Machine Vision and Applications (2014) 25:823–844
DOI 10.1007/s00138-014-0598-1

ORIGINAL PAPER

Attributed hypergraph matching on a Riemannian manifold

J. M. Wang · S. W. Chen · C. S. Fuh

Received: 19 September 2012 / Revised: 19 July 2013 / Accepted: 23 January 2014 / Published online: 20 March 2014

Abstract If we consider a matching that preserves high-order relationships among points in the same set, we can introduce a hypergraph-matching technique to search for correspondence according to high-order feature values. While graph matching has been widely studied, there is limited research available regarding hypergraph matching. In this paper, we formulate hypergraph matching in terms of tensors. Then, we reduce the hypergraph matching to a bipartite matching problem that can be solved in polynomial time. We then extend this hypergraph matching to attributed hypergraph matching using a combination of different attributes with different orders. We perform analyses that demonstrate that this method is robust when handling noisy or missing data and can achieve inexact graph matching. To the best of our knowledge, while attributed graph-matching and hypergraph-matching have been heavily researched, methods for attributed hypergraph matching have not been proposed before.

Keywords Graph matching · Hilbert space · Riemannian manifold · Inexact graph matching · Attributed hypergraph matching

J. M. Wang · C. S. Fuh
Department of Computer Science and Information Engineering, National Taiwan University, No. 1, Sec. 4, Roosevelt Rd., Taipei 10617, Taiwan, ROC

S. W. Chen (✉)
Department of Computer Science and Information Engineering, National Taiwan Normal University, No. 88, Sec. 4, Tingzhou Rd., Wenshan District, Taipei 116, Taiwan, ROC
e-mail: swchen2012@gmail.com

1 Introduction

Matching two sets of feature points is a key element in many computer vision tasks, such as feature tracking, image alignment and stitching, and stereo fusion. This problem is closely related to weighted bipartite graphs. One may consider the two sets of feature points as two sets of nodes, and associate a weight value (called similarity) to an edge between two nodes belonging to different sets. Finding a maximum weighted bipartite matching will give a matching with the maximal sum of the edge values. This problem has been reduced to the maximal flow problem [1] and can be solved in polynomial time. Many additional algorithms have been proposed to solve these problems [2], such as the Bellman–Ford algorithm, Dijkstra's algorithm, and the Fibonacci heap.

Bipartite matching is perhaps the simplest matching problem, because it only concerns information of the node (feature point) itself. Graph-matching techniques are developed with the intent to match the feature points in different sets and to preserve binary relationships between points in the same set. Here, we regard the feature points as nodes and the relationships of point pairs as edges, and then construct a graph for a set of feature points. Graph matching is performed to establish the mapping between the nodes of two graphs while preserving their edge, i.e., if an edge is mapped to another edge in the other graph, those nodes will be linked by the corresponding edges.

Graph-matching methods can be roughly divided into exact graph matching and inexact graph matching [3,4]. In exact graph matching, two graphs may have some identical parts. The challenge is then to find an isomorphism such that we can map a node and an edge from one graph to another in terms of their associated values. According to the size of the identical part, we may have a graph isomorphism, a subgraphisomorphism, or a maximum common subgraph

isomorphism. The graph isomorphism problem is an unsolved problem in relation to whether it is NP or NP-complete, and we believe that it cannot be solved in polynomial time. Tree-based search algorithms [5] are the standard processing methods for isomorphism testing, and most of them require exponential time in the worst case. Some constrained graphs, such as trees [6], permutation graphs, and planar graphs [7], have been invoked to develop polynomial time algorithms [8] for the graph isomorphism problem.

Unlike graph isomorphism, subgraph isomorphism has been reduced to maximum common subgraph isomorphism, which has been proved to be an NP-complete problem. Both are believed to be impossible to solve in polynomial time. In practice, the maximum common subgraph isomorphism problem is more common as the feature points extracted from an image may be different from those points extracted from another image. Tree-based search algorithms [9] can be applied to solve the subgraph isomorphism problem in exponential time. To solve this problem in polynomial time, some approximation or suboptimal algorithms [10] have been proposed for constrained graphs (e.g., sparse graphs), which have then been solved iteratively in bounded polynomial time.

In computer vision, however, feature values often have spatial or temporal deviations, which cause the matching to be inexact matching rather than an exact graph matching. Inexact matching algorithms are designed with a tolerance for errors and noise, therefore they can be applied to determine the identical parts of two graphs in a more practical application than is possible with exact graph matching. It is for this reason that inexact graph matching is also called error-tolerant graph matching [11]. It can be regarded as the maximum common graph matching in some form and has been proved to be an NP-complete problem [12].

In recent years, various algorithms designed for inexact graph matching have been proposed. For example, tree-based search algorithms [13] are guaranteed to find the optimal solution in exponential time. To allow a solution to be found in polynomial time, we may choose some constraints for the graphs or accept a suboptimal solution. However, various applications have led to differing algorithms. Artificial neural networks [14], relaxation labeling [15], genetic search [16], the spectral method [17], and the kernel method [18] are key classes of inexact graph matching algorithms. One may refer to the papers [3] and [4] to find more information about inexact graph matching algorithms.

Traditional graph matching only considers about nodes and edges, which are related to unary features and binary relationships, respectively. In computer vision, the unary feature could be color or position value related to the feature point itself, while a binary relationship could be distance or orientation value computed from one pair of feature points. Feature points are extracted from one image frame and matched to those extracted in the successive frame. The feature values may or may not be deviated spatially or temporally. For example, the position feature is illumination-invariant, but the color feature is not. Moreover, the orientation feature is invariant to scale but the distance feature is not. If we can develop an algorithm based on some kind of invariant feature, our proposed algorithm will be robust to that kind of deformation.

Since we can design features with many kinds of invariance from unary and binary relationships, some applications still require high-order relationships [19,20]. Relating the search for correspondence using high-order relationships to hypergraph matching problem [21] is very straightforward. Searching for correspondence using high-order relationships is equivalent to hypergraph matching [21] or high-order graph matching [22]. While an adjacency matrix represents a graph, a tensor can be used to represent hypergraph. Extending the formulation from the adjacency matrix to a tensor-based formulation is very straightforward. However, solving the tensor formulation of hypergraph matching is not so simple [22].

Except for increasing the order of relationships, we can extend graph matching to attributed graph matching [23,24], where each node and edge is associated with a set of attribute values in an attributed graph. When exploring the correspondence not only the structure, but also the attribute value should be considered. This approach is very useful in computer vision applications where we want to match feature points in different images [25]. However, considering different kinds of attributes at the same time encourages inconsistency due to inconsistent feature types [26].

In this study, we consider attributed hypergraph matching, where feature points are associated with many kinds of feature values, both high order and low order. The hypergraph is represented using several tensors in different orders, and the attributed hypergraph matching is formulated in terms of least squares of tensors. The correspondence is finally shown in a permutation matrix. In Sect. 3, bipartite matching is addressed and formulated. Traditional graph matching is then discussed in Sect. 4. The extension from graph matching to hypergraph matching and attributed hypergraph matching are discussed in Sects. 5 and 6, respectively. Finally, the robustness of the proposed method is investigated in the final section of this paper.

The main contributions are summarized as follows. First, to the best of our knowledge, although some attributed graph-matching and hypergraph-matching algorithms have been proposed before, no algorithm focusing on attributed hypergraph matching, as our method does, has been proposed previously. Second, the hypergraph in general case is considered, unlike most of the previous works concerned about special graphs to deal with the inexact graph matching because it is believed to be NP-hard problem. Third, attribute values

are transformed and represented in the same Hilbert space to prevent the issues relating to inconsistency of feature type. Fourth, the hypergraph matching was reduced to bipartite graph matching in polynomial time, therefore it required much less processing time than other methods. Finally, under some assumptions, we can obtain optimal solution of hypergraph matching.

2 Related works

Let $V = \{V_1, \ldots, V_n\}$ and $V' = \{V'_1, \ldots, V'_{n'}\}$ be two sets of points, where n and n' are the number of points in V and V', respectively. We may solve their correspondence by looking for an $n \times n'$ assignment matrix Z such that its (i, j)th element $Z_{i,j} = 1$ when V_i corresponds to V'_j and it satisfies $Z\underline{\mathbf{1}} \leq \underline{\mathbf{1}}$ and $Z^T\underline{\mathbf{1}} \leq \underline{\mathbf{1}}$, where $\underline{\mathbf{1}}$ is a column vector in the form $\underline{\mathbf{1}} = [1, 1, \ldots, 1]^T$. Such a problem is also called an assignment problem and it can be solved by bipartite matching in polynomial time. If the correspondences are established by considering their binary relationships, $E_{i,j} = (V_i, V_j)$ and $E'_{i,j} = (V'_i, V'_j)$, simultaneously, this problem is known as a quadratic assignment problem and has no known polynomial-time algorithm for solving it. As those nodes and edges form a graph $G = (V, E = \{E_{i,j}\})$ in mathematics, we may say that the quadratic assignment problem is a graph-matching problem. One may refer to the book [27] to find more information about assignment problems.

Approximate methods for the quadratic assignment problem can be categorized as those expressed by an affinity matrix and those expressed by a weighted adjacency matrix. In the first category, the affinity matrix $A \in \mathbf{R}^{nn' \times nn'}$, whose element $A_{i,j}$ is the pairwise affinity of a potential assignment $i = (V_i, V'_{i'})$ to $j = (V_j, V'_{j'})$, is used for representation. Solving the problem yields the following binary optimization problem:

$$\mathbf{z}^* = \arg\max_{z}(\mathbf{z}^T A\mathbf{z}), \quad \mathbf{Z}^* \in \{0, 1\}^{nn'},$$

where $\mathbf{z}$ is a row-wise vectorized replica of Z. This formulation was first shown in [28] and can be solved via spectral relaxation:

$$w^* = \arg\max_{w} \frac{w^T A w}{w^T w}.$$

Here, w^* can be solved by computing the leading eigenvalue and corresponding eigenvector of A. Given the continuous solution w^*, one may discretize w^* to derive z^*. By the Perron–Frobenius theorem, the leading eigenvector of A is known to exist if A is symmetric and non-negative. The power iteration method is a useful technique to compute the leading eigenvector iteratively.

To extend the above works to high-order relationship, the affinity tensor $T \in \mathbf{R}^{nn' \times nn' \times nn' \times \cdots}$, whose element is the affinity between potential assignments, is used for formulation of hypergraph matching. A hypergraph $G = (V, E)$ is a generalization of a graph where each edge, called hyperedge, can connect any number of nodes. According to the order of relationship, we can define the order of the affinity tensor and the number of nodes connected to one hyperedge. Each hyperedge can be associated with the value of relationship among the connected nodes. To solve the hypergraph matching, one may unfold T into a matrix and apply the aforementioned methods proposed in matrix space [19], but some relationships would be lost in such a reduction.

Power iteration has been extended to tensor power iterations [22,29]. However, as the order of the tensor increases, the number of entries of the tensor increases exponentially. If the tensor is not sparse, constructing the tensor and computing the leading eigenvector will be time-expensive operations. The leading eigenvector could be used to infer the assignment matrix as it provides the optimal rank one approximation for the matrix (as in the Eckart–Young Theorem) [19]. However, this property does not hold for tensors.

The affinity matrix T can be though as the weighted adjacency matrix of an association graph and Cho et al. [30] proposed a random walk algorithm to solve the matching problem in this association graph. This work is then generalized to higher order [31], i.e., the random walk algorithm is applied to an association hypergraph that is represented by the affinity tensor. Power iteration and random walk algorithms use the expression with bulky processing space and solve the problem iteratively, which cause their methods are time and space consuming. Besides, their methods may be trapped in local minimum.

In the second category, graph nodes represent the points in the set and graph edges denote the relationships between two points in the same set. The sets of points, V and V', and their corresponding set of relationships are then represented as two weighted graphs using two weighted adjacency matrices. We denote these two matrices as $A \in \mathbf{R}^{n \times n}$ and $A' \in \mathbf{R}^{n' \times n'}$, respectively. Here, the assignment matrix Z is used as the permutation matrix P, and the matching problem may then be expressed as a least square problem as follows:

$$P^* = \arg\min_{P} \left\| A - PA'P^T \right\|, \tag{1}$$

where $||\cdot||$ means some norm (the Frobenius norm in our research). Obviously, this expression uses less processing space than the previous one. Besides, computing affinity matrix requires much more time $O(n^2)$ than computing weighted adjacency matrix $O(n)$. As the order of relationship is increased, computing affinity tensor will require unacceptable processing time $O(n^{2d})$.

Spectral methods are another technique for solving Eq. (1). These can be applied because of the observation that the eigenvectors relating to the adjacency of a graph are invariant with respect to node permutation [4]. The graduated assign-

ment algorithm [32] may be one of the first algorithms to solve the quadratic assignment problem by relaxing P^* to a leading eigenvector of A. This algorithm combines the quadratic assignment problem and sparsity, and has proved to be a very successful algorithm [19].

While a weighted adjacency matrix defined for graph, we can define a tensor, whose elements are the values of hyperedges, for hypergraph. Matrix decompositions, such as singular value decomposition [33] or eigen decomposition [34], are widely used to solve quadratic assignment problems in matrix space. The corresponding decompositions for a tensor could be CANDECOMP/PARAFAC (CP) decompositions or Tucker decompositions [35]. The CP decomposition decomposes a tensor as a sum of rank-one tensors, while the Tucker decomposition is a principal component analysis for tensors. However, as discussed by Kolda and Bader [35], there is no straightforward algorithm to determine the rank of a specific given tensor, or the tensor decomposition.

Our method has the following advantages: At first, our method took a significantly short processing time (less than 1 s) to perform a hypergraph matching while obtaining reasonable results. In the second, this speed allows us to employ a higher number of attributes and higher orders of relationships for hypergraph matching. The higher number of attributes helps our method to adapt to noise, while the higher orders of relationships allow our method to be invariant to some image distortion (i.e., scale, rotation, or translation). Finally, even graph matching is supposed to be NP-hard problem, we can obtain the optimal solution in polynomial time under some assumptions.

3 Bipartite matching

To determine the correspondence between the feature points of V and those of V', we look for an $n \times n'$ permutation matrix P such that its (i, j)th element $P_{i,j} = 1$ when V_i corresponds to V'_j. In general, n is not equal to n'. However, dummy feature points can be added to the smaller set to give the same size n for both sets V and. This matching problem can be formulated as the minimization of the following equation [36]:

$$P^* = \arg\min_P \left\| U - PU' \right\| \tag{2}$$

where $\boldsymbol{U}$ and $\boldsymbol{U}'$ are two $n \times 1$ vectors constructed from V and V', respectively, and in each vector, the ith element U_i contains the feature value of V_i. Each element of P can be either 1 or 0, and only one element is 1 in each row and each column. The permutation matrix P with element $P_{i,j} = 1$ will permute U'_j to subtract U_i. Since $\boldsymbol{U}$ and $\boldsymbol{U}'$ associate the nodes of the graph without edges, Eq. (2) can express the solution of the graph matching.

To satisfy Eq. (2), the element U'_j should be as close as possible to the element U_i when P_{ij} is set as 1. In a realistic situation, noise exists in the feature values. Therefore, U_i and U'_j will not be exactly equal even if they represent the same feature point. To address the issue of noise, a similarity matrix S can be introduced for solving the problem. Each element, $S_{i,j}$, of S is computed as the degree of the similarity between U_i and U'_j. The permutation matrix P can be solved by the following maximization:

$$P^* = \arg\min_P \sum_{i,j} S_{i,j} P_{i,j}.$$

The above problem is a well-known maximum weight bipartite matching problem, and can be solved in polynomial time [1]. Many methods, such as the relaxation method, the Hopfield network method [17], the Hungarian algorithm [37], and the Dijkstra's algorithm, have been proposed for solving this problem.

In this paper, $S_{i,j}$, is defined as $S_{i,j} = k(d(i, j))$, where $k(\cdot)$ is a decreasing function (e.g., $k(x) = 1/x$) and $d(i, j)$ is the degree of dissimilarity:

$$d(i, j) = \left\| U_i - U'_j \right\|. \tag{3}$$

The vector norm $||\cdot||$ is applied here instead of the absolute value, because we may assign a feature vector to a node in many applications. Finally, P is the solution of the bipartite matching with a maximum sum of weights, where V and V' are the two sets of nodes, and $S_{i,j}$ is the weighted value of the edge between nodes V_i and V'_j. Note that we will reduce hypergraph matching to bipartite matching in polynomial time. Since the solution of maximum weight bipartite matching can be solved in polynomial time, we will be able to solve hypergraph matching in polynomial time too.

4 Graph matching

To decide the correspondence between different sets of feature points and to consider the binary relationships of point pairs at the same time, we construct a graph for each set of feature points and apply the method of graph matching to solve the correspondence problem. All the feature points in one set, $V = \{V_1, \ldots, V_n\}$, are considered as the nodes in the graph G, and the binary relationship between V_i and V_j can be thought of the edge $E_{i,j}$ between nodes. We assign a feature value $A_{i,j}$ to edge $E_{i,j}$ and a feature value A_{ii} to node V_i, and then construct an weighted adjacency matrix $A = [A_{i,j}]$. Feature points in V' can be a graph represented by another weighted adjacency matrix A'. In our application, A and A' are not necessarily symmetric.

Solving the correspondence between the feature points in different images can be formulated as the minimization problem [24] defined in Eq. (1). By assigning a feature value to an

edge, the graph becomes an attributed graph. Since A and A' represent the attributed graphs, Eq. (1) formulates the problem of an attributed graph matching [38]. In previous works, many algorithms have been proposed to solve the attributed graph matching [3,4], but they encounter difficulties, such as an exponential processing time [9], only applying in special cases [10], or falling into local minimum traps.

Let us consider a permutation matrix P. Suppose $A^* = PA'P^T$, and the permutation matrix P, with elements $P_{i,j} = 1$, will permute $A'_{j,:}$ and $A'_{:,j}$ to $A^*_{i,:}$ and $A^*_{:,i}$, respectively. If Eq. (1) is satisfied, $A_{i,:}$ and $A_{:,i}$ should be very similar to $A^*_{i,:}$ and $A^*_{:,i}$, respectively. Since the elements in $A^*_{i,:}$ and $A^*_{:,i}$, come from $A'_{j,:}$ and $A'_{:,j}$, we can expect the elements in $A'_{j,:}$ and $A'_{:,j}$ to be close to those in $A_{i,:}$ and $A_{:,i}$ except for sequence.

For convenience, let $A(i)$ denote the bag containing the elements of $A_{i,:}$ and $A_{:,i}$, and $A'(j)$ denote the bag containing the elements of $A'_{i,:}$ and $A'_{:,i}$. If $A(i)$ and $A'(j)$ are equal, we will set $P_{i,j}$ to 1 in most cases. In a realistic situation, however, noises exist in the feature values and, thus, $A(i)$ and $A'(j)$ cannot be exactly equal even if they represent the same feature point. We use the similarity matrix S introduced above with $S_{i,j}$ computed as $S_{i,j} = k(d(i,j))$, where the distance function $d(i,j)$ provides the degree of dissimilarity between $A(i)$ and $A'(j)$. When we have S, the permutation matrix P can be solved using the methods discussed in the previous section.

Instead of measuring dissimilarity by comparing the elements in the bag directly, we compare their continuous distribution described below to tolerate the noise. Many harmonic analysis methods, such as the Hilbert transform or the Fourier transform, can be applied for describing the continuous distribution. In this study, we approximate of the elements in a bag with the distribution by a Fourier series $f'(x)$:

$$f'(x) = \sum_{n=-\infty}^{\infty} c_n \mathrm{e}^{in\mu x}, \tag{4}$$

with:

$$c_n = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(x)\mathrm{e}^{-in\mu x}\mathrm{d}x, \tag{5}$$

where $f(x)$ is a function that is periodic on the interval $[-L, L]$ and $\mu = \frac{\pi}{L}$. In our application, we normalize the values of the elements in a bag [$(A(i)$ or $A'(j)$ in our case] to be between $-\pi$ and π, and set:

$$f(x) = \begin{cases} \frac{n_k}{|A(i)|} & \text{if } x = a_k \\ 0 & \text{otherwise} \end{cases}, \tag{6}$$

where a_k is an element of set $A(i)$, and n_k is the number of elements with value a_k. If we treat $f(x)$ as a function that is periodic on the interval $[-\pi, \pi]$, Eq. (5) can be simplified as:

$$c_n = \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x)\mathrm{e}^{-inx}\mathrm{d}x,$$

and its discrete form would be:

$$\begin{aligned} c_n &= \frac{1}{2\pi} \sum_{k=1}^{|A(i)|} \frac{1}{|A(i)|} \mathrm{e}^{-ina_k} = \frac{1}{2\pi} \frac{1}{|A(i)|} \sum_{k=1}^{|A(i)|} \mathrm{e}^{-ina_k} \\ &= C \sum_{k=1}^{|A(i)|} \mathrm{e}^{-ina_k}, \end{aligned} \tag{7}$$

where $C = \frac{1}{2\pi} \frac{1}{|A(i)|}$. The computation of Eq. (7) requires time complexity $O(|A(i)|)$.

The above function $f(x)$ is associated with a sequence of Fourier series, $\{F_N(x)\}$, where $N = 0, 1, 2, \ldots$, defined by:

$$F_N(x) = \sum_{n=-N}^{N} c_n \mathrm{e}^{inx}.$$

For any small positive real number, we can always choose N such that:

$$\|f(x) - F_{N'}(x)\| < \varepsilon$$

for $N' > N$. Finally, any square-integrable function can be expressed as a series:

$$f(x) = \sum_{n=-\infty}^{\infty} c_n \mathrm{e}^{inx}. \tag{8}$$

Thus, the sequence of Fourier series converges. The functions e^{inx} form the orthogonal basis of a space, satisfying the conditions for a Hilbert space. Under these basis functions, we can represent the function $f(x)$ by a vector $(\ldots, c_{-1}, c_0, c_1, \ldots)$ or by components (c_n) in a Fourier coefficient space.

The Fourier series approximation, $F_N(x) = f'(x) = \sum_{n=-N}^{N} c_n \mathrm{e}^{inx}$, is then used instead of Eq. (4). We defined a Hilbert space H, the $f'(x)$ now can be represented as a vector $\boldsymbol{v} = (c_{-N}, \ldots, c_0, \ldots, c_N)$ in H. Figure 1 shows a diagram of the above transformation. With such an approximation, the distribution of the elements can accommodate data noise and even a loss of data. The total time complexity required for computing the Fourier series approximation is $O(N|A(i)|)$. Since N is a small constant number (8 in our experiments), the time complexity is assumed to be $O(|A(i)|)$.

Let us use $f_i(x)$, $f'_i(x)$, and $\boldsymbol{v}_i$ to denote the discrete function, the Fourier series approximation, and the vector, respectively, transformed from $A(i)$. Note that $f_j(x)$, $f'_j(x)$, and $\boldsymbol{v}_j$ are similarly transformed from $A'(j)$. The degree of dissimilarity between $A(i)$ and $A'(j)$ corresponds to the difference between $f_i(x)$ and $f_j(x)$, which can be computed by the

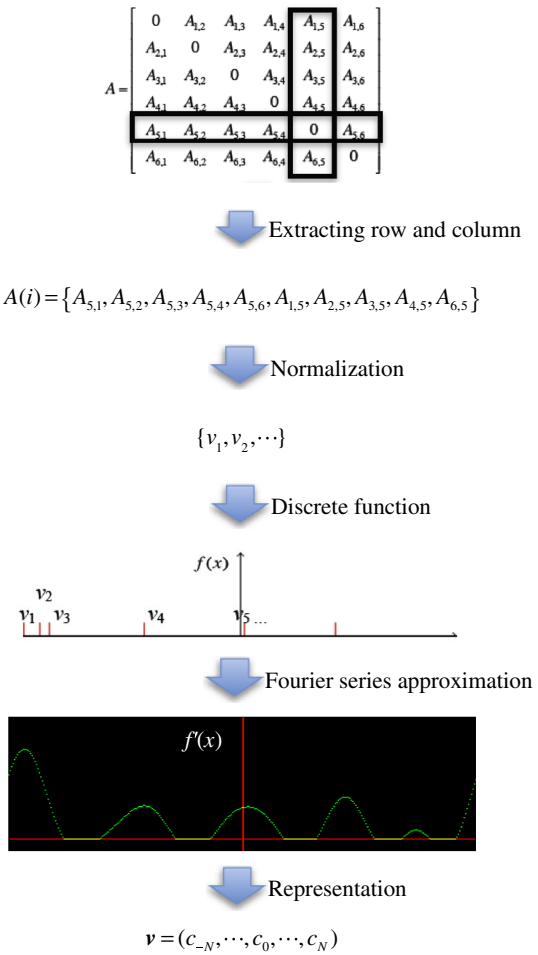


Fig. 1 Transformation from $A(i)$ to $f'(x)$, where $A(i)$ is a bag with the elements in the ith row and column of matrix A, $f(x)$ is the discrete distribution of $A(i)$, and $f'(x)$ is the Fourier series approximation for $f(x)$. Note that the elements A_{ii} can be zero if we do not consider the feature values of nodes

difference between $f_i'(x)$ and $f_j'(x)$ to tolerate the noise. Finally, we compute the distance between $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ to reflect the difference between $f_i'(x)$ and $f_j'(x)$ for convenience.

To compute the true distance between $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$, we introduce a Riemannian manifold M embedded in H. The metric $g_{\boldsymbol{v}}$ on M is a family of inner products:

$$g_{\boldsymbol{v}} : T_{\boldsymbol{v}}M \times T_{\boldsymbol{v}}M \to R, \boldsymbol{v} \in M,$$

where $T_{\boldsymbol{v}}M$ refers to the tangent space consisting of all the tangent vectors at one point $\boldsymbol{v}$ on M. Regarding the space H, the components of the metric tensor G at point $\boldsymbol{v}$ are given by:

$$g_{mn}(v) = g_{\boldsymbol{v}}\left(\left(\frac{\partial M}{\partial e^{imx}}\right)_{\boldsymbol{v}}, \left(\frac{\partial M}{\partial e^{inx}}\right)_{\boldsymbol{v}}\right).$$

With the metric tensor G, the inner product between two tangent vectors $\boldsymbol{v}_1 \in T_{\boldsymbol{v}}M$ and $\boldsymbol{v}_2 \in T_{\boldsymbol{v}}M$ can be computed by $g_{\boldsymbol{v}}(\boldsymbol{v}_1, \boldsymbol{v}_2) = \boldsymbol{v}_1^T G \boldsymbol{v}_2$. Let us consider another point $\boldsymbol{v}' = \boldsymbol{v} + d\boldsymbol{v}$, where $d\boldsymbol{v} = (0, \ldots, 0, dc_m, 0, \ldots, 0)$. The distance, $||d\boldsymbol{v}||$, between $\boldsymbol{v}'$ and $\boldsymbol{v}$ defined on the manifold can be computed by:

$$\|d\boldsymbol{v}\|^2 = d\boldsymbol{v}^2 = d\boldsymbol{v}^T G d\boldsymbol{v} = g_{mm}(\boldsymbol{v}) \cdot dc_m \cdot dc_m. \tag{9}$$

Suppose $f_{v'}'(x)$ and $f_{\boldsymbol{v}}'(x)$ are the Fourier series approximations defined by $\boldsymbol{v}'$ and $\boldsymbol{v}$. We calculate the difference, $|\mathrm{d}\boldsymbol{v}|$, between $f_{v'}'(x)$ and $f_{\boldsymbol{v}}'(x)$ using the following equation:

$$\begin{aligned}\|\mathrm{d}\boldsymbol{v}\| &= \int \left(f_{v'}'(x) - f_{\boldsymbol{v}}'(x)\right)\mathrm{d}x = \int dc_m \mathrm{e}^{imx}\mathrm{d}x \\ &= dc_m \int \mathrm{e}^{imx}\mathrm{d}x.\end{aligned} \tag{10}$$

By combining Eqs. (9) and (10), $g_{mm}(\boldsymbol{v})$ equals to $\left(\int \mathrm{e}^{imx}\mathrm{d}x\right)^2$. Let $\left(\int \mathrm{e}^{imx}\mathrm{d}x\right)^2 = (C_m)^2$, where C_m can be obtained from the inverse Fourier transform on the frequency $\frac{m}{2\pi}$.

Next, let us replace $d\boldsymbol{v}$ with $(0, \ldots, 0, dc_m, 0, \ldots, 0, d_n, 0, \ldots, 0)$. Equation (9) can be rewritten as:

$$\begin{aligned}\|\mathrm{d}\boldsymbol{v}\|^2 &= \mathrm{d}\boldsymbol{v}^T G d\boldsymbol{v} = g_{mm}(\boldsymbol{v}) \cdot \mathrm{d}c_m \cdot \mathrm{d}c_m \\ &\quad + 2g_{mn}(\boldsymbol{v}) \cdot \mathrm{d}c_m \cdot \mathrm{d}c_n + g_{nn}(\boldsymbol{v}) \cdot \mathrm{d}c_n \cdot \mathrm{d}c_n \\ &= C_m^2 (\mathrm{d}c_m)^2 + 2g_{mn}(\boldsymbol{v}) \cdot \mathrm{d}c_m \cdot \mathrm{d}c_n + C_n^2 (\mathrm{d}c_n)^2,\end{aligned} \tag{11}$$

and Eq. (10) will become:

$$\begin{aligned}\|\mathrm{d}v\| &= \int \left(f_{v'}'(x) - f_{\boldsymbol{v}}'(x)\right)\mathrm{d}x \\ &= \mathrm{d}c_m \int \mathrm{e}^{imx}\mathrm{d}x + \mathrm{d}c_n \int \mathrm{e}^{inx}\mathrm{d}x\end{aligned}$$

and

$$\begin{aligned}\|\mathrm{d}v\|^2 &= \left(\mathrm{d}c_m \int \mathrm{e}^{imx}\mathrm{d}x\right)^2 \\ &\quad + 2\mathrm{d}c_m\mathrm{d}c_n \int \mathrm{e}^{imx}\mathrm{d}x \int \mathrm{e}^{inx}\mathrm{d}x \\ &\quad + \left(\mathrm{d}c_n \int \mathrm{e}^{inx}\mathrm{d}x\right)^2 \\ &= (\mathrm{d}c_m)^2 C_m^2 + 2\mathrm{d}c_m\mathrm{d}c_n C_m C_n + (\mathrm{d}c_n)^2 C_n^2.\end{aligned} \tag{12}$$

By combining Eqs. (11) and (12), we can obtain $g_{mn}(\boldsymbol{v}) = C_m C_n$. Finally, we have the metric tensor G that can be computed beforehand.

Let $r : [a, b]$ be a continuously differentiable curve in M. We define $\mathrm{dis}(r(a), r(b))$ as the length of the shortest path (i.e., a geodesic) r between a and b:

$$\begin{aligned}\mathrm{dis}(r(a), r(b)) &= \int_a^b \left\| r'(s) \right\| \mathrm{d}s = \int_a^b \sqrt{r'(s)^T G r'(s)} \mathrm{d}s \\ &= \int_a^b \sqrt{r'(s)^T \boldsymbol{C}\boldsymbol{C}^T r'(s)} \mathrm{d}s = \int_a^b r'(s)^T \boldsymbol{C} \mathrm{d}s \\ &= \int_a^b r'(s)^T \mathrm{d}s \cdot \boldsymbol{C} = \Delta r^T \boldsymbol{C} = \sqrt{\Delta r^T G \Delta r}\end{aligned} \tag{13}$$

where $\boldsymbol{C} = (C_{-N}, \ldots, C_0, \ldots, C_N)$ and $\Delta r = r(b) - r(a)$. The elements of the similarity matrix S are defined by $S_{i,j} = k(d(i, j))$, where $d(i, j)$ is given by Eq. (13), $d(i, j) = \mathrm{dis}(\boldsymbol{v}_i, \boldsymbol{v}_j)$. Constructing S will require time complexity $O(n^3)$ where $n = |A(i)|$. Using the method relating to Eq. (2) proposed in the previous section, we can obtain P^* from S.

The graph with fewer nodes is padded with dummy nodes to give an equal number of nodes in each graph. In the matrices, the feature values of the dummy nodes and their corresponding edges are set to zero. To handle the zero values in the matrix, the corresponding values of $f(x)$ are set to be a very small number. After graph matching, each node will match one of the nodes in the other graph. Dummy nodes will match either a dummy node or a redundant node in the other graph.

Based on the definition, $A(i)$ contains the weights of the connected edges of node i. Its corresponding expressions $\boldsymbol{v}_i$ can be thought of as the feature vectors of node i. In other words, we transform the edge values $A(i)$ of the binary relationship to the feature vector, $\boldsymbol{v}_i$, of node i. The same situation occurs in the transformation from $A'(j)$ to $\boldsymbol{v}_j$. Solving Eq. (1) will be equivalent to solving Eq. (2) if we set $U_i = \boldsymbol{v}_i$ and replace Eq. (3) with Eq. (13). We transform the graph matching to a bipartite matching. Figure 2 shows the overview of the transformation. In the next section, the same idea provides a method for transforming a matching that considers *d-ary* relationships to a bipartite matching.

5 Hypergraph matching

As a weighted adjacency matrix can be used to represent an attributed graph, an attributed tensor can be used to represent an attributed hypergraph with *d-ary* affinities, each of which is a relationship of d nodes [22]. Let us take a matching of ternary relationships for example. As the matrix is applied to represent the binary relationship of the nodes, a third-order tensor T is applied to represent the ternary relationship of the nodes. A third-order tensor T can be regarded as a three-dimensional array, and its element $T_{i,j,k}$ contains the weight among the ith, jth, and kth nodes. The element $T_{i,j,k}$ will store the weight value of an edge if two of the indexes (i, j, or k) are equal, while $T_{i,i,i}$ will store the feature value of single node i as the indexes are the same.

With respect to the Cartesian basis $\{\mathbf{e}_i = (\ldots, 0, 1, 0, \ldots)^{\mathrm{T}}\}$, where all the elements are 0 except that the ith element is 1, a third-order tensor can be represented by a linear combination of the Cartesian components:

$$T = \sum_{i,j,k} T_{i,j,k} \mathbf{e}_i \otimes \mathbf{e}_j \otimes \mathbf{e}_k, \tag{14}$$

where $\mathbf{e}_i \otimes \mathbf{e}_j \otimes \mathbf{e}_k$ is the dyadic product of three vectors $\mathbf{e}_i$, $\mathbf{e}_j$, and $\mathbf{e}_k$. The dimension of $\{\mathbf{e}_i\}$ is the same as the number of nodes in our case, and the dyadic product of three vectors is extended from the dyadic product of two vectors. As the dyadic product of two vectors is shown as a matrix, $\mathbf{e}_i \otimes \mathbf{e}_j \otimes \mathbf{e}_k$ is a three-dimensional array with 0 in all elements except for the (i, j, k) element (equal to 1). In the following expression,

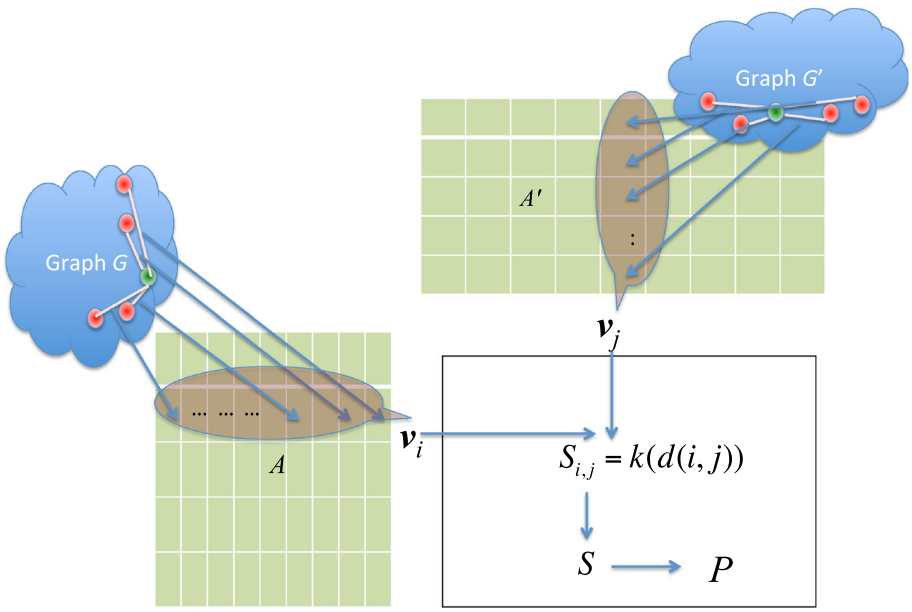


Fig. 2 Overview of the transformation from graph matching to bipartite matching

the summation sign of the triple will be suppressed (using the Einstein summation convention), and a tensor expressed by Eq. (14) will be denoted by

$$T = T_{i,j,k}\mathbf{e}_i \otimes \mathbf{e}_j \otimes \mathbf{e}_k.$$

Solving the correspondence between the feature points in different images can be formulated as the following minimization problem with the above triple:

$$P^* = \arg\min_P \left\| T - T'_{i,j,k}(P\mathbf{e}_i) \otimes (P\mathbf{e}_j) \otimes (P\mathbf{e}_k) \right\|, \quad (15)$$

where P is the permutation matrix and $P\mathbf{e}_i$ will equal $\mathbf{e}_j$ if $P_{i,j} = 1$. The permutation matrix can permute the slices $T'_{j,:,:}$, $T'_{:,j,:}$, and $T'_{:,:,j}$ to subtract $T_{i,:,:}$, $T_{:,i:}$, and $T_{:,:,i}$, respectively, in Eq. (15). A slice $T'_{j,:,:}$ can be considered as a matrix, which can have the inner product applied with the permutation matrix, $PT'_{j,:,:}P^T$, as it is expressed in graph matching. To minimize Eq. (15), the permuted slice $PT'_{j,:,:}P^T$, $PT'_{:,j,:}P^T$, $PT'_{:,:,j}P^T$ must similar to $T_{i,:,:}$, $T_{:,i,:}$, $T_{:,:,i}$, respectively. As discussed in the previous section, since T and T' are associated with hypergraphs, Eq. (15) expresses the solution for a hypergraph matching.

Let us consider $T(i)$ as a bag of elements in $T_{i,:,:}$, $T_{:,i:}$, and $T_{:,:,j}$, and consider $T'(j)$ as a bag of elements in $T'_{j,:,:}$, $T'_{:,j,:}$, and $T'_{:,:,j}$. The higher the similarity between $T(i)$ and $T'(j)$, the higher the probability that $P_{i,j}$ will be to 1. Using the process described earlier, $T(i)$ and $T'(j)$ can be transformed and expressed as the feature vectors $\boldsymbol{v}_i$ and $\boldsymbol{v}_j$ using Eq. (5). The time complexity for a transformation is $O(n^2)$. After setting $r(a) = \boldsymbol{v}_i$ and $r(b) = \boldsymbol{v}_j$, we can measure the dissimilarity using Eq. (13) and construct S for solving the permutation matrix in Eq. (2). Extending the tensor order from three to d is straightforward. The difference is the time complexity, which is $O(n^{d-1})$ for computing the vector and $O(n^{d+1})$ for computing P^*.

5.1 Proof

Using the above transformation, a hypergraph matching can be reduced to a bipartite matching in polynomial time. As the latter can be solved in polynomial time, so can the proposed hypergraph matching. Since graph matching is generally an NP-hard problem, the solution should be an approximation. However, we have a proof to show that the processing result is the global minimum of Eq. (15).

It is known that bipartite matching can be solved in polynomial time, which means that we can have global minimum of Eq. (1) using those methods mentioned before. Let vector $\boldsymbol{U}$ consist of elements $\boldsymbol{v}_i$. Then, Eq. (1) can be expanded to the following:

$$P^* = \arg\min_P \|\boldsymbol{U} - P\boldsymbol{U}'\| = \arg\min_P \sqrt{\sum_j \|\boldsymbol{v}_j - \boldsymbol{v}_{\pi(j)}\|^2}, \quad (16)$$

where $\pi(j)$ refers to the permutation of element j. As in Eq. (8), $\boldsymbol{v}_j$ and $\boldsymbol{v}_{\pi(j)}$ refer to the Fourier series $f'_j(x)$ and $f'_{\pi(j)}(x)$, respectively. Minimizing the norm, $\|\boldsymbol{v}_j - \boldsymbol{v}_{\pi(j)}\|$ is equivalent to minimizing the expected value $E\left[f'_j(x) - f'_{\pi(j)}(x)\right]$. Thus, by substituting this for the norm, Eq. (16) can be expanded as:

$$\begin{aligned}
&\arg\min_P \sqrt{\sum_j \|\boldsymbol{v}_j - \boldsymbol{v}_{\pi(j)}\|^2} \\
&= \arg\min_P \sqrt{\sum_j E\left[f'_j(x) - f'_{\pi(j)}(x)\right]^2} \\
&= \arg\min_P \sqrt{\sum_j E\left[\sum_{n=-N}^{N}\left(C\sum_{k=1}^{|T(j)|} \mathrm{e}^{-ina_{j,k}}\right)\mathrm{e}^{inx} - \sum_{n=-N}^{N}\left(C\sum_{k=1}^{|T'(\pi(j))|} \mathrm{e}^{-ina'_{\pi(j),\pi(k)}}\right)\mathrm{e}^{inx}\right]^2},
\end{aligned} \quad (17)$$

where $a_{j,k}$ is the kth element in bag $T(j)$. Equation (17) can be further rearranged to:

$$
\begin{aligned}
&\arg\min_P \sqrt{\sum_j E\left[\sum_{n=-N}^{N}\left(C\sum_{k=1}^{|T(j)|}\mathrm{e}^{-ina_{j,k}}\right)\mathrm{e}^{inx}-\sum_{n=-N}^{N}\left(C\sum_{k=1}^{|T'(\pi(j))|}\mathrm{e}^{-ina'_{\pi(j),\pi(k)}}\right)\mathrm{e}^{inx}\right]^2}\\
&=\arg\min_P \sqrt{\sum_j E\left[\sum_{n=-N}^{N}\left(\sum_{k=1}^{|T(j)|}\left(\mathrm{e}^{-ina_{j,k}}-\mathrm{e}^{-ina'_{\pi(j),\pi(k)}}\right)\right)\mathrm{e}^{inx}\right]^2}\\
&=\arg\min_P \sqrt{\sum_j E\left[\sum_{k=1}^{|T(j)|}\left(\sum_{n=-N}^{N}\left(\mathrm{e}^{-ina_{j,k}}-\mathrm{e}^{-ina'_{\pi(j),\pi(k)}}\right)\mathrm{e}^{inx}\right)\right]^2}\\
&=\arg\min_P \sqrt{\sum_j E\left[\sum_k\left(\sum_{n=-N}^{N}\mathrm{e}^{-ina_{j,k}}\mathrm{e}^{inx}-\sum_{n=-N}^{N}\mathrm{e}^{-ina'_{\pi(j),\pi(k)}}\mathrm{e}^{inx}\right)\right]^2}
\end{aligned}
\tag{18}
$$

where we assume that $|T(j)| = |T'(\pi(j)|$. As defined in Eq. (7), $\sum_{n=-N}^{N}\mathrm{e}^{-ina_{j,k}}\mathrm{e}^{inx}$ is the Fourier series approximation of:

$$
f(x)=\begin{cases}1 & \text{if } x=a_{j,k}\\ 0 & \text{otherwise}\end{cases},
$$

with $\sum_{n=-N}^{N}\mathrm{e}^{-ina'_{\pi(j),\pi(k)}}\mathrm{e}^{inx}$ defined similarly. Both $\sum_{n=-N}^{N}\mathrm{e}^{-ina_{j,k}}\mathrm{e}^{inx}$ and $\sum_{n=-N}^{N}\mathrm{e}^{-ina'_{\pi(j),\pi(k)}}\mathrm{e}^{inx}$ are Gaussian functions with the same standard deviation. The subtraction of these two Gaussian functions can be performed as the subtraction of two random variables that are normally distributed with mean values $a_{j,k}$ and $a_{\pi(j),\pi(k)}$, respectively. Let us define two random variables, $X_{j,k}\equiv\sum_{n=-N}^{N}\mathrm{e}^{-ina_{j,k}}\mathrm{e}^{inx}$ and $X_{(j),\pi(k)}(x)\equiv\sum_{n=-N}^{N}\mathrm{e}^{-ina'_{\pi(j),\pi(k)}}\mathrm{e}^{inx}$. These can be substituted into Eq. (18), giving:

$$
\begin{aligned}
&\arg\min_P \sqrt{\sum_j E\left[\sum_k\left(\sum_{n=-N}^{N}\mathrm{e}^{-ina_{j,k}}\mathrm{e}^{inx}-\sum_{n=-N}^{N}\mathrm{e}^{-ina'_{\pi(j),\pi(k)}}\mathrm{e}^{inx}\right)\right]^2}\\
&=\arg\min_P \sqrt{\sum_j E\left[\sum_k\left(X_{j,k}-X_{\pi(j),\pi(k)}\right)\right]^2}\\
&=\arg\min_P \sqrt{\sum_j E\left[\sum_k DX_{j,k}\right]^2}
\end{aligned}
\tag{19}
$$

where $DX_{j,k}=X_{j,k}-X_{\pi(j),\pi(k)}$. We assume that each $DX_{j,k}$ is independent to one other. Then, Eq. (19) can be written as:

$$
\arg\min_P \sqrt{\sum_j E\left[\sum_k DX_{j,k}\right]^2}=\arg\min_P \sqrt{\sum_j\sum_k E\left[DX_{j,k}\right]^2}.
\tag{20}
$$

Since the expected value $E\left[DX_{j,k}\right]$ shows the difference between $a_{j,k}$ and $a_{\pi(j),\pi(k)}$, Eq. (20) becomes:

$$
\begin{aligned}
&\arg\min_P \sqrt{\sum_j\sum_k E\left[DX_{j,k}\right]^2}\\
&=\arg\min_P \left\|T-T'_{i,j,k}(P\mathbf{e}_i)\otimes(P\mathbf{e}_j)\otimes(P\mathbf{e}_k)\right\|.
\end{aligned}
\tag{21}
$$

Finally, we have $P^*=\arg\min_P\left\|T-T'_{i,j,k}(P\mathbf{e}_i)\otimes(P\mathbf{e}_j)\otimes(P\mathbf{e}_k)\right\|$, proving that our processing result satisfies Eq. (15).

5.2 Error propagation and discernibility

Most of the previous research has focused on the matching of an attributed graph constructed using node values (unary features) and edge values (binary relationships). In hypergraph matching, we try to solve the matching problem involving high-order relationships [26]. Such matching has the advantages of increasing the geometric invariance and more models can explain [22]. However, sometimes high-order feature values have greater deviation than expected. When we construct a d-order tensor for hypergraph matching, the absence of one pair influences the $d\times(d-1)\times n^{d-2}$ feature values of a bag. To prove this, let us consider a d-order tensor $T_{i_1,i_2,\ldots,i_d}$ and a specific node i. The bag for computing feature vector $\boldsymbol{v}_i$ contains d elements in $T_{:,\ldots,:,i,:,\ldots,:}$. For each $T_{:,\ldots,:,i,:,\ldots,:}$, one of the remaining $(d-1)$ indexes indicates the corresponding node whose feature vector will be influenced. This leaves $d-2$ indexes to set as one of the n index values (n^{d-2} cases).

With respect to the bag size n^{d-1}, the ratio of the changed elements to the non-changed elements in a bag is:

$$
\frac{d\times(d-1)}{n}
$$

under one missing pair. When we are missing k pairs, the ratio would be:

$$
\frac{k\times d\times(d-1)}{n}=r\times d\times(d-1),
$$

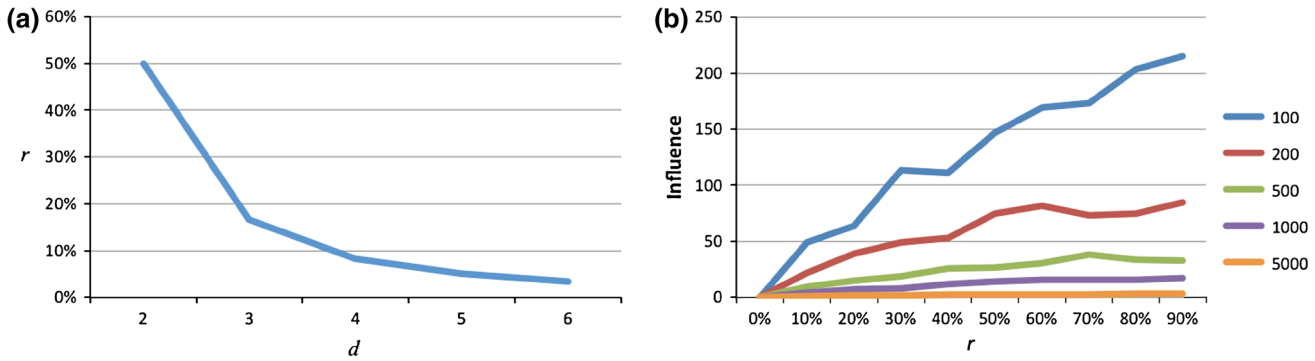


Fig. 3 **a** The relationship between r and d when the rate of influence reaches 100 %. **b** The degree of influence (y-axis) to the ratio (x-axis) of the number of feature values being changed

where $r = \frac{k}{n}$ is defined as the rate of missing nodes. As the order d (>1) is increased, the influenced elements of a bag increase exponentially. This influence leads to low tolerance for missing nodes. Figure 3a shows the relationship between r and d when the rate of influence reaches 100 %. We can see that while the tensor order is >4, the missing rate cannot be >10 %.

Increasing the order of the tensor leads to another difficulty: the decreasing degree of discernibility. As long as the number of elements in a bag is very large, those bags will become harder to distinguish. To demonstrate this, we randomly produce k feature values and considered these values as the elements of a bag. This bag is then computed to the feature vector $\boldsymbol{v}_{\text{ori}}$ using Eq. (5). Given a missing rate of r, $r \times k$ feature values of the bag are randomly selected and replaced using new random values. Then, a new bag is produced and can be computed with a new feature vector $\boldsymbol{v}_{\text{new}}$. Their dissimilarity is computed as $\text{dis}(v_{\text{ori}}, v_{\text{new}})$ using Eq. (13), and indicates the degree of discernibility. Figure 3b shows the degree of discernibility in relation to r. We see that while the number of elements is $>1{,}000$, the two bags cannot be distinguished, even if they are 30 % different. Such issues are easy to encounter in hypergraph matching, for example, if $d = 3$ and $n > 32$, then the number (32^3) of elements in the bag will be $>1{,}000$; if $d = 4$ and $n > 10$, then the number (10^3) of elements in the bag will also be $>1{,}000$.

Some of the above issues can be addressed. First, if only a few nodes are designed for hypergraph matching, the propagation of errors will be reduced. Second, to avoid the influence of noise, the hypergraph matching can only be applied in a case where there are no missing nodes or data noise. Third, constructing a tensor that is very sparse helps to reduce the effects of error propagation. This can be achieved by applying constraints as proposed by Duchenne et al. [22]. Finally, we can select a suitable tensor order according to the order of the relationship, and use the attributed hypergraph-matching method proposed in the next section to combine different orders of relationships.

6 Attributed hypergraph matching

Attributed graph matching is defined as:

$$P^* = \arg\min_P \left(\sum_{k=1}^{m} w_k \left\| T_k - T'_{k,i_1,\cdots,i_{d_k}} (P\mathbf{e}_{i_1}) \otimes \cdots \otimes (P\mathbf{e}_{i_{d_k}}) \right\| \right), \tag{22}$$

where d_k refers to the order of the tensor relating to the d_k-ary relationship, and w_k is the weight for describing the importance of the kth feature. If $d_k = 1$, then $\left\| T - T_{i_1(P\mathbf{e}_{i_1})} \right\|$ will equal the form of $\left\| U - PU' \right\|'$ relating to Eq. (2). Similarly, if $d_k = 2$, then $\left\| T - T'_{i_1,i_2} (P\mathbf{e}_{i_1}) \otimes (P\mathbf{e}_{i_2}) \right\|$ will equal the form of $\left\| A - PA'P^T \right\|$ relating to Eq. (1). As we have the solution for each part of the Eq. (22) (hypergraph matching), i.e.,

$$P^* = \arg\min_P \left\| T_k - T'_{k,i_1,\ldots,i_{d_k}} (P\mathbf{e}_{i_1}) \otimes \cdots \otimes (P\mathbf{e}_{i_{d_k}}) \right\|, \tag{23}$$

we solve this problem first and then combine all the solutions. Recall that we can compute the similarity matrix S_k for Eq. (23) at first. Then the combination is defined as:

$$S = \sum_{k=1}^{m} w_k S_k, \tag{24}$$

and the final solution P^* is derived from S using Eq. (2).

However, previous research [19] shows that combining multiple solutions may increase the occurrence of problems due to degradation and bias if one of the error solutions dominates the result. A reason for this could be that the solutions are solved from different types of equations and instances,

and do not transform the solutions into the same representation. In our case, the similarity is defined as the same type, $k(d(i, j))$, and the instance is extracted from the same feature domain, the coefficients of the approximated Fourier series, so that we can directly combine them using Eq. (24) to minimize Eq. (22). The weight w_k is also designed to avoid issues relating to degradation and bias, which will be discussed in detail below.

In the following discussion, we will focus on one feature and ignore the index k to allow for a compact representation of symbols. In our computation, the value of $S_{i,j}$ will be large when $A(i)$ and $A'(j)$ contain the feature values of the same node, and most of $S_{i,j}$ are very small since $A(i) \neq A'(j)$ in most cases. We can assume that such S should have elements with a larger standard deviation, σ. To prove this, two matrices A and A' are designed in the following tests. Each element in matrix A is given a random number. The matrix A' is computed by $PAP^T + N$, where P is a given permutation matrix and N is a noise matrix. An element of N is computed by multiplying ε with a random variable, which is uniformly distributed on the interval between -0.5 and 0.5. The similarity matrix S is then computed from A and A'. Figure 4 shows several S values with σ in a matrix of size 50×50. We see that a larger ε correlates to a smaller σ.

The feature values with more noise are supposed to be less reliable for computing P. The σ value helps us to understand the unknown noise and to decide the reliability of the feature. However, σ is too sensitive when noise is added. In Fig. 4, the σ value is too small to be discernible when ε is >0.2. In this paper, we replace the σ value by the degree of orthogonality g as the importance of the feature.

We also consider a permutation matrix that is an orthogonal matrix whose columns and rows are orthonormal vectors. If a matrix is orthogonal, its transpose will be equal to its inverse, which requires $PP^T = I$, where I is the identity matrix. Since S may be derived to produce P, S should be an orthogonal matrix in some way. To this end, we compute $I' = SS^T$, and normalize I' to I^* by:

$$I^*_{i,j} = I'_{i,j} \Big/ \sqrt{\sum_j I'^2_{i,j}}.$$

The degree of orthogonality [39], g, can be defined by $g = Tr(I^*)/n$, where $Tr(I^*)$ is the trace of I^* and n is the number of nodes. Figure 4 also shows the g values relating to ε. In the best case, $S_{i,j}$ performs a Dirac delta function, $S_{i,j} = \delta(d(i, j))$, which would occur while searching a graph isomorphism of exact graph matching. We have $g = 1$ in this case. In the worst case, there is no relationship between $A(i)$ and $A'(j)$, and the elements of S are distributed uniformly. The $\hat{I}_{i,j}$ is also distributed uniformly and we have a lower bound of $g = 1/\sqrt{n}$ after computing.

In our application, the weight w_k of the kth feature is computed by:

$$w_k = \frac{g - 1/\sqrt{n}}{1 - 1/\sqrt{n}} \tag{25}$$

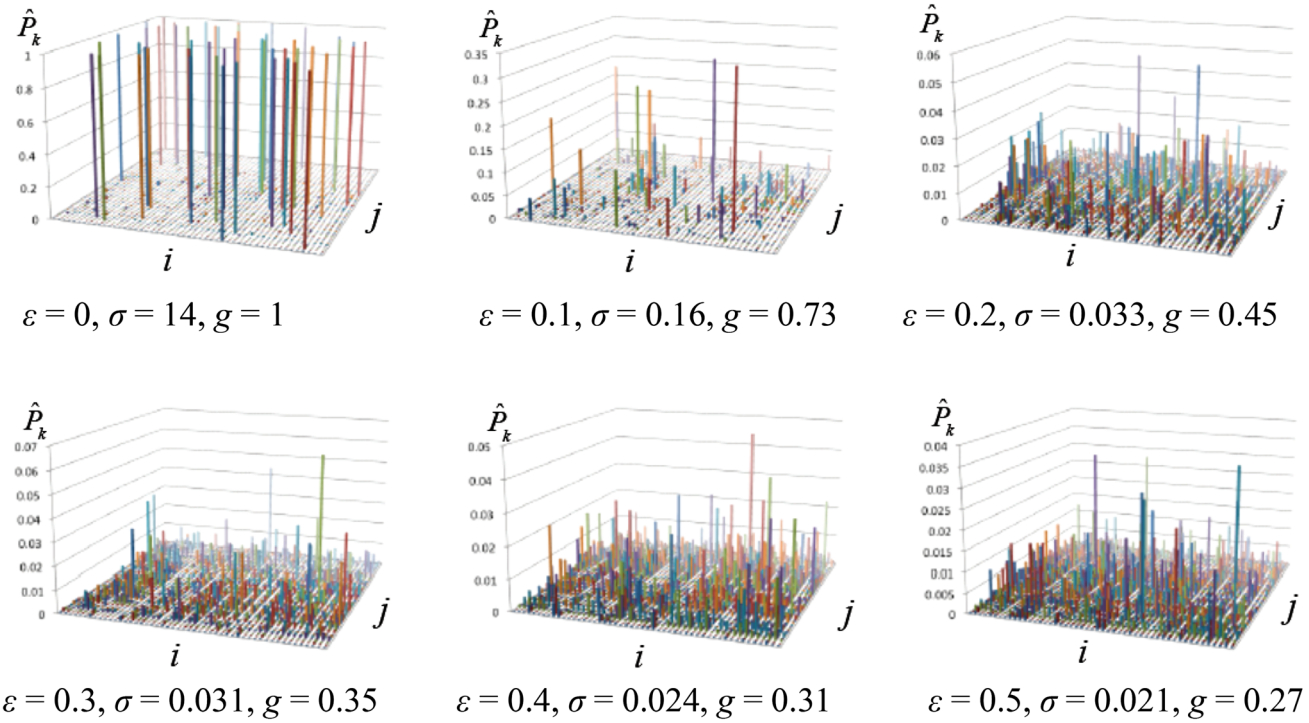


Fig. 4 The standard deviation σ of elements of an unconstrained permutation matrix, and the orthogonality g value related the different ε value

Table 1 The processing times of each method by the number of nodes (*underlining* indicates the quickest processing time for that number of nodes)

	Our method	[40]	[24]
Processing time (10 nodes)	0.0009 s	0.0005 s	0.0017 s
Processing time (50 nodes)	0.02 s	0.14 s	0.68 s
Processing time (100 nodes)	0.07 s	4.5 s	1.7 s
Processing time (200 nodes)	0.3 s	74.6 s	12.9 s
Processing time (300 nodes)	0.67 s	377 s	93.7 s

instead of σ because of the following reasons:

1. The g value is in the interval between 0 and 1, while the interval of the σ value depends on the application.
2. The g value can be computed even when the number of nodes is very small, while σ cannot be reliably computed with a small number of nodes.
3. Orthogonality is more important than the distribution of matrix elements.

In computer vision, the unary relationship of feature values could be the RGB values of the feature point, and the binary relationships could be the Euclidean distance or the orientation from one feature point to another. The ternary relationship can be designed as the three angles of a triangle, which has the advantages of scale and rotation invariance. More examples of ternary features are given in reference [22]. The higher order feature is often represented in the whole structure, in which the tensor is often very sparse.

7 Experimental results

We compare our proposed method with those proposed and compared in references [24] and [40] using a 2 GHz Intel Core 2 Duo machine with an Xcode compiler in MacOS X. Since the two methods to be compared dealt with on graphs, only first-order and second-order tensors are considered here. Since we do not have the source code of the method in [24], We implement the program ourselves. As discussed in reference [24], the processing method is dominated by an algorithm that extracts the permutation matrix from the similarity matrix, and the resulting time complexity is $O(n^3)$. Table 1 shows the minimum processing time based on fifty trials. According to this table, although the method presented in [40], (with $O(n^4)$) outperformed other methods with few nodes, our method has less processing time than the other methods in most cases.

We designed three experiments to test the proposed algorithm. In the first, some simulations of tensors of different orders are designed to demonstrate the robustness of our hypergraph matching method. In the second experiment, we apply attributed hypergraph matching to some synthetic data to test the performance. In the third, the feature points extracted from a real image are tested for applicability. We designed several features for use as the attributes in attributed hypergraph matching. We show that the weights can reflect the importance of the attributes. Finally, we test local relationships to show that the partially connected graph has higher reliability than the fully connected graph.

7.1 Hypergraph matching

In this experiment, we use a tensor whose elements are assigned randomly between $-\pi$ and π. This tensor is analogous to the T in Eq. (15). A permutation matrix P' is constructed by randomly assigning 1 and 0 to its elements and satisfying the constraints of the permutation matrix, which are to have only one element equal to 1 in each row and each column. The permutation matrix P' is used to permute T to obtain T' according to $T' = T_{i_1,\ldots,i_{d_k}}(P\mathbf{e}_{i_1}) \otimes \cdots \otimes (P\mathbf{e}_{i_{d_k}})$, so P'^T serves as the ground truth of P computed by our algorithm. We also construct a noise tensor, N, whose order is the same as T and whose elements are defined via a random variable, uniformly distributed on the interval between -0.5π to 0.5π, multiplied by a magnitude parameter ε. The probability of a correct correspondence is calculated using the average of 500 trials for a given ε, and N is changed for each trial.

In the first test, T' is produced with noise, $T' = T_{i_1,\ldots,i_{d_k}}(P\mathbf{e}_{i_1}) \otimes \cdots \otimes (P\mathbf{e}_{i_{d_k}}) + N$, given different ε values. Figure 5 shows the probability p of correct correspondence related to the ε values given different numbers of nodes (5, 10, 20, 50, or 100). Figure 5a–d is computed using the formulation of first-order, second-order, third-order, and forth-order tensors, respectively. In most cases, fewer nodes will lead to a higher probability of correct correspondence under the same ε value, except for bipartite matching (using a first-order tensor) with five nodes, which tends to have $p = 40\,\%$. This value is greater than the expectation value of 20 % for p in blind matching.

Except for the case of bipartite matching, p mainly decreases when ε reaches 20 %. For example, consider the binary relationships. We can compute the distance between two points as the binary relationship in a 320×240 image,

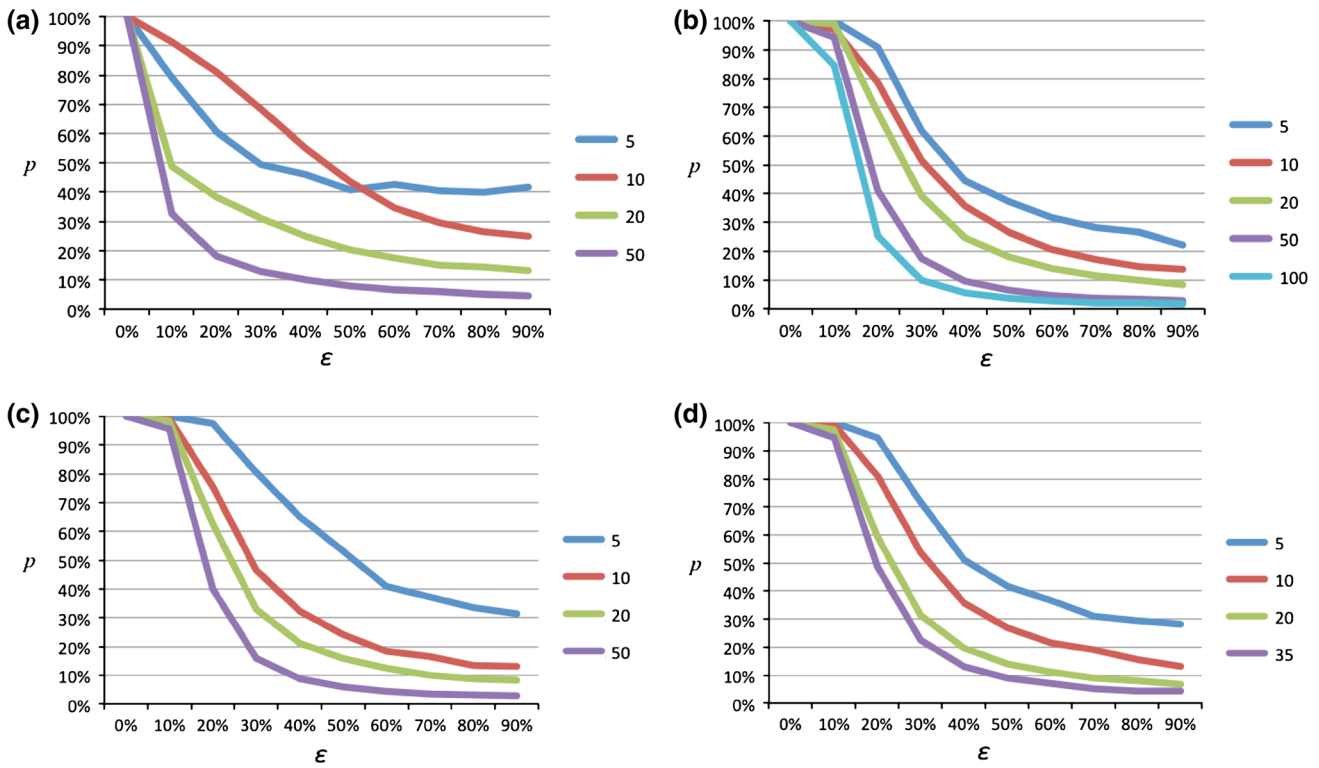


Fig. 5 Probability of correct correspondence related to the noise ratio ε given different numbers of nodes. Each figure is computed from the hypergraph matching expressed by **a** a first-order tensor (feature vector), **b** a second-order tensor (adjacency matrix), **c** a third-order tensor, and **d** a fourth-order tensor

and the resulting distance will be between 0 and 400. After mapping the interval ($-\pi$ and π) according to our testing, the deviation will be between -20 to $+20$ pixels if the ε value is 20 %. The distance between points seldom changes so dramatically in stereo images or successive images. Our method is robust in terms of handling applications where noise exists in the feature values. Furthermore, our method is robust to the number of nodes. In Fig. 5b, we see that even when the node number is 100, the p value is still as high as when matching with 50 nodes.

In the second test, we replace several entries of T', computed by $T' = T_{i_1,\ldots,i_{d_k}}(P\mathbf{e}_{i_1}) \otimes \cdots \otimes (P\mathbf{e}_{i_{d_k}})$, with new random values to simulate the case of missing nodes. While a node i is selected to act as the missing node, the element of $T_{:,\ldots,:,i,:,\ldots,:}$ is assigned a new value. The ratio of the replaced entries to the total entries is given as r, and the identical part of both graphs represented by T and T' to one graph is $1-r$. The probability p of correct correspondence is defined as the ratio between the number of correct matching nodes and the number of identical parts.

Figure 6 shows the p values corresponding to different r values. In bipartite matching (Fig. 6a), the probability of correct correspondence is over 98 %, even when the number of nodes is 100, which means that the bipartite matching can tolerate missing nodes. As mentioned in Sect. 4, using high-order features leads to a decreased tolerance for missing nodes. Corresponding to Fig. 6b, c, and d, the orders of the features are binary, ternary, and quaternary. We can see that the p value decreases significantly as the order of the feature and the missing rate are increased.

7.2 Attributed hypergraph matching

In attributed hypergraph matching [Eq. (22)], the weight values, w_k, play an important role in combining the unconstrained permutation matrices computed from different features. The orthogonality of each unconstrained permutation matrix is computed as the weight value. In this test, we show the relationship between the weight w_k and the probability p of correct correspondence. There are twenty nodes in the graph, and the w_k value is calculated as the average of 500 trials for a given ε. Figure 7 shows the relationship between p and g. When the p value is small, the corresponding w_k value is also small. Our proposed w_k can reflect the importance of the corresponding $\hat{P}_k$ computed in Sect. 5.

Combining the different feature information formulated in Eq. (22) helps to increase the tolerance for deviation and missing nodes. Figure 8 shows the results of attributed hypergraph matching with noise. In this test, we fix the number of nodes with different feature numbers m (1, 2, 3, 4, and 5) and

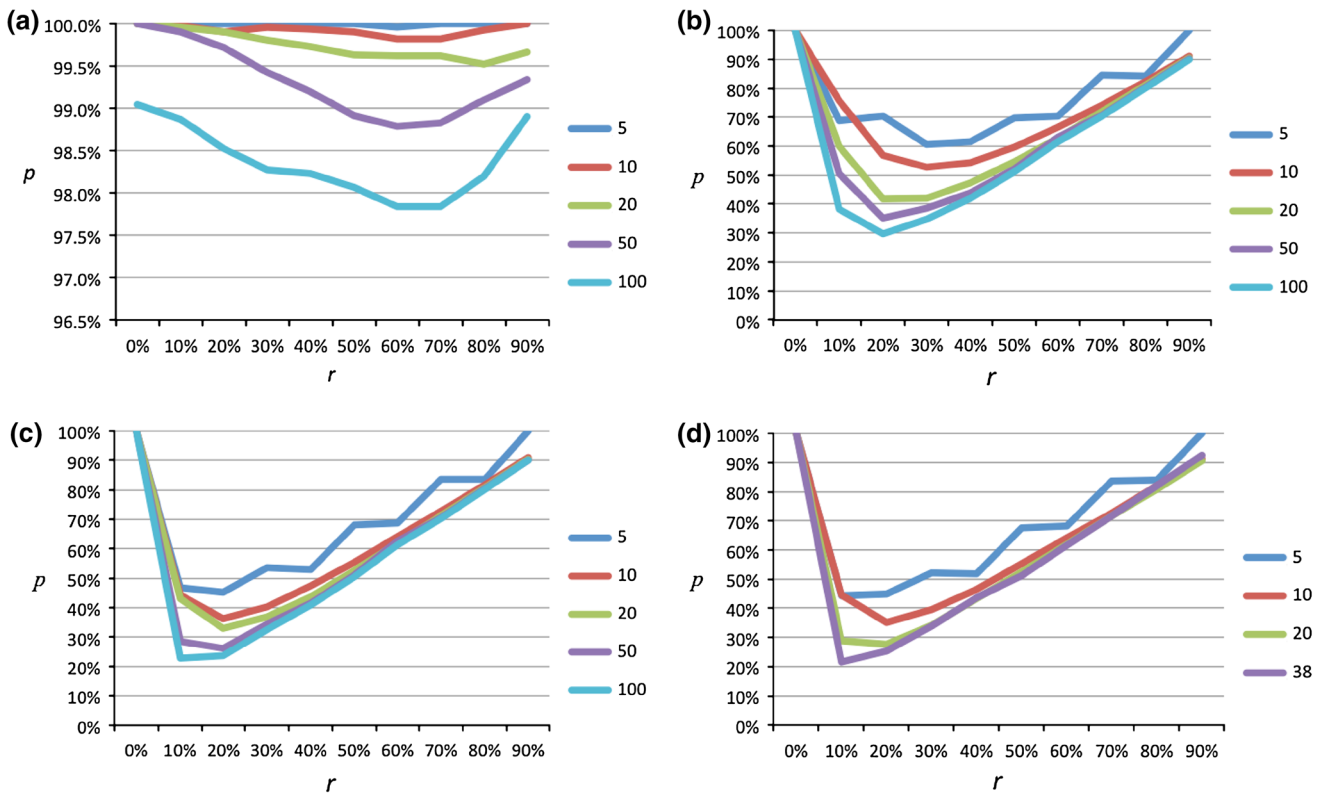


Fig. 6 Probability of correct correspondence versus the missing node rate r given varying numbers of nodes. Each figure is computed from the formulation in terms of **a** a first-order tensor (feature vector), **b** a second-order tensor (adjacency matrix), **c** a third-order tensor, and **d** a fourth-order tensor

compute the probability p of correct correspondence relating to the different ε values for noise. Different numbers of nodes are fixed according to the feature type, where the node numbers, 20, 50, 20, and 5 are selected to test the computation formulated using first-order, second-order, third-order, and forth-order tensors, respectively. The experimental results show that the p value can increase as more attributes are considered.

7.3 Weight adjustment

In this experiment, we extract the feature points from two images, one of which is transformed from the other. We selected translation, rotation, scaling, occlusion, and intensity changes as our transformation methods because they are commonly used in computer vision. Several feature values are computed and used as the attributes of the attributed hypergraph. We apply our attributed hypergraph matching methodology to search the correspondence of the feature points in both images and each weight of an attribute is computed by Eq. (25).

Five feature values are computed for one feature point. first, the coordinate (x, y) is separated into the attributes of x-position and y-position. The third attribute is intensity. The distance between a pair of feature points is defined as the fourth attribute. The final attribute is the angles of a triangle such that the nodes are three of the feature points. Any one feature cannot be invariant to all the transformations. For example, the distance between pairs can be invariant to translation but variant to scaling, and position can be invariant to occlusion but invariant to translation.

Figure 9 shows the weight values under different transformations. The experimental results show that our weight adjustment method can reflect the importance of an attribute. When the image is translated horizontally or vertically, the weights of the x-position and y-position are reduced. The rotation-invariant features—intensity, distance between pairs, and angle of the triangle—play key roles after a rotation transformation of the image. The ternary feature, the angle of the triangle, is also scale-invariant, therefore it still has a high importance in image scaling.

Figure 10 shows the correspondences of feature points. There are 311 feature points extracted from the image (Fig. 10a). After applying translation, rotation, scaling, occlusion, and intensity changes to the image, each feature point is connected with its corresponding feature points using a line. Some outliers are connected to dummy feature points and shown as lines with dark colors. In the case of occlusion, many feature points in the transformed image are miss-

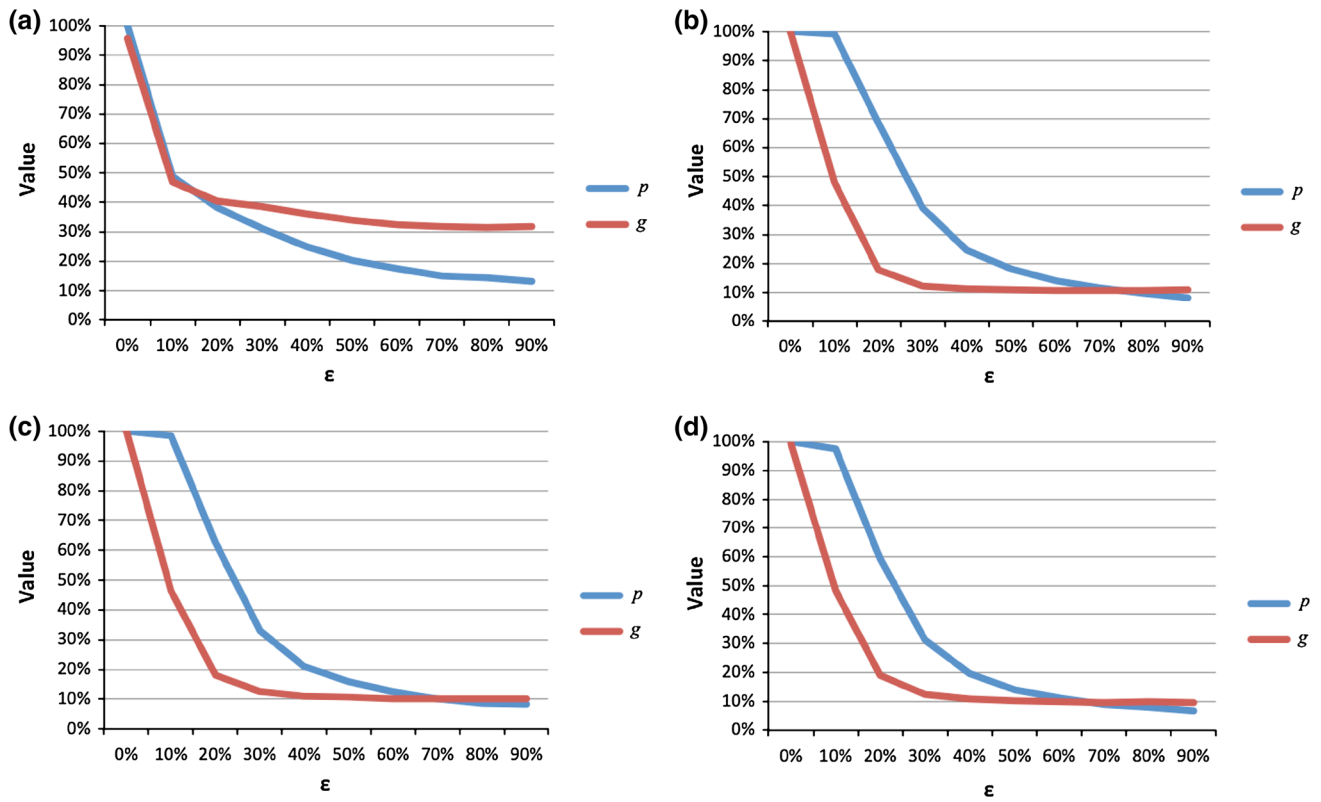


Fig. 7 The relationship between p and g with 20 nodes for **a** a first-order tensor (feature vector), **b** a second-order tensor (adjacency matrix), **c** a third-order tensor, and **d** a fourth-order tensor

ing, therefore there are more outliers shown in the image. A similar situation is shown in the case of changing intensity. When the intensity of an image is changed, so is its contrast. This influences the extraction of feature points and the corresponding results.

In this test, five matching results are obtained according to single feature. Two more results are obtained from Eq. (22) using first the same weight values and then different weight values as determined by our weight adjustment method. By comparing the weight and accuracy values (Figs. 9 and 11), we see that a larger weight value often leads to higher accuracy of matching. Combining the different types of feature information formulated in Eq. (22) helps to increase the accuracy as shown in the last two matching results. Our proposed method gives higher accuracy when compared with the method without weight adjustment, particularly for cases involving rotation and scaling.

In Fig. 12, we apply our proposed method to the real world data. The first pair of images was captured while the camera was rotated. The correspondence of feature points shows the rotation of the camera. In the second row of images, camera was shift in some direction. We can see that the upper feature points have smaller shift distances than the bottom points, which means that objects in the bottom part are closer to the camera than the objects in the upper part. The third row of images was captured from a static camera. Feature points on the moving objects are matched successfully even though the objects in the scene moved in different directions. Figure 13 is obtained from CMU Image Data Base:http://vasc.ri.cmu.edu/idb/html/motion/, where the house has 3D motion. The correspondence of feature points can be obtained by applying our matching algorithm.

7.4 Local relationships

Binary features are often defined according to the difference between two nodes. For example, let us take the distance feature, which is computed by subtracting one node's position from the other node's position. Using this, we can construct second-order tensors for matching. However, computing a permutation matrix using such tensors may invoke error propagation and low discernibility as discussed above. To solve those problems, one may reduce the number of nodes or set a threshold to make the tensor sparse [22]. Here, we take a binary feature, for example, but the same idea can be applied to a high-order tensor.

The node number of the graph is selected to be 100. While the adjacency matrix is constructed, a threshold is given for eliminating the larger entry values. This process is reasonable because the relationship between two nodes with a larger

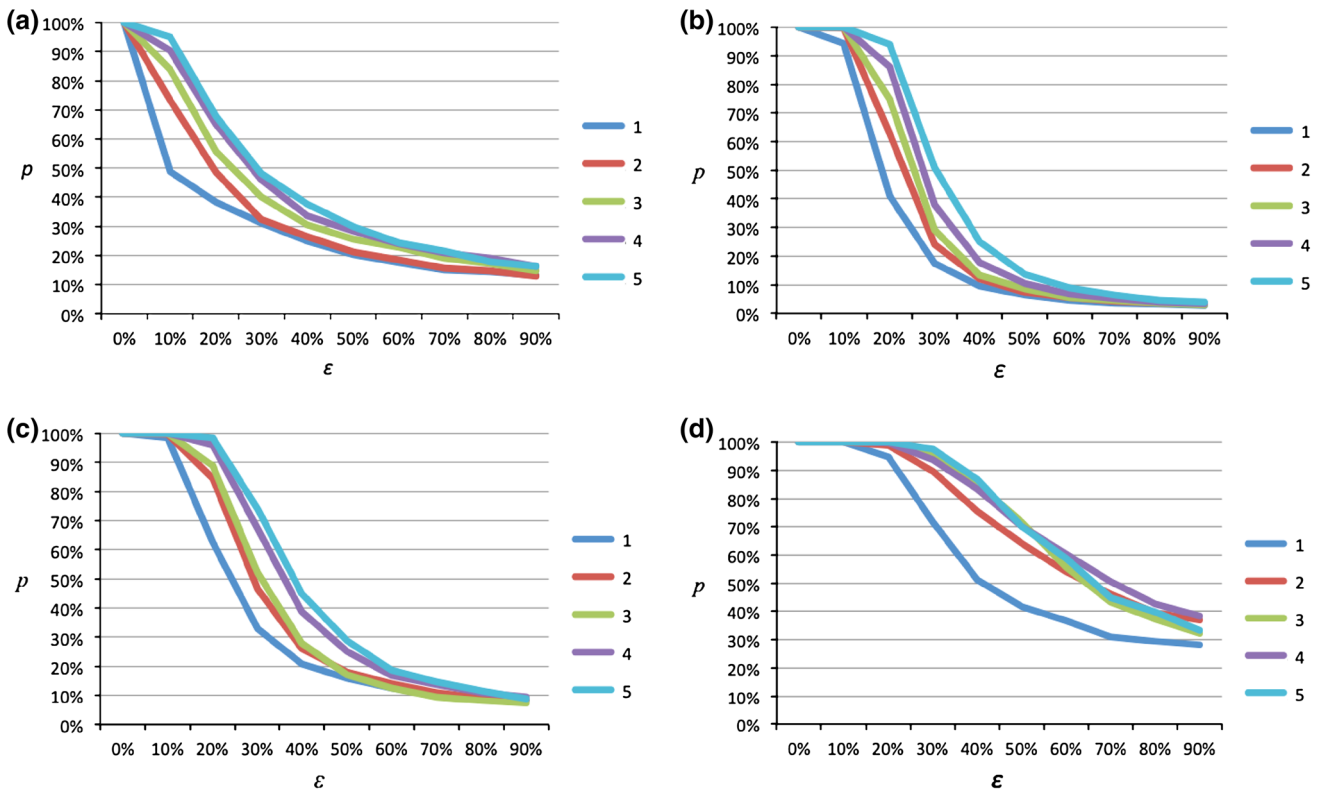


Fig. 8 Probability of correct correspondence related to noise ratio ε given a different number of features. They are formulated by **a** a first-order tensor (feature vector) with 20 nodes, **b** a second-order tensor (adjacency matrix) with 50 nodes, **c** a third-order tensor with 20 nodes, and **d** a fourth-order tensor with 5 nodes. The p value can increase as more attributes are considered

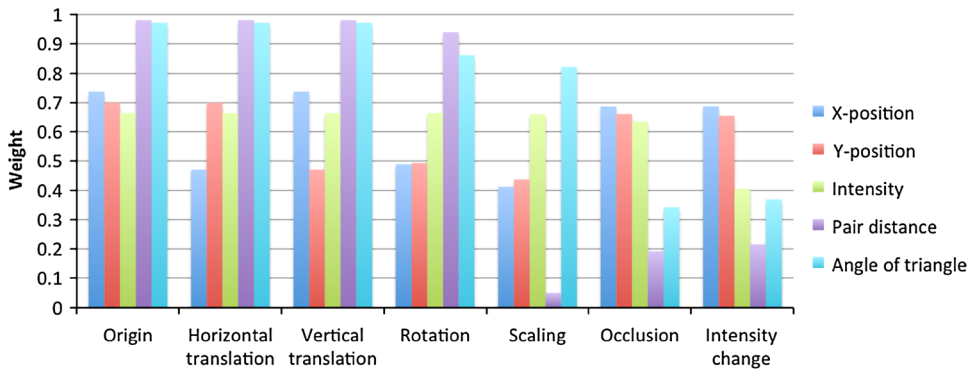


Fig. 9 The weights of attributes adjusted under different transformations

distance should be low. A fully (100 %) connected graph is equivalent to the experiment designed in hypergraph matching. Graphs that are only partially (60, 30, and 10 %) connected are designed and the probability of correct correspondence is calculated as the average of 500 trials for a given noise ratio ε. Figure 14a shows the experimental results.

This test shows that the partially connected graph is robust when handling applications with noise present in the binary feature values. For example, a 10 % connected graph can increase the rate of correct matching from 94 to 98 % under a ε value of 10 %, and from 17 to 28 % under a ε value of 30 %. Compared with a fully connected graph, a partially connected graph has fewer edges. A graph with very few edges is similar to a graph that only has nodes, which leads to graph matching considering unary features. Figure 14b illustrates such a situation, where the 2 % connected graph has the highest rate of correspondence.

7.5 Comparisons

To the best of our knowledge, although some attributed graph-matching [24] and hypergraph-matching [19] algorithms have previously been proposed, none have included algorithms focusing on attributed hypergraph matching, as

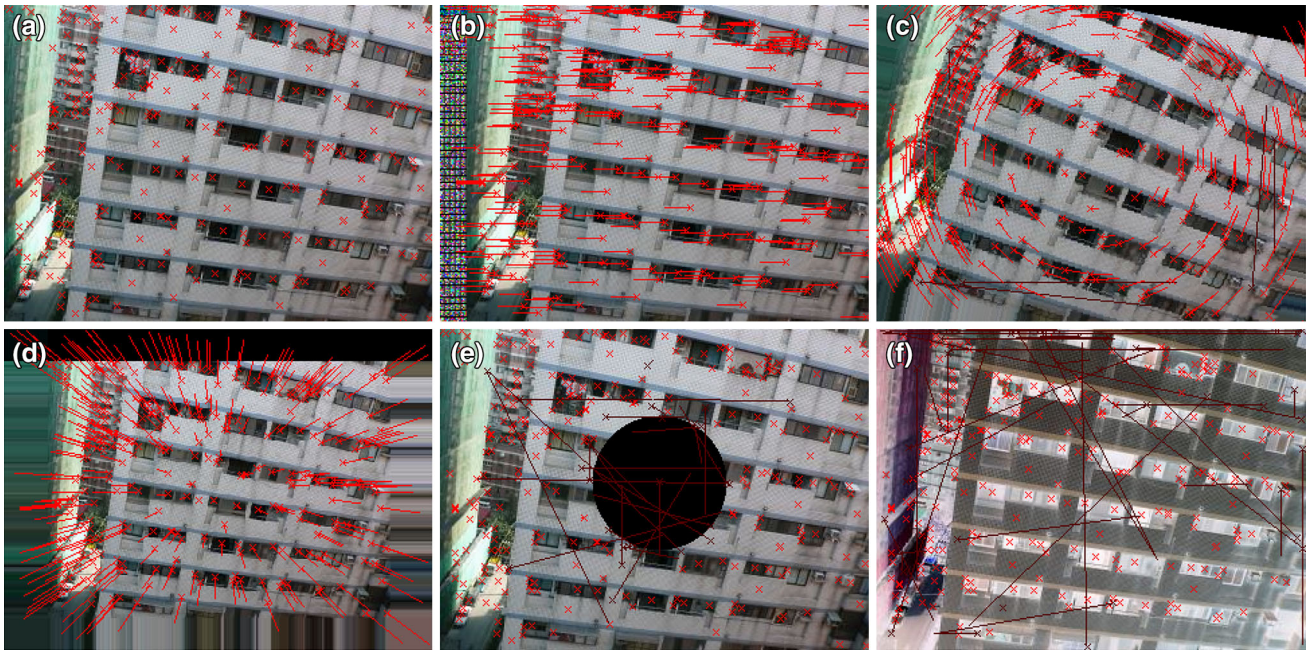
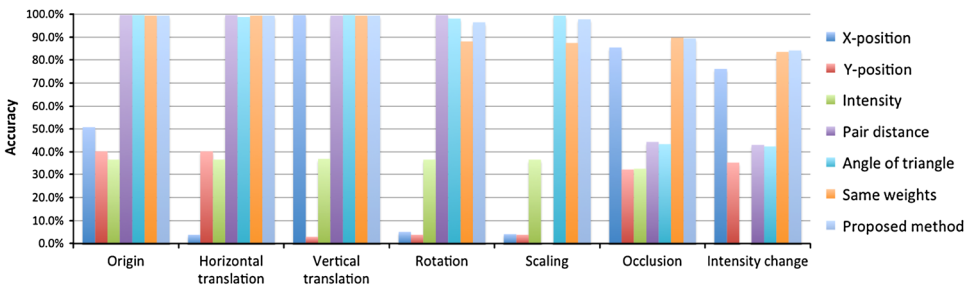


Fig. 10 Correspondence of feature points extracted from two images, where **a** is original image and the other one has been transformed via **b** horizontal translation, **c** rotation, **d** scaling, **e** occlusion, and **f** a change of intensity

Fig. 11 The accuracy of matching using different attributes under different transformations. The last two attributes refer to methods described in this article



our method does. Thus, these experiments are designed to contrast our processed results with those of van Wyk et al. [24] and of Chertok and Keller [19].

7.5.1 A. Part 1

Two experiments were designed for comparing with RKHS Interpolator-based graph-matching method [24]. These alternative methods that are compared in [24] include polynomial transform graph matching, linear programming graph matching (LPGM), Eigen-decomposition graph matching, graduated assignment graph matching (GAGM), conjugate gradient graph matching (CGGM), and spherical approximation graph matching (SAGM). See reference [24] for more detail regarding these algorithms.

In the first experiment, ten nodes with three binary and three unary attributes are simulated. In other words, we construct three weighted adjacency matrices and three attributed vectors, which are converted to diagonal matrices, to simulate the reference graph G'. All attribute values are distributed uniformly between 0 and 1. A permutation matrix, whose transpose is also used as the ground truth, is generated randomly and used to permute the matrices to obtain duplicate graphs. The elements of the noise matrix are defined as random variables, uniformly distributed on the interval between -0.5 and 0.5. These are then multiplied by a magnitude parameter ε, and added to the duplicated attributed matrices to form the simulation of the duplicate graph G. The permutation matrix for permuting G' to G is then computed and compared to the ground truth to obtain the correct correspondence.

Figure 15a shows the probability p of correct correspondence relating to the ε values given the different algorithms. The processing result obtained by our proposed method, i.e., AHM, overlaps the figure extracted from Ref. [24]. As discussed in Sect. 5.2, the higher number of features improves our algorithm's tolerance for added noise. The process of weight adjustment also enhances those attributes with high reliability. This is why our method outperforms the other algorithms, as shown in the figure below.

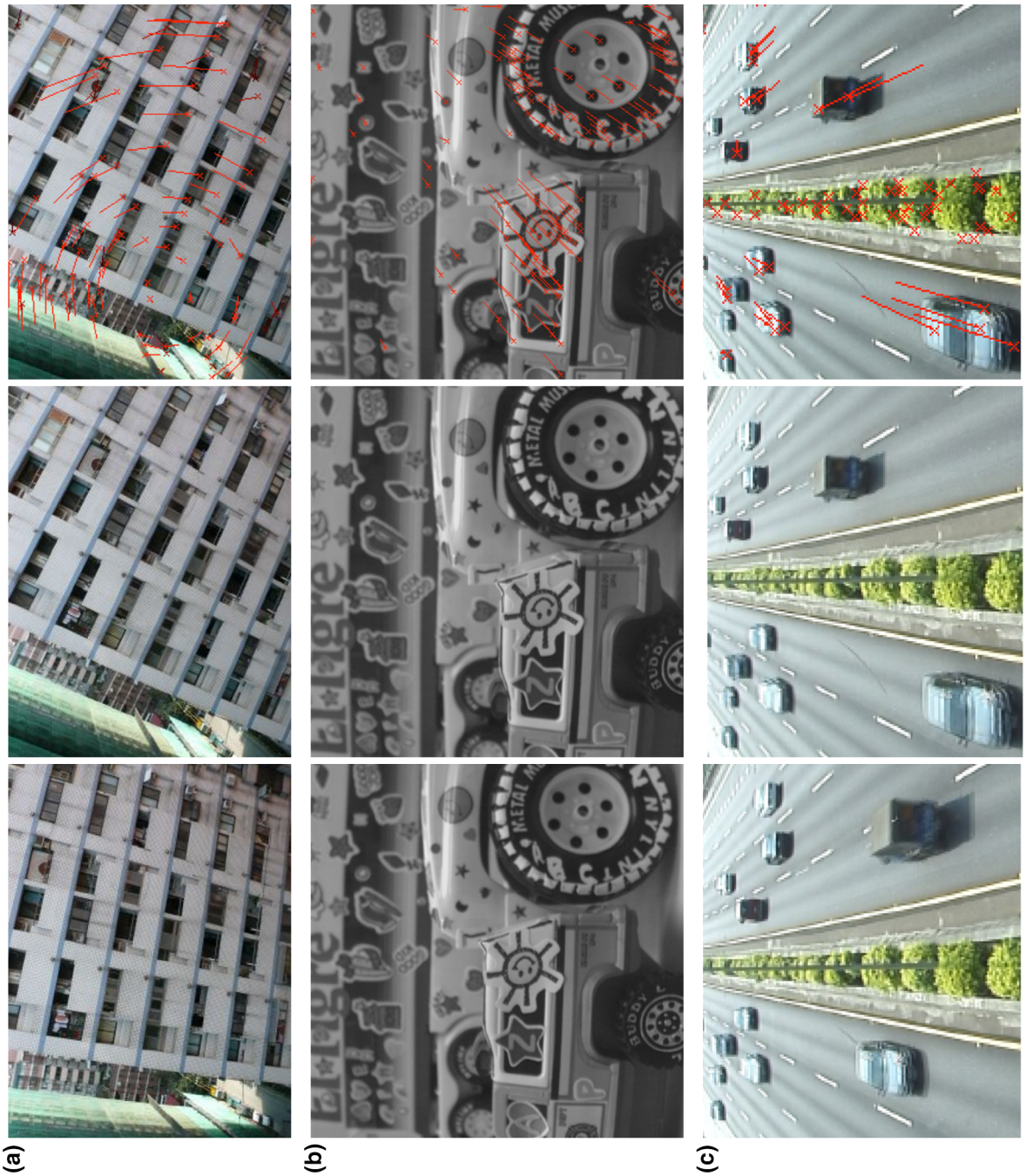


Fig. 12 Correspondence of feature points extracted from two real images. Column **a** shows the previous images, column **b** shows the current images, and column **c** shows the matching between feature points

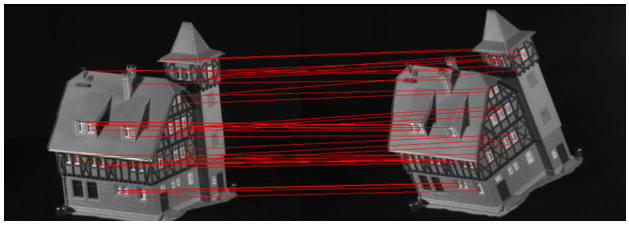


Fig. 13 Correspondence of feature points extracted from house with 3D motion

In the second experiment, the same procedure is used, but the node number is set to 30. Figure 15b displays the results, showing that the correct correspondence decreases as the node number increases. Our method did not perform well with an ε value smaller than 50 %, but outperformed most of the other methods in the case of a larger ε value. Although our method is outperformed by some methods, such methods require high levels of space and time complexity. As discussed by van Wyk et al. [24], the GAGM, CGGM, and SAGM algorithms require $O(n^4)$ in space complexity, while the AHM requires only $O(n^2)$. The GAGM, CGGM, and LPGM require $O(n^4)$, $O(n^6)$, and $O(n^6)$ in time complexity, respectively, while our proposed method requires only $O(n^3)$.

7.5.2 B. Part 2

We designed the second part of the experiment according to that used by Chertok and Keller [19]. In their experiments, a set of 20 random points are generated and defined as the source set. Their ternary attributes are computed to construct a third-order tensor. A permutation matrix, whose transpose is also the ground truth, is generated randomly and used to permute the elements of the source set to obtain a target set. We add Gaussian noise to the attribute values of the target set. Given different standard deviations σ of the Gaussian distribution of the noise, we apply hypergraph matching (or high-order matching used by van Chertok and Keller [19]) to compute the permutation matrix (or the correspondence between the points).

Figure 16 shows that our processing result overlaps with the results of the other methods, summarized below, described by Chertok and Keller [19]. The FA+TE are full affinity (FA) tensors (TE) applied for computing. The FA+M2 and FA+M1 are full affinity tensors marginalized into a matrix (M2) and vector (M1), respectively. These methods are all types of the Tensor-High Order Assignment (HOA) algorithm [19]. ProbOpt is a Probabilistic Graph and Hypergraph Matching [21], Spectral is a spectral technique for correspondence problems using pairwise constraints [28], and GradAssign is a graduated assignment algorithm for graph matching [32]. It is worth noting that Spectral and GradAssign were designed to solve permutations for second-order tensors (i.e., matrices).

The results using ten potential matches per point are reported in Fig. 16a. These are for a partial matching, where each point is assumed to have relationship values with half of the other points. This figure shows that AHM outperforms the other methods when the σ value is <0.03. Although AHM is outperformed by most of the methods in partial connection cases, it outperforms most of the other methods in the full connection case as discussed below.

In Fig. 16b, there is no pruning of potential assignments of each point, which makes the problem more difficult. In this case, AHM outperforms the previous matching algorithms. Tensor-HOA methods [19] slightly outperform AHM when the standard deviation of the Gaussian noise is >0.04. Even so, we have provided a very fast algorithm to deal with hypergraph matching. Tensor-HOA is stated to require polynomial time complexity, but the corresponding algorithm required 3 s to compute the result on a computer running an Intel Core 2 Duo. Our method, as tested before, required 0.02 s on a similar computer type. Hence, we have provided a very fast processing algorithm with only slight error to deal with the problem of hypergraph matching.

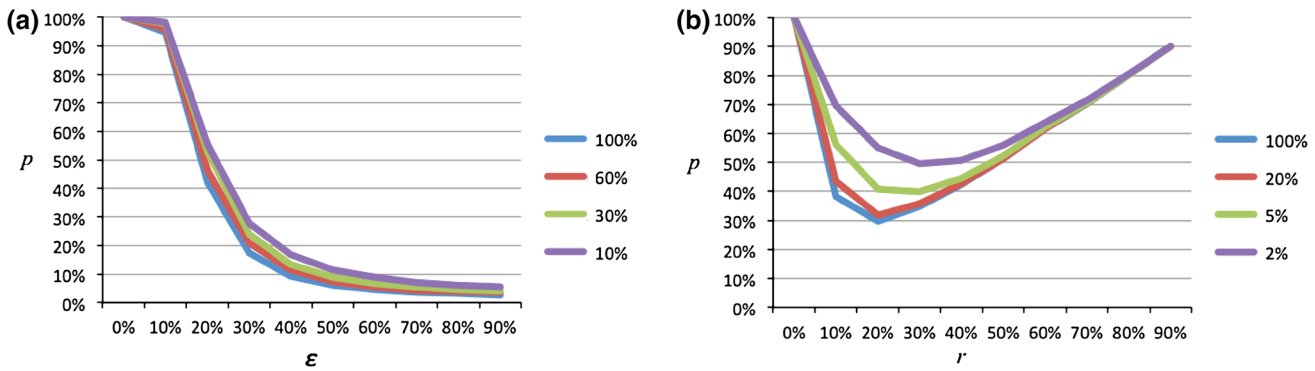


Fig. 14 Probability of correct correspondence related to **a** the noise ratio ε given different (100, 60, 30, and 10 %) partially connected graphs and **b** to the missing node ratio r given different partially (100, 20, 5, and 20 %) connected graphs

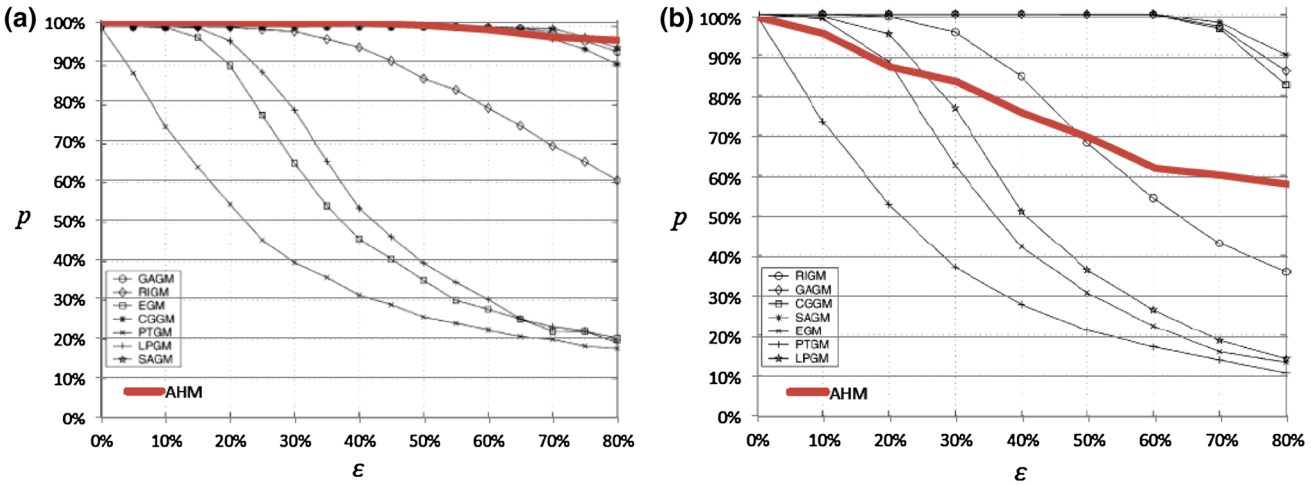


Fig. 15 Our processing result overlaps the figure extracted from reference [24]. The probability p of correct correspondence, in relation to the ε values, given different algorithms, with **a** node number 10 and **b** node number 30

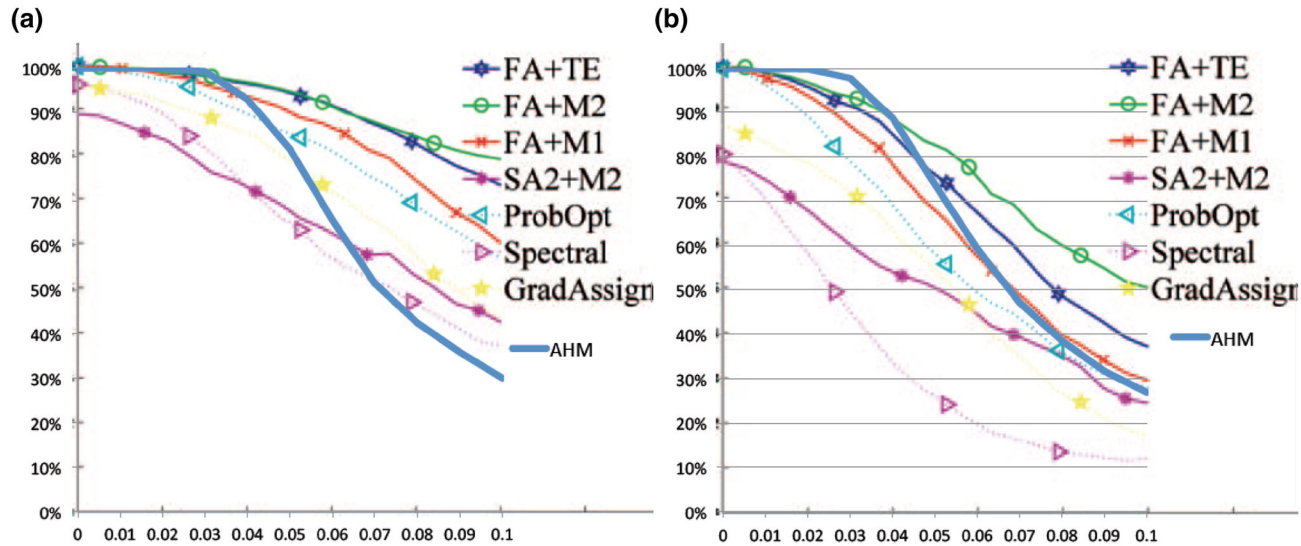


Fig. 16 Our processing result overlaps the figure extracted from reference [19]. The probability of correct correspondence (Y-axis) related to the σ values (X-axis) given different algorithms, where **a** each point is potentially matched to half of the other points and **b** each point is potentially matched to all the other points

The next test is designed for partial matching. Twenty points are randomly generated and added to a varying number of outliers. Due to the low degree of discernibility, our method using a high-order tensor cannot perform well for partial matching. Fortunately, weight adjustment helps to enhance the weight of the first-order tensor for matching. Figure 17 shows that our matching result has a high accuracy in comparison to a Tensor-HOA algorithm.

8 Conclusion

Searching for the correspondences between feature points plays an important role in computer vision task. In this paper, we proposed an attributed hypergraph-matching technique in terms of tensors. Using tensors to represent features means that we can apply not only unary and binary relationships for matching, but also higher-order relationships. However, combining the information from different attribute types to make a decision may cause inconsistency issues. To address this, we represent the attribute values in a Hilbert space and compute their difference on a Riemannian manifold. This processing method helps us to construct a matching with a high robustness for handling data noise, deviation, and inconsistency between attributes.

Our method has the following advantages: At first, our method took a significantly short processing time (<1 s) to perform a hypergraph matching, while obtaining reasonable

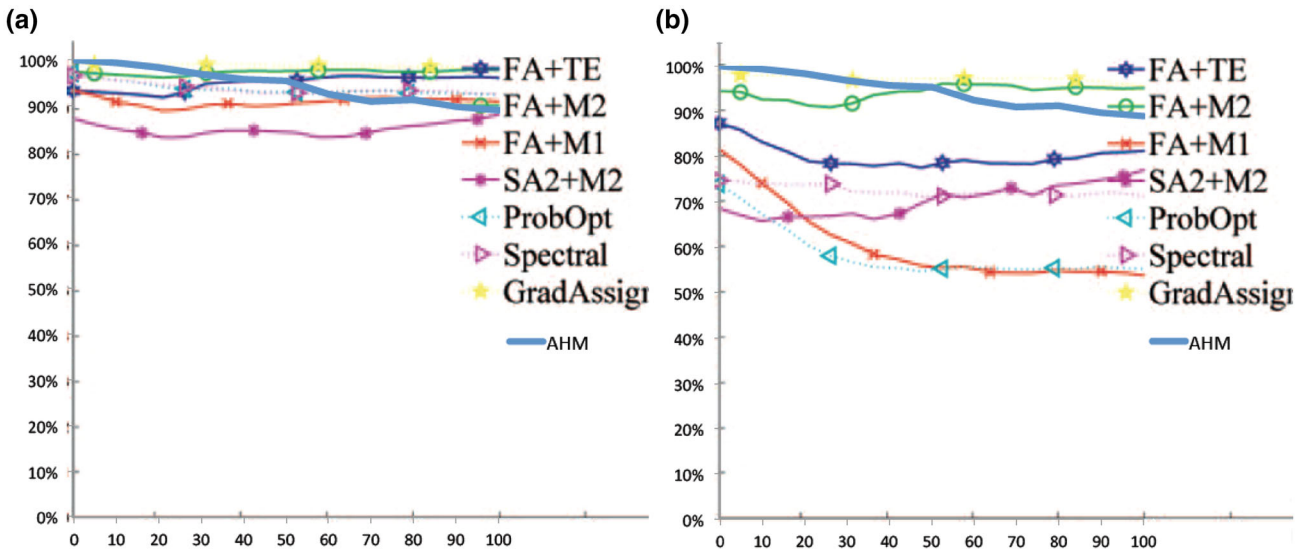


Fig. 17 Our processing result overlaps the figure extracted from Ref. [19]. Graph-matching simulations with added outliers, where: **a** each point is potentially matched to half of the other points, and **b** each point is potentially matched to all the other points

results. In the second, this speed allows us to employ a higher number of attributes and higher orders of relationships for hypergraph matching. The higher number of attributes helps our method to adapt to noise, while the higher orders of relationships allow our method to be invariant to some image distortion (i.e., scale, rotation, or translation). Finally, even graph matching is supposed to be NP-hard problem; we can obtain the optimal solution in polynomial time under some assumptions.

References

1. Papadimitriou, C.H.: Computational complexity. Addison-Wesley, USA (1995)
2. Russell, S., Norvig, P.: Artificial intelligence—a modern approach, 2nd edn. Prentice Hall, New Jersey (2003)
3. Conte, D., Foggia, P., Sansone, C., Vento, M.: Thirty years of graph matching in pattern recognition. Int. J. Pattern Recognit. Artif. Intell. **18**(3), 265–298 (2004)
4. Riesen, K., Jiang, X., Bunke, H.: Exact and inexact graph matching: methodology and applications. Managing and mining graph data, pp. 217–247. Springer, New York (2010)
5. Cordella, L.P., Foggia, P., Sansone, C., Vento, M.: A (Sub)graph isomorphism algorithm for matching large graphs. IEEE Trans Pattern Anal. Mach. Intell. **26**(20), 1367–1372 (2004)
6. Campbell, D.M., Radford, D.: Tree isomorphism algorithms: speed vs. clarity. Math Assoc. Am. **64**(4), 252–261 (1991)
7. Filotti, I.S., Mayer, J.N.: A polynomial-time algorithm for determining the isomorphism of graphs of fixed genus. In: Proceedings of the Twelfth Annual ACM Symposium on Theory of Computer, pp. 236–243. ACM, New York (1980)
8. McKay, B.D.: Practical graph isomorphism. Congr. Numerantium **30**, 45–87 (1981)
9. Ullmann, J.R.: An algorithm for subgraph isoorphism. J. Assoc. Comput. Mach. **13**(1), l31–42 (1976)
10. Feige, U.: Approximating maximum clique by removing subgraphs. SIAM J. Discret. Math. **18**(2), 219–225 (2005)
11. Bunke, H.: Error-tolerant graph matching: a formal framework and algorithms. Lect. Notes Comput. Sci. **1451**(1998), 1–14 (1998)
12. Abdulkader, A.M.: Parallel algorithm for labeled graph matching. PhD Thesis, Colorado School of Mines (1998)
13. Shapiro L.G., Haralick R.M.: Structural descriptions and inexact matching. IEEE Trans. Pattern Anal. Mach. Intell. **3**(5), 504–519 (1981)
14. Jain, B.J., Wysotzki, F.: Solving inexact graph isomorphism problems using neural networks. Neurocomputing **63**, 45–67 (2005)
15. Wilson, R.C., Hancock, E.R.: Structural matching by discrete relaxation. IEEE Trans. Pattern Anal. Mach. Intell. **19**(6), 634–648 (1997)
16. Cross, A.D.J., Wilson, R.C., Hancock, E.R.: Inexact graph matching using genetic search. Pattern Recognit. **30**(6), 953–970 (1997)
17. Freeman, J.A., Skapura, D.M.: Chapter 4: The BAM and the Hopfield memory. Neural networks algorithms, applications, and programming techniques, pp. 127–168. Addison Wesley, USA (1991)
18. Lebrun, J., Gosselin, P.H., Philipp-Foliguet, S.: Inexact graph matching based on kernels for object retrieval in image databases. Image Vis. Comput. **29**(1), 716–729 (2011)
19. Chertok, M., Keller, Y.: Efficient high order matching. IEEE Trans. Pattern Anal. Mach. Intell. **32**(12), 2205–2215 (2010)
20. Bunke, H., Dickinson, P., Karetzl, M., Neuhaus, M., Stettler, M.: Matching of hypergraphs—algorithms, applications, and experiments. Appl. Pattern Recognit. **91**, 131–154 (2008)
21. Zass, R., Shashua, A.: Probabilistic graph and hypergraph matching. IEEE Conference on Computer Vision and Pattern Recognition, pp. 1–8. Alaska (2008)
22. Duchenne, O., Bach, F., Kweon, I.S., Ponce, J.: A tensor-based algorithm for high-order graph matching. IEEE Trans. Pattern Anal. Mach. Intell. **33**(12), 2383–2395 (2011)
23. Tsai, W.H., Fu, K.S.: Error-correcting isomorphism for attributed relational graphs for pattern analysis. IEEE Trans. Syst. Man Cybern. **SMC–9**, 757–768 (1979)
24. van Wyk, M.A., Durrani, T.S., van Wyj, B.J.: A RKHS interpolator-base graph matching algorithm. IEEE Trans. Pattern Anal. Mach. Intell. **24**(7), 988–995 (2002)
25. van Wyk M.A., Clark J.: An algorithm for approximate least-squares attributed graph matching. In: Problems in applied mathematics and computational intelligence, pp. 67–72 (2001)

26. Ho, K., Hull, J.J., Srihari, S.N.: Decision combination in multiple classifier systems. IEEE Trans. Pattern Anal. Mach. Intell. **16**(1), 66–75 (1994)
27. Burkard, R.E., Dell'Amico, M., Martello, S.: Assignment problems. Society for Industrial and Applied Mathematics, USA (2009)
28. Leordeanu, M., Hebert, H.: A spectral technique for correspondence problems using pairwise constraints. IEEE Int. Conf. Comput. Vis. **2**, 1482–1489 (2005)
29. Leordeanu, M., Zanfir, A., Sminchisescu, C.: Semi-supervised learning and optimization for hypergraph matching. In: IEEE International Conference on Computer Vision, pp. 2274–2281. Barcelona (2011)
30. Cho, M., Lee, J., Lee, K.M.: Reweighted random walks for graph matching. In: European Conference on Computer Vision, pp. 1633–1640. Crete, Greece (2010)
31. Lee, J., Cho, M., Lee, K.M.: Hyper-graph matching via reweighted random walks. In: IEEE Conference on Pattern Recognition, pp. 1633–1640. Providence, RI (2011)
32. Gold, S., Rangarajan, A.: A graduated assignment algorithm for graph matching. IEEE Trans. Pattern Anal. Mach. Intell. **18**(4), 377–388 (1996)
33. Luo, B., Hancock, E.: Structural graph matching using the EM algorithm and singular value decomposition. In: The Asian Conference on Computer Vision, pp. 1–6. Melbourne (2002)
34. Umeyama, S.: An Eigendecomposition approach to weighted graph matching problems. IEEE Trans. Pattern Anal. Mach. Intell. **10**(5), 695–703 (1988)
35. Kolda, T.G., Bader, B.W.: Tensor decompositions and applications. SIAM Rev. **51**(3), 455–500 (2009)
36. Lu, J., Caelli, T., Yang, J.: A graph decomposition approach to least squares attributed graph matching. International Conference on Pattern Recognition, vol. 2, pp. 471–474. Cambridge (2004)
37. Kuhn, H.W.: The Hungarian method for the assignment problem. Naval Res. Logist. Q. **2**, 83–97 (1955)
38. Zheng, K.J., Peng, Jg, Ying, S.H.: A new approach to weighted graph matching. IEICE Trans. Inf. Syst. **E92–D**(8), 1580–1583 (2009)
39. Deng, L.Y., Lin, D.K.J., Wang, J.: A measurement of multi-factor orthogonality. Stat. Probab. Lett. **28**, 203–209 (1996)
40. Wang, J.M., Fuh, C.S., Chen, S.W.: Video stabilization for a hand-held camera based on 3D motion model. In: IEEE International Conference on Image Processing, pp. 3477–3480. Cairo (2009)

J. M. Wang received the B.Sc. degree and the M.Sc. degree in information and computer education from National Taiwan Normal University, Taipei, Taiwan, in 1996 and 2001, respectively, and Ph.D. degrees in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 2013. He is currently a teacher at the National Tao-yuan Agricultural & Industrial Vocational High School, Taoyuan, Taiwan. His areas of research interests include digital image processing, computer vision, and pattern recognition.

S. W. Chen received the M. Sc. and Ph.D. degrees in computer science and engineering from Michigan State University, East Lansing, Michigan, in 1985 and 1989, respectively. In 1990, he was a researcher in the Advanced Technology Center of the Computer and Communication Laboratories at the Industrial Technology Research Institute, Hsinchu, Taiwan, Republic of China. He is currently a full professor of the Department of Computer Science and Information Engineering at the National Taiwan Normal University, Taipei, Taiwan, Republic of China. He was elected to the grades of IEEE Senior Member and the IET Fellow in 1997 and 2012, respectively. Since 1991, he has been the Director of Image Processing and Computer Vision – Intelligent Transportation Systems (IPCV-ITS) Lab at the University. His areas of research interest include pattern recognition, image processing, computer vision, and Intelligent Transportation Systems.

C. S. Fuh received the BS degree in computer science and information engineering from National Taiwan University, Taipei, Taiwan, in 1983, the MS degree in computer science from the Pennsylvania State University, University Park, PA, in 1987, and the PhD degree in computer science from Harvard University, Cambridge, MA, in 1992. He was with AT&T Bell Laboratories and engaged in performance monitoring of switching networks from 1992 to 1993. He was an associate professor in Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan from 1993 to 2000 and then promoted to a full professor. His current research interests include digital image processing, computer vision, pattern recognition, mathematical morphology, and their applications to defect inspection, industrial automation, digital still camera, digital video camcorder, surveillance camera, and camera module such as color interpolation, auto exposure, auto focus, auto white balance, color calibration, and color management.