Fast and Scalable Approximate Spectral Matching for Higher-Order Graph Matching

Soonyong Park, Sung-Kee Park, and Martial Hebert

Abstract—This paper presents a fast and efficient computational approach to higher-order spectral graph matching. Exploiting the redundancy in a tensor representing the affinity between feature points, we approximate the affinity tensor with the linear combination of Kronecker products between bases and index tensors. The bases and index tensors are highly compressed representation of the approximated affinity tensor, requiring much smaller memory than in previous methods which store the full affinity tensor. We compute the principal eigenvector of the approximated affinity tensor using the small bases and index tensors without explicitly storing the approximated tensor. In order to compensate for the loss of matching accuracy by the approximation, we also adopt and incorporate a marginalization scheme that maps a higher-order tensor to matrix as well as a one-to-one mapping constraint into the eigenvector computation process. The experimental results show that the proposed method is faster and requiring smaller memory than the existing methods with little or no loss of accuracy.

Index Terms—Higher-order graph matching, spectral relaxation, approximation algorithm.

1 Introduction

Establishing consistent correspondences between two or more sets of features is a fundamental problem in computer vision tasks. Those include object recognition [1], object category discovery [2], image retrieval [3], structure from motion [4], camera self-calibration [5]. Most of the tasks start by assuming that feature points were extracted and their correspondences were detected. There are various difficulties associated with this problem: huge dimensionality of the search space, the existence of outliers and occlusions, deformations in scale and location of feature due to viewpoint change, and variation of feature descriptor due to illumination change. It is therefore challenging to detect perfect reliable correspondences.

The correspondence problem can be well defined as graph matching. Given a set of feature points, graph nodes can represent the points and graph edges can encode the relationships between two points. The purpose of graph matching is then to find correspondences between the nodes of two feature graphs such that both the unary information (e.g., feature descriptors) on the nodes and the relation information (e.g., distance or angle) associated with the edges are preserved as much as possible.

Graph matching can be formulated as the optimization problem of a certain cost function that incorporates both geometric relationships between feature points and their descriptors. First-order methods [6], [7] mainly focus on node-to-node (or unary) similarities based on local feature descriptors. However, they could fail to establish reliable correspondences due to image ambiguities such as indistinguishable local appearance or repeated textures. Second-order methods [8], [9], [10] find correspondences by considering how well edge-to-edge (or pairwise) relationships such as distances between feature points are preserved. The pairwise relationships, however, are not invariant to scale change as well as not enough to enforce entire geometric consistency. More recently, the problem of higher-order graph matching has also been studied [11], [12], [13], [14] to overcome the limitation of second-order methods, by considering higher-order relationships between tuples of feature points.

Most recent works [12], [13], [14] on higher-order graph matching typically extend the spectral matching method [9] using pairwise relationships to higher-order relationships (mostly third-order). The higher-order relationships are represented by an affinity tensor and the correspondences are then given by discretizing the principal eigenvector of the tensor. These methods, however, has two problems of scalability. First, the affinity tensor is huge, requiring very large memory to store, and thus it could limit the maximum data sizes, as well as its complete construction is impracticable and unwarrantable for large data. Second, the construction of the affinity tensor as well as the eigenvector computation takes long processing time due to huge size of the tensor.

In this paper, we address the scalability problems of higher-order graph matching inspired by the spectral matching method [9]. We show how to perform fast and efficient higher-order graph matching on a huge affinity tensor by our proposed HOFASM (Higher-Order FAst Spectral

- *Soonyong Park is with the Future IT R&D Center, Samsung Advanced Institute of Technology (SAIT), Republic of Korea.*
  E-mail: soonyong.park@gmail.com
- *Sung-Kee Park is with the Center for Bionics/Biomedical Research Institute, Korea Institute of Science and Technology (KIST), Republic of Korea.*
  E-mail: skee@kist.re.kr
- *Martial Hebert is with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, United States.*
  E-mail: hebert@ri.cmu.edu

graph Matching) algorithm. The main contributions are as follows:

1) We observe that there exists high redundancy in the affinity tensor used for general higher-order graph matching. Based on the observation, we approximate the tensor by the linear combination of the Kronecker product of bases and index tensors which are highly compressed representation of the approximated affinity tensor, requiring much smaller memory than in previous works.
2) We develop an efficient algorithm to compute the principal eigenvector of the approximated affinity tensor by using the compressed representation without explicitly storing the approximated tensor.
3) We adopt a partial marginalization scheme [12] and a one-to-one mapping constraint [15] for the eigenvector computation process. Thus, the proposed algorithm compensates for the loss of matching accuracy by the approximation and produces a reliable performance to large deformation noises and outliers.

Note that the work presented in this paper is an extension of our second-order method introduced in [16] to higher-order graph matching. In [16], we approximated the affinity matrix with the linear combination of Kronecker products between bases and index matrices. In addition, we proposed a power method to compute the principal eigenvector of the approximated affinity matrix. The requirements for the extension are threefold. First, there are many tuples of feature points whose corresponding angles are the same or close each other, such that the affinity tensor has many redundancies. Second, the affinity tensor can be approximated as the linear combination of Kronecker products between bases and index tensors. Third, the tensor-vector multiplication is implementable to compute the principal eigenvector of the approximated affinity tensor.

This paper is organized as follows. Section 2 reviews the previous works on graph matching. Section 3 presents generalized method for higher-order graph matching. Section 4 describes our proposed method of approximating the affinity tensor for higher-order graph matching with few bases and index tensors, and explains our proposed method of computing the principal eigenvector using the tensors. Section 5 provides experimental evaluations and comparisons. Finally, we conclude in Section 6.

2 Related Work

Graph matching has been formulated as an optimization problem. According to the types of objective functions, graph matching algorithms can be categorized into first-order, second-order, and higher-order methods. Furthermore, more recent algorithms including both second-order and higher-order methods are generally based on the Integer Quadratic Programming (IQP) formulation. Since the IQP problem is NP-complete [17], different relaxation approaches have been applied to solve the problem.

First-order methods represent the objective function as a bipartite graph having a form of cost matrix, in which each node represents some measure of similarity between two feature points (e.g., the Euclidean distance between local image descriptors [18]). One of the first of such algorithms is the hungarian method proposed by Kuhn [6]. This method locates the set of one-to-one correspondences in the cost matrix, where the correspondences minimize the overall cost. Scott and Longuet-Higgins [19] solved the one-to-one matching by maximizing the inner product of two matrices, one of which is a Gaussian affinity matrix representing the proximities between feature points and the other is the pairing matrix obtained by singular value decomposition of the affinity matrix. Albarelli *et al.* [20] formulated the matching problem as a non-cooperative game where the set of possible correspondences correspond to game strategies. The matching results in this method correspond to evolutionary stable states. These methods, however, could fail in the presence of image ambiguities such as indistinguishable local appearance or repeated textures.

Second-order methods try to deal with the difficulty of first-order methods, in which the objective function is defined by a matrix representing the affinity between pairs of feature points. The idea is as follows: Given two sets of feature points P and Q where $i \in P$ and $i' \in Q$, if the pair (i, j) is consistent with the pair (i', j'), then the pairwise relationships of (i, j) and (i', j') should have high similarity. The graduated assignment algorithm proposed by Gold and Rangarajan [21] is one of the first of pairwise matching methods solving the IQP problem by the spectral relaxation with power iteration, in which Sinkhorn's bistochastic normalization [15] is employed to satisfy two-way assignment constraints for one-to-one mapping. Maciel and Costeira [22] formulated pairwise matching as a global optimization solution for the IQP problem by maximizing all possible correlation computed with unary information of feature points, in which a concave objective function is built and the discrete search domain is relaxed into its convex hull. Berg *et al.* [8] presented fast-approximate techniques to address NP-complete problem of the IQP whose cost function has terms based on similarity of both unary and pairwise information, in which the minimum of a linear bounding problem is first found and then a locally minimum assignment is detected with a manner of local gradient descent. Leordeanu and Hebert [9] used similar cost function to [8] while they approximated the IQP problem as spectral relaxation, in which the cost function is defined as a matrix representing the geometric affinity between pairs of feature points and the correspondences are then given by the principal eigenvector of the affinity matrix and a corresponding discretization. Cour *et al.* [10] proposed similar spectral matching algorithm to [9], in which they modified the cost function with affine constraints and incorporated one-to-one mapping constraints into power iteration step with bistochastic normalization. Cho *et al.* [23] proposed a random walk view for graph matching, in which they incorporated one-to-one mapping constraints by a reweighting jump scheme. Cho and Lee [24] proposed a progressive framework for graph matching, which combines probabilistic graph progression with graph matching. Using a Bayesian manner, this algorithm recursively re-estimates the most plausible target graphs based on the current graph matching result. Zhou and De la Torre [25] presented the factorized graph matching (FGM) algorithm, in which the affinity matrix is factorized as a Kronecker product of smaller matrices. This method is quite related to our second-order method [16] in the context of avoiding the computation of large and sparse affinity matrix. They utilize the Kronecker product operation to represent the affinity matrix with smaller matrices. The differences between them are as follows. The FGM method factorizes the affinity matrix as a Kronecker product of smaller matrices: source graph's incidence matrix, target graph's incidence matrix, node affinity matrix, and edge affinity matrix. These matrices encode the composition of the whole affinity matrix without an approximation. Our second-order method [16], on the other hand, approximates the affinity matrix with the linear combination of Kronecker products between bases and index matrices. These matrices are highly compressed representation of the approximated affinity matrix, that require smaller memory than FGM. All these second-order methods are limited to the affinity matrix embedding pairwise relationships between feature points with unary information, where the pairwise relationships are rotation-invariant but not scale-invariant as well as affine-invariant.

Recent higher-order methods define the objective function by a tensor representing higher-order affinity between tuples of feature points. The affinity tensor encodes higher-order geometric invariants such as scale and affine invariants. The hyper-graph matching algorithm proposed by Zass and Shashua [11] marginalizes the affinity tensor into a one-dimensional vector and refines the vector by projecting it onto the space of soft assignment vectors. Chertok and Keller [12] extended the hyper-graph matching [11] to higher-order spectral matching, in which the affinity tensor is partially marginalized down to a matrix. The matches are given by soft assignment refinement of the principal eigenvector of the marginalized matrix and a corresponding discretization. Duchenne *et al.* [13] extended the spectral matching method [9] to higher-order relationships by using a multi-dimensional power iteration for tensors. They approximated the IQP problem with the ℓ^1-norm relaxation by the Hadamard product and unit-norm rows normalization. Lee *et al.* [14] generalized their second-order method [23] using the random walk approach to higher-order graph. All these existing higher-order methods, which solve the IQP problem with an affinity tensor as an input, show good performance in accuracy. However, they have the problem of scalability: the affinity tensor is very huge, and thus the construction of the tensor as well as the eigenvector computation takes long processing time. To handle the increased computational complexity and storage, different approximation schemes for the affinity tensor have also been introduced. Zass and Shashua [11] sampled a number of triangles per feature point and only measured affinities between sampled triangles of the source and target images. Duchenne *et al.* [13] sampled a number of triangles per feature point such as [11], and found a number of nearest neighbors of the source image for each of sampled triangles of the target image. Then they computed the tensor value with a truncated Gaussian kernel. Chertok and Keller [12] made the affinity tensor sparse with a randomly sparsifying scheme. In this paper, we solve the scalability problem with the use of both index and bases tensors as well as an efficient power method utilizing the tensors.

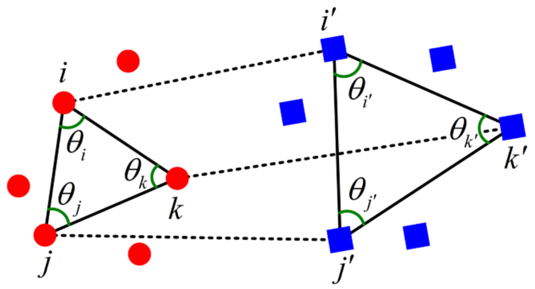


Fig. 1: Higher-order affinity based on tuples of feature points. The affinity can be computed by comparing the angles of triangles. Higher-order matching based on triplets is invariant to scale and affine changes.

3 Generalized Higher-Order Graph Matching

Generalized higher-order matching is an extension of the spectral matching [9] using pair-to-pair comparison. Given two sets of feature points P and Q, each having n_P and n_Q feature points, correspondences are defined as a set C of pairs (or assignments) (i, i'), where $i \in P$ and $i' \in Q$. For each m pairs of assignments $(\omega_1, ..., \omega_m)$, where $\omega_k = (i_k, i'_k)$, there is an affinity $\Omega(\omega_1, ..., \omega_m)$ that measures how compatible the feature points $(i_1, ..., i_m)$ in P with the feature points $(i'_1, ..., i'_m)$ in Q. This m-order affinity Ω can be represented by an m-dimensional tensor T such that

$$T(\omega_1, ..., \omega_m) = \Omega(\omega_1, ..., \omega_m).$$

For simplicity, we will focus from now on third-order affinity $(m = 3)$ but extending to higher-order is straightforward. In order to achieve scale and affinity invariance, we exploit triplets of feature points as shown in Fig. 1. We define the affinity Ω based on the difference in corresponding angles as follow:

$$\Omega(\omega_1, \omega_2, \omega_3) = 4.5 - \frac{1}{3}\left(\frac{(\theta_i-\theta_{i'})^2+(\theta_j-\theta_{j'})^2+(\theta_k-\theta_{k'})^2}{2\sigma_\theta^2}\right). \tag{1}$$

This affinity represents the amount of deformation between three candidate correspondences $\omega_1 = (i, i')$, $\omega_2 = (j, j')$, and $\omega_3 = (k, k')$. Then the affinity tensor T is determined by

$$T(\omega_1, \omega_2, \omega_3) = \begin{cases} \Omega(\omega_1, \omega_2, \omega_3), & \text{if } |\theta_i - \theta_{i'}| < 3\sigma_\theta \text{ and} \\ & \quad |\theta_j - \theta_{j'}| < 3\sigma_\theta \text{ and} \\ & \quad |\theta_k - \theta_{k'}| < 3\sigma_\theta; \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

where σ_θ is a parameter to control the sensitivity of the matching: larger σ_θ allows more deformations in corresponding triangles, and more triplet relationships to have positive scores in T. The tensor T is nonnegative as well as super-symmetric, i.e., invariant under permutations of indices such that

$$T(\omega_1,\omega_2,\omega_3) = T(\omega_1,\omega_3,\omega_2) = T(\omega_2,\omega_1,\omega_3) \\ = T(\omega_2,\omega_3,\omega_1) = T(\omega_3,\omega_1,\omega_2) = T(\omega_3,\omega_2,\omega_1).$$

This means that it is enough to compute $\frac{(n_P n_Q)^m}{m!}$ elements of T instead of computing $(n_P n_Q)^m$ elements.

Given the affinity tensor, we want to find the best cluster of assignments (i,i') that maximizes the matching score:

$$S(x) = \sum_{\omega_1,\omega_2,\omega_3 \in C} T(\omega_1,\omega_2,\omega_3)x(\omega_1)x(\omega_2)x(\omega_3), \quad (3)$$

where the resulting score $S(x)$ is the summation of affinity values associated with all triplets of correspondences in a mapping x. The cluster can then be represented by a binary indicator vector x with an element for each assignment $\omega = (i,i')$, such that $x(\omega) = 1$ if i in P is matched with i' in Q and 0 otherwise. Using the tensor-vector product notation in [26], (3) can be simply represented by

$$S(x) = T \otimes_1 x \otimes_2 x \otimes_3 x, \quad (4)$$

where the index d in the notation $\otimes_d$ represents that we multiply T and x on the d-th dimension of T. Then, the vector x^* maximizing the score $S(x)$ is given by the solution of an Integer Quadratic Programming (IQP):

$$x^* = \arg\max_x S(x). \quad (5)$$

Relaxing the integral constraints on x by interpreting the value of $x(\omega)$ as the association of ω with the best cluster, and requiring the norm of x to be 1, the solution of (5) is given by the principal eigenvector of T associated with the largest eigenvalue. Since T contains only nonnegative elements, the elements of x^* will be in $[0,1]$. This is the relaxed and continuous vector, so we need to binarize x^* to find discrete assignments. The binarization is performed by repeating the following greedy steps:

1) Find $\omega^* = \arg\max_{\omega \in L}(x^*(\omega))$, where L is the set of all tentative correspondences. If $x^*(\omega^*) = 0$, stop the binarization. Otherwise set $x(\omega^*) = 1$ and remove ω^* from L.
2) Remove from L all potential correspondences which conflict with $\omega^* = (i,i')$. These have the form of $(i,*)$ or $(*,i')$.
3) If L is empty, return the solution x. Otherwise go back to step 1.

There is a scalability issue, however, in computing the principal eigenvector of T. Note that T is a three-dimensional tensor whose size is $n_P n_Q \times n_P n_Q \times n_P n_Q$, which means that the number of nonzero elements in T can grow in proportion to n_Q^6 at maximum, assuming $n_P \sim n_Q$. Thus, constructing T explicitly in memory is prohibitive. In the next section, we show how to solve this problem.

4 Proposed Methods

In this section, we propose HOFASM (Higher-Order FAst Spectral graph Matching), a fast, scalable, and accurate approximation method for higher-order graph matching with spectral relaxation. HOFASM consists of two stages.

1) **Affinity Tensor Approximation** Compute the approximated affinity tensor $\hat{T}$.
2) **Eigenvector Computation** Compute the principal eigenvector of $\hat{T}$.

We first describe our proposed method of approximating the affinity tensor, and the fast method for computing the principal eigenvector of the approximated affinity tensor.

4.1 Affinity Tensor Approximation

As we described in previous sections, due to its heavy storage requirement, materializing T explicitly and computing the principal eigenvector of T are impracticable or unwarrantable. To solve this problem, we exploit the redundancy pattern in the affinity tensor. We can observe the pattern by identifying that some areas of the affinity tensor have almost the same elements which we formally describe as follows.

Lemma 1 (Redundancy in Affinity Tensor)**.** *Let $i_1, i_2, j_1, j_2, k_1, k_2$ be the feature points in P, with the corresponding angles following the conditions $\theta_{i_1} = \theta_{i_2}$, $\theta_{j_1} = \theta_{j_2}$, and $\theta_{k_1} = \theta_{k_2}$. The six points are discriminative, with the exception that i_1, j_1, and k_1 can be the same. Let i', j', k' be the arbitrary points in Q, satisfying $i' \neq j' \neq k'$. Let $a_n = (i_n, i')$, $b_n = (j_n, j')$, $c_n = (k_n, k')$, for $n = 1, 2$. Then, the sub-tensor of T involving i_1, j_1, and k_1 is identical to the sub-tensor of T involving i_2, j_2, and k_2. That is, $T(a_1, b_1, c_1) = T(a_2, b_2, c_2)$.*

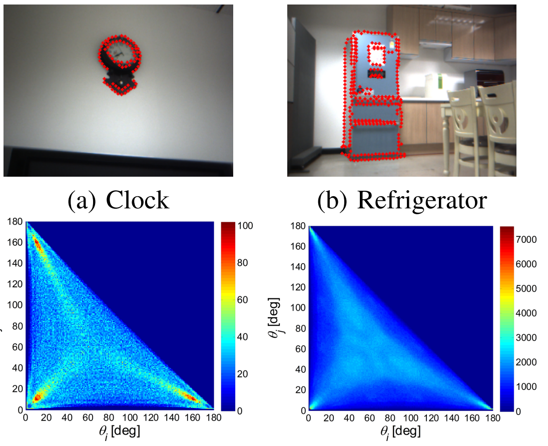
Proof: Let $|\theta_{i_1} - \theta_{i'}| < 3\sigma_\theta$, $|\theta_{j_1} - \theta_{j'}| < 3\sigma_\theta$, and $|\theta_{k_1} - \theta_{k'}| < 3\sigma_\theta$. Then,

$$\begin{aligned} T(a_1,b_1,c_1) &= 4.5 - \tfrac{1}{3}\left(\tfrac{(\theta_{i_1}-\theta_{i'})^2 + (\theta_{j_1}-\theta_{j'})^2 + (\theta_{k_1}-\theta_{k'})^2}{2\sigma_\theta^2}\right) \\ &= 4.5 - \tfrac{1}{3}\left(\tfrac{(\theta_{i_2}-\theta_{i'})^2 + (\theta_{j_2}-\theta_{j'})^2 + (\theta_{k_2}-\theta_{k'})^2}{2\sigma_\theta^2}\right) \\ &= T(a_2,b_2,c_2) \end{aligned}$$

where we used the assumptions $\theta_{i_1} = \theta_{i_2}$, $\theta_{j_1} = \theta_{j_2}$, and $\theta_{k_1} = \theta_{k_2}$ in the second equality. The same analysis applies to the case when $|\theta_{i_1} - \theta_{i'}| \geq 3\sigma_\theta$, $|\theta_{j_1} - \theta_{j'}| \geq 3\sigma_\theta$, or $|\theta_{k_1} - \theta_{k'}| \geq 3\sigma_\theta$. □

Lemma 1 means that if there are two triplets of feature points in P whose corresponding angles are the same, then the sub-tensor of T involving the two triplets are exactly the same. Furthermore, if $|\theta_{i_1} - \theta_{i_2}|$, $|\theta_{j_1} - \theta_{j_2}|$, and $|\theta_{k_1} - \theta_{k_2}|$ are very small, Lemma 1 implies that $T(a_1, b_1, c_1) \sim T(a_2, b_2, c_2)$. Thus, if we have many triplets of feature points whose corresponding angles are the same or close, then we have many redundancies in the tensor T.

To make the observation into something useful, it is necessary to check whether there are many triplets of feature



(a) Clock (b) Refrigerator (c) Angle-triplet distribution of clock (d) Angle-triplet distribution of refrigerator

Fig. 2: [Best viewed in color.] Objects and their angle-triplet distributions. (a-b): Red dots are feature points sampled from extracted edges. (c-d): The color bar indicates the number of triplets with same corresponding angles. We use 1° for the bin width in this figure.

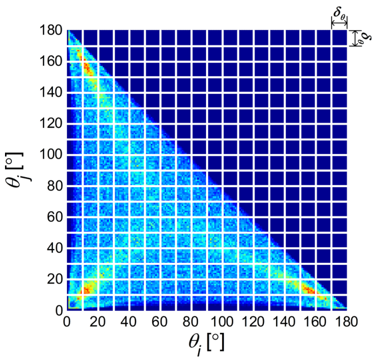


Fig. 3: Angle-triplet distribution of the clock in Fig. 2(a), divided by bins whose width δ_θ is 10. All the angles within a bin are approximated to the center angle of the bin.

points with same corresponding angles in real images. For this purpose, we analyze the angle-triplet distribution from real images in Fig. 2. Note that inside angles of a triangle add up to 180 degrees, such that $\theta_k = 180 - (\theta_i + \theta_j)$ and $\theta_i + \theta_j < 180$. We can see that the distribution is highly symmetric, and many triplets of feature points have same or similar corresponding angles.

Given this observation, our proposed idea is to approximate the elements of T by using approximate angles $\hat{\theta}_i$, $\hat{\theta}_j$, and $\hat{\theta}_k$ instead of exact angles θ_i, θ_j, and θ_k. Let θ_{max} be the maximum corresponding angle of the feature points in P. We divide the total angle range into $K = \lceil \frac{\theta_{max}}{\delta_\theta} \rceil$ bins, where δ_θ is the width of a bin and $\lceil x \rceil$ is the smallest integer not less than x. Then, all the angles within the range from $\delta_\theta \cdot n$ to $\delta_\theta(n+1)$ is approximated by $\hat{\theta}_i = \frac{\delta_\theta(2n+1)}{2}$. That is, θ_i, θ_j, and θ_k are approximated by

$$\hat{\theta}_i = \delta_\theta(\lfloor \frac{\theta_i}{\delta_\theta} \rfloor + \frac{1}{2}),\ \hat{\theta}_j = \delta_\theta(\lfloor \frac{\theta_j}{\delta_\theta} \rfloor + \frac{1}{2}),\ \text{and}$$
$$\hat{\theta}_k = 180 - (\hat{\theta}_i + \hat{\theta}_j) \quad s.t.\ \hat{\theta}_i + \hat{\theta}_j \le 180, \tag{6}$$

where the function $\lfloor x \rfloor$ maps a real number x to the largest integer not greater than x. For example, see Fig. 3 for the angle-triplet distribution of the clock object shown in Fig. 2(a). We divide the total angle range of each axis into 18 bins of width 10. All the angles within a bin are approximated to the center angle of the bin: e.g., all the angles from 20° to 30° are approximated to 25°.

How do these approximations of θ_i, θ_j, θ_k by $\hat{\theta}_i$, $\hat{\theta}_j$, $\hat{\theta}_k$ change the affinity tensor T? Let T_{ijk} be the $n_Q \times n_Q \times n_Q$ sub-tensor of T, containing the rows $(i-1)n_Q + 1 : i \cdot n_Q$, the columns $(j-1)n_Q + 1 : j \cdot n_Q$, and the tubes $(k-1)n_Q + 1 : k \cdot n_Q$. Let H_{ijk} be the $n_P \times n_P \times n_P$ index tensor indicating the location of sub-tensor T_{ijk} in T: $H_{ijk}(a, b, c) = 1$, if $i = a$, $j = b$, $k = c$, and 0 otherwise. If we think of filling T with $n_Q \times n_Q \times n_Q$ blocks, then T_{ijk} is the area of T covered with the block having the block row id i, the block column id j, and the block tube id k. The affinity tensor T can then be represented by the linear combination of the Kronecker product [27] of H_{ijk} and T_{ijk}, for $i \neq j \neq k$:

$$T = \sum_{i=0}^{n_P-1} \sum_{j=0}^{n_P-1} \sum_{k=0}^{n_P-1} H_{ijk} \otimes T_{ijk}, \tag{7}$$

where the Kronecker product of two tensors H_{ijk} and T_{ijk} of dimensions $n_P \times n_P \times n_P$ and $n_Q \times n_Q \times n_Q$, respectively, is a tensor with dimensions $n_P n_Q \times n_P n_Q \times n_P n_Q$. Fig 4 shows an example of constructing the affinity tensor with the index tensor and the sub-tensors.

Recall from (1) and (2) that for three points $i, j, k \in P$ whose corresponding angles are $\theta_i, \theta_j, \theta_k$, the (i', j', k')-th element $T_{ijk}(i', j', k')$ of T_{ijk} is determined by

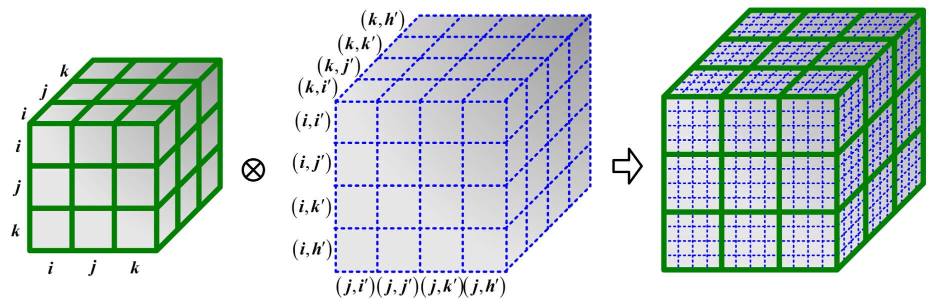
$$\Omega_{ijk}(i', j', k') = 4.5 - \tfrac{1}{3}\left(\tfrac{(\theta_i - \theta_{i'})^2 + (\theta_j - \theta_{j'})^2 + (\theta_k - \theta_{k'})^2}{2\sigma_\theta^2}\right),$$
$$T_{ijk}(i', j', k') = \begin{cases} \Omega_{ijk}(i', j', k'), & \text{if } |\theta_i - \theta_{i'}| < 3\sigma_\theta \text{ and} \\ & |\theta_j - \theta_{j'}| < 3\sigma_\theta \text{ and} \\ & |\theta_k - \theta_{k'}| < 3\sigma_\theta; \\ 0 & \text{otherwise.} \end{cases} \tag{8}$$

where the sub-tensor is not super-symmetric such that

$$T_{ijk}(i', j', k') \neq T_{ijk}(i', k', j') \neq T_{ijk}(j', i', k')$$
$$\neq T_{ijk}(j', k', i') \neq T_{ijk}(k', i', j') \neq T_{ijk}(k', j', i').$$

The proposed idea is then to approximate T_{ijk} with $\hat{T}_{ijk}$ whose (i', j', k')-th element $\hat{T}_{ijk}(i', j', k')$ is determined by

$$\hat{\Omega}_{ijk}(i', j', k') = 4.5 - \tfrac{1}{3}\left(\tfrac{(\hat{\theta}_i - \theta_{i'})^2 + (\hat{\theta}_j - \theta_{j'})^2 + (\hat{\theta}_k - \theta_{k'})^2}{2\sigma_\theta^2}\right),$$
$$\hat{T}_{ijk}(i', j', k') = \begin{cases} \hat{\Omega}_{ijk}(i', j', k'), & \text{if } |\hat{\theta}_i - \theta_{i'}| < 3\sigma_\theta \text{ and} \\ & |\hat{\theta}_j - \theta_{j'}| < 3\sigma_\theta \text{ and} \\ & |\hat{\theta}_k - \theta_{k'}| < 3\sigma_\theta; \\ 0 & \text{otherwise.} \end{cases} \tag{9}$$



(a) Index tensor (b) Sub-tensor (c) Affinity tensor

Fig. 4: [Best viewed in color] The $n_P n_Q \times n_P n_Q \times n_P n_Q$ affinity tensor (c) can be represented by the linear combination of the Kronecker product of the $n_P \times n_P \times n_P$ index tensor (a) and the $n_Q \times n_Q \times n_Q$ sub-tensors (b) ($n_P = 3$ and $n_Q = 4$ in this example). The $n_P^{n_P}$ cubes, bounded by green lines, in the affinity tensor contain corresponding sub-tensors whose locations are indicated by the index tensor. In addition, the 3-tuple (i, j, k) derived from the i-th row, j-th column, and k-th tube of the index tensor denotes a triplet of feature points in P, where $i \neq j \neq k$. The row, column, or tube (i, i') of the sub-tensor represents a pair derived from the two sets of feature points $(i \in P, i' \in Q)$.

We note that the proposed approximation scheme in (9) is only applied to the affinity function given in (1), which uses approximated angles $\hat{\theta}_i$, $\hat{\theta}_j$, and $\hat{\theta}_k$ instead of exact angles θ_i, θ_j, and θ_k. It means that the approximation scheme cannot be applied to affinity functions using pairwise distance d_{ij} between feature points such as $\Omega(\omega_1, \omega_2, \omega_3) = \exp(-\gamma(\|d_{ij} - d_{i'j'}\| + \|d_{ik} - d_{i'k'}\| + \|d_{jk} - d_{j'k'}\|))$.

Since there are $(K = \lceil \frac{\theta_{max}}{\delta_\theta} \rceil)$ distinct $\hat{\theta}_i$s and $\hat{\theta}_j$s, the approximation by (6) and (9) creates $(N = \frac{K(K+1)}{2})$ distinct $n_Q \times n_Q \times n_Q$ sub-tensors of $\hat{T}$. For example, the number of sub-tensors derived from Fig. 3 is 171, in which $K = 18$. Let $B = \{B_n | 0 \leq n \leq N-1\}$ be the set of such tensors. Then, B_n is created using the approximated angles:

$$\hat{\theta}_i = \delta_\theta(k_1 + \frac{1}{2}),\ \hat{\theta}_j = \delta_\theta(k_2 + \frac{1}{2}),\ \hat{\theta}_k = 180 - (\hat{\theta}_i + \hat{\theta}_j)$$
$$s.t.\ k_1 + k_2 \leq K-1 \text{ and } 0 \leq k_1, k_2 \leq K-1, \tag{10}$$

where the index n of B_n is determined by

$$n = k_1 + Kk_2 - \frac{k_2(k_2-1)}{2}. \tag{11}$$

Let H_n be the $n_P \times n_P \times n_P$ index tensor containing only the index of the tensor B_n in $\hat{T}$. That is, H_n's (i, j, k)-th element $H_n(i, j, k)$ is 1 if $\hat{T}_{ijk} = B_n$, and 0 otherwise. Then, $\hat{T}$ can be represented by the linear combination of the Kronecker product of H_n and B_n, for $0 \leq n \leq N-1$:

$$\hat{T} = \sum_{n=0}^{N-1} H_n \otimes B_n. \tag{12}$$

Note that H_n and B_n tensors are highly compressed representation of the $\hat{T}$ tensor. Storing H_n and B_n requires much smaller space than explicitly storing the $\hat{T}$ matrix, as we will see in Section 4.1.1.

A remaining question is, how accurate is the approximation of $T_{ijk}(i', j', k')$ by $\hat{T}_{ijk}(i', j', k')$? Since $\theta_i \sim \hat{\theta}_i$, $\theta_j \sim \hat{\theta}_j$, and $\theta_k \sim \hat{\theta}_k$, the tensor elements $T_{ijk}(i', j', k')$ and $\hat{T}_{ijk}(i', j', k')$ are expected to be close each other. We provide the bound of the approximation as follows.

Lemma 2 (Bound of Approximation)**.** *Assume* $|\theta_i - \theta_{i'}| < 3\sigma_\theta - \frac{\delta_\theta}{2}$, $|\theta_j - \theta_{j'}| < 3\sigma_\theta - \frac{\delta_\theta}{2}$, *and* $|\theta_k - \theta_{k'}| < 3\sigma_\theta - \frac{\delta_\theta}{2}$. *Then,* $|T_{ijk}(i', j', k') - \hat{T}_{ijk}(i', j', k')| \leq \frac{1.5\delta_\theta}{\sigma_\theta}$.

Proof: $|T_{ijk}(i', j', k') - \hat{T}_{ijk}(i', j', k')|$

$$= \left| \left(4.5 - \frac{1}{3}\left(\frac{(\theta_i-\theta_{i'})^2+(\theta_j-\theta_{j'})^2+(\theta_k-\theta_{k'})^2}{2\sigma_\theta^2}\right)\right) - \left(4.5 - \frac{1}{3}\left(\frac{(\hat{\theta}_i-\theta_{i'})^2+(\hat{\theta}_j-\theta_{j'})^2+(\hat{\theta}_k-\theta_{k'})^2}{2\sigma_\theta^2}\right)\right) \right|$$
$$= \left| \frac{(\theta_i+\hat{\theta}_i-2\theta_{i'})(\theta_i-\hat{\theta}_i)}{6\sigma_\theta{}^2} + \frac{(\theta_j+\hat{\theta}_j-2\theta_{j'})(\theta_j-\hat{\theta}_j)}{6\sigma_\theta{}^2} + \frac{(\theta_k+\hat{\theta}_k-2\theta_{k'})(\theta_k-\hat{\theta}_k)}{6\sigma_\theta{}^2} \right| \leq \frac{18\sigma_\theta \frac{\delta_\theta}{2}}{6\sigma_\theta{}^2} = \frac{1.5\delta_\theta}{\sigma_\theta},$$

where the inequality is derived from the facts that $|\theta_i - \hat{\theta}_i| \leq \frac{\delta_\theta}{2}$, $|\theta_j - \hat{\theta}_j| \leq \frac{\delta_\theta}{2}$, *and* $|\theta_k - \hat{\theta}_k| \leq \frac{\delta_\theta}{2}$ *which follow from the definitions of* $\hat{\theta}_i$, $\hat{\theta}_j$, *and* $\hat{\theta}_k$ *in* (6). □

Lemma 2 says that the approximated tensor does not differ much from the original tensor, under the mild assumption. We verify the accuracy of the approximation with the experiments in Section 5.

4.1.1 Storage Estimation

In this section, we estimate the required memory in the generalized higher-order graph matching and in our approximation. From the definitions of T in (2) and T_{ijk} in (8), the tensor T is super-symmetric, but its sub-tensor T_{ijk} is not super-symmetric. Thus, the maximum number of nonzero elements inside a block is given by $\prod_{i=0}^{m-1}(n_Q - i) = n_Q(n_Q-1)(n_Q-2)$, where $m = 3$. Applying the same analysis to T and saving only its upper-diagonal elements, the maximum number of nonzero elements in the tensor T to store in memory is given by

$$\text{Maximum nonzeros in } T = \frac{n_P(n_P-1)(n_P-2)}{6} n_Q(n_Q-1)(n_Q-2).$$

Let ρ be the density of the tensor T, meaning the number of edges divided by the maximum possible number of edges. Assuming $n_P \sim n_Q$, it follows that

$$\begin{aligned}&\text{Nonzeros in } T\\ &= \rho\frac{n_P(n_P-1)(n_P-2)}{6}n_Q(n_Q-1)(n_Q-2) \propto O(n_Q^6).\end{aligned}$$

In contrast, storing $\hat{T}$ requires ($N = \frac{K(K+1)}{2}$) distinct bases tensors B_n with dimensions $n_Q \times n_Q \times n_Q$, and index tensors H_n with dimensions $n_P \times n_P \times n_P$, for $0 \le n \le N-1$. Notice that the number of nonzero elements in all the index tensors, in total, is at most n_P^3. Thus, the maximum number of nonzero elements to store $\hat{T}$ is given by

$$\begin{aligned}&\text{Maximum nonzeros to store } \hat{T}\\ &= Nn_Q(n_Q-1)(n_Q-2) + n_P^3.\end{aligned}$$

Assuming the $\hat{T}$ has the same density as T, and $n_P \sim n_Q$,

$$\begin{aligned}&\text{Nonzeros to store } \hat{T}\\ &\le \rho N n_Q(n_Q-1)(n_Q-2) + n_P^3 \propto O(n_Q^3).\end{aligned}$$

Thus, the cubic growth rate of our approximation is much smaller than the sextic growth rate of the generalized higher-order graph matching. We verify this growth rate in Section 5.1.2.

4.2 Eigenvector Computation

The standard method to compute the principal eigenvector of a matrix is called the power method [28]. In the power method, we multiply the matrix with a randomly initialized vector repeatedly until the normalized vector converges. The converged vector is the principal eigenvector of the matrix. If the matrix is sparse and the number of its nonzero elements is n_Z, each iteration of the power method requires $O(n_Z)$ operations. Extending the power method for a matrix to a tensor is straightforward.

To find the maximum score of the higher-order cost function in (3), we apply a generalization of the power method proposed in [29]. This algorithm was also recently used in [13]. While this method is not guaranteed to get the global maximum, it converges to a local maximum for tensors that result in convex functions of x [26]. Nevertheless, it almost always provides a good solution.

A naive method to compute the principal eigenvector of $\hat{T}$ is to explicitly materialize $\hat{T}$ and run a standard power method to compute the principal eigenvector. However, such method lacks scalability since the materialized tensor $\hat{T}$ is very large. Our proposed idea, called bases power method, is to use the compressed representation of $\hat{T}$ for the power method. Recall that we expressed $\hat{T}$ using the linear combination of Kronecker product of H_n and B_n tensors in (12). The key observation is that we can perform the power method on $\hat{T}$ using H_n and B_n. Applying the power method on $\hat{T}$ requires the tensor-vector multiplication $\hat{T} \otimes_1 v \otimes_2 v \otimes_3 v$ for a vector v. Since $\hat{T} = \sum_{n=0}^{N-1} H_n \otimes B_n$, we can instead compute

$$\begin{aligned}\hat{T}v &= \left(\sum_{n=0}^{N-1} H_n \otimes B_n\right) \otimes_1 v \otimes_2 v \otimes_3 v\\ &= \sum_{n=0}^{N-1} (H_n \otimes B_n) \otimes_1 v \otimes_2 v \otimes_3 v\end{aligned}$$

for computing $\hat{T} \otimes_1 v \otimes_2 v \otimes_3 v$.

Algorithm 1: Bases Power Method

```
Input : Bases tensors $B_n$ and index tensors $H_n$,
        for $0 \le n \le N-1$
Output: Principal eigenvector $x$ of $\hat{T}$
1  begin
2      $v^{(0)} \leftarrow$ random vector with $||v^{(0)}|| = 1$;
3      $q \leftarrow 0$;
4      while $v^{(q)}$ does not converge do
5          $q \leftarrow q+1$;
6          $z \leftarrow 0$;
7          for n=0...N-1 do
8              $\hat{T}_n \leftarrow H_n \otimes B_n$;
9              $z \leftarrow z + \hat{T}_n \otimes_1 v^{(q-1)} \otimes_2 v^{(q-1)} \otimes_3 v^{(q-1)}$;
10         end
11         $v^{(q)} \leftarrow z/||z||$;
12     end
13     $x \leftarrow v^{(q)}$;
14 end
```

Algorithm 1 describes our proposed bases power method. In line 2, a random vector with norm 1 is initialized. Then, the vector is multiplied with the tensor $\hat{T}$ repeatedly until convergence, in line 4 to 12. Note that $\hat{T}$ is never explicitly constructed: only the bases tensors B_n and index tensors H_n are used to compute the tensor-vector multiplication. We also note that the H_n and B_n tensors are accessed sequentially (not randomly) in line 7-10, which makes the algorithm suitable for a parallel implementation on GPUs [30].

4.2.1 *Accuracy Compensation*

In this section, we introduce an approach to compensate for the loss of matching accuracy that may be caused by the approximation method proposed in Section 4.1. The approach incorporates partial marginalization scheme that maps the affinity tensor to a matrix and a one-to-one mapping constraint into the bases power method.

The affinity tensor is nonnegative and super-symmetric, and we want to obtain its principal eigenvector. In the second-order graph matching case [9], the solution is given by the eigenvector of the affinity matrix associated with the largest eigenvalue according to the Rayleigh quotient theorem [31]. Since the affinity matrix contains only nonnegative elements, the elements of its principal eigenvector is in $[0, 1]$ by Perron-Frobenius theorem [32]. The principal eigenvector can be numerically computed by the power method, which converges to the global maximum. However,

Algorithm 2: Bases Power Method with Marginalization and Bistochastic Normalization

```
Input  : Bases tensors $B_n$ and index tensors $H_n$,
         for $0 \le n \le N-1$
Output: Indicator vector $x$
begin
    $v^{(0)} \leftarrow$ random vector with $||v^{(0)}|| = 1$;
    $q \leftarrow 0$;
    while $v^{(q)}$ does not converge do
        $q \leftarrow q + 1$;
        $z \leftarrow 0$;
        for n=0...N-1 do
            $\hat{T}_n \leftarrow H_n \otimes B_n$;
            Marginalize:
            $M_n \leftarrow \sum_i \hat{T}_n(i,j,k)$;
            $z \leftarrow z + M_n v^{(q-1)}$;
        end
        $v^{(q)} \leftarrow \exp(\beta z / \max z)$;
        while $V^{(q)}$ does not converge do
            Normalize across all rows:
            $V^{(q)}(i,j) \leftarrow V^{(q)}(i,j) / \sum_{i=0}^{n_P-1} V^{(q)}(i,j)$;
            Normalize across all columns:
            $V^{(q)}(i,j) \leftarrow V^{(q)}(i,j) / \sum_{j=0}^{n_Q-1} V^{(q)}(i,j)$;
        end
        $v^{(q)} \leftarrow v^{(q)} / ||v^{(q)}||$;
    end
    $x \leftarrow v^{(q)}$;
end
```

we are not able to apply this attribute to tensors: the principal eigenvector of tensors is not the global maximum [33].

Chertok and Keller [12] suggested a method to avoid computing the principal eigenvector of tensors due to its numerical inaccuracy. Instead, they partially marginalize an affinity tensor T down to a matrix M as follows:

$$M = \sum_i T(i,j,k), \tag{13}$$

where the matrix M is symmetric whose principal eigenvector identifies with the principal eigenvector of T. Note that the principal eigenvector of matrices is the global maximum and numerically stable [28].

We now want to impose a one-to-one mapping constraint into the resultant vector from the power method: a feature point in P must correspond to only one feature point in Q and vice versa. This constraint can be satisfied using the bistochastic normalization method proposed by Sinkhorn [15], in which it is proven that a strictly positive square matrix will converge to a strictly positive doubly stochastic matrix by the iterative process of alternately normalizing the rows and columns. The doubly stochastic matrix constrains the norm of each row and column to 1.

Gold and Rangarajan [21] proposed an efficient algorithm that extends Sinkhorn's method [15] to an assignment problem in combinational optimization. Given a positive matrix Z, their method associates an element $V(i,j) \in \{0,1\}$ of an assignment matrix V with each $Z(i,j)$, such that

$$\sum_{i=0}^{n_P-1} V(i,j) = 1 \; \forall i \quad \text{and} \quad \sum_{j=0}^{n_Q-1} V(i,j) = 1 \; \forall j.$$

Then, the solution is given by the matrix V that maximizes the mapping score:

$$S(V) = \sum_{i=0}^{n_P-1} \sum_{j=0}^{n_Q-1} V(i,j) Z(i,j). \tag{14}$$

Algorithm 2 incorporates the marginalization in (13) and the bistochastic normalization in (14) into the bases power method. Note that the matrix V is the row-wise converted replica of the vector v. In line 10, the tensor $\hat{T}$ is marginalized into a matrix. Line 13 makes assignments having high affinity become higher, and makes assignments having small affinity become smaller. In this paper, we empirically set the inflation parameter β as 30. The bistochastic normalization is performed in line 14 to 19, in which the one-to-one mapping constraint is enforced. In the next section, we show that the matching accuracy resulting from Algorithm 2 is better than Algorithm 1.

5 Experimental Results

In this section, we evaluate the performance of the proposed approaches described in Algorithm 1 and Algorithm 2 by conducting experiments on random synthetic points and real images. Following [11] and [12], we first use synthetic points to simulate a real matching problem. It allows us to quantify the matching accuracy, memory size, and running time of the proposed algorithms, and compare our methods to the state-of-the-art methods. Table 1 summarizes the algorithmic setups of previous state-of-the-art methods, including our HOFASM method (Algorithm 1) and its modification HOFASM+MG+BN method (Algorithm 2). We also perform different experiments on the registration of real images to show the applicability of the proposed methods.

5.1 Synthetic points matching

We perform experiments on synthetic points matching to answer the following questions:

Q1 **Accuracy.** What is the accuracy of the proposed methods compared to previous state-of-the-art methods?

Q2 **Memory and Running Time.** How much do the proposed methods reduce the required memory and the running time?

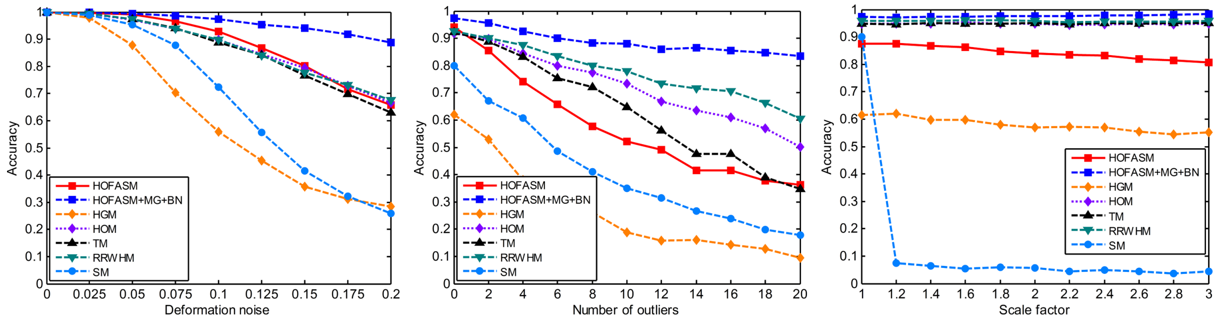
Q3 **Parameter.** What is the trade-off between the accuracy and the running time/memory size on different values of the bin width parameter in the proposed methods?

For the experiments, we randomly sample source points with the normal distribution $N(0,1)$ in the 2D plane. To generate target points, we perturb the source points with the Gaussian noise $N(0,\sigma^2)$ on their positions. We use

TABLE 1: Different graph matching methods

Notation	Affinity	Marginalization	One-to-one mapping constraint
HOFASM	T approximated	none	none
HOFASM + MG[a]+ BN[b]	T approximated	M (matrix)	Bistochastic normalization
HGM [11]	T full	v (vector)	Bistochastic normalization
HOM [12]	T full	M (matrix)	Bistochastic normalization
TM [13]	T full	none	ℓ^1-norm relaxation
RRWHM [14]	T full	none	Personalized page rank
SM [9]	M full (matrix)	none	none

[a] Marginalization
[b] Bistochastic normalization



(a) Accuracy w.r.t noise (b) Accuracy w.r.t outliers (c) Accuracy w.r.t scale

Fig. 5: Accuracy performance with varying the deformation noise, number of outliers, and scale.

$\sigma_\theta = 1$ for all the experiments, and bin width $\delta_\theta = 5$ for the experiments in Section 5.1.1 and Section 5.1.2, and vary the bin width for the experiments in Section 5.1.3.

5.1.1 Accuracy

How accurate are our proposed methods, compared with the state-of-the-art methods? For the evaluation, we measure the matching accuracy on different spatial deformations: noise, outliers, and scale. The matching accuracy is defined as the ratio between the number of correctly matched pairs and the number of total matched pairs. We use the following deformations:

- *Varying noise.* We generate 30 source points. Then, we perturb the source points with the deformation noise σ to create target points, where we vary σ.
- *Varying outliers.* We generate 20 source points and disturb them with the deformation noise $\sigma = 0.1$ to create target points. Then, we add n_O outliers to both the source and target sets, where we vary n_O. The outliers are created with the Gaussian noise $N(0,1)$.
- *Varying scale.* We generate 30 source points. To create target points, we perturb the source points with the deformation noise $\sigma = 0.05$. Then, we multiply the target points by the scale factor δ_S on their positions to induce scale change, where we vary δ_S. Finally, we add $n_O = 5$ outliers to both the source and target sets.

In each deformation, we repeat the same experiment 100 times and show the mean accuracy over the different trials. In Figs. 5a - 5c. We can see that our HOFASM+MG+BN method outperforms the other methods. This also identifies that the marginalization and bistochastic normalization described in Section 4.2.1 can compensate for the loss of matching accuracy in our HOFASM method. The HGM method does not show good performance due to the loss of information by marginalizing the affinity tensor down to a vector. The SM method, second-order method, shows unsatisfied performance when many deformations are added, because many pairwise relationships become similar. We also can see that the higher-order methods using the triplets affinity measure are scale invariant as shown in Fig. 5c, while the second-order method is vulnerable to the scale change.

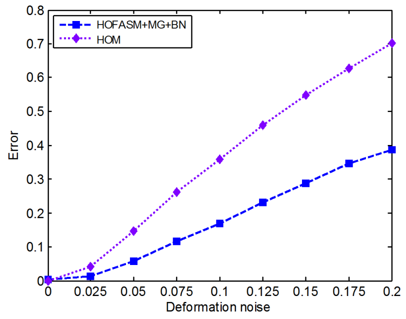
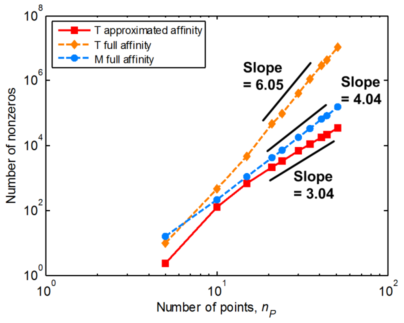
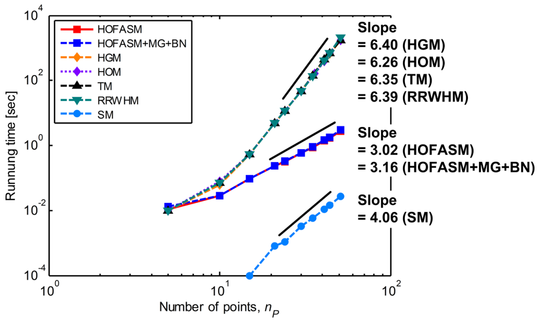


Fig. 6: Perturbation error comparison between HOFASM+MG+BN and HOM [12]. Note that the difference between the ideal indicator vector and the indicator vector from HOFASM+MG+BN is smaller than that of HOM.



(a) Memory vs. data size



(b) Running time vs. data size

Fig. 7: Performances on memory usage and running time with varying number of points. The x and y axes are in log scale. The growth rates 3.04 and 6.05 of the memory sizes of the approximated affinity tensor and the full affinity tensor, respectively, are close to the estimated cubic and sextic growth rates in Section. 4.1.1

.

Our HOFASM+MG+BN method applies the approach of the HOM method [12] to marginalize the approximated affinity tensor down to a matrix, which provides a computationally efficient approximation with HOM. As shown in Fig. 5, HOFASM+MG+BN improves the matching accuracy on different deformations in comparison with HOM. At this point, an interesting question is whether the improvement is due to the approximation of the angles such as (6). Let v^* be the principal eigenvector of the marginalized matrix M^* of the ideal affinity tensor T^* in which only correct correspondences form triplet links. Let x^* be the indicator vector obtained from v^* by the bistochastic normalization. In a similar way, let $\hat{x}$ be the indicator vector derived from the approximated affinity tensor $\hat{T}$ and x be the indicator vector derived from the original affinity tensor T which is a slightly perturbed version of T^* due to deformations. To show the effect of angle approximation on the matching accuracy, we compare the perturbation error of the indicator vectors: $\|\hat{x}-x^*\|$ for HOFASM+MG+BN and $\|x-x^*\|$ for HOM. We obtain the ideal tensor T^* experimentally, in matching set of points used in the experiment of Fig. 5a. We randomly rotate and translate a set of points, without adding Gaussian noise, to obtain other set. We then measure the mean differences with the indicator vectors resulting from HOFASM+MG+BN and HOM in the experiment of Fig. 5a. Fig. 6 shows that the difference between the ideal indicator vector x^* and the indicator $\hat{x}$ from HOFASM+MG+BN is smaller than that of HOM. This explains that the angle approximation stabilizes the indicator vector derived from $\hat{T}$ in the presence of deformations.

5.1.2 Memory and running time

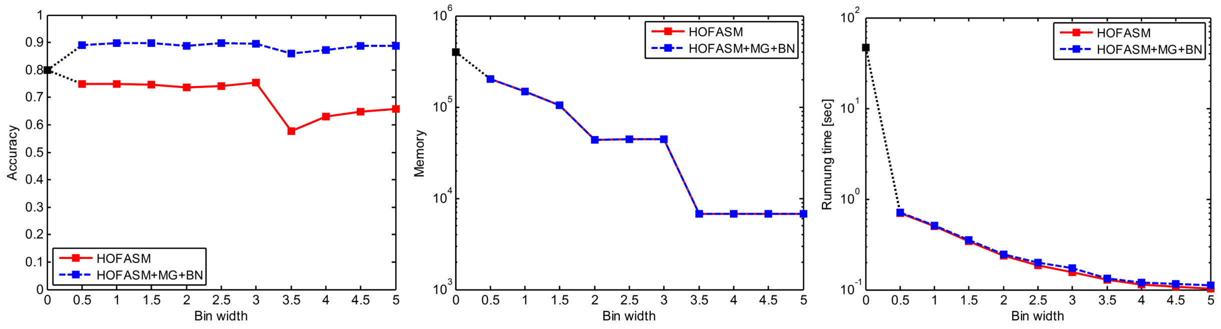
How much do our proposed methods save the required memory and the running time? To compare with the state-of-the-art methods, we measure the memory size and running time as we grow the data sizes. Specifically, we generate n_P source points and disturb them with the Gaussian noise $N(0, 0.1^2)$ to create target points. As we see in Fig. 7a, the required memory in the approximated affinity tensor grows much slowly than in the full affinity tensor as well as the full affinity matrix of the second-order method. The growth rates 3.04 and 6.05 of the memory sizes of the approximated affinity tensor and the full affinity tensor, respectively, are very close to the estimated cubic and sextic growth rates in Section. 4.1.1.

The result in Fig. 7b shows that the running times of HOFASM and HOFASM+MG+BN grow slower than the state-of-the-art methods. However, the growth rate of the running time of each method, except for HOFASM and SM, is larger than their corresponding growth rate of the memory size. The reason is that the power method used in each method, except for HOFASM and SM, requires additional operations such as marginalization and one-to-one mapping enforcement, while the power methods of HOFASM and SM still need similar number of floating point operations to the memory size.

5.1.3 Parameter

How does the bin width parameter affect the performance of the proposed methods? For the evaluation, we measure the matching accuracy, the required memory, and the running time with varying the bin width. We generate 30 source points and perturb them with the Gaussian noise $N(0, 0.2^2)$ to create target points. Fig. 8 shows the trade-off results with regard to the bin width parameter. The result on the bin width 0 is from the HOM method [12], and all other results are from the the proposed methods. Notice that the results from HOFASM and HOFASM+MG+BN in Fig. 8b is exactly the same each other, since they use identical approximated affinity tensor.

We see that increasing the bin width slightly decrease the accuracy, while decreasing the memory requirement dramatically. However, their changing patterns are similar each other. In Fig. 8a, the accuracy decreasing rate of HOFASM between bin width 3 and 3.5 is relatively large. We can see similar result in Fig. 8b. The large accuracy decreasing rate of HOFASM stems from the corresponding memory decreasing rate. On the other hand, the corresponding accuracy decreasing rate of HOFASM+MG+BN



(a) Accuracy (b) Memory size (c) Running time

Fig. 8: The effect of the bin width parameter on the performances of HOFASM and HOFASM+MG+BN. The result on the bin width 0 is from the HOM method [12], and all other results are from the HOFASM and HOFASM+MG+BN methods.

is relatively small since the marginalization and bistochastic normalization are still working on compensation for the loss of matching accuracy. The running time also decreases as the bin width increase, but not as quickly as the rate of the memory size decrease since the number of floating point operations in the power methods of HOFASM and HOFASM+MG+BN does not change much.

As we described in Section 5.1.1, thanks to the angle approximation, the HOFASM+MG+BN method outperforms the HOM method. We can see the effect of angle approximation on the matching accuracy again in Fig. 8a, in which HOFASM+MG+BN improves over HOM as the bin width goes over 0.5. Since the HOM method uses full affinity tensor, its performances on memory use and running time are relatively low as shown in Figs. 8b - 8c, compared to HOFASM+MG+BN.

5.2 Image matching

We perform two kinds of feature matching experiments. We first analyze the matching of the $House$ image sequence. Next, we experiment with the matching of natural images. We use $\sigma_\theta = 1$ for all the following experiments, and bin width $\delta_\theta = 5$ for the experiments in Section 5.2.1, and two bin widths $\delta_\theta = 5$ and $\delta_\theta = 0.5$ for the experiments in Section 5.2.2.

5.2.1 Sequence matching

The CMU $House$ dataset[1] is widely used to evaluate the performance of different matching methods. This sequence of image frames is taken from the same object. The scale is always approximately the same, but the point of view varies along the sequence. Thus the deformation between image frames is related to affine transformation. Then the lager the sequence gap between the frames is, the larger the relative deformation as well as the more difficult to match correctly it is. In order to construct ground truth data, 30 feature points are manually labeled and tracked over all image frames. We match the first image frame to the frames succeeding it. Figs. 9a-d show a comparative example indicating correspondences founded by each method. Fig. 9e depicts the overall accuracy performance comparison. Our HOFASM and HOFASM+MG+BN methods show the perfect matching performance in this experiment. We can also see that the second-order method is hard to deal with affine deformations.

5.2.2 Natural image matching

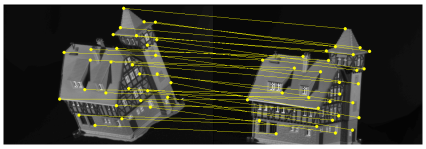
We carry out two feature matching experiments using natural images. Since the number of feature points is large, it is very time consuming to use all the triplets of feature points. So we only sample the n nearest neighbors per point in both source and target images. Then we build $t = \frac{n(n-1)}{2}$ triplets and compute their descriptors such as shown in Fig. 1. We store the triplet descriptors in a kd-tree [34]. For each of the triplets in source image, we find the k nearest neighbors in target image. We take $n = 20$ and $k = 400$ in this experiment. Then we construct the affinity tensor with the k nearest neighbors and start the power method.

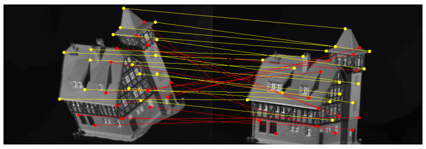
The first experiment tests on 4 pairs of source and target images taken from the Oxford VGG database[2]. Following [12], we first extract feature points in each image and compute their SIFT descriptors [35]. Then, we find correspondences with the lowest SIFT descriptor distances. We remove outliers in the correspondences with RANSAC [36] and collect inliers for candidate matches. In this experiment, the affinity tensor is constructed with these candidate matches. Namely, the candidate matches are used as the set of feature points for higher-order graph matching. Fig. 10 shows the matching image pairs. Each target image is taken by changing the view point from its source image, in which the affine deformation is added relative to the source image. In addition, we downscale the target images by a scale factor of 2.

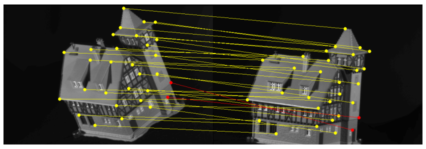
Table 2 describes the number of feature points in each image, and compares the memory size between full affinity tensor and approximated affinity tensor. As shown in this table, approximated affinity tensor requires $3\times$ to $541\times$

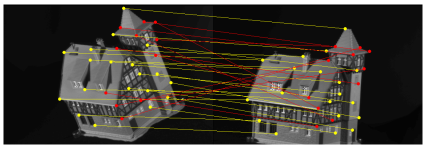
1. http://vasc.ri.cmu.edu//idb/html/motion/house/index.html

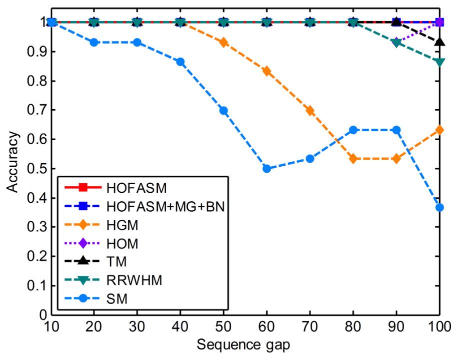
2. http://www.robots.ox.ac.uk/~vgg/data/


(a) HOFASM, HOFASM+MG+BN, HOM, TM: 30 correct matches


(b) HGM: 16 correct matches


(c) RRWHM: 28 correct matches

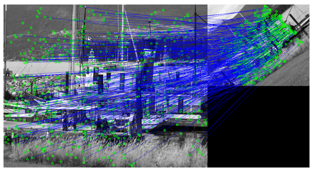

(d) SM: 19 correct matches

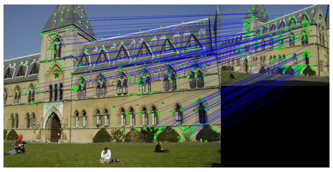


(e) Accuracy w.r.t sequence gap

Fig. 9: [Best viewed in color] Sequence matching results on the *House* dataset. (a-d): Correspondences founded by HOFASM, HOFASM+MG+BN, and the state-of-the-art methods when the sequence gap is 90. Yellow and red lines indicate the correct and incorrect matches, respectively. (e): Accuracy performance curve with varying the sequence gap.

(a) Graffiti


(b) Boat


(c) University library

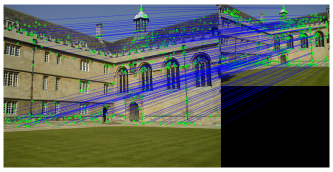

(d) Wadham college

Fig. 10: [Best viewed in color] Matching image pairs from the Oxford VGG database. We downscale the target images by a scale factor of 2. The green circles indicate the feature points used. The number of feature points is described in Table 2. The blue lines denote an example set of candidate matches randomly selected.

TABLE 2: Size of the data and its affinity tensor

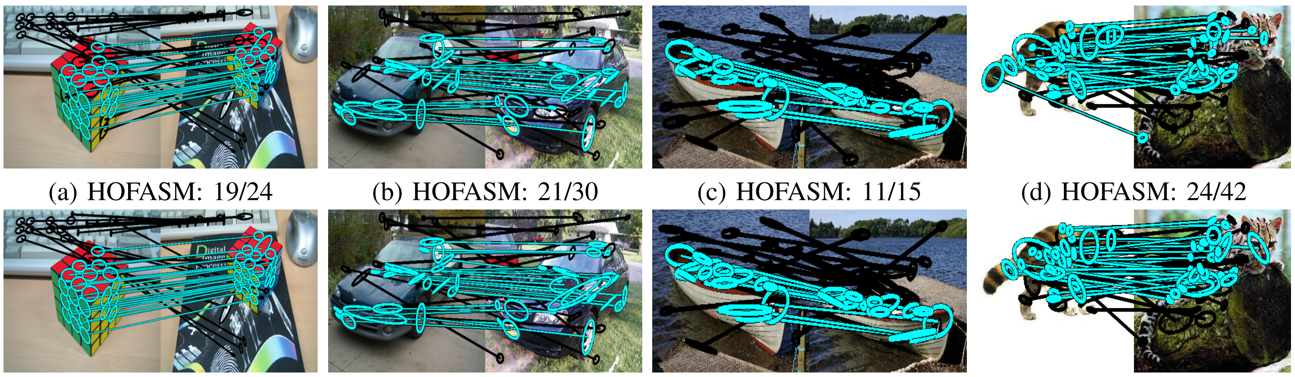
Image	**Number of feature points**	**Number of nonzeros in full affinity tensor**	**Number of nonzeros in approximated affinity tensor**	
			$\delta_\theta = 5$	$\delta_\theta = 0.5$
Graffiti	308	11,775,252	37,270 (316× smaller)	3,115,459 (4× smaller)
Boat	711	36,385,560	67,282 (541× smaller)	6,496,317 (6× smaller)
University library	309	10,429,788	38,007 (274× smaller)	3,180,635 (3× smaller)
Wadham college	491	21,076,190	54,652 (386× smaller)	5,113,338 (4× smaller)

TABLE 3: Performance comparison on matching accuracy

Image	**HOFASM**		**HOFASM+MG+BN**		**HGM**	**HOM**	**TM**	**RRWHM**
	$\delta_\theta = 5$	$\delta_\theta = 0.5$	$\delta_\theta = 5$	$\delta_\theta = 0.5$				
Graffiti	0.05	0.28	0.86	**0.88**	0.16	0.73	0.72	0.26
Boat	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
University library	0.68	0.82	**0.90**	0.89	0.60	0.86	0.86	0.57
Wadham college	0.72	0.78	0.74	0.87	0.73	0.77	**1**	0.80

TABLE 4: Performance comparison on running time in seconds

Image	**HOFASM**		**HOFASM+MG+BN**		**HGM**	**HOM**	**TM**	**RRWHM**
	$\delta_\theta = 5$	$\delta_\theta = 0.5$	$\delta_\theta = 5$	$\delta_\theta = 0.5$				
Graffiti	**1.10**	3.25	1.50	3.60	9.18	10.97	22.31	10.90
Boat	**4.03**	6.98	5.70	6.95	23.35	25.89	26.25	26.42
University library	**1.02**	3.39	1.38	3.71	7.78	8.39	21.42	8.32
Wadham college	**2.08**	4.95	2.55	5.39	13.17	15.69	14.15	16.41



(a) HOFASM: 19/24 (b) HOFASM: 21/30 (c) HOFASM: 11/15 (d) HOFASM: 24/42

(e) HOFASM+MG+BN: 22/24 (f) HOFASM+MG+BN: 23/30 (h) HOFASM+MG+BN: 14/15 (i) HOFASM+MG+BN: 26/42

Fig. 11: [Best viewed in color] Example results of image matching on the dataset used in [14]. The cyan lines denote correct matches. Note that we use 5° for the bin width in this figure.

TABLE 5: Average matching accuracy on the dataset used in [14] (30 pairs)

Method	**HOFASM**		**HOFASM+MG+BN**		**HGM**	**HOM**	**TM**	**RRWHM**
	$\delta_\theta = 5$	$\delta_\theta = 0.5$	$\delta_\theta = 5$	$\delta_\theta = 0.5$				
Avg. accuracy (%)	46.45	47.39	53.54	55.62	45.87	50.63	48.09	50.02

smaller memory than full affinity tensor. The comparison on accuracy performance is presented in Table 3. The results show that our proposed methods are able to match real images with little or no loss of accuracy compared with the state-of-the-art methods. We can also see that the marginalization and bistochastic normalization applied in HOFASM+MG+BN are able to improve the matching accuracy of HOFASM. Table 4 compares the performance on running time in seconds. HOFASM is 6× to 22× faster than the state-of-the-art methods. HOFASM+MG+BN is slightly slower than HOFASM because the additional operations for accuracy compensation.

The second experiment tests on an image dataset[3] used in [14]. This dataset consists of 30 image pairs that are mostly collected from Caltech-101 and MSRC datasets. For each image pair, this dataset provides detected MSER features [37], initial matches, and manually labeled ground truth feature pairs. Following [14], we construct the affinity tensor using the given initial matches. Some example results from HOFASM and HOFASM+MG+BN are shown in Fig. 11. Table 5 compares the performance on average matching accuracy. In this experiment result, we can verify again that our proposed methods are able to match real images with little or no loss of accuracy compared with the others.

6 Conclusion

In this paper, we propose HOFASM (Higher-Order FAst Spectral graph Matching), an approximate higher-order spectral graph matching algorithm for correspondence problems with large feature sets whose affinity tensor contains huge number of nonzero elements. The main contributions are the followings.

1) **Approximation.** Exploiting the high redundancy in the affinity tensor used for the state-of-the-art higher-order graph matching methods, we propose an accurate approximation of the tensor by the linear combination of Kronecker products between bases and index tensors which require much smaller memory than in previous works.
2) **Fast Algorithm.** Using the bases and index tensor, we develop an efficient algorithm to compute the principal eigenvector of the approximated tensor.
3) **Accuracy Compensation.** Applying the marginalization and bistochastic normalization, we compensate for the loss of matching accuracy by the approximation.
4) **Experiments.** Extensive experiments show that our proposed methods are faster, and require smaller memory than the state-of-the-art methods, with little or no loss of accuracy.

3. http://cv.snu.ac.kr/research/~RRWHM/

References

[1] M. Brown and D. G. Lowe, "Recognising panoramas," in *Internat. Conf. on Computer Vision (ICCV)*, 2003, pp. 1218–1227.

[2] M. Leordeanu, M. Hebert, and R. Sukthankar, "Beyond local appearance: Category recognition from pairwise interactions of simple features," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2007.

[3] R. C. Veltkamp and M. Tanase, "Content-based image retrieval systems: A survey," Tech. Rep., 2000.

[4] G. J. Brostow, J. Shotton, J. Fauqueur, and R. Cipolla, "Segmentation and recognition using structure from motion point clouds," in *European Conference on Computer Vision (ECCV)*, 2008, pp. 44–57.

[5] E. E. Hemayed, "A survey of camera self-calibration," in *IEEE Conference on Advanced Video and Signal Based Surveillance*, 2003, pp. 351–357.

[6] H. W. Kuhn, "The hungarian method for the assignment problem," in *Naval Research Logistics Quarterly*, vol. 2, 1955, pp. 83–97.

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

[8] A. C. Berg, T. L. Berg, and J. Malik, "Shape matching and object recognition using low distortion correspondences," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 26–33.

[9] M. Leordeanu and M. Hebert, "A spectral technique for correspondence problems using pairwise constraints," in *Internat. Conf. on Computer Vision (ICCV)*, 2005, pp. 1482–1489.

[10] T. Cour, P. Srinivasan, and J. Shi, "Balanced graph matching," in *Conf. Neural Information Processing Systems (NIPS)*, 2006, pp. 313–320.

[11] R. Zass and A. Shashua, "Probabilistic graph and hypergraph matching," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2008.

[12] M. Chertok and Y. Keller, "Efficient high order matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 2205–2215, 2010.

[13] O. Duchenne, F. Bach, I.-S. Kweon, and J. Ponce, "A tensor-based algorithm for high-order graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, pp. 2383–2395, 2011.

[14] J. Lee, M. Cho, and K. M. Lee, "Hyper-graph matching via reweighted random walks," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 1633–1640.

[15] R. Sinkhorn, "A relationship between arbitrary positive matrices and doubly stochastic matrices," *The Annals of Mathematical Statistics*, vol. 35, no. 2, pp. 876–879, 1964.

[16] U Kang, M. Hebert, and S. Park, "Fast and scalable approximate spectral graph matching for correspondence problems," *Information Sciences*, vol. 220, no. 20, pp. 306–318, 2013.

[17] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.

[18] K. Mikolajczyk and C. Schmid, "A performance evaluation of local descriptors," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630, 2005.

[19] G. L. Scott and H. C. Longuet-Higgins, "An algorithm for associating the features of two images," in *Biological Sciences*, vol. 244, no. 1309, 1991, pp. 21–26.

[20] A. Albarelli, S. R. Bulo, A. Torsello, and M. Pelillo, "Matching as a non-cooperative game," in *Internat. Conf. on Computer Vision (ICCV)*, 2009, pp. 1319–1326.

[21] S. Gold and A. Rangarajan, "A graduated assignment algorithm for graph matching," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, no. 4, pp. 377–388, 1996.

[22] J. Maciel and J. Costeira, "A global solution to sparse correspondence problems," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 25, no. 2, pp. 187–199, 2003.

[23] M. Cho, J. Lee, and K. M. Lee, "Reweighted random walks for graph matching," in *European Conference on Computer Vision (ECCV)*, 2010, pp. 492–505.

[24] M. Cho and K. M. Lee, "Progressive graph matching: Making a move of graphs via probabilistic voting," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 398–405.

[25] F. Zhou and F. De la Torre, "Factorized graph matching," in *IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2012, pp. 127–134.

[26] E. Kofidis, Phillip, and A. Regalia, "On the best rank-1 approximation of higher-order supersymmetric tensors," *SIAM Journal of Matrix Analysis and Applications*, vol. 23, pp. 863–884, 2002.

[27] C. D. Martin, "Higher-order kronecker products and tensor decompositions," Ph.D. thesis, Cornell University, August 2005.

[28] G. H. Golub and C. F. V. Loan, "Matrix computations," *Johns Hopkins University Press*, 1996.

[29] J. V. Lieven De Lathauwer, Bart De Moor, "On the best rank-1 and rank-(r1,r2,. . .,rn) approximation of higher-order tensors," *SIAM Journal of Matrix Analysis and Applications*, vol. 21, no. 4, pp. 1324–1342, 2000.

[30] J. Krüger and R. Westermann, "Linear algebra operators for GPU implementation of numerical algorithms," *ACM Transactions on Graphics*, vol. 22, pp. 908–916, 2003.

[31] K. Lange, "Numerical analysis for statisticians," *Springer*, 1999.

[32] R. A. Horn and C. R. Johnson, "Matrix analysis," *Cambridge University Press*, 1985.

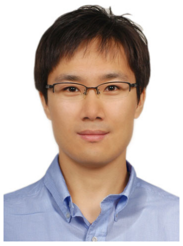
[33] T. G. Kolda, "A counterexample to the possibility of an extension of the eckart-young low-rank approximation theorem for the orthogonal rank tensor decomposition," *SIAM Journal on Matrix Analysis and Applications*, pp. 762–767, 2003.

[34] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Communications of the ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[35] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[36] A. M. L. Peter J. Rousseeuw, *Robust Regression and Outlier Detection*. Wiley, 2003.

[37] J. Matas, O. Chum, U. Martin, and T. Pajdla, "Robust wide baseline stereo from maximally stable extremal regions," in *British Machine Vision Conference (BMVC)*, 2002, pp. 384–393.



Soonyong Park received the B.S. and M.S degrees in mechanical engineering from Kyunghee University, Seoul, Korea, in 2001 and 2003, respectively. He received the Ph.D. degree in electrical and electronic engineering from Yonsei University, Seoul, Korea, in 2010. From 2001 to 2010, he was a graduate student researcher and a postdoctoral fellow with the Cognitive Robotics Center, Korea Institute of Science and Technology (KIST), Seoul, Korea. He was a postdoctoral fellow with the Robotics Institute, Carnegie Mellon University (CMU), in 2011. He is now a research staff member in the Future IT R&D Center at Samsung Advanced Institute of Technology (SAIT). His research interests include computer vision, machine learning, and mobile robot navigation.

Sung-Kee Park is a principal research scientist for Korea Institute of Science and Technology (KIST). He received the B.S. degree and M.S. degree in mechanical design and production engineering from Seoul National University, Seoul, Korea, in 1987 and 1989, respectively. He received the Ph.D. degree (2000) from Korea Advanced Institute of Science and Technology (KAIST), Korea, in the area of computer vision. Since then, he has been studying the field of robot vision and cognitive robotics at KIST. During his period at KIST, he held a visiting position in the Robotics Institute at Carnegie Mellon University (CMU) in 2005, where he did research on object recognition. Now he is working for the center of Bionics at KIST. His recent work has been on object recognition, visual navigation, video surveillance, human-robot interaction, and socially assistive robot.

Martial Hebert is a Professor, Robotics Institute at Carnegie Mellon University (CMU). His research interests include computer vision, especially recognition in images and video data, model building and object recognition from 3D data, and perception for mobile robots and for intelligent vehicles. His group has developed approaches for object recognition and scene analysis in images, 3D point clouds, and video sequences with application to robotics, surveillance, and assistive systems. In the area of machine perception for robotics, his group has developed techniques for people detection, tracking, and prediction, for understanding the environment of ground vehicles from sensor data, for control of AUVs.