Fast Detection of Dense Subgraphs with Iterative Shrinking and Expansion

Hairong Liu, Longin Jan Latecki, *Senior Member, IEEE*, and Shuicheng Yan, *Senior Member, IEEE*

Abstract—In this paper, we propose an efficient algorithm to detect dense subgraphs of a weighted graph. The proposed algorithm, called the shrinking and expansion algorithm (SEA), iterates between two phases, namely, the expansion phase and the shrink phase, until convergence. For a current subgraph, the expansion phase adds the most related vertices based on the average affinity between each vertex and the subgraph. The shrink phase considers all pairwise relations in the current subgraph and filters out vertices whose average affinities to other vertices are smaller than the average affinity of the result subgraph. In both phases, SEA operates on small subgraphs; thus it is very efficient. Significant dense subgraphs are robustly enumerated by running SEA from each vertex of the graph. We evaluate SEA on two different applications: solving correspondence problems and cluster analysis. Both theoretic analysis and experimental results show that SEA is very efficient and robust, especially when there exists a large amount of noise in edge weights.

Index Terms—Dense subgraph, correspondence, point set matching, maximum common subgraph, cluster analysis

1 Introduction

Graph is an important representation for many real-world objects, such as the Internet, the shape of natural objects, and traffic maps. Most of these objects have no corresponding vectorial representations. Even for data in vectorial form, many algorithms are essentially based on graph representations such as graph-based image segmentation [1].

Although edges represent pairwise relations, the graph as a whole can represent very complex relations. A set of vertices that constitute a complex relation usually forms a dense subgraph. Dense subgraphs identify cliques of vertices that are highly related to each other. Such cohesiveness of pairwise relations is unlikely to be produced by accident and is also not easily disturbed by noises and outliers. Thus, a dense subgraph may robustly indicate key patterns underlying the graph. For example, in the World Wide Web, dense subgraphs might be communities or link spam [2]; in telephone call graphs, dense subgraphs might be groups of friends or families. In these situations, the graphs are usually sparse globally, but have many dense subgraphs of different sizes. These dense subgraphs are the natural focal points for studying graph structure and extracting the underlying meaningful patterns.

Dense subgraphs are widely used in many fields. For example, the community structure [3], which appears in social and biological networks, is a kind of dense subgraph. The maximal clique [4], which plays a fundamental role in many graph problems, is also a kind of dense subgraph. In machine learning, the one-class clustering/classification problem [5], which finds small and coherent subsets of points within a given dataset, is in fact meant to find dense subgraphs.

Obviously, directly enumerating all dense subgraphs is time prohibitive. Due to the commonality of the dense subgraphs, many approximate algorithms have been proposed for computing dense subgraphs [4], [6], [7], [8]. Among them the most famous work was done by Motzkin and Straus [9], who proved that solving a dense subgraph (or, equivalently, a maximal clique) problem on an unweighted graph is equivalent to finding the maxima of a quadratic function on the simplex. This result was extended to weighted graphs by Pravan and Pelillo [10]. These algorithms usually have high complexity, both in time and space; at the same time, usually they can only find part of dense subgraphs.

In this paper, we propose the shrinking and expansion algorithm (SEA) that can enumerate significant dense subgraphs with low time and memory complexity. The proposed algorithm is a direct extension of the dominant set method (DS) [10], with significant improvements in effectiveness and speed. The algorithmic flow of SEA is similar to mean shift algorithm [11], a well-known nonparametric feature space analysis technique. Both algorithms can start from any starting point and optimize their target functions along certain increasing trajectories. Mean shift operates directly on the feature space, and its aim is to find the modes of the data, while SEA operates on the affinity graphs, and its aim is to detect dense subgraphs. Since both modes of the vectorial data and dense subgraphs of the affinity graphs reveal the patterns underlying data, SEA can be considered to be a complementary method of mean shift.

The main contributions of this paper are manyfold. 1) We propose the SEA, which can efficiently detect the dense subgraph from any starting point. 2) Based on SEA, we propose an algorithm to enumerate significant dense subgraphs and show its applications in correspondence

- *H. Liu and S. Yan are with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore 119615. E-mail: lhrbss@gmail.com, eleyans@nus.edu.sg.*
- *L.J. Latecki is with Department of Computer and Information Sciences, Temple University, Philadelphia, PA 19122. E-mail: latecki@temple.edu.*

Manuscript received 7 July 2012; revised 3 Nov. 2012; accepted 16 Dec. 2012; published online 4 Jan. 2013.
Recommended for acceptance by T. Cootes.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-2012-07-0514.
Digital Object Identifier no. 10.1109/TPAMI.2013.16.

problems. 3) Similar to mean shift, the SEA provides a natural tool for cluster analysis, and we show its excellent performance in clustering affinity data with large amount of noises and outliers.

The remainder of the paper is organized as follows: We first define graph density in Section 2, then we present the SEA and discuss implementation issues in Section 3. The algorithm for enumerating dense subgraphs in presented in Section 4. In Section 5, the experimental evaluations on two general problems, the correspondence problem and cluster analysis, are demonstrated. In Section 6, concluding remarks are made.

2 Graph Density and Dense Subgraph

In this section, we first define a unique coordinate for each subgraph and then give the definition of graph density.

2.1 Embedding Subgraph in Simplex

A graph G is represented as $G = (V, E, w)$, where $V = \{v_1, \dots, v_n\}$ is a set of n vertices, $E \subseteq V \times V$ is the edge set, and $w : E \to \mathbb{R}_+$ is the (nonnegative) weight function over edge set. Vertices in G correspond to data points, edges represent pairwise relations, and edge-weight reflects the strength of pairwise relation. As is customary, we represent graph G with the corresponding weighted adjacency matrix, A, which is also called the affinity matrix. More specifically, $A = (a_{ij})$ is an $n \times n$ symmetric matrix, where $a_{ij} = w(v_i, v_j)$ if $(v_i, v_j) \in E$, and $a_{ij} = 0$ otherwise. Clearly, if there are no self-loops, all the diagonal elements of A are zeros. In this paper, we only consider graphs with no self-loops.

Let $I = \{1, \dots, n\}$ be the index set of the vertex set V. For any subset $B \subseteq I$, a subgraph G_B of G with the vertex set $V_B = \{v_i | i \in B\}$ is introduced and the corresponding edge set is $E_B = \{(v_i, v_j) \mid (v_i, v_j) \in E, i \in B, j \in B\}$. We denote the set of all subgraphs of graph G as $\mathcal{G}$.

Let $\Delta = \{x \in \mathbb{R}^n : x \geq 0 \text{ and } \|x\|_1 = 1\}$ be a simplex, where $\|x\|_1 = \sum_{i=1}^n |x_i|$ is the ℓ_1 norm of $x = (x_1, x_2, \dots, x_n)^T$. We define an embedding $H : \mathcal{G} \to \Delta$ as $H(G_T) = x$ such that $x_i = \frac{1}{m}$ if $v_i \in V_T$ and $x_i = 0$ otherwise, where m is the number of vertices in G_T. We will denote $H(\mathcal{G})$ as $\Delta_{\mathcal{G}}$. We observe that each coordinate of a point $x \in \Delta_{\mathcal{G}}$ is either zero or $\frac{1}{m}$ for some positive integer m, i.e., $\Delta_{\mathcal{G}} = \{x \in \Delta \mid \exists m > 0\ \forall i = 1, \dots, n\ x_i = \frac{1}{m} \text{ or } x_i = 0\}$.

The indices of all nonzero components of $x \in \Delta$ constitute its *support*, denoted as $\sigma(x) = \{i | x_i \neq 0\}$. According to our definition, each subgraph $G_{\sigma(x)}$ has a unique coordinate $x \in \Delta_{\mathcal{G}}$, and each point $x \in \Delta_{\mathcal{G}}$ represents a unique subgraph of G; thus, in the following text, we will refer to a subgraph and its coordinate in $\Delta_{\mathcal{G}}$ interchangeably.

In particular, each node v_i of graph G can be treated as a one-node subgraph of G. Hence, we identify $H(v_i) \in \Delta$ with a vertex e_i of simplex Δ, i.e., $H(v_i) = e_i$ is a point in simplex Δ with the ith coordinate equal to one and all other coordinates equal to zero.

2.2 Graph Affinity

We define the affinity value between two points $x, y \in \Delta_{\mathcal{G}}$ as

$$a(x, y) = \sum_{i,j} x_i a_{ij} y_j = x^T A y. \tag{1}$$

Note that $a(e_i, e_j) = a_{ij}$, which is consistent with the definition of the weight of graph edge.

Suppose the number of vertices in the subgraphs $G_{\sigma(x)}$ and $G_{\sigma(y)}$ are m_1 and m_2, respectively. Then,

$$a(x, y) = \frac{1}{m_1 m_2} \sum_{i \in \sigma(x), j \in \sigma(y)} a_{ij}. \tag{2}$$

That is, $a(x, y)$ is the average affinity between the subgraph $G_{\sigma(x)}$ and $G_{\sigma(y)}$. As a special case, we obtain the *(sub)graph density*:

$$g(x) = a(x, x) = \sum_{i,j} x_i a_{ij} x_j = x^T A x, \tag{3}$$

which is the average affinity of the subgraph $G_{\sigma(x)}$. Note that this definition is the same as the definition of DS [10]; the only difference is that we restrict $x \in \Delta_{\mathcal{G}}$. Since $g(x)$ reflects the strength of overall internal pairwise relations in the subgraph $G_{\sigma(x)}$, we use it as a criterion to detect a dense subgraph.

2.3 Continuous Relaxation and KKT Points

Our intuition is that a dense subgraph should have large average affinity. Since $\Delta_{\mathcal{G}}$ is a discrete set, it is hard to search for $x \in \Delta_{\mathcal{G}}$ with large $g(x)$. To overcome this problem, we relax x into the continuous space Δ, and solve the following standard quadratic optimization problem (StQP) [12]:

$$\begin{cases} \text{maximize} & g(x) = x^T A x, \\ \text{subject to} & x \in \Delta. \end{cases} \tag{4}$$

That is, we extend the domain of function g to Δ and search local maxima of $g(x), x \in \Delta$ instead. Any local maximizer x^* of $g(x), x \in \Delta$ indicates a potential dense subgraph. If $x^* \in \Delta_{\mathcal{G}}$, then the dense subgraph is $G_{\sigma(x^*)}$; otherwise, we can find the $\tilde{x} \in \Delta_{\mathcal{G}}$ closest to x^* to identify the dense subgraph $G_{\sigma(\tilde{x})}$ (see Section 4.1).

In [10], the properties of local maximizers of (4) have been analyzed. Here, we give a brief summary.

By adding Lagrangian multipliers λ and $\mu_1, \dots, \mu_n$, $\mu_i \geq 0$ for all $i = 1, \dots, n$, we can obtain its Lagrangian function:

$$L(x, \lambda, \mu) = g(x) - \lambda\left(\sum_{i=1}^n x_i - 1\right) + \sum_{i=1}^n \mu_i x_i. \tag{5}$$

Any local maximizer x^* must satisfy the Karush-Kuhn-Tucker (KKT) conditions [13], i.e., the first-order necessary conditions for local optimality. That is,

$$\begin{cases} 2(Ax^*)_i - \lambda + \mu_i = 0, \ i = 1, \dots, n, \\ \sum_{i=1}^n x_i^* \mu_i = 0. \end{cases} \tag{6}$$

Since both x_i^* and μ_i are nonnegative for all $i = 1, \dots, n$, $\sum_{i=1}^n x_i^* \mu_i = 0$ is equivalent to saying that if $x_i^* > 0$, then $\mu_i = 0$. Hence, based on simple algebraic calculations, the KKT conditions can be rewritten as

$$(Ax^*)_i \begin{cases} = \lambda/2, & i \in \sigma(x^*), \\ \leq \lambda/2, & i \notin \sigma(x^*). \end{cases} \tag{7}$$

Any point $x^* \in \Delta$ satisfying the KKT conditions (7) is called a *KKT point*, and all KKT points constitute a set, denoted by

Ψ. Obviously, Ψ contains all local maximizers of the function $g(x) = x^T Ax, x \in \Delta$. Thus, by searching for $x^* \in \Psi$ with large $g(x^*)$ we can obtain potential dense subgraphs.

The size of Ψ is usually very large and it is impossible to obtain the whole set. In real applications, our aim is to enumerate dense subgraphs with large densities, which correspond to true underlying patterns, and the dense subgraphs with small densities are usually meaningless.

Note that for the ith coordinate of the vector Ax^* the following holds:

$$(Ax^*)_i = \sum_j a_{ij} x_j^* = e_i^T Ax^* = a(e_i, x^*), \tag{8}$$

which is the affinity value between x^* and the vertex e_i of simplex Δ.

To lay the theoretical groundwork for our algorithm, we further analyze the properties of the KKT point of (4), especially the relation between the KKT points of G and the KKT points of its subgraphs. We call $r_i(x^*) = a(e_i, x^*)$ the *reward* at e_i. Since $g'(x) = 2Ax$, $r_i(x)$ is in fact half of the partial derivative $\frac{\partial g(x)}{\partial x_i}$. Equation (7) has an obvious geometric meaning, which is summarized in the following theorem.

Theorem 1. *If x^* is a KKT point of (4), then 1) the rewards at the vertices belonging to the subgraph $G_{\sigma(x^*)}$ are identical to $g(x^*)$, and 2) the rewards at the vertices not belonging to the subgraph $G_{\sigma(x^*)}$ are not larger than $g(x^*)$. At the same time, if a point x^* satisfies both "1" and "2," then it is a KKT point of (4).*

Proof. According to (7):

$$g(x^*) = x^{*T} Ax^* = \sum_i x_i^* (Ax^*)_i = \sum_i x_i^* \lambda/2 = \lambda/2. \tag{9}$$

Hence, if x^* is a KKT point of (4), the two conditions in (7) imply the conditions "1" and "2." According to the definition of KKT points, if a point x^* satisfies both "1" and "2," then it is a KKT point. □

Theorem 1 states that if x^* is a KKT point, then the vertices in graph G enter into an equilibrium. In this equilibrium, all the vertices belonging to the subgraph $G_{\sigma(x^*)}$ have the same reward $g(x^*)$, and the rewards at other vertices are not larger than $g(x^*)$. Fig. 1a illustrates such a scenario: x^* is a KKT point, $\sigma(x^*) = \{v_1, v_2, v_3, v_4\}$, they all lie on the sphere $\{y | a(x^*, y) = g(x^*)\}$, and other vertices lie in the space $\{y | a(x^*, y) \leq g(x^*)\}$.

If x^* is a KKT point of G_B, where G_B is a subgraph of G induced by the index set $B \subseteq I$, then it is easy to judge whether x^* is also a KKT point of G or not. In fact, we have the following corollary.

Corollary 1. *If x^* is a KKT point of G_B and $S = \{i | a(x^*, e_i) > g(x^*), i \in I\}$, then x^* is also a KKT point of G if and only if S is empty.*

This corollary is a direct result of Theorem 1; hence its proof is omitted.

Note that $a(x^*, e_i) = g(x^*)$ when $i \in \sigma(x_B^*)$, thus the vertices in S have stronger relations with x^* than the vertices in $\sigma(x_B^*)$. Obviously, all vertices in S violate

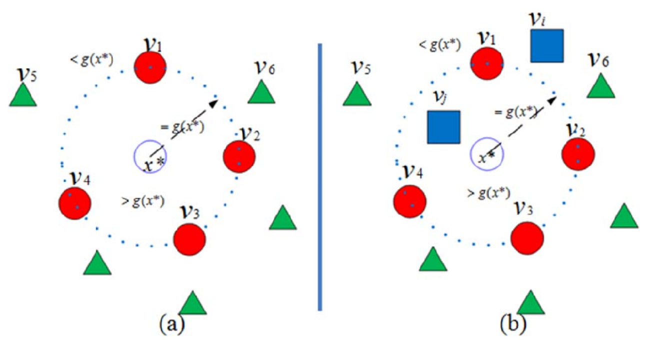


Fig. 1. (a) Geometric explanation of a KKT point. If x^* is a KKT point, all vertices (red points) belonging to $G_{\sigma(x^*)}$ are on the sphere $\{y|a(x^*, y) = g(x^*)\}$ and all other vertices are within the space $\{y|a(x^*, y) \leq g(x^*)\}$. (b) The relations between the KKT points of a graph and the KKT points of its subgraphs. If x^* is a KKT point of the subgraph excluding two square vertices from G, whether it is the KKT point of graph G depends on the rewards at these two square vertices.

the KKT conditions (7). In Fig. 1b, $S = \{v_j\}$, according to Corollary 1, x^* is not a KKT point of the whole graph G.

In many situations, graph G is very large, that is, it has many vertices and edges. However, its dense subgraphs are usually limited to small subsets of vertices of G, that is, $G_{\sigma(x^*)}$ is a small subgraph of G if x^* is a KKT point. In such a case, when computing x^* we can limit the computation on small subgraphs of G, thus greatly reducing the time complexity.

3 Shrinking and Expansion Algorithm

In this section, we present the SEA, which can efficiently locate a KKT point of StQP (4) from an initialization $x(0)$. The characteristic of this algorithm is as follows: It always works on a small subgraph $G_B \subseteq G$, and adaptively updates G_B until a KKT point of G has been located. Each iteration of SEA includes two phases: the shrink phase and the expansion phase. In the shrink phase, a KKT point x_B^* of current subgraph G_B is obtained, and G_B shrinks to its subgraph $G_{\sigma(x_B^*)}$, while in the expansion phase, the vertices that have strong relations with the subgraph G_B are added and form the new subgraph $G_{B'}$. These two phases iterate until no vertex can be added in the expansion phase. In both phases, the density function $g(x)$ always increases, and $g(x)$ is upper bounded; thus the convergence of this algorithm is guaranteed.

3.1 Shrink Phase

In the shrink phase, the aim is to efficiently locate a KKT point of the current subgraph G_B. There are many such algorithms, and we utilize the most popular one, replicator dynamics [14]. Given an initialization $x_B(0)$, the corresponding local solution x_B^* of StQP (4) with $A = A_B$ can be efficiently computed by the discrete-time version of the first-order replicator equation:

$$(x_B)_i(t+1) = (x_B)_i(t) \frac{(A_B x_B(t))_i}{x_B(t)^T A_B x_B(t)}, \ i \in B. \tag{10}$$

It can be observed that Δ_B is invariant under these dynamics, which means that every trajectory starting in Δ_B will remain in Δ_B. Moreover, it has been proven in [14] that, when A_B is symmetric and with nonnegative entries, the objective function $g(x) = x^T A_B x$ strictly increases along a nonconstant trajectory, and its asymptotically stable points are in one-to-one correspondence with strict local solutions of StQP (4).

Note that (10) has a nice property: If $(x_B)_i(t) = 0$, then $(x_B)_i(t+1) = 0$ and $(x_B)_i(t)$ does not affect the computation of $(x_B)_j(t), j \neq i$, which means that during the evolution procedure, replicator dynamic (10) can drop vertices; thus the current graph $G_{\sigma(x_B(t))}$ shrinks. When the KKT point x_B^* is detected, the current graph shrinks to the subgraph $G_{\sigma(x_B^*)}$. x_B^* is the KKT point of the subgraph G_B, but may not be the KKT point of graph G. In the latter case, we must expand the current subgraph, which is described in the following section.

3.2 Expansion Phase

When we get a KKT point x_B^* of the subgraph G_B, we extend it to x^* by adding zeros to the components whose indices are in the set $I \setminus B$. According to Corollary 1, we can judge whether x^* is a KKT point of G by the set S. If S is empty, then x^* is already a KKT point of G and no further step is required; otherwise, we need to find an update vector Δx such that $g(x^* + \Delta x) > g(x^*)$.

When S is not empty, to further increase $g(x)$ we may add the vertices in S to the current subgraph. The reward $r_i(x^*) = a(x^*, e_i)$ provides a natural filtering tool to find out which vertices should be considered, and $g(x^*)$ is a natural threshold. To construct S, we only need to compute the rewards at the neighbors of the current subgraph. The size of S depends on the threshold $g(x^*)$, and as $g(x^*)$ increases, the size of S quickly decreases. When $g(x^*)$ is small (usually appearing in the first few iterations), the size of S may be large. To control the time complexity, we must control the size of current subgraph. Hence, we only add some vertices with relatively larger rewards into the current subgraph. Suppose the index set of selected vertices is Z; we can define a vector γ with

$$\gamma_i = \begin{cases} 0, & i \notin Z, \\ a(x^*, e_i) - g(x^*), & i \in Z. \end{cases} \tag{11}$$

Suppose $s = \sum_i \gamma_i$, $\zeta = \sum_i \gamma_i^2$, and $\omega = \sum_{i,j \in Z} \gamma_i a_{ij} \gamma_j$. When Z is not empty, $s > 0$ and $\zeta > 0$. We update x^* along the direction

$$b = \begin{cases} -x_i^* s, & i \in \sigma(x^*) \\ \gamma_i, & i \notin \sigma(x^*) \end{cases},$$

that is, to decrease the possibilities of vertices belonging to current subgraph and increase the possibilities of vertices with large rewards.

Suppose $g(x^*) = \bar{\lambda}$, then $(Ax^*)_i = \bar{\lambda}$ for $i \in \sigma(x^*)$. Our goal is to maximize the following difference:

$$\begin{aligned} & g(x^* + tb) - g(x^*) \\ &= (1 - ts)^2 \bar{\lambda} + 2(1 - ts) \sum_i (\bar{\lambda} + \gamma_i) t \gamma_i + \sum_{i,j} \gamma_i a_{ij} \gamma_j t^2 - \bar{\lambda} \\ &= 2t(1 - ts)(\zeta + \bar{\lambda} s) - ts(2 - ts)\bar{\lambda} + \omega t^2 \\ &= -(\bar{\lambda} s^2 + 2s\zeta - \omega)t^2 + 2\zeta t. \end{aligned} \tag{12}$$

When $\bar{\lambda} s^2 + 2s\zeta - \omega \leq 0$, the maximal increase from $g(x^*)$ to $g(x^* + tb)$ is obtained at $t^* = \frac{1}{s}$ since $t \leq \frac{1}{s}$, which follows from the fact that $x_i^* \geq 0$ and $x_i^* - t x_i^* s \geq 0$ for $i \in \sigma(x^*)$.

When $\bar{\lambda} s^2 + 2s\zeta - \omega > 0$, the increase from $g(x^*)$ to $g(x^* + tb)$ will reach the maximum at $t^* = \min\{\frac{1}{s}, \frac{\zeta}{\bar{\lambda} s^2 + 2s\zeta - \omega}\}$. Therefore, we set the update vector to

$$\Delta x = t^* b, \tag{13}$$

which is called the *expansion vector*.

In our implementation, we usually set a threshold K_1 on the size of Z and add the vertices with the K_1 largest rewards to Z. As $g(x^*)$ increases, the size of S decreases, and when the size of S is smaller than K_1, we set $Z = S$. Since during the whole procedure of SEA, $g(x)$ is continuously increasing, the final solution is always a KKT point of the whole graph.

The update from x^* to $x^* + \Delta x$ not only increases the value of $g(x)$, but also expands the support $\sigma(x^*)$ to its neighborhood, and forms the new current subgraph $G_{\sigma(x^* + \Delta x)}$. In this procedure, the vertices having strong relations with the current subgraph are added. Since such relations are the composite effects of multiple pairwise relations, they are robust to the noises in pairwise relations; thus, they are very reliable.

3.3 Iterative Shrinking and Expansion

The SEA iterates between the shrink phase and the expansion phase, as summarized in Algorithm 1.

Algorithm 1. Shrinking and Expansion Algorithm (SEA)

```
1: Input: The weighted adjacency matrix $A$ of graph $G$,
   the start point $x(0)$ and the parameter $K_1$.
2: Set $x = x(0)$.
3: repeat
4:     Evolve $x$ toward a KKT point of the subgraph $G_{\sigma(x)}$
       by replicator dynamics (10);
5:     Compute the rewards at the neighbors of $G_{\sigma(x)}$,
       construct $S$ and build the set $Z$, whose size is
       controlled by $K_1$;
6:     if $Z \neq \phi$, compute the expansion vector and update
       $x$; if $Z = \phi$, $x$ is already a KKT point of graph $G$;
7: until $x$ is a KKT point of graph G
8: Output: A KKT point $x^*$.
```

Algorithm 1 is an EM-style procedure; the expansion phase expands the current subgraph to its neighborhood, thus providing a larger lower bound of $g(x)$, which corresponds to a KKT point of current subgraph, while the shrink phase evolves toward this lower bound and guarantees to reach this lower bound. These two steps iterate until a KKT point of the whole graph is reached. In the expansion phase, we control the number of vertices and always add the nearest neighbors to the current subgraph, while in the shrink phase, some vertices may be dropped, and only a very compact cluster of vertices is retained. Thus, our SEA always operates on small subgraphs; hence it is very efficient, both in time and memory. Since in real applications the graph is usually very sparse, the main computational load lies in the shrink phase, which evolves toward a KKT point of the current subgraph. Suppose the number of edges in the subgraph is h and the number of iterations for the replicator equation is t_1, then the time complexity of the shrink phase is $O(t_1 h)$ and the space complexity is $O(h)$. The total time complexity of SEA algorithm is then $O(t_1 t_2 h)$, where t_2 is the number of iterations for the shrink and expansion phases.

4 Enumerating Dense Subgraphs

When a KKT point x^* is obtained, the consequent problem is how to recover the corresponding dense subgraph. If $x^* \in \Delta_{\mathcal{G}}$, then the dense subgraph is $G_{\sigma(x^*)}$. However, usually $x^* \notin \Delta_{\mathcal{G}}$. Thus, we need to find a $\tilde{x} \in \Delta_{\mathcal{G}}$ that approximates x^* to identify the corresponding dense subgraph $G_{\sigma(\tilde{x})}$. There exist many ways to define what approximation means and we select a particularly simple one. We require that $\sigma(\tilde{x}) \subseteq \sigma(x^*)$ and $g(\tilde{x}) = \tilde{x}^T A \tilde{x}$ is large.

The KKT point x^* can be interpreted as having probabilistic meanings: x_i^* is the probability of the dense subgraph containing vertex v_i. Thus, we adopt a greedy but efficient algorithm to calculate $\tilde{x}$ and recover the dense subgraph $G_{\sigma(\tilde{x})}$, which is summarized in Algorithm 2. It adds vertices to the dense subgraph one by one (from large to small) according to the values of the corresponding components in x^* until the density of the subgraph reaches maximum. Note that in step 4 of Algorithm 2, at iteration i, the number of vertices in the set $C \cup \{v\}$ is i since one vertex is added in each iteration. Thus, each component of $\tilde{x}$ is equal to $\frac{1}{i}$ to let $\tilde{x} \in \Delta_G$.

Algorithm 2. Recover corresponding dense subgraph from a KKT point

```
1: Input: The weighted adjacency matrix $A$ of graph $G$,
   the KKT point $x^*$.
2: Sort the components of $x^*$ in descending order, set
   $C = \emptyset$ and $f = -\infty$;
3: for $i = 1 \cdots n$ do
4:     Construct $\tilde{x}$ where $\tilde{x}_j = \frac{1}{i}$ if $j \in C \cup \{v\}$, where $v$ is
       the $i$-th component of $x^*$; otherwise $\tilde{x}_j = 0$.
5:     If $\tilde{x}^T A \tilde{x} > f$, then set $f = \tilde{x}^T A \tilde{x}$ and add $v$ into $C$;
       otherwise, break;
6: end for
7: Output: the point $\tilde{x} \in \Delta_{\mathcal{G}}$.
```

To enumerate important dense subgraphs, we need to tessellate the simplex Δ to generate multiple initializations. Obviously, there are many methods to accomplish this task. In this paper, we utilize a simple and natural one, that is, taking every vertex as a starting point. In total, there are n initializations for a graph with n vertices.

5 Applications

The SEA is a nonparametric graph analysis technique. Although it has a parameter K_1, this parameter only controls the complexity of the algorithm. This algorithm is application independent, and thus can be used to develop algorithms for a wide variety of tasks.

In this paper, we focus on two general tasks, correspondence problem and cluster analysis. In the correspondence problem, all the correct correspondences are compatible, thus, the correct correspondence configurations naturally form significant dense subgraphs of the graph whose vertices represent possible correspondences. In cluster analysis, all the vertices belonging to the basin of attraction of the same KKT point naturally form a cluster, and the dense subgraph corresponding to this KKT point, forms the core of this cluster.

5.1 Correspondence Problem

Establishing feature correspondences between two images is a long standing fundamental problem in computer vision. The general correspondence problem can be described as follows: Given two sets of feature points obtained from two images, P and Q, with n_P and n_Q feature points, respectively, the task is to obtain the correct correspondences between them.

Because of the importance of the correspondence problem, there exist a huge number of papers addressing this problem. Most of them formulate the correspondence problem into the problem of minimizing the energy function of a Markov random field [15], [16], [17], [18], [19], [20]. Since the Markov random field is well studied, this kind of formulation has a solid theoretical foundation. However, this kind of formulation requires that every point in one image must have a correspondence in the other image, which is usually not true in many real-world applications. At the same time, there may be multiple correspondence configurations. Due to these two issues, Markov random field-based formulations cannot lead to satisfactory results in many applications.

Horaud and Skordas [21] first formulated the correspondence problem into the problem of finding maximal cliques in the *correspondence graph*, which naturally eliminates the requirement that every point in one image must have a correspondence in the other image; thus it is inherently more suitable for many applications, especially when the number of outliers is large. Pavan and Pelillo [10] proposed finding the maximal clique by replicator equation. They also utilized this formulation to match free trees [22], and in their recent work [23], they proposed a new iterative method to obtain the DS. Since these methods operate directly on the whole graph, they are inefficient on large graphs.

Each correspondence $c_i \in C$ is a pair $(P_i, Q_{i'})$, where $P_i \in P$ and $Q_{i'} \in Q$. For a correspondence $c_i = (P_i, Q_{i'})$, the *similarity function*, denoted by $f_1(P_i, Q_{i'})$, measures the similarity of the feature points P_i and $Q_{i'}$. Since the correct correspondences generally have similar features, we may only consider a much smaller set $M = \{c | c \in C, f_1(c) > \varepsilon\}$, where ε is a manually set threshold. For two correspondences $c_i = (P_i, Q_{i'})$ and $c_j = (P_j, Q_{j'})$, the *compatibility function*, denoted by $f_2(c_i, c_j)$, measures the compatibility of the correspondences c_i and c_j. Note that both f_1 and f_2 can take various forms, depending on the need of applications.

Based on set M, the correspondence graph G is constructed as follows: Each vertex of G represents a correspondence in M, and the weight of the edge between node i and node j is set to $f_2(i, j)$. The correspondence graph G usually has a large number of vertices (the number of possible correspondences); however, the number of vertices in the dense subgraphs (correct correspondences) is usually very small. Our proposed SEA is especially suitable to this kind of graphs; thus it is a powerful tool for the correspondence problem.

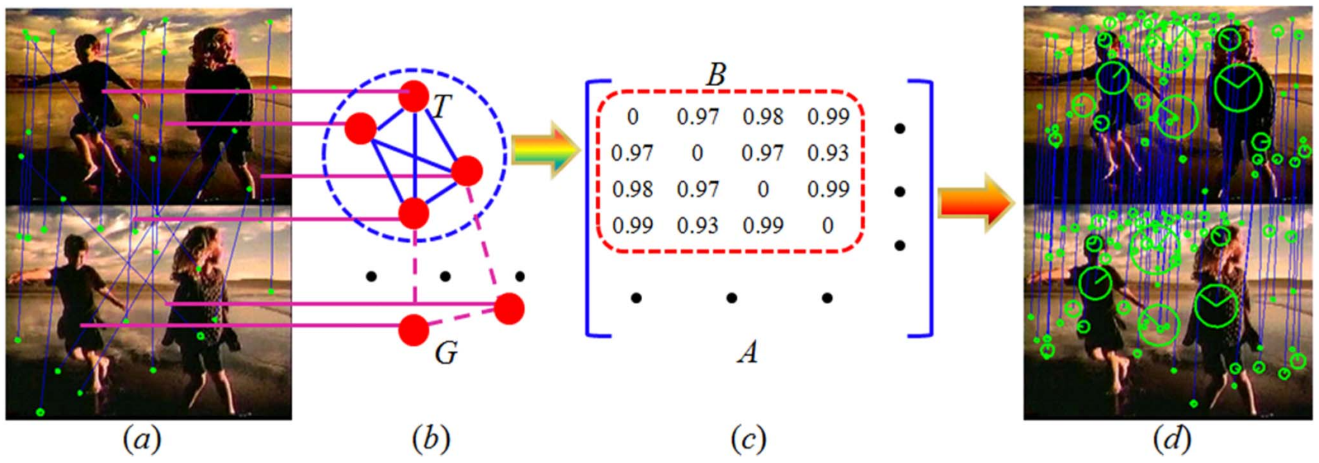


Fig. 2. Illustration of the main idea for correspondence problem. First, find all candidate correspondences shown in (a) by SIFT features (for clarity, only a small subset of the candidate correspondences are shown), and then form the correspondence graph G in (b) and weighed adjacency matrix A in (c). The correct correspondences shown in (d) form a dense subgraph T of G, and thus correspond to the dense block B of A after some permutations.

For simplicity, we regard each vertex as a starting point. Thus, for the graph G with n vertices, we can get n KKT points. Since the correspondence configurations correspond to significant dense subgraphs, we usually select K_2 largest KKT points to further analyze whether they correspond to true correspondence configurations or not. Note that among these K_2 largest KKT points, there may be duplicate ones or overlapping ones that corresponding to the same correspondence configuration, which is easy to verify.

Since a KKT point x^* represents the core of a cluster, we can judge how much a vertex v_i belongs to this cluster by its reward $r_i(x^*) = a(x^*, e_i)$. In the correspondence problem, since each vertex represents a possible correspondence, we can recover the correct correspondences from the vertices with large rewards, and usually we consider K_3 largest rewards vertices. The algorithm to recover the correspondence configuration from a KKT point x^* is summarized in Algorithm 3, which is similar as the spectral method (SM) in [24].

Algorithm 3. Recover correspondences from a KKT point x^*

```
1: Input: The KKT point $x^*$, the parameter $K_3$.
2: Compute the reward $r_i(x^*)$ for each vertex $v_i$ and
   initialize a set $C_1$ to be empty;
3: for $i = 1, \ldots, K_3$ do
4:     Select the $i$th largest reward vertex, and check
       whether the correspondence corresponding to this
       vertex is compatible with all correspondences
       already in $C_1$, that is, the edges connecting this
       vertex to vertices in $C_1$ have large weights. If yes,
       then add this vertex into the set $C_1$;
5: end for
6: Output: The correspondence configuration $C_1$;
```

Based on Algorithms 1 and 3, the overall algorithm for correspondence problem is then described in Algorithm 4. Note that in this application, we do not really need dense subgraphs; thus we skip Algorithm 2 and directly recover correspondences from KKT points. However, in other applications we may need Algorithm 2 to reconstruct dense subgraphs.

Algorithm 4. SEA-based Correspondence Algorithm

```
1: Input: Two feature point sets $P$ and $Q$, the similarity
   function $f_1$ and the compatibility function $f_2$, the
   threshold $\varepsilon$, the parameters $K_1$, $K_2$, and $K_3$.
2: According to $f_1$ and $\varepsilon$, construct the correspondence
   set $M$;
3: According to $M$ and $f_2$, construct the correspondence
   graph $G$;
4: for $i = 1, \ldots, n$ do
5:   Evolve from the vertex $i$, and obtain a KKT point $x^*$
     by Alg. 1;
6: end for
7: Select the $K_2$ largest KKT points and add them into the
   set $S_1$;
8: For each KKT point in $S_1$, recover the correspondence
   configuration $C_1$ by Algorithm 3.
9: Output: All correspondence configurations.
```

The main flowchart is illustrated in Fig. 2. Differently from many existing algorithms, this algorithm can naturally deal with large amount of outliers, and can also simultaneously detect multiple correspondence configurations; thus it is very suitable for many applications.

5.1.1 Point Set Matching

We compare our method with seven methods, namely, the SM [24], the DS [10], the IPFP method [25], the ESS game method (ESS) [23], the tensor matching method (TM) [26], the RRWH method [27], and the HGM method [28]. The codes for five methods, namely, SM, IPFP, TM, RRWH, and HGM, are downloaded from the web, and the codes for other methods were implemented by us. For IPFP, we run it from different initializations, namely, uniform initialization (IPFP), the result of SM (SM+IPFP), the result of DS (DS+IPFP), and the result of ESS (ESS+IPFP). In total, there are 11 methods.

The experimental setting is described as follows: First, generate a dataset Q of 2D model points by randomly selecting n_Q^i inliers in a given region of the plane, then obtain the corresponding inliers in P by independently disturbing the n_Q^i points from Q with white Gaussian noise $N(0, \sigma)$, and then rotate and translate the whole dataset Q with a random rotation and translation. Next, we add n_Q^o and n_P^o outliers in Q and P, respectively, by randomly selecting points in the same region as the inliers from Q and P, respectively, from the same random uniform distribution over the x-y coordinates. The range of the x-y point coordinates in Q is $256\sqrt{n_Q/10}$ to enforce an approximately constant density of 10 points over a 256×256 region, as the number of points varies. The total number of points in Q and P is $n_Q = n_Q^i + n_Q^o$ and $n_P = n_P^i + n_P^o$. The parameter σ controls the level of deformation between two point sets, while n_P^o and n_Q^o control the numbers of outliers in P and Q, respectively.

Since the points themselves are not distinctive, we set $f_1(c_i) = 1$, $i = 1, \ldots, n$, and set $\varepsilon = 0$; thus $M = C$. We rely fully on the geometric consistency information to find the correspondences, and the compatibility function is defined as follows:

$$f_2(c_i, c_j)(s) = \begin{cases} 4.5 - \dfrac{(l_{ij} - s l_{i'j'})^2}{2\sigma_d^2}, & \text{if } |l_{ij} - s l_{i'j'}| < 3\sigma_d, \\ 0, & \text{otherwise,} \end{cases} \tag{14}$$

where l_{ij} is the distance between P_i and P_j, $l_{i'j'}$ is the distance between $Q_{i'}$ and $Q_{j'}$, and s is the scale factor. The parameter σ_d controls the sensitivity of the function value to deformations. The larger the σ_d is, the more deformations we can accommodate, but also the more pairwise relationships between incorrect assignments will get positive values.

For fair comparisons, we first do the experiments on the same scale, that is, we fix $s = 1$. We also keep the sensitivity parameter fixed, $\sigma_d = 5$. All algorithms ran on the same datasets over 30 trials for each value of the varying parameter, and the mean performance curves are plotted. We score the performances of all methods by counting how many matches agree with the ground truths. Fig. 3 illustrates the performance curves of all methods. In the first row, we vary the noise σ from 0 to 20 (in step of 1), and in the second row, we change the number of outliers. Obviously, five methods, namely, SM+IPFP, IPFP, TM, RRWH, and our proposed method, are robust to both noises and outliers. Note that our method does not utilize the matching constraint (one-to-one) in the process of finding dense subgraphs, while the other four methods incorporate the matching constraint (one-to-one) in the optimization procedure. This is why these methods perform better than our method when noise is large. Incorporating correct matching constraints into the optimization procedure can increase the performance; however, it also limits the flexility of the matching methods since the matching constraints may not exist or may be not be known in advance. Our method is nearly not affected by outliers, while SM, ESS, and DS are sensitive to outliers. Note that these four methods operate on the same matrix A and optimize the same quadratic function $f(x) = x^T Ax$. The differences may come from three aspects. First, the constraints on x are different. In SM, the constraint is $|x|_2 = 1$, while in our method, the constraint is $|x|_1 = 1$. Since our aim is to find

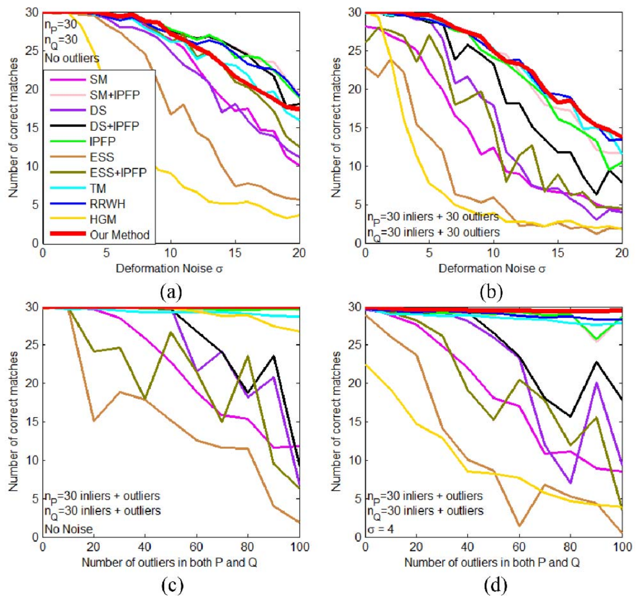


Fig. 3. Performance curves of 11 methods, with each method shown in a distinct color. Top row: The performance curves as the deformation noises vary; in (a) there are no outliers and in (b) there are 30 outliers. Bottom row: The performance curves as the number of outliers change; in (c) there is no deformation noise and in (d) $\sigma = 4$.

dense subgraphs, the ideal maximizers of (4) should be sparse. The constraint $|x|_1 = 1$ usually leads to sparse results, while the constraint $|x|_2 = 1$ usually results in nonsparse results. Second, the strategies for initialization are different. Both ESS and DS utilize only one initialization, that is, the center of the simplex Δ (all vertices have the same probability), while our method adopts a systematic way of initializations. Since there are many local maxima, a systematic way of initializations greatly increases the chance of obtaining correct local maxima. Third, the optimization methods are different. From the same initialization, different optimization methods probably evolve to different local maxima, which are clearly demonstrated by the performance curves of ESS and DS. Obviously, the optimization method adopted by the ESS method is generally inferior to DS. HGM seems to be robust to outliers; however, it is very sensitive to noise. For IPFP, different initializations lead to different results. Both IPFP and SM+IPFP perform well; however, DS+IPFP and ESS+IPFP perform badly.

In Fig. 4a, we fix the deformation parameter $\sigma = 5$, keep the number of inliers and outliers equal, and change the total number of points in P and Q. The performance is measured by matching rate, which is the ratio of the number of obtained correct correspondences to the number of all correct correspondences. As the figure shows, the performances of all algorithms improve as the number of points increases; this is because as the size of the high-order relation increases, it is more robust to noises and outliers. In Fig. 4b, we also test the sensitivity of all methods to parameter σ_d. In this experiment, both the number of inliers and that of outliers are 30. Obviously, our method works remarkably well under different σ_d, while other methods are sensitive to σ_d. We set the number of inliers to 30 and change the number of outliers in both P and Q from 10 to

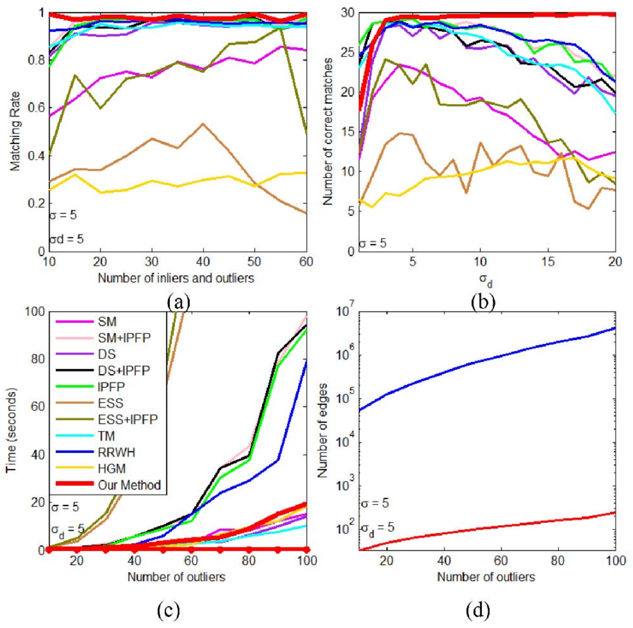


Fig. 4. (a) The performance curves as the number of inliers and outliers change. (b) The sensitivity to the parameter σ_d. (c) The time complexity. (d) The average and total number of edges with large weights in the correspondence graph G.

100. The curves of time cost are shown in Fig. 4c. Since our method runs n times, we also plotted the average time of our method in the red star curve. Note that the average time of our method is the time to obtain one local maximizer of (4). Since SEA always operates on small subgraphs, it is obviously very efficient. When the number of points in each point set is 130, our algorithm runs SEA $n = 16{,}900$ times; however, the total time of our algorithm is still much lower than IPFP. Since the time complexity of our algorithm is linear in the number of initializations, our algorithm can be further speeded up by reducing the number of initializations. ESS and ESS+IPFP are computationally expensive due to the slow convergence of ESS procedure. SM+IPFP, DS+IPFP, and IPFP are also slow since IPFP procedure is not fast. The number of edges whose weights are larger than 2.25 (the maximal weight is 4.5) is plotted in blue in Fig. 4d, and the average number of edges per vertex is plotted in red. Obviously, the number of edges with large weight is several orders of magnitude larger than the number of edges within true correspondences, which is only $30 \times 29/2 = 435$. This means that in such cases, the strong pairwise relations can be easily produced by noises and outliers and are thus not reliable. However, the high-order relation is very reliable since it is the ensemble of all its internal pairwise relations.

We also do the experiments in complex matching situations. For IPFP, we only adopt uniform initialization since it is simple and good enough. In Figs. 5a and 5b, the point sets P and Q have one-to-two matchings, that is, P contains a cluster of points and Q contains two similar copies of this cluster. In Figs. 5c and 5d, the point sets P and Q have two-to-two matchings, that is, there are four similar clusters, two in P and two in Q, respectively. Each cluster has 30 points; thus, there are 60 correct correspondences in the one-to-two matchings and 120 correct correspondences

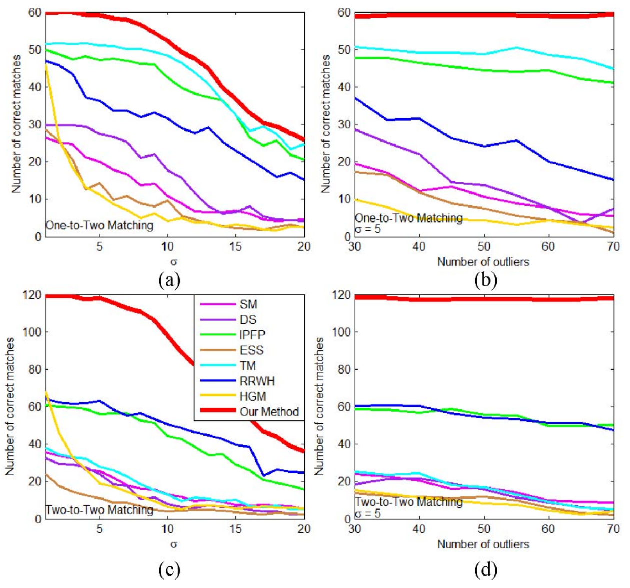


Fig. 5. Performance curves of our proposed method on one-to-two matchings and two-to-two matchings. First row: One-to-two matchings. Second row: Two-to-two matchings.

in the two-to-two matchings. For the three methods which incorporate matching constraints in the optimization process, the adopted matching constraints are reported in Table 1. Obviously, for the one-to-two experiments, the constraint should be one-to-many, and for the two-to-two experiments there should be no constraint. For RRWH, since it inherently assumes a one-to-one matching constraint, we can only use one-to-one constraint. Both IPFP and TM can adopt a one-to-many constraint; however, IPFP does not work if no constraint is utilized; thus, in the two-to-two experiment, we also adopt the one-to-many constraint for IPFP. Fig. 5 clearly demonstrates that our method performs well in complex matching situations. In the one-to-two experiments, both IPFP and TM perform well since the correct constraint is utilized; however, they both perform badly in the two-to-two experiments since the constraint is wrong or no constraint is utilized. RRWH always performs badly since the wrong constraint is used. Theoretically, TM without matching constraints is very similar to SM, and this point is verified by the fact that TM and SM have similar performance curves in the two-to-two experiment.

Finally, we conduct an experiment on multiscale point set matching, which is demonstrated in Fig. 6. The first row shows P and Q. P contains three parts, after adding noise ($\sigma = 5$), one copy of the red part is directly added into Q (red dot) and another copy of the red part is scaled by 0.5, then added into Q (red plus). The green part has been added with noises, scaled by 2, and then added into Q (green star). Both P and Q are then mixed with some

TABLE 1
Matching Constraints Adopted by Different Methods in the Experiment of Fig. 5

	IPFP	TM	RRWH
One-to-Two Exp.	one-to-many	one-to-many	one-to-one
Two-to-Two Exp.	one-to-many	no constraint	one-to-one

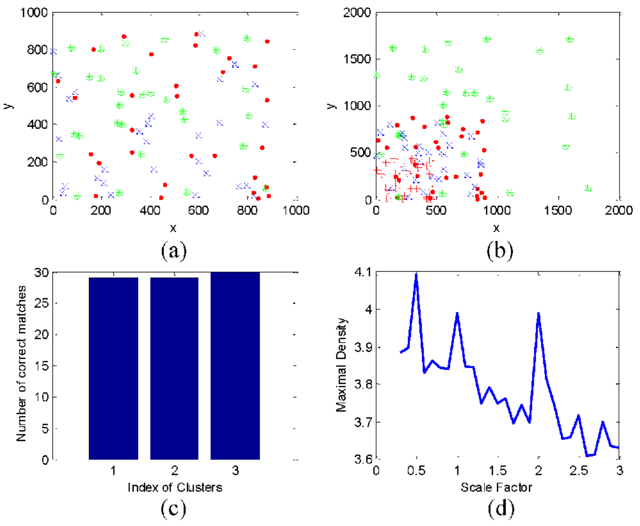


Fig. 6. Correspondence configurations on multiscales. First row: P and Q. Second row: Number of correct matches for three correspondence configurations (left) and the maximal density as scale factor varies.

outliers. We detect the correspondences at different scale factors, ranging from 0.3 to 3 (in step of 0.1). Fig. 6c shows that our method can correctly detect the three correspondence configurations. We also plot the maximal density as a function of the scale factor, which is shown in Fig. 6d. As expected, it correctly indicates at which scale factor there exist correct correspondence configurations. More specifically, at the scale 0.5, 1, and 2, this curve reaches large local maxima.

In all the above experiments, we set K_1 to the number of inliers; $K_2 = 200$, that is, we only examine the 200 largest KKT points; $K_3 = 1{,}000$, that is, for each KKT point, we check the 1,000 vertices most related to it to find compatible correspondences (if the number of all vertices is less than 1,000, then all vertices are checked). Note that the performance is not sensitive to these parameters except when they are set to too small values.

5.1.2 Near Duplicate Image Retrieval

In this section, we show an application of our method on near-duplicate image retrieval, which plays an important role in many real-world multimedia applications. The experiment is conducted on the Columbia database, which contains 150 near-duplicate pairs and 300 non-duplicate images (600 images in total). For fair comparison, we first rank all images using global features, as done in [29], then rerank the images in the top 50 based on the number of correspondences.

We use the SIFT features [30] and utilize the algorithm suggested by Lowe [30] to construct the candidate correspondence set M, that is, a point P_i in the first image is matched to a point $Q_{i'}$ in the second image only if their distance multiplied by a threshold is not greater than the distance of P_i to other points in the second image. In our experiments, this threshold is set to 1.4. The compatibility function is the same as in (14).

For near-duplicate images, there should be dense correspondences in them. Thus, for simplicity we only count the correspondences in the support of the largest dense subgraph, that is, $K_2 = 1$ and K_3 is equal to the size

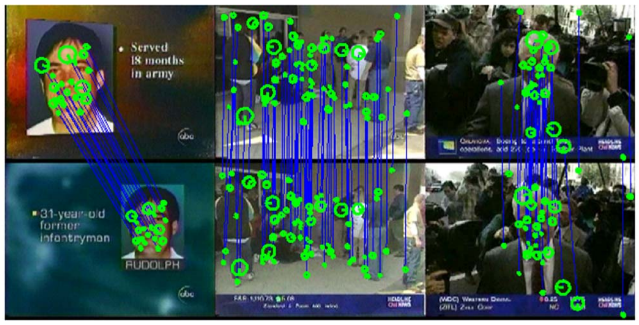


Fig. 7. Correspondences between near duplicate images.

of the support. We set $K_1 = 30$, and for each pair of images, we search 11 scales. Fig. 7 demonstrates the correspondences detected on three near-duplicate images. Such dense correspondences usually indicate similar objects or scenes. In Fig. 8, the retrieval performance is plotted and compared with the NIM method [29], OOS-PCA-SIFT method [31], and visual keywords method [32]. Obviously, our method outperforms all other methods and gets the best cumulative accuracies (ratio between correctly retrieved images in the top retrieved images and total number of query images), which verifies that our method can correctly detect correspondences in real images.

5.1.3 Maximum Common Subgraph

In this section, we evaluate our method on the maximum common subgraph problem, a well-known NP-hard yet very important problem [33]. Our aim is to find the maximum common subgraph in two labeled graphs, E and F, and this common subgraph must be a connected graph. For a node in E, it can correspond to any node in F with the same label. For a correspondence $(E_i, F_{i'})$, the correspondence $(E_j, F_{j'})$ which is consistent with it must satisfy either of the following two criteria: 1) E_i is adjacent to E_j and $F_{i'}$ is adjacent to $F_{j'}$, or 2) E_i is not adjacent to E_j and $F_{i'}$ is not adjacent to $F_{j'}$. Since the common subgraph must be a connected graph, in the expansion phase we just need to consider the neighbors satisfying the first criterion, which will not affect the final result, but can greatly speed up our algorithm since it largely reduces the number of neighbors to be considered.

We compare our method with the Durand-Pasari algorithm [34], which is based on the well-known reduction

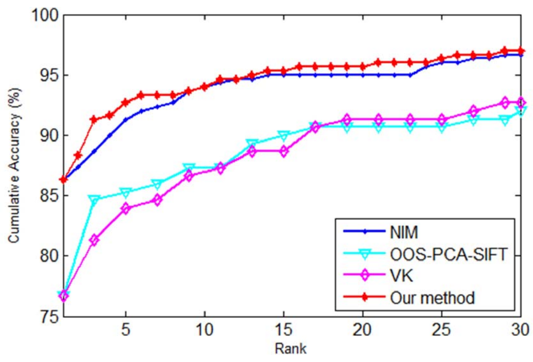


Fig. 8. Comparison of cumulative accuracy of near duplicate image retrieval on the Columbia database.

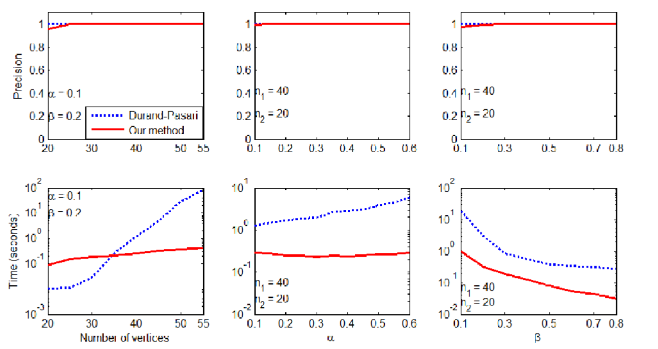


Fig. 9. Performance curves of our method versus the Durand-Pasari algorithm. The performance of our method is shown in red solid curves, while the performance of the Durand-Pasari algorithm is shown in blue dotted curves.

of the search of the maximum common subgraph to the problem of finding a maximal clique in the correspondence graph. The same as with our method, the Durand-Pasari algorithm also seeks to obtain the dense subgraphs of the correspondence graph. However, the Durand-Pasari algorithm is an enumeration method, which guarantees to find the maximum common subgraph, but with inherently very high time complexity.

We conduct the experiments on randomly connected graphs. Suppose the number of nodes in both graphs is n_1 and the number of nodes in the maximum common subgraph is n_2. The number of edges is controlled by a density parameter α, that is, the number of edges is about $\alpha n_1(n_1 - 1)$. We also assign a label to each node of the graph, with the number of labels being controlled by a parameter β, that is, the number of labels is βn_1. Since candidate correspondences are established between nodes with the same labels, obviously the larger β is, the easier the problem is.

We conduct three experiments, the results of which are shown in the first, second, and third columns of Fig. 9, respectively. The top row shows the precision, which is the ratio between the number of correctly detected correspondences and the number of nodes in the maximum common subgraph, and the bottom row shows the time complexity. In the first experiment, we increase the number of nodes of both graphs E and F from 20 to 55, with the number of nodes in the maximum common subgraph being about half of the number of nodes in the graph. In the second and third experiments, we fix $n_1 = 40$ and $n_2 = 20$, and change α and β, respectively. Each experiment is repeated 10 times on randomly generated graphs to obtain the average results. As the experimental results demonstrate, on small graphs the Durand-Pasari algorithm is more efficient; this is because the search space is small and our method must run from many initializations. However, as the number of nodes increases, the time complexity of the Durand-Pasari algorithm increases exponentially and quickly becomes time prohibitive. In contrast, our method increases slowly, which is very efficient for large graphs. As for the precision, the Durand-Pasari algorithm always finds the maximum common subgraphs, while our method cannot. However, as the results indicate, our method in fact has very high probability (nearly equal to 1) of finding the maximum

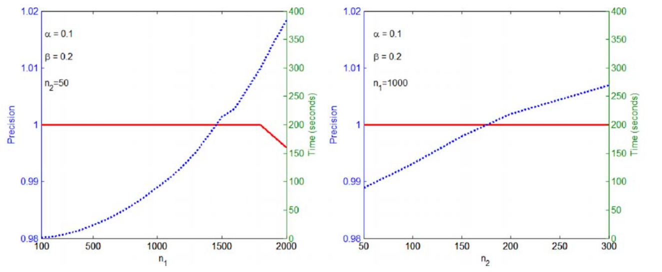


Fig. 10. Performance curves of our method on large graphs. The red solid lines show the precision, while the blue dotted lines show the time cost.

common subgraph; even when the common subgraph is not found, the obtained solution usually has a large common part with the maximum common subgraph. As the number of nodes in the maximum common subgraph increases, the mode it corresponds to becomes more and more significant, and then we have a much higher chance of detecting it.

We also test our proposed method on large graphs. In the left of Fig. 10, we fix the size of the maximum common subgraph to 50, and increase the size of both E and F from 100 to 2000. In the right figure, we fix the size of both E and F to 1000, and increase the size of maximum common subgraph from 50 to 300. The red solid curves show the precision and the blue dotted lines demonstrate the time complexity. Obviously, our method has very high probability of obtaining the correct results; at the same time, it can deal with large graphs very efficiently.

5.2 Cluster Analysis

Just as is mean shift clustering, SEA is a natural clustering tool, and all the vertices evolving toward the same KKT points should belong to the same cluster.

We first consider the problem of extracting dense clusters from cluttered background. Because many points should not belong to any clusters, assigning each point into a cluster usually leads to bad results. Our method, on the contrary, appears to be particularly suited for such applications since it allows one to extract as many clusters as desired, while leaving the remaining points (namely, the clutter) ungrouped. At the same time, it can automatically reveal the number of coherent groups.

First, we conduct an experiment on the toy dataset shown in Fig. 11a, which contains two dense clusters of Gaussian random points surrounded by uniformly distributed clutter points. We compare our method with k-means [35] and spectral clustering (SC) [36], two representative clustering methods, based on vectorial representation and affinity data, respectively. For k-means and SC, we specify that the number of clusters is 3, while for our method, we choose the largest two clusters and regard other small clusters as background. The clustering results of k-means, SC, and our method are illustrated in Figs. 11b, 11c, and 11d, respectively. Since k-means tends to partition the data into spherical clusters, it works poorly in such a situation. Both SC and our method can separate the two dense clusters from the background; however, our method can automatically reveal that there exist two clusters.

We also conduct an experiment on the affinity data from shape matching. The database is the MPEG-7 shape database

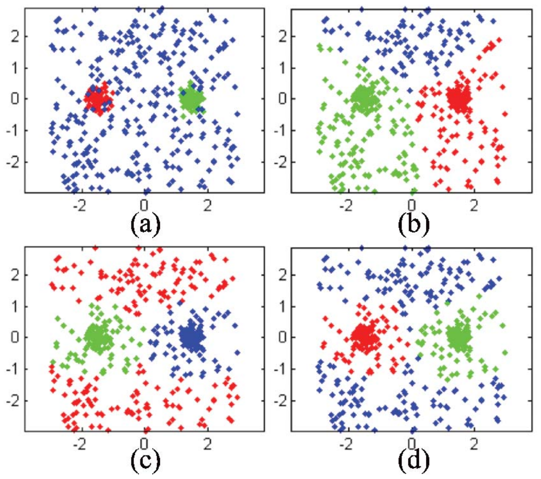


Fig. 11. Clustering on data with uniform distributed background points: (a) the dataset, (b) clustering result of k-means, (c) clustering result of SC, and (d) clustering result of our method.

[37]. There are 70 categories and each category contains 20 shapes. For each pair of shapes, we calculate their matching score (affinity value) using the IDSC method [37], and obtain a $1{,}400 \times 1{,}400$ affinity matrix. Such affinity data has no corresponding vectorial representation; at the same time, it usually contains large noises since for different pairs of shapes, their matching scores are computed independently, and the matching method may produce incorrect results on some pairs of shapes. We compare our method to six methods, namely, SC, affinity propagation (AP) [38], power iteration clustering (PIC) [39], 1-spectral clustering (1-SC) [40], game theoretic clustering (GC) [41], and ensemble clustering (EC) [42]. The result is shown in Table 2. For EC, there is a parameter ε which controls the least number of elements in each detected cluster. We illustrate the results of EC when $\varepsilon = \frac{1}{10}$ and $\varepsilon = \frac{1}{20}$, that is, the least size of cluster is 10 and 20, respectively. The performance of clustering is measured by both purity and normalized mutual information. For SC, PIC, and 1-SC, the number of clusters can be directly specified. For our method, when adding all vertices evolving to the same KKT point into a cluster, we iteratively merge the two closest clusters until the number of clusters reaches 70. For both GC and EC, we iterate the process of detecting a cluster, then delete the vertices belonging to this cluster and detect clusters on the remaining graph, following [41]. To obtain the final clusters, we also iteratively merge two closet clusters until the number of clusters reaches 70. GC can be simply considered as EC with the least size of cluster being 1. AP can only approximate the real number of clusters. Obviously, our method outperforms all other methods, and a possible explanation is that the affinity matrix contains noises and SEA is inherently noise-resistent. At the same time, our method spends much less time. Note that AP needs to search an appropriate preference value; thus it runs the clustering algorithm many times and takes a very long time. For EC, by adjusting the parameter ε, the result is improved. The precision increases from 66.43 to 70.79 percent when $\varepsilon = \frac{1}{10}$, then up to 72.71 percent when $\varepsilon = \frac{1}{20}$. Generally speaking, these two methods perform poorly on this dataset, probably due to the "detect-and-delete" strategy from [41], which allows errors to accumulate and hence does not work well when the number of clusters is large.

TABLE 2
Clustering Results on Shape Matching Affinity Data

	SC	AP	PIC	1-SC	GC	EC($\frac{1}{10}$)	EC($\frac{1}{20}$)	SEA
Clusters	70	64	70	70	70	70	70	70
Purity(%)	73	76.5	67.14	41.43	66.43	70.79	72.71	**85.36**
NMI(%)	88.86	88.27	84.61	69.40	86.84	84.45	85.53	**91.41**
Time	9.44	473.28	3.21	22.21	32.10	2.45	1.37	**0.95**

The time is measured in seconds.

6 Conclusions

We proposed the SEA to detect dense subgraphs. Based on SEA, we derived a general algorithm for the correspondence problem, and demonstrated its excellent performance on point set matching, near duplicate image retrieval, and the maximum common subgraph problem. The SEA can also automatically delineate the cluster structure underlying the data, and we showed its application on two clustering tasks. Clearly, our method can be applied to many other tasks that can be formulated as seeking dense subgraphs, such as object detection, graph matching, and community detection in social networks. We are currently working toward a more efficient way of initialization, which can reduce the number of initializations but still retain a high probability of obtaining all significant dense subgraphs.

References

[1] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 22, no. 8, pp. 888-905, Aug. 2000.
[2] M. Girvan and M. Newman, "Community Structure in Social and Biological Networks," *Proc. Nat'l Academy of Sciences,* vol. 99, no. 12, pp. 7821-7826, 2002.
[3] J. Chen and Y. Saad, "Dense Subgraph Extraction with Application to Community Detection," *IEEE Trans. Knowledge and Data Eng.,* vol. 24, no. 7, pp. 1216-1230, July 2012.
[4] Q. Ouyang, P. Kaplan, S. Liu, and A. Libchaber, "DNA Solution of the Maximal Clique Problem," *Science,* vol. 80, pp. 446-448, 1997.
[5] K. Crammer, P. Talukdar, and F. Pereira, "A Rate-Distortion One-Class Model and Its Applications to Clustering," *Proc. Int'l Conf. Machine Learning,* pp. 184-191, 2008.
[6] G. Palla, I. Derényi, I. Farkas, and T. Vicsek, "Uncovering the Overlapping Community Structure of Complex Networks in Nature and Society," *Nature,* vol. 435, no. 7043, pp. 814-818, 2005.
[7] A. Clauset, M. Newman, and C. Moore, "Finding Community Structure in Very Large Networks," *Physical Rev. E,* vol. 70, no. 6, pp. 66-71, 2004.
[8] M. Zaki, S. Parthasarathy, M. Ogihara, and W. Li, "New Algorithms for Fast Discovery of Association Rules," *Proc. Int'l Conf. Knowledge Discovery and Data Mining,* vol. 20, pp. 283-286, 1997.
[9] T. Motzkin and E. Straus, "Maxima for Graphs and a New Proof of a Theorem of Turán," *Canadian J. Math.,* vol. 17, no. 4, pp. 533-540, 1965.
[10] M. Pavan and M. Pelillo, "Dominant Sets and Pairwise Clustering," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 1, pp. 167-172, Jan. 2007.
[11] D. Comaniciu and P. Meer, "Mean Shift: A Robust Approach toward Feature Space Analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 5, pp. 603-619, May 2002.
[12] I. Bomze, "Branch-and-Bound Approaches to Standard Quadratic Optimization Problems," *J. Global Optimization,* vol. 22, no. 1, pp. 17-37, 2002.
[13] H. Kuhn and A. Tucker, "Nonlinear Programming," *Proc. Berkeley Symp. Math. Statistics and Probability,* pp. 481-492, 1951.
[14] J. Weibull, *Evolutionary Game Theory.* The MIT Press, 1997.
[15] J. Maciel and J. Costeira, "A Global Solution to Sparse Correspondence Problems," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 25, no. 2, pp. 187-199, Feb. 2003.

[16] T. Caetano, T. Caelli, D. Schuurmans, and D. Barone, "Graphical Models and Point Pattern Matching," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 10, pp. 1646-1663, Oct. 2006.

[17] H. Jiang, M. Drew, and Z. Li, "Matching by Linear Programming and Successive Convexification," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 6, pp. 959-975, June 2007.

[18] B. Georgescu and P. Meer, "Point Matching under Large Image Deformations and Illumination Changes," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 26, no. 6, pp. 674-688, June 2004.

[19] A. Cross and E. Hancock, "Graph Matching with a Dual-Step EM Algorithm," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, no. 11, pp. 1236-1253, Nov. 1998.

[20] M. Zaslavskiy, F. Bach, and J. Vert, "A Path Following Algorithm for the Graph Matching Problem," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 31, no. 12, pp. 2227-2242, Dec. 2009.

[21] R. Horaud and T. Skordas, "Stereo Correspondence through Feature Grouping and Maximal Cliques," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 11, no. 11, pp. 1168-1180, Nov. 1989.

[22] M. Pelillo, "Matching Free Trees with Replicator Equations," *Proc. Advances in Neural Information Processing Systems Conf.,* pp. 865-872, 2002.

[23] A. Albarelli, S. Bulo, and M. Pelillo, "Matching as a Non-Cooperative Game," *Proc. IEEE Int'l Conf. Computer Vision,* 2009.

[24] M. Leordeanu and M. Hebert, "A Spectral Technique for Correspondence Problems Using Pairwise Constraints," *Proc. IEEE Int'l Conf. Computer Vision,* pp. 1482-1489, 2005.

[25] M. Leordeanu, M. Hebert, and R. Sukthankar, "An Integer Projected Fixed Point Method for Graph Matching and Map Inference," *Proc. Advances in Neural Information Processing Systems Conf.,* vol. 1, no. 3, p. 4, 2009.

[26] O. Duchenne, F. Bach, I. Kweon, and J. Ponce, "A Tensor-Based Algorithm for High-Order Graph Matching," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 1980-1987, 2009.

[27] M. Cho, J. Lee, and K. Lee, "Reweighted Random Walks for Graph Matching," *Proc. European Conf. Computer Vision,* pp. 492-505, 2010.

[28] R. Zass and A. Shashua, "Probabilistic Graph and Hypergraph Matching," *Proc. IEEE Conf. Computer Vision and Pattern Recognition,* pp. 1-8, 2008.

[29] J. Zhu, S. Hoi, M. Lyu, and S. Yan, "Near-Duplicate Keyframe Retrieval by Nonrigid Image Matching," *Proc. ACM Int'l Conf. Multimedia,* pp. 41-50, 2008.

[30] D. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," *Int'l J. Computer Vision,* vol. 60, no. 2, pp. 91-110, 2004.

[31] X. Wu, W. Zhao, and C. Ngo, "Near-Duplicate Keyframe Retrieval with Visual Keywords and Semantic Context," *Proc. ACM Int'l Conf. Image and Video Retrieval,* pp. 169-176, 2007.

[32] W. Zhao, C. Ngo, H. Tan, and X. Wu, "Near-Duplicate Keyframe Identification with Interest Point Matching and Pattern Learning," *IEEE Trans. Multimedia,* vol. 9, no. 5, pp. 1037-1048, Aug. 2007.

[33] D. Conte, C. Guidobaldi, and C. Sansone, "A Comparison of Three Maximum Common Subgraph Algorithms on a Large Database of Labeled Graphs," *Proc. Fourth IAPR Int'l Conf. Graph Based Representations in Pattern Recognition,* vol. 2726, pp. 130-141, 2003.

[34] P. Durand, R. Pasari, J. Baker, and C. Tsai, "An Efficient Algorithm for Similarity Analysis of Molecules," *Int'l J. Chemistry,* vol. 2, no. 17, pp. 1-16, 1999.

[35] T. Kanungo, D. Mount, N. Netanyahu, C. Piatko, R. Silverman, and A. Wu, "An Efficient K-Means Clustering Algorithm: Analysis and Implementation," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 24, no. 7, pp. 881-892, July 2002.

[36] A. Ng, M. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an Algorithm," *Proc. Advances in Neural Information Processing Systems Conf.,* vol. 2, pp. 849-856, 2002.

[37] H. Ling and D. Jacobs, "Shape Classification Using the Inner-Distance," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 29, no. 2, pp. 286-299, Feb. 2007.

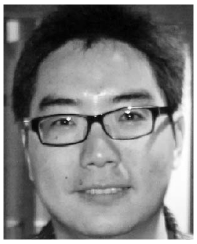
[38] B. Frey and D. Dueck, "Clustering by Passing Messages between Data Points," *Science,* vol. 315, no. 5814, pp. 972-976, 2007.

[39] F. Lin and W. Cohen, "Power Iteration Clustering," *Proc. Int'l Conf. Machine Learning,* 2010.

[40] M. Hein and T. Bühler, "An Inverse Power Method for Nonlinear Eigenproblems with Applications in 1-Spectral Clustering and Sparse PCA," *Proc. Advances in Neural Information Processing Systems Conf.,* 2010.

[41] S. Bulò and M. Pelillo, "A Game-Theoretic Approach to Hypergraph Clustering," *Proc. Advances in Neural Information Processing Systems Conf.,* vol. 22, pp. 1571-1579, 2009.

[42] H. Liu, L. Latecki, and S. Yan, "Robust Clustering as Ensembles of Affinity Relations," *Proc. Advances in Neural Information Processing Systems,* 2010.



Hairong Liu is currently a research fellow in the Department of Electrical and Computer Engineering at the National University of Singapore. His research interests include computer vision and machine learning, with a focus on matching and graph analysis. He received the Best Paper Award from ICME '10, and he is a reviewer for CVPR, ICCV, TIP, TCSVT, and TPAMI.

Longin Jan Latecki is a professor of computer science at Temple University, Philadelphia. His main research interests include shape representation and similarity, object detection and recognition in images, robot perception, machine learning, and digital geometry. He has published 200 research papers and books. He is an editorial board member of *Pattern Recognition* and the *International Journal of Mathematical Imaging*. He received the annual Pattern Recognition Society Award together with Azriel Rosenfeld for the best article published in the journal *Pattern Recognition* in 1998. He is the recipient of the 2000 Olympus Prize, the main annual award from the German Society for Pattern Recognition. He is a senior member of the IEEE.

Shuicheng Yan is currently an assistant professor in the Department of Electrical and Computer Engineering at the National University of Singapore. His research interests include computer vision, multimedia, and machine learning, and he has authored or coauthored more than 200 technical papers. He is an associate editor of *IEEE TCSVT*. He received the Best Paper Awards from ACM MM '10, ICME '10, and ICIMCS '09, the winner of the prize for the classification task in PASCAL VOC '10, the honorable mention prize for the detection task in PASCAL VOC '10, and a 2010 TCSVT Best Associate Editor Award. He is a senior member of the IEEE.