m^d)$. Lemma 1 first shows how to increase the multiplicity by $q_k = q_k + 1$, and further to m by distorting the affinity tensor $\mathbf{H}$. In other words, $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ will converge with respect to its subpart $(\mathbf{x}_1^k, \mathbf{x}_2^k, \ldots, \mathbf{x}_m^k)$. This step is visualized by the horizontally expanding green rectangle in Fig. 10. Then Lemma 2 shows how to further enforce convergence for the whole $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ to a fixed point in permutation space. Thus Lemma 2 establishes Theorem 2. This step is shown in Fig. 10 by the vertically expanding orange rectangle.

Lemma 1: For the n m-tuple $\{(j_1^d, j_2^d, \ldots, j_m^d)\}_{d=1}^n$ derived from the m-circle sequence $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ generated by HDGA, by distorting the affinity tensor by HADGA, the largest multiplicity q_k can increase to m. In other words, the subpart $(\mathbf{x}_1^k, \mathbf{x}_2^k, \ldots, \mathbf{x}_m^k)$ converges to a fixed point.

Proof: For the kth subpart $(\mathbf{x}_1^k, \mathbf{x}_2^k, \ldots, \mathbf{x}_m^k)$ associated with the largest multiplicity q_k, in its scalar form, suppose $j_1^k = j_2^k = \cdots = j_{q_k}^k \neq j_{q_k+1}^k \cdots j_m^k$ without loss of generality. We further suppose the element 1 in the first q_k subparts $\mathbf{x}_1^k, \mathbf{x}_2^k, \ldots, \mathbf{x}_{q_k}^k$ is at position a, i.e., $j_1^k = j_2^k = \cdots = j_{q_k}^k = a$, and $j_{q_k+1}^k = b$. Such settings indicate the first $q_k + 1$ solutions out of the m consecutive ones assign node k in $\mathcal{G}_1$ to node a in $\mathcal{G}_2$ and the (q_k+1)th solution assigns node k to node b (for $a \neq b$). Accordingly, for the (q_k+1)th solution $\mathbf{x}_{q_k+1}$, suppose it assigns node l to node a, i.e., $j_{q_k+1}^l = a$ (for $l \neq a$). Note swapping from $\{l \leftrightarrow a, k \leftrightarrow b\}$ to $\{l \leftrightarrow b, k \leftrightarrow a\}$ for $\mathbf{x}_{q_k+1}$ will increase the multiplicity from q to $q+1$ with respect to $(\mathbf{x}_1^k, \mathbf{x}_2^k, \ldots, \mathbf{x}_m^k)$, while the current affinity tensor $\mathbf{H}$ disallows this swapped configuration as the score in (2) will be smaller which contradicts the condition that the relaxed objective score is at its peak and stops climbing for the given m-circle sequence. We compare the function score difference before and after this swapping: in general, $2n^{2m-2}$ terms in the objective function that relate to the mapping $k \leftrightarrow a$ or $l \leftrightarrow b$ in the (q_k+1)th solution will be affected. On the other hand, the new configuration for the q_k+1 solutions having the same match $k \leftrightarrow a$ will cover $n^{2(m-q_k-1)}$ terms. Thus a loose bound for the score decrease is $2n^{2m-2}$ when $\mathbf{H}$'s element falls in $[0, \lambda_{q_k}]$. In order to make the swap become score ascending, we perturb $\mathbf{H}$ by $\mathbf{H} = \mathbf{H} + \lambda_{q_k+1}\mathbf{H}^{q_k+1}$ and set $\lambda_{q_k+1} > (2n^{2m-2}\lambda_{q_k}/n^{2(m-q_k-1)})$. Remember that at each iteration the Hungarian method in HDGA will return the global optimal solution, thus the swap will be realized using the perturbed $\mathbf{H}$. Repeat such perturbation from q to m step by step along the same subpart $(\mathbf{x}_1^k, \mathbf{x}_2^k, \ldots, \mathbf{x}_m^k)$ till $q_k = m$, then it establishes Lemma 1. ■

Lemma 2: For the m-circle $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ generated by HDGA, if it converges to a fixed point regarding the subpart $(\mathbf{x}_1^k, \mathbf{x}_2^k, \ldots, \mathbf{x}_m^k)$, then by distorting the affinity tensor $\mathbf{H}$ by HADGA, $(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m)$ will converge to a fixed point in the permutation matrix space.

Proof: Given the kth subpart sequence whose multiplicity is m in its scalar form $j_1^k = j_2^k = \ldots = j_m^k = a$, it assigns node k

to node a in the other graph. Suppose the largest multiplicity of the rest n-1 subparts $\{j_1^d, j_2^d, \ldots, j_m^d\}_{d=1,\neq k}^n$ is q. Disturbing the $\mathbf{H}$ by $\mathbf{H} = \mathbf{H} + \lambda_{q+1}\mathbf{H}^{q+1}$ will not affect the fixed configuration $k \leftrightarrow a$ while increases q to $q+1$ by Lemma 1. This is because swapping out one of the m number of $k \leftrightarrow a$ will not increase the score after adding $\mathbf{H}^{q+1}$, thus the node matching for $k \leftrightarrow a$ will keep unchanged. Repeat the above step until the multiplicity of each subpart sequence reaches m, which establishes Lemma 2. ∎

Acknowledgment

The authors would like to thank the anonymous reviewers for valuable discussion and feedback.

References

[1] B. Yao and L. Fei-Fei, "Action recognition with exemplar based 2.5d graph matching," in *Proc. ECCV*, Florence, Italy, 2012, pp. 173–186.

[2] S. Xiang, F. Nie, and C. Zhang, "Contour matching based on belief propagation," in *Proc. ACCV*, Hyderabad, India, 2006, pp. 489–498.

[3] J. Carreira, A. Kar, S. Tulsiani, and J. Malik, "Virtual view networks for object reconstruction," in *Proc. CVPR*, Boston, MA, USA, 2015, pp. 2937–2946.

[4] W. Wang *et al.*, "Finding coherent motions and semantic regions in crowd scenes: A diffusion and clustering approach," in *Proc. ECCV*, Zürich, Switzerland, 2014, pp. 756–771.

[5] S. Xiang, F. Nie, Y. Song, and C. Zhang, "Contour graph based human tracking and action sequence recognition," *Pattern Recognit.*, vol. 41, no. 12, pp. 3653–3664, 2008.

[6] J. Wu, S. Pan, X. Zhu, C. Zhang, and X. Wu, "Positive and unlabeled multi-graph learning," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2016.2527239.

[7] X. Li, G. Cui, and Y. Dong, "Graph regularized non-negative low-rank matrix factorization for image clustering," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2016.2585355.

[8] X. Peng, Z. Yu, Z. Yi, and H. Tang, "Constructing the l2-graph for robust subspace learning and subspace clustering," *IEEE Trans. Cybern.*, to be published, doi: 10.1109/TCYB.2016.2536752.

[9] Y. Dong, D. Tao, and X. Li, "Nonnegative multiresolution representation-based texture image classification," *ACM Trans. Intell. Syst. Technol.*, vol. 7, no. 1, p. 4, 2015.

[10] M. Vento, "A long trip in the charming world of graphs for pattern recognition," *Pattern Recognit.*, vol. 48, no. 2, pp. 291–301, 2015.

[11] P. Foggia, G. Percannella, and M. Vento, "Graph matching and learning in pattern recognition in the last 10 years," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 33, no. 1, 2014, Art. no. 1450001.

[12] J. Yan *et al.*, "A short survey of recent advances in graph matching," in *Proc. ICMR*, New York, NY, USA, 2016, pp. 167–174.

[13] G. Fielding and M. Kam, "Weighted matchings for dense stereo correspondence," *Pattern Recognit.*, vol. 33, no. 9, pp. 1511–1524, 2000.

[14] R. Zass and A. Shashua, "Probabilistic graph and hypergraph matching," in *Proc. CVPR*, Anchorage, AK, USA, 2008, pp. 1–8.

[15] 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, Dec. 2011.

[16] J. Lee, M. Cho, and K. M. Lee, "Hyper-graph matching via reweighted random walks," in *Proc. CVPR*, Colorado Springs, CO, USA, 2011, pp. 1633–1640.

[17] F. Zhou and F. De la Torre, "Factorized graph matching," in *Proc. CVPR*, Providence, RI, USA, 2012, pp. 127–134.

[18] 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, Apr. 1996.

[19] F. Nie, X. Wang, M. I. Jordan, and H. Huang, "The constrained Laplacian rank algorithm for graph-based clustering," in *Proc. AAAI*, Phoenix, AZ, USA, 2016, pp. 1969–1976.

[20] M. Cho, K. Alahari, and J. Ponce, "Learning graphs to match," in *Proc. ICCV*, Sydney, NSW, Australia, 2013, pp. 25–32.

[21] S. Gold and A. Rangarajan, "Softmax to softassign: Neural network algorithms for combinatorial optimization," *J. Artif. Neural Netw.*, vol. 2, no. 4, pp. 381–399, 1995.

[22] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *Proc. ICCV*, Beijing, China, 2005, pp. 1482–1489.

[23] J. Munkres, "Algorithms for the assignment and transportation problems," *J. Soc. Ind. Appl. Math.*, vol. 5, no. 1, pp. 32–38, 1957.

[24] R. Jonker and A. Volgenant, "A shortest augmenting path algorithm for dense and sparse linear assignment problems," *Computing*, vol. 38, no. 4, pp. 325–340, 1987.

[25] M. Leordeanu, M. Hebert, and R. Sukthankar, "An integer projected fixed point method for graph matching and map inference," in *Proc. NIPS*, Vancouver, BC, Canada, 2009, pp. 1114–1122.

[26] M. Cho, J. Lee, and K. M. Lee, "Reweighted random walks for graph matching," in *Proc. ECCV*, Heraklion, Greece, 2010, pp. 492–505.

[27] Y. Tian *et al.*, "On the convergence of graph matching: Graduated assignment revisited," in *Proc. ECCV*, Florence, Italy, 2012, pp. 821–835.

[28] J. Yan *et al.*, "Discrete hyper-graph matching," in *Proc. CVPR*, Boston, MA, USA, 2015, pp. 1520–1528.

[29] J. Lee, M. Cho, and K. M. Lee, "A graph matching algorithm using data-driven Markov chain Monte Carlo sampling," in *Proc. ICPR*, Istanbul, Turkey, 2010, pp. 2816–2819.

[30] Y. Suh, M. Cho, and K. M. Lee, "Graph matching via sequential Monte Carlo," in *Proc. ECCV*, Florence, Italy, 2012, pp. 624–637.

[31] K. Adamczewski, Y. Suh, and K. M. Lee, "Discrete tabu search for graph matching," in *Proc. ICCV*, Santiago, Chile, 2015, pp. 109–117.

[32] M.-C. Chang and B. B. Kimia, "Measuring 3D shape similarity by graph-based matching of the medial scaffolds," *Comput. Vis. Image Understand.*, vol. 115, no. 5, pp. 707–720, 2011.

[33] G. H. Golub and C. F. Van Loan, *Matrix Computations*. Baltimore, MD, USA: Johns Hopkins Univ. Press, 1996.

[34] J. Demmel, I. Dumitriu, and O. Holtz, "Fast linear algebra is stable," *Numerische Math.*, vol. 108, no. 1, pp. 59–91, 2007.

[35] T. Cour, P. Srinivasan, and J. Shi, "Balanced graph matching," in *Proc. NIPS*, Vancouver, BC, Canada, 2006, pp. 313–320.

[36] A. Rangarajan, A. Yuille, and E. Mjolsness, "Convergence properties of the softassign quadratic assignment algorithm," *Neural Comput.*, vol. 11, no. 6, pp. 1455–1474, 1999.

[37] L. G. Shapiro and R. M. Haralick, "Structural descriptions and inexact matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-3, no. 5, pp. 504–519, Sep. 1981.

[38] W. J. Christmas, J. Kittler, and M. Petrou, "Structural matching in computer vision using probabilistic relaxation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 17, no. 8, pp. 749–764, Aug. 1995.

[39] A. Sanfeliu and K.-S. Fu, "A distance measure between attributed relational graphs for pattern recognition," *IEEE Trans. Syst., Man, Cybern.*, vol. SMC-13, no. 3, pp. 353–362, May/Jun. 1983.

[40] H. Bunke, "Error correcting graph matching: On the influence of the underlying cost function," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 21, no. 9, pp. 917–922, Sep. 1999.

[41] A. Solé-Ribalta, F. Serratosa, and A. Sanfeliu, "On the graph edit distance cost: Properties and applications," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 26, no. 5, 2012, Art. no. 1260004.

[42] K. Riesen and H. Bunke, "Approximate graph edit distance computation by means of bipartite graph matching," *Image Vis. Comput.*, vol. 27, no. 7, pp. 950–959, 2009.

[43] F. Serratosa, "Fast computation of bipartite graph matching," *Pattern Recognit. Lett.*, vol. 45, pp. 244–250, Aug. 2014.

[44] F. Serratosa, "Speeding up fast bipartite graph matching through a new cost matrix," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 29, no. 2, 2015, Art. no. 1550010.

[45] F. Serratosa, "Computation of graph edit distance: Reasoning about optimality and speed-up," *Image Vis. Comput.*, vol. 40, pp. 38–48, Aug. 2015.

[46] F. Serratosa and X. Cortés, "Graph edit distance: Moving from global to local structure to solve the graph-matching problem," *Pattern Recognit. Lett.*, vol. 65, pp. 204–210, Nov. 2015.

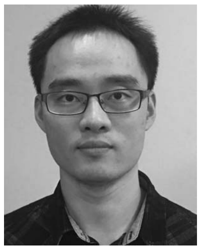
[47] Q. Ngoc, A. Gautier, and M. Hein, "A flexible tensor block coordinate ascent scheme for hypergraph matching," in *Proc. CVPR*, Boston, MA, USA, 2015, pp. 5270–5278.

[48] M. Bansal and K. Daniilidis, "Geometric polynomial constraints in higher-order graph matching," *arXiv 1405.6261*, 2014. [Online]. Available: https://arxiv.org/abs/1405.6261

[49] S. Park, S.-K. Park, and M. Hebert, "Fast and scalable approximate spectral matching for higher order graph matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 3, pp. 479–492, Mar. 2013.

[50] A. K. C. Wong and M. You, "Entropy and distance of random graphs with application to structural pattern recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-7, no. 5, pp. 599–609, Sep. 1985.

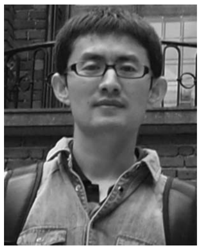
[51] M. Chertok and Y. Keller, "Efficient high order matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, no. 12, pp. 2205–2215, Dec. 2010.
[52] P. A. Regalia and E. Kofidi, "The higher-order power method revisited: Convergence proofs and effective initialization," in *Proc. ICASSP*, Istanbul, Turkey, 2000, pp. 2709–2712.
[53] M. Leordeanu, A. Zanfir, and C. Sminchisescu, "Semi-supervised learning and optimization for hypergraph matching," in *Proc. ICCV*, Barcelona, Spain, 2011, pp. 2274–2281.
[54] J. Yan, J. Wang, H. Zha, X. Yang, and S. Chu, "Consistency-driven alternating optimization for multigraph matching: A unified approach," *IEEE Trans. Image Process.*, vol. 24, no. 3, pp. 994–1009, Mar. 2015.
[55] J. Yan *et al.*, "A matrix decomposition perspective to multiple graph matching," in *Proc. ICCV*, Santiago, Chile, 2015, pp. 199–207.
[56] M. Cho, J. Sun, O. Duchenne, and J. Ponce, "Finding matches in a haystack: A max-pooling strategy for graph matching in the presence of outliers," in *Proc. CVPR*, Columbus, OH, USA, 2014, pp. 2091–2098.
[57] J. Yan *et al.*, "Joint optimization for consistent multiple graph matching," in *Proc. ICCV*, Sydney, NSW, Australia, 2013, pp. 1649–1656.
[58] 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, Jun. 2009.
[59] J. Yan *et al.*, "Graduated consistency-regularized optimization for multi-graph matching," in *Proc. ECCV*, Zürich, Switzerland, 2014, pp. 407–422.
[60] J. Yan, M. Cho, H. Zha, X. Yang, and S. M. Chu, "Multi-graph matching via affinity optimization with graduated consistency regularization," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 38, no. 6, pp. 1228–1242, Jun. 2016.
[61] F. Viksten, P.-E. Forssén, B. Johansson, and A. Moe, "Comparison of local image descriptors for full 6 degree-of-freedom pose estimation," in *Proc. ICRA*, Kobe, Japan, 2009, pp. 2779–2786.



Junchi Yan (M'10) received the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China.

He is with the Shanghai Key Laboratory of Trustworthy Computing, and the School of Computer Science and Software Engineering, East China Normal University, Shanghai, and IBM Research—China, Beijing 201203, China. His current research interests include computer vision and machine learning.

Dr. Yan was a recipient of the IBM Master Inventor Award, the IBM Research Division Award, the China Computer Federation Doctoral Dissertation Award, and the ACM China Doctoral Dissertation Nomination Award. He is a member of ACM.



Changsheng Li received the B.E. degree from the University of Electronic Science and Technology of China, Chengdu, China, in 2008, and the Ph.D. degree in pattern recognition and intelligent system from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2013.

He is an Algorithm Expert with Alibaba Group, Beijing. His current research interests include machine learning and data mining.

Yin Li is currently pursuing the Doctoral degree with the School of Interactive Computing, Georgia Institute of Technology, Atlanta, GA, USA.

He is currently researching on understanding videos captured from wearable cameras, also known as First Person Vision. His current research interests include computer vision, behavioral imaging, and mobile health.

Mr. Li was a co-recipient of the Best Student Paper Awards at Mobihealth 2014 and the IEEE Face and Gesture 2015.

Guitao Cao received the B.S. and M.S. degrees from Shandong University, Jinan, China, in 1991 and 2001, respectively, and the Ph.D. degree from Shanghai Jiao Tong University, Shanghai, China, in 2006.

She is an Associate Professor with the School of Computer Science and Software Engineering, East China Normal University, Shanghai. She has over 30 publications, including the IEEE Transactions on Biomedical Engineering. Her current research interests include image processing and pattern recognition, and media analysis and understanding.