Hyper-graph Matching via Reweighted Random Walks

Jungmin Lee Minsu Cho Kyoung Mu Lee
Department of EECS, ASRI, Seoul National University, 151-742, Seoul, Korea
jmlee.vision@gmail.com chominsu@gmail.com kyoungmu@snu.ac.kr
http://cv.snu.ac.kr

Abstract

Establishing correspondences between two feature sets is a fundamental issue in computer vision, pattern recognition, and machine learning. This problem can be well formulated as graph matching in which nodes represent feature points while edges describe pairwise relations between feature points. Recently, several researches have tried to embed higher-order relations of feature points by hyper-graph matching formulations. In this paper, we generalize the previous hyper-graph matching formulations to cover relations of features in arbitrary orders, and propose a novel state-of-the-art algorithm by reinterpreting the random walk concept on the hyper-graph in a probabilistic manner. Adopting personalized jumps with a reweighting scheme, the algorithm effectively reflects the one-to-one matching constraints during the random walk process. Comparative experiments on synthetic data and real images show that the proposed method clearly outperforms existing algorithms especially in the presence of noise and outliers.

1. Introduction

In computer vision, pattern recognition, and machine learning research, establishing correspondences between two feature sets is a fundamental and important issue [13, 14, 11, 3]. Given two sets of features extracted from images, the task is to find each feature's corresponding feature in the other set while preserving the relations with other features. This task is widely used in various vision problems such as scene registration, object recognition, feature tracking, and two- or three-dimensional shape matching. It is challenging to find perfect correspondences due to various reasons: imperfect feature descriptor, deformation in feature location due to viewpoint change or class variation, and outlier problem. The correspondence problem is well formulated as graph matching. Each graph is constructed with nodes representing features, and edges describing relations between two features. Correspondences are established by determining the mapping between two graphs, which preserves as much attributes as possible. Since the graph matching task is proven to be NP-hard [11], we need to find an approximated solution.

Conventional graph matching approaches mainly focus [7, 19, 12, 11, 5, 4, 10] on pairwise similarity between two correspondences such as distances among feature points. Pairwise relations, however, are not enough to incorporate the information about the entire geometrical structure of features. To overcome the limitation of pairwise similarity, several researchers [20, 6, 2] have tried to embed higher-order information into their problem formulations. They define the correspondence task as the hyper-graph matching problem and use hyper-edges in the graph to represent higher-order relations of feature points. However, the existing methods are not able to achieve a robust solution since the mapping constraints (such as one-to-one) are not effectively incorporated but mainly considered at their final discretization steps.

In this paper, we generalize the previous hyper-graph matching formulations to cover relations of features in arbitrary orders, and propose a novel state-of-the-art algorithm by reinterpreting the random walk concept on the hyper-graph in a probabilistic manner. In our formulation, relations in different orders are embedded altogether in a recursive manner, yielding a single higher-order similarity tensor. The hyper-graph matching problem is solved by ranking on an association hyper-graph via random walks. In particular, adopting personalized jumps with a reweighting scheme [4] into the random walk process, the algorithm effectively reflects matching constraints to produce a robust performance to large deformation and outlier noise. In experimental evaluations on both synthetic and real data, the performance is quantitatively evaluated and compared to those of existing hyper-graph matching methods [20, 6]. The proposed algorithm achieves a state-of-the-art performance showing robustness to both deformation and outlier noises.

1.1. Related Works

Since the graph matching problem is known as NP-hard, various approaches have tried to solve the problems by relaxed approximations [7, 19, 12, 11, 5, 4, 10]. Gold and Rangarajan proposed the Graduated Assignment (GA) [7], Wyk and Wyk introduced the Successive Projection Graph Matching (SPGM) [19], and Leordeanu *et al.* designed the Integer Projected Fixed Point (IPFP) [12] method. GA, SPGM, and IPFP are iterative methods while the mapping constraints are considered during their optimization process. Leordeanu and Hebert presented the Spectral Matching (SM) [11] and Cour *et al.* proposed the Spectral Matching with Affine Constraint (SMAC) [5]. SM and SMAC utilize the spectral property to calculate rank-1 approximation of the *affinity matrix*. Lee *et al.* [10] solved the graph matching problem by stochastic sampling, which simulates the mapping solution space. Cho *et al.* [4] provided a novel random walk view for graph matching and incorporated mapping constraints by a reweighting jump scheme. All these approaches [7, 19, 12, 11, 5, 4, 10] are restricted to the normal graph embedding unary and pairwise relations.

On the other hand, recent hyper-graph matching methods [20, 6, 2] incorporate higher-order similarity measures to achieve more accurate results. Zass and Shashua proposed Hyper-Graph Matching (HGM) [20], which introduces a novel view that the matching problem and its corresponding solution are related by the Kronecker product. Duchenne *et al.* proposed Tensor Matching (TM) [6], which can be interpreted as a higher-order extension of SM. It takes rank-1 approximation of the affinity tensor as a solution by using higher-order tensor power iteration. Chertok and Keller [2] also focused on rank-1 approximation of the affinity tensor. They calculated rank-1 vector by taking the leading left singular vector of the unfolded affinity tensor. However, all these existing higher-order approaches [20, 6, 2] are unable to effectively incorporate the matching constraints during their rank-1 approximation stage. The results of [4] show that incorporating the mapping constraints in the approximation has an important role in graph matching. Reinterpreting the random walk approach of [4] in the domain of hyper-graphs, the proposed algorithm effectively reflects the one-to-one matching constraints during random walks for hyper-graph matching.

2. Generalized Hyper-graph Matching

A hyper-graph $G = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ consists of nodes $v \in \mathcal{V}$, hyper-edges $e \in \mathcal{E}$, and attributes $\mathbf{a} \in \mathcal{A}$ associated with the hyper-edges. As illustrated in Fig. 1, a hyper-edge e encloses a subset of nodes with size $\delta(e)$ from $\mathcal{V}$, where $\delta(e)$ denotes the order of each hyper-edge. The goal of the hyper-graph matching problem is to establish mapping between nodes of two hyper-graphs $G^P = (\mathcal{V}^P, \mathcal{E}^P, \mathcal{A}^P)$ and $G^Q = (\mathcal{V}^Q, \mathcal{E}^Q, \mathcal{A}^Q)$.

Our *generalized* formulation covers matching of hyper-graphs having hyper-edges of any order $\delta(e)$. Suppose a set of all possible node correspondences $\mathcal{C} = \mathcal{V}^P \times \mathcal{V}^Q$, and k-

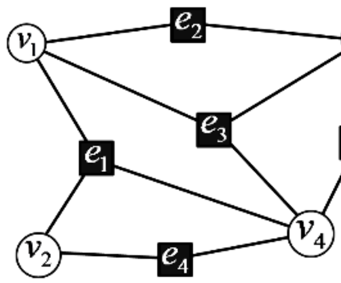


Figure 1. An example of hyper-graph with four nodes $\{v_1, \cdots, v_4\}$ and five edges $\{e_1, \cdots, e_5\}$ in circles and rectangles, respectively. Orders of edges might vary from edge to edge. The three edges that connect two nodes are $e_2(v_1, v_3)$, $e_4(v_2, v_4)$, and $e_5(v_3, v_4)$. Two hyper-edges connect three nodes, $e_1(v_1, v_2, v_4)$ and $e_3(v_1, v_3, v_4)$, respectively.

tuples $c_{w_1} = (v^P_{p_1}, v^Q_{q_1}), \cdots, c_{w_k} = (v^P_{p_k}, v^Q_{q_k}) \in \mathcal{C}$ among them. For k-th order hyper-graph matching, the similarities of the k-tuples are measured by comparing attributes of two k-th order hyper-edges $e^P_{p_1,\cdots,p_k}$ and $e^Q_{q_1,\cdots,q_k}$, which mean the hyper-edges connecting $v^P_{p_1}, \cdots, v^P_{p_k}$ and $v^Q_{q_1}, \cdots, v^Q_{q_k}$, respectively. Denoting the k-th order similarity function by $\Omega_k(\cdot,\cdot)$, the k-th order similarity of the k-tuple is measured by $\Omega_k(\mathbf{a}^P_{p_1,\cdots,p_k}, \mathbf{a}^Q_{q_1,\cdots,q_k})$. Generalizing this, we also consider other types of lower-order hyper-edges as well as the k-th order hyper-edges and formulate them into the k-th order tensor form in a recursive manner as follows:

$$\begin{aligned}\mathbf{H}^{(k)}_{w_1,\cdots,w_k} &= \Omega_k(\mathbf{a}^P_{p_1,\cdots,p_k}, \mathbf{a}^Q_{q_1,\cdots,q_k}) \\ &\quad + \lambda^{(k-1)} \sum_{l=1}^{k} \mathbf{H}^{(k-1)}_{\{w_1,\cdots,w_k\}\setminus w_l}, \\ \mathbf{H}^{(1)}_{w_i} &= \Omega_1(\mathbf{a}^P_{p_i}, \mathbf{a}^Q_{q_i}),\end{aligned} \tag{1}$$

where $\lambda^{(k)}$ represents the weighting factor of k-th order similarity value and the superscript on $\mathbf{H}$ denotes the dimension of a tensor. Note that subscripts are used to represent element indices in vectors, matrices, and tensors. Here, we define overall order δ which is the maximum order value among all hyper-edges:

$$\delta = \max_{e \in \{\mathcal{E}^P \cup \mathcal{E}^Q\}} \delta(e). \tag{2}$$

The overall order δ implies that we can measure similarities up to the order of δ. Thus, the resulting similarity tensor $\mathbf{H}^{(\delta)}$ becomes a single high-order tensor, which contains similarity information of all the orders. In the rest of the paper, we abbreviate $\mathbf{H}^{(\delta)}$ as $\mathbf{H}$.

The solution of hyper-graph matching is determined as a subset of candidate correspondences $\mathcal{C}$ and efficiently represented using a binary assignment matrix $\mathbf{X} \in \{0,1\}^{n^P \times n^Q}$,

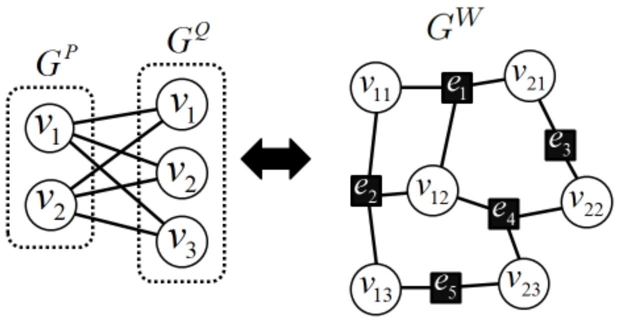


Figure 2. An example of an association hyper-graph. There are two and three nodes in the graph G^P and G^Q, respectively. There are six possible correspondences between two graphs (left). An association hyper-graph consists of six nodes, which represent candidate correspondences. There are five hyper-edges which connect some of candidate correspondences. For example, the hyper-edge weight e_1 is defined by the similarity of v_{11}, v_{12}, and v_{21} (right).

where n^P and n^Q denote the numbers of nodes in G^P and G^Q, respectively. If $v_p^P \in \mathcal{V}^P$ matches to $v_q^Q \in \mathcal{V}^Q$, $\mathbf{X}_{p,q} = 1$, otherwise $\mathbf{X}_{p,q} = 0$. In graph matching problems, it is natural to impose two-way constraints (i.e., one-to-one constraints) that make $\mathbf{X}$ a permutation matrix:

$$\mathbf{X}\mathbf{1}_{n^Q\times 1} \leq \mathbf{1}_{n^P\times 1}, \quad \mathbf{X}^T\mathbf{1}_{n^P\times 1} \leq \mathbf{1}_{n^Q\times 1}, \tag{3}$$

where $\mathbf{1}_{n\times 1}$ denotes an all-ones vector with size n and the inequalities hold for every element. For a convenient representation, the vectorized version $\mathbf{x}$ of the matrix $\mathbf{X}$ is used. The hyper-graph matching score is defined for an assignment $\mathbf{x}$ as follows:

$$S(\mathbf{x}) = \sum_{w_1,\cdots,w_\delta} \mathbf{H}_{w_1,\cdots,w_\delta}\mathbf{x}_{w_1}\cdots\mathbf{x}_{w_\delta}. \tag{4}$$

The resulting score $S(\mathbf{x})$ is the summation of similarity values associated with all δ-tuples of correspondences in a mapping $\mathbf{x}$. Since $\mathbf{H}$ itself also contains similarity values of all the lower-orders, $S(\mathbf{x})$ amounts to the summation of all similarity values in all orders. Using the tensor product [16], it is simply represented by

$$S(\mathbf{x}) = \mathbf{H}\otimes_1 \mathbf{x}\cdots\otimes_\delta \mathbf{x}. \tag{5}$$

Then, the goal of the hyper-graph matching problem becomes finding the assignment vector $\mathbf{x}^*$ (or assignment matrix $\mathbf{X}^*$), which maximizes the matching score function $S(\mathbf{x})$ under the constraints of Eq.(3);

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} S(\mathbf{x}). \tag{6}$$

3. Proposed Method

This section interprets the hyper-graph matching problem based on random walks, and explains our algorithm in the framework.

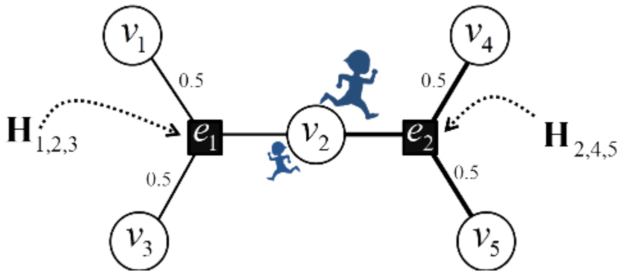


Figure 3. Description of the random walks along hyper-edges. A random walker in the node v_2 travels along the hyper-edge e_1 with the probability $\mathbf{H}_{1,2,3}/(\mathbf{H}_{1,2,3}+\mathbf{H}_{2,4,5})$, or e_2 with $\mathbf{H}_{2,4,5}/(\mathbf{H}_{1,2,3}+\mathbf{H}_{2,4,5})$. Each hyper-edge covers two other nodes except the node v_2 and the random walker randomly chooses one of the two nodes. For example, if we assume that $\mathbf{H}_{2,4,5}$ is greater than $\mathbf{H}_{1,2,3}$, the random walker has higher chance to travel along the hyper-edge e_2 and to select v_4 or v_5.

3.1. Association Hyper-graph

To solve the hyper-graph matching problem in a random walk view, it is necessary to define an association graph [4]. As illustrated in Fig. 2, we construct an association hyper-graph $G^W = (\mathcal{V}^W, \mathcal{E}^W, \mathcal{A}^W)$ by considering each candidate correspondence $c_w = (v_p^P, v_q^Q)$ as a node $v_w \in \mathcal{V}^W$. Here, a random walk from a node v_{w_1} to another node v_{w_2} on this graph G^W implies a walk from a correspondence c_{w_1} to another correspondence c_{w_2} between G^P and G^Q. In the basis hyper-graphs G^P and G^Q, two hyper-edges $e^P_{p_1,\cdots,p_\delta}$ and $e^Q_{q_1,\cdots,q_\delta}$ are derived from δ-tuple of correspondences $c_{w_1} = (v^P_{p_1}, v^Q_{q_1}), \cdots, c_{w_\delta} = (v^P_{p_\delta}, v^Q_{q_\delta})$. Hence, a hyper-edge $e_{w_1,\cdots,w_\delta}$ in the association hyper-graph G^W embeds the similarity value of the δ-tuple, which is defined by Eq. (1) and assigned as attribute weights. The construction of G^W allows us to solve the hyper-graph matching problem by ranking the nodes $v \in \mathcal{V}^W$ in G^W. The higher score a node in G^W has, the better correspondence between G^P and G^Q it represents.

3.2. Random Walks on Hyper-graph

In the random walk [17] framework on a normal graph (i.e., $\delta = 2$), the random walker on a node v_i travels to another node v_j with the probability proportional to the edge weight of $e_{i,j}$, which corresponds to $\mathbf{H}_{i,j}$ ($\mathbf{H}$ becomes a matrix) in the case of our formulation. Classical random walks make the transition matrix $\mathbf{P}$ by normalizing each row of the $\mathbf{H}$: $\mathbf{P} = \mathbf{D}^{-1}\mathbf{H}$, where $\mathbf{D}_{ii}$ is i-th row sum of $\mathbf{H}$. We generalize this idea for higher-order cases (i.e., $\delta > 2$) [21]. In a hyper-graph, two steps are required to move the random walker from one node to another one as illustrated in Fig. 3. First, the random walker on a node moves along one of its connected hyper-edges with a probability proportional to the weight of that hyper-edge. Second, the random walker uniformly selects another node from

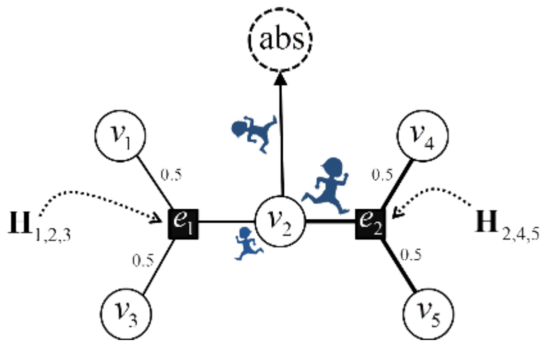


Figure 4. Description of the absorbing node. The absorbing node soaks weights from every node. For example, it soaks $d_{\max} - d_2$ of weight from the node v_2. As a result, a random walker in the node v_2 moves to the absorbing node with the probability $1 - d_2/d_{\max}$, or travels along hyper-edges (e_1 or e_2) with the probability $d_2/d_{\max}$.

chosen hyper-edge's nodes and moves to that node. Additionally, we may assume that $\mathbf{H}$ is *super-symmetric* so that $\mathbf{H}$ is invariant under any permutation of $\{c_{w_1}, \cdots, c_{w_\delta}\}$. It is compatible with the fact that the similarity of the δ correspondence tuple is independent of their sequences. Thus, the transition tensor and the updating rule for the hyper-graph random walks can be described as follows:

$$\mathbf{P}_{w_1,\cdots,w_\delta} = \mathbf{H}_{w_1,\cdots,w_\delta}/(\mathbf{H} \otimes_2 \mathbf{1} \cdots \otimes_\delta \mathbf{1})_{w_1}, \quad (7)$$

$$\mathbf{x}^{(t+1)} = \mathbf{P} \otimes_2 \mathbf{x}^{(t)} \cdots \otimes_\delta \mathbf{x}^{(t)}, \quad (8)$$

where $\mathbf{1}$ is all one vector and $\mathbf{x}^{(t)}$ is distribution of $\mathbf{x}$ at t-th iteration. Note that the division in Eq. (7) is element-wise, and $(\mathbf{H} \otimes_2 \mathbf{1} \cdots \otimes_\delta \mathbf{1})_{w_1}$ corresponds to the sum of all hyper-edge weights for node v_{w_1}. The resulting tensor $\mathbf{P}$ is *stochastic* in the probabilistic sense.

In the association hyper-graph, every node represent a candidate correspondence. There are not only true correspondences (*inliers*) but also false ones (*outliers*). Every node in the graph has one vote (outgoing probabilities are summed to one). When using the standard normalization of Eq. (7) in our matching problem, the probabilities relevant to outlier nodes are likely to be scaled up since the outlier nodes usually have small weights in the association hyper-graph.

3.3. Affinity-preserving Random Walks

To address this issue by extending the approach of [4], an *absorbing node* is added to the association hyper-graph as shown in Fig. 4. The absorbing node is designed to soak $d_{\max} - d_w$ of weight from the node v_w, where d_w represents the degree (sum of the weight values associated with a certain node) of the node v_w and $d_{\max}$ represents the maximum value among d_w which are defined as follows:

$$d_w = \sum_{w_2,\cdots,w_\delta} \mathbf{H}_{w,w_2,\cdots w_\delta} = (\mathbf{H} \otimes_2 \mathbf{1} \cdots \otimes_\delta \mathbf{1})_w, \quad (9)$$

Algorithm 1 Reweighted Random Walks Hyper-graph Matching

```
1:  Given similarity tensor $\mathbf{H}$, reweight $\alpha$, and inflation $\beta$
2:  Initialize the mapping matrix $\mathbf{X}$ uniformly
3:  $d_{\max} = \max_w \sum_{w_2,\cdots,w_\delta} \mathbf{H}_{w,w_2\cdots,w_\delta}$
4:  Derive the transition tensor $\mathbf{P} = \mathbf{H}/d_{\max}$
5:  **repeat**
6:      ( Affinity-preserving random walking )
7:      $\bar{\mathbf{x}} = \mathbf{P} \otimes_2 \mathbf{x} \cdots \otimes_\delta \mathbf{x}$
8:      ( Reweighting with the two-way constraints )
9:      $\mathbf{y} = \exp(\beta\bar{\mathbf{x}}/\max\bar{\mathbf{x}})$: element-wise exponential
10:     **repeat**
11:         Normalize across rows:
12:         $\mathbf{Y}_{p,q} = \mathbf{Y}_{p,q}/\sum_{q=1}^{n^Q} \mathbf{Y}_{p,q}$
13:         Normalize across columns:
14:         $\mathbf{Y}_{p,q} = \mathbf{Y}_{p,q}/\sum_{p=1}^{n^P} \mathbf{Y}_{p,q}$
15:     **until** $\mathbf{Y}$ converges
16:     $\mathbf{y} = \mathbf{y}/\|\mathbf{y}\|_1$
17:     ( Random walking with reweighted jumps )
18:     $\mathbf{x} = \alpha\bar{\mathbf{x}} + (1-\alpha)\mathbf{y}$
19:     $\mathbf{x} = \mathbf{x}/\|\mathbf{x}\|_1$
20: **until** $\mathbf{x}$ converges
21: Discretize $\mathbf{X}$ by the matching constraints
```

$$d_{\max} = \max_w d_w, \quad (10)$$

where subscript $(\cdot)_k$ represents k-th element in a vector. Adding the absorbing node enables the random walks to transit with probability proportional to the original weight value, that is the *affinity-preserving* property [4]. Since the probability of the random walker staying in the absorbing node is not our concerns in the matching problem, we can simply rewrite Eq. (7) as follows:

$$\mathbf{P} = \mathbf{H}/d_{\max}. \quad (11)$$

3.4. Reweighting Jumps

While we now have the affinity-preserving random walk framework on the hyper-graph, the one-to-one mapping constraints in Eq. (3) are not considered during the random walk process. To impose the matching constraints on the random walks, we employ the personalized PageRank approach [9]:

$$\mathbf{x}^{(t+1)} = \alpha\mathbf{P} \otimes_2 \mathbf{x}^{(t)} \cdots \otimes_\delta \mathbf{x}^{(t)} + (1-\alpha)\mathbf{r}, \quad (12)$$

where, $\mathbf{r}$ is the personalized vector and α means a bias between random walking and personalized jumps. It means that the random walker travels along with its hyper-edges with a probability α, or jumps according to the probability distribution $\mathbf{r}$ with a probability $1-\alpha$. We impose

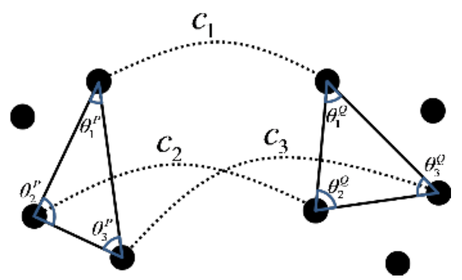


Figure 5. An example of calculating higher-order similarity tensor. Three candidate correspondences form two triangles. Higher-order similarity tensor can be calculated by comparing corresponding angles. See Eq. 14.

the one-to-one constraints on the random walks using the reweighting jump $\mathbf{r}$. Adopting a reweighting function $f(\cdot)$, the reweighted random walks is formulated by

$$\begin{aligned} \mathbf{x}^{(t+1)} &= \alpha \mathbf{P} \otimes_2 \mathbf{x}^{(t)} \cdots \otimes_\delta \mathbf{x}^{(t)} \\ &+ (1-\alpha) f(\mathbf{P} \otimes_2 \mathbf{x}^{(t)} \cdots \otimes_\delta \mathbf{x}^{(t)}). \end{aligned} \tag{13}$$

Here, the reweighting function $f(\cdot)$ guides the current solution to move toward a solution that satisfies the mapping constraints of Eq. (3). We define $f(\cdot)$ as two steps following the approach of [4]; the inflation and the Sinkhorn bistochastic normalization [18]. The inflation step makes nodes with a high probability become higher, and makes nodes with a small probability become smaller. The bistochastic normalization step makes the current state distribution become more likely to be a permutation matrix, which satisfies the one-to-one constraints. The entire procedure of proposed the affinity-preserving reweighted random walks method on the hyper-graph is described in Algorithm 1. The final step makes the converged distribution into a binary permutation matrix, which can be done by any discretization method. The Hungarian [15] method is known to be optimal for this linear assignment problem.

4. Experiments

In this section, we evaluate the performances of several existing hyper-graph matching approaches by conducting experiments on both synthetic point-set data and real image data. For HGM [20] and TM [6] methods, we use the authors' MATLAB codes. We termed ours RRWHM, which is the abbreviation of Reweighted Random Walks Hyper-graph Matching. Our RRWHM is also implemented in MATLAB, and the parameters, α and β, are empirically tuned as 0.2 and 30, respectively. Each quantitative result in our synthetic experiments is acquired from averages of 100 random trials.

4.1. Synthetic Points-set Matching

In the point-set matching test, deformation noise and outliers are added to simulate a real matching problem. First, we randomly generate n_{in} inlier points using the normal distribution $N(0,1)$ in 2D domain P, which become nodes in the graph G^P. To generate points in the second domain Q, the Gaussian deformation noise $N(0,\sigma^2)$ is independently added to the n_{in} points. Outlier noise is simulated using additional n_{out} random Gaussian points (i.e., $N(0,1)$) to each domain.

In the synthetic point-set matching experiment, it is impossible to directly compare two points by the unary similarity ($\delta = 1$) since there is no distinctive descriptor for a single point. For pairwise cases ($\delta = 2$), we can compare distances of two point sets to compute the pairwise similarity values. However, this measure is very sensitive to the scale change (i.e., invariant up to translation and rotation). To achieve scale invariance, we exploit triplets of correspondences ($\delta = 3$) as shown in Fig. 5, where the sine values of three angles in each triangle are compared each other [6].

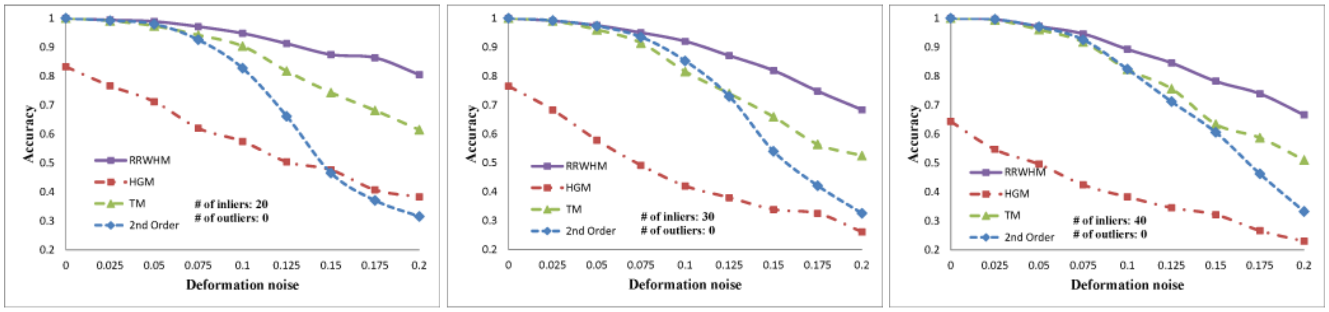
$$\mathbf{H}_{w_1,w_2,w_3} = \exp\left[-\frac{1}{\sigma_s}\sum_{k=1}^{3}|\sin(\theta^P_{w_k}) - \sin(\theta^Q_{w_k})|\right], \tag{14}$$

where $\theta^P_{w_k}$ and $\theta^Q_{w_k}$ denote angles of nodes related to the correspondence w_k in the domain P and Q, respectively. In the entire experiment, we empirically set $\sigma_s = 0.5$ for the best performance.

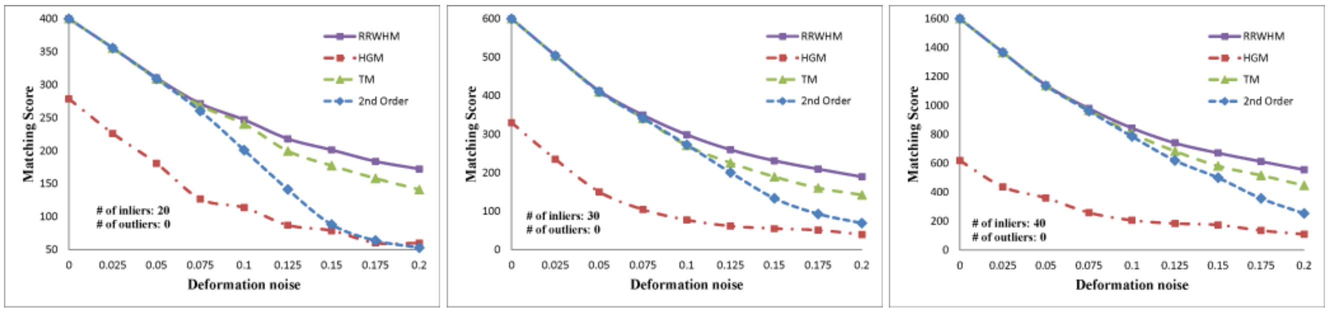
When constructing the higher-order similarity tensor, the storage size of the tensor should be considered. Suppose we have m by m point-matching problem, then there will be m^2 possible candidate correspondences. For the $\delta = 3$ case, the size of the resulting full-affinity tensor will be $\binom{m^2}{3}$, an order of $O(m^6)$, which demands a huge amount of memory. Thus, in this work, to make the higher-order similarity tensor sparse, we sample $n = n_{\text{in}} + n_{\text{out}}$ triangles per point and find n^2 nearest neighbors using the *approximate nearest neighbor* searching algorithm [1, 6].

The experimental results are plotted in Figs. 6 and 7. To compare the robustness of the higher-order feature, we also evaluate the performance of the state-of-the-art pairwise graph matching algorithm [4] denoted as *2nd Order*. In the deformation test, we vary the deformation noise σ from 0 to 0.2 with the interval 0.025 while n_{in} is fixed to 20, 30, and 40. In the outlier noise test, the number of outliers n_{out} varies from 0 to 20 with the interval 2, while σ is set to 0, 0.1, and 0.2.

The performances of each method are measured in both accuracy and scores. The accuracy represents the ratio between the number of correctly selected correspondences and that of the inlier points n_{in}. The matching score is defined by $S(\mathbf{X})$ in Eq. (4). In most experiments, the proposed



(a) Accuracy with 20, 30, and 40 inlier points (left from right).



(b) Matching score with 20, 30, and 40 inlier points (left from right).

Figure 6. Performance according to the deformation change. The top row shows accuracy and the bottom row shows matching score.

method outperforms the others in the sense of both the accuracy and the matching score. HGM does not gain a good performance due to the information loss at its marginalization step, and TM does not deal with the presence of outliers or large amount of deformation noise in computing the principal eigenvector of the affinity tensor $\mathbf{H}$. To the contrary, RRWHM clearly outperforms them since its affinity-preserving property and reweighting scheme enable to avoid adverse effects from deformation and outliers. Without deformation noise, the second-order method of [4] performs best in accuracy on the outlier experiment as shown at the left-top of Fig. 7. This is mainly because pairwise distance measure has a more robust characteristic to outliers for ideal case without deformation. The second-order method [4], however, does not show satisfactory performance in the presence of deformation noise in all the other experiments.

4.2. Image Matching Problem

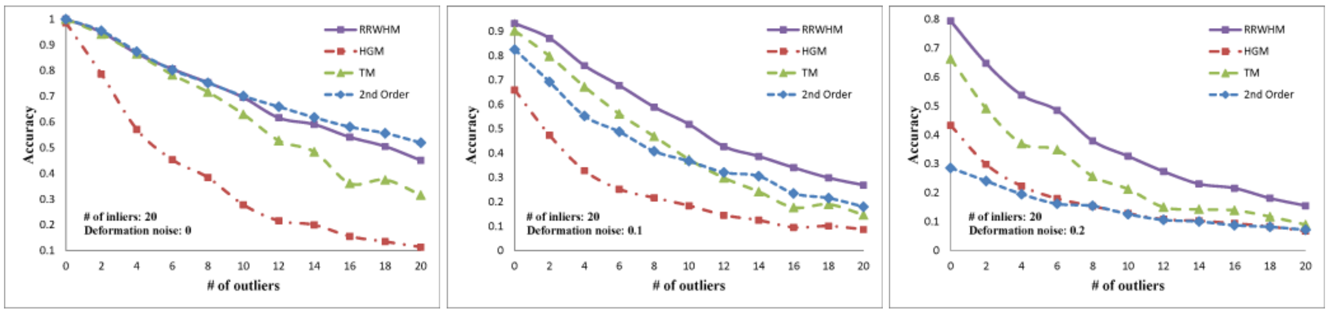
We perform two feature matching experiments using real images. In the first image experiment, we perform a feature tracking task on the *House* image sequence[1]. 30 feature points are manually tracked over all image sequence (110 images) to construct the ground-truth data. This image sequence is taken from the same object but the viewpoint varies along the sequence, that is, the deformation noise in real situation. The larger the sequence gap is, the larger the deformation noise imposed to each point is. The performance of each method is tested along every sequence gap (from 10 to 100 with interval 10). In this experiment, the third-order similarity measure is used to calculate higher-order similarity tensor $\mathbf{H}_{w_1,w_2,w_3}$ in the same way as Eq. (14). The sparseness of $\mathbf{H}$ is also acquired by following the same procedure of Sec. 4.1. A comparative example is shown in Fig. 8 (a)-(d), and the overall performance comparison is plotted in Fig. 8 (e). While HGM and TM gradually degrade with the increasing sequence gaps, RRWHM gives almost perfect accuracy in this experiment.

In the second image experiment, we test on 30 image pairs taken from MSRC v2 dataset[2] and Caltech 101 dataset[3]. The feature points are extracted automatically using the MSER feature detector [14] so that in this experiment outlier feature points from background clutters are present unlike the previous *House* sequence experiment. Then, 500 correspondences with the lowest SIFT descriptor distances [13] are collected for candidate matches. The higher-order similarity tensor $\mathbf{H}$ is constructed according to Eq. (14) using these 500 matches. For the sparseness of $\mathbf{H}$, elements with lower values than 0.1 are discarded. For evaluation, we manually labelled ground truths for all candidate correspondences. Some comparative examples

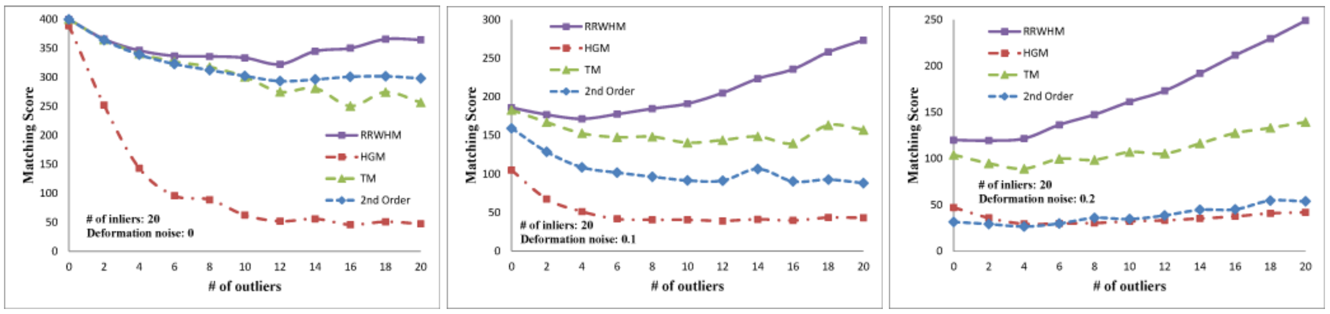
[1]CMU Image Data Base: http://vasc.ri.cmu.edu/idb/html/motion/

[2]MSRC v2: http://research.microsoft.com/en-us/projects/objectclassrecognition/

[3]Caltech 101: http://www.vision.caltech.edu/Image_Datasets/Caltech101/



(a) Accuracy with 0, 0.1, and 0.2 deformation noise (left from right).



(b) Matching score with 0, 0.1, and 0.2 deformation noise (left from right).

Figure 7. Performance according to the outlier change. The top row shows accuracy and the bottom row shows matching score.

are shown in Fig. 9, and the average performance comparison on the dataset is summarized in Table 1. Note that average matching accuracy values are underestimated since our ground truths are generously labelled not considering matching constraints. The results show our RRWHM outperforms the others also in this real image experiment with outliers.

Table 1. Performance on the real image dataset (30 pairs)

Methods	HGM	TM	RRWHM
Avg. Accuracy (%)	41.36	41.12	**45.03**
Avg. Relative Score (%)	81.50	81.57	**98.38**

More information and the source code will be provided at our project site: `http://cv.snu.ac.kr/research/~RRWHM/`.

5. Conclusion

We introduced a generalized formulation of the hyper-graph matching problem to make it possible to embed similarity measures of arbitrary orders, and proposed a state-of-the-art hyper-graph matching method based on a random walk view. Extending the reweighted random walks [4] to hyper-graphs, our RRWHM achieves a robust performance against deformation and outlier noises. Experimental evaluations demonstrate that our method clearly outperforms the previous hyper-graph matching approaches [20, 6] on both synthetic and real problems.

References

[1] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman, and A. Wu. An optimal algorithm for approximate nearest neighbor searching. *ACM*, 1994. 1637

[2] M. Chertok and Y. Keller. Efficient high order matching. *PAMI*, 2010. 1633, 1634

[3] M. Cho, J. Lee, and K. M. Lee. Feature correspondence and deformable object matching via agglomerative correspondence clustering. 2009. 1633

[4] M. Cho, J. Lee, and K. M. Lee. Reweighted random walks for graph matching. *ECCV*, 2010. 1633, 1634, 1635, 1636, 1637, 1638, 1639

[5] T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. *NIPS*, 2007. 1633, 1634

[6] O. Duchenne, F. Bach, I. Kweon, and J. Ponce. Tensor-based algorithm for high-order graph matching. *CVPR*, 2009. 1633, 1634, 1637, 1639

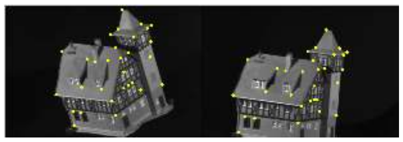
[7] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *PAMI*, 1996. 1633, 1634

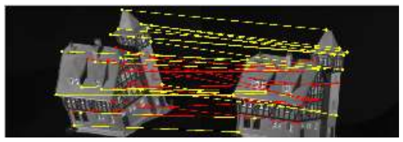
[8] L. Grady. Random walks for image segmentation. *PAMI*, 2006.

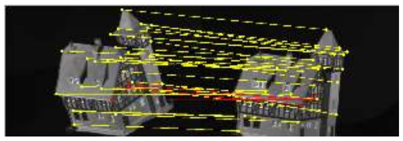
[9] T. Haveliwala. Topic-sensitive pagerank. *WWW*, 2002. 1636

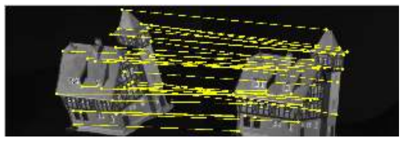
[10] J. Lee, M. Cho, and K. M. Lee. A graph matching algorithm using data-driven markov chain monte carlo sampling. *ICPR*, 2010. 1633, 1634

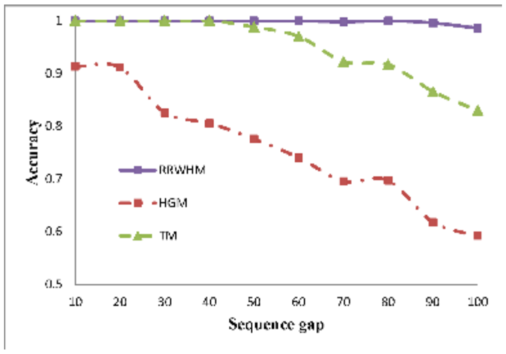
[11] M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. *ICCV*, 2005. 1633, 1634


(a) An example input pair.


(b) HGM: 17 correct matches out of 30.

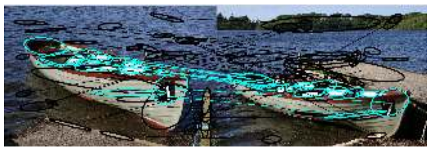

(c) TM: 27 correct matches out of 30.

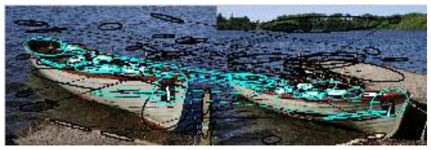

(d) RRWHM: 30 correct matches out of 30.

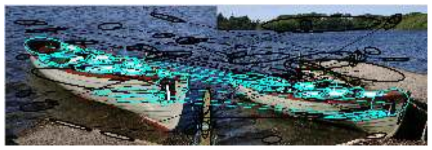


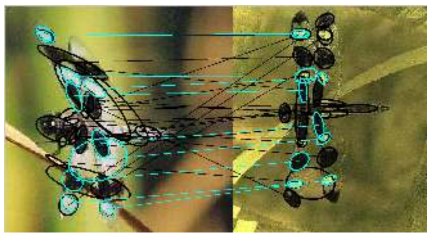
(e) Accuracy according to the sequence gap.

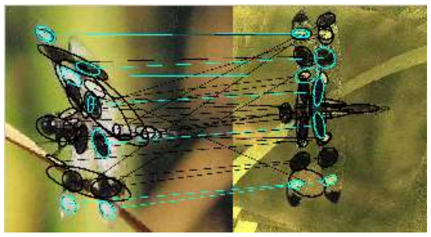
Figure 8. Experiments on images from *House* object with different viewpoints. (a) is an input pair. (b), (c), and (d) are the matching results when the sequence gap is 80. Correct and incorrect matches are denoted as yellow and red lines, respectively. (e) is the performance plot according to the sequence gap (view change). These figures are best viewed in color.

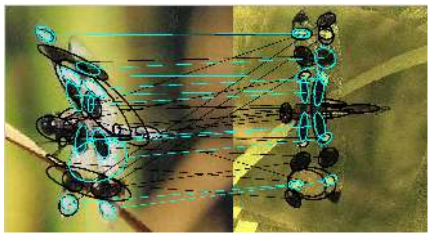

(a) HGM: 12 correct matches out of 15

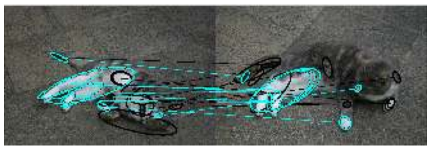

(b) TM: 10 correct matches out of 15

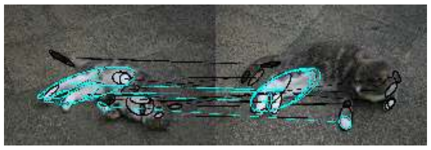

(c) RRWHM: 14 correct matches out of 15


(d) HGM: 10 correct matches out of 15


(e) TM: 8 correct matches out of 15


(f) RRWHM: 12 correct matches out of 15


(g) HGM: 8 correct matches out of 11


(h) TM: 5 correct matches out of 11

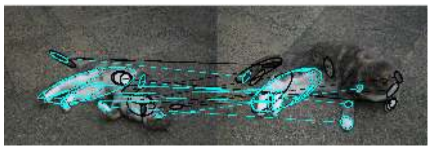

(i) RRWHM: 9 correct matches out of 11

Figure 9. Experiments on images from *MSRC v2* and *Caltech 101* dataset. Correct matches for the objects are denoted as cyan lines. These figures are best viewed in color.

[12] M. Leordeanu, M. Hebert, and R. Sukthankar. An integer projected fixed point method for graph matching and map inference. *NIPS*, 2009. 1633, 1634

[13] D. Lowe. Object recognition from local scale-invariant features. 1999. 1633, 1638

[14] J. Matas, O. Chum, M. Urban, and T. Pajdla. Robust wide baseline stereo from maximally stable extremal regions. 2002. 1633, 1638

[15] J. Munkres. Algorithms for the assignment and transportation problems. *SIAM*, 1957. 1637

[16] P. A. Regalia and E. Kofidis. The higher-order power method revisitiedl: Convergence proofs and effenctive initialization. *ICASSP*, 2000. 1635

[17] E. Seneta. *Non-negative Matrices and Markov Chains*. Springer, 1981. 1635

[18] R. Sinkhorn. *A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices*. Annals of Mathematical Statistics, 1964. 1637

[19] B. J. van Wyk and M. A. van Wyk. A pocs-based graph matching algorithm. *PAMI*, 2004. 1633, 1634

[20] R. Zass and A. Shashua. Probabilistic graph and hypergraph matching. *CVPR*, 2008. 1633, 1634, 1637, 1639

[21] D. Zhou, J. Huang, and B. Schlkopf. Learning with hypergraphs: Clustering, classification, and embedding. In *NIPS*, 2006. 1635