Clustering and Embedding Using Commute Times

Huaijun (John) Qiu and Edwin R. Hancock

Abstract—This paper exploits the properties of the commute time between nodes of a graph for the purposes of clustering and embedding and explores its applications to image segmentation and multibody motion tracking. Our starting point is the lazy random walk on the graph, which is determined by the heat kernel of the graph and can be computed from the spectrum of the graph Laplacian. We characterize the random walk using the commute time (that is, the expected time taken for a random walk to travel between two nodes and return) and show how this quantity may be computed from the Laplacian spectrum using the discrete Green's function. Our motivation is that the commute time can be anticipated to be a more robust measure of the proximity of data than the raw proximity matrix. In this paper, we explore two applications of the commute time. The first is to develop a method for image segmentation using the eigenvector corresponding to the smallest eigenvalue of the commute time matrix. We show that our commute time segmentation method has the property of enhancing the intragroup coherence while weakening intergroup coherence and is superior to the normalized cut. The second application is to develop a robust multibody motion tracking method using an embedding based on the commute time. Our embedding procedure preserves commute time and is closely akin to kernel PCA, the Laplacian eigenmap, and the diffusion map. We illustrate the results on both synthetic image sequences and real-world video sequences and compare our results with several alternative methods.

Index Terms—Commute time, clustering, embedding, spectral graph theory, image segmentation, motion tracking.

1 Introduction

Graph spectral methods have played an important role in the image segmentation and data clustering literature [2], [24], [26], [27], [31], [37]. Spectral graph theory [4] is concerned with characterizing the structural properties of graphs using information conveyed by the eigenvalues and eigenvectors of the Laplacian matrix (the degree matrix minus the adjacency matrix). One of the most important tasks that arise in the analysis of graphs is that of how information diffuses with time across the edges connecting nodes. This process can be characterized using the heat equation [20]. The solution of the heat equation, or heat kernel, can be found by exponentiating the Laplacian eigensystem with time. The heat kernel contains a considerable amount of information concerning the distribution of paths on the graph. For instance, it can be used to compute the lazy random walk on the nodes of the graph, since the lazy random walk is the limit of the heat kernel in the continuous time limit. It may also be used to determine hitting times or commute times under the random walk between pairs of nodes. The *hitting time* $O(u,v)$ of a random walk on a graph is defined as the expected number of steps before node v is visited, commencing from node u. The *commute time* $CT(u,v)$, on the other hand, is the expected time for the random walk to travel from node u to reach node v and then return. An alternative but closely related characterization of the graph is the discrete Green's function (or pseudoinverse of the Laplacian), which captures the distribution of sources in the heat flow process. Not surprisingly, there is a direct link between commute times and the Green's function [5].

The aim of this paper is to explore whether commute time can be used as a means of data clustering and embedding. The intuition that motivates this study is that since commute time reflects the combined effect of all possible weighted paths between a pair of nodes, it is more robust to structural disturbance. Hence, the commute time can lead to a measure of cluster cohesion that is less sensitive to edge deletions and insertions than the simple use of edge weight alone, which underpins algorithms such as the normalized cut [37]. Specifically, the affinity of nodes conveyed by commute time is large for pairs of nodes residing in a cluster and small for those falling outside the cluster. It has been shown [42] that the reason that some methods succeed in solving the grouping problem is because they lead to an affinity matrix with a strong block structure. In fact, this block structure can be further amplified by the commute times [11].

1.1 Related Literature

We will explore two applications of commute time: The first of these is for image segmentation and, the second, is for multibody motion tracking. In this section, we review the related literature.

1.1.1 Segmentation, Clustering, and Embedding

There are two quantities that are commonly used to define the utility in graph-theoretic methods for grouping and clustering. The first of these is the association, which is a measure of total edge linkage within a cluster and is useful in defining clump structure. The second is the cut, which is a measure of linkage between different clusters and can be used to split extraneous nodes from a cluster. Several methods use eigenvectors to extract clusters. Some of the earliest work was

- *H. Qiu is with the Queen Mary Vision Laboratory, Department of Computer Science, Queen Mary, University of London, London, UK E1 4NS. E-mail: john@dcs.qmul.ac.uk.*
- *E.R. Hancock is with the Department of Computer Science, University of York, Heslington, York, UK YO10 5DD. E-mail: erh@cs.york.ac.uk.*

Manuscript received 13 Feb. 2006; revised 13 Sept. 2006; accepted 29 Dec. 2006; published online 30 Jan. 2007.
Recommended for acceptance by A. Srivastava.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number TPAMI-0152-0206.
Digital Object Identifier no. 10.1109/TPAMI.2007.1103.

done by Scott and Longuet-Higgins [36] who developed a method for refining the block structure of the affinity matrix by relocating its eigenvectors. At the level of image segmentation, several authors have used algorithms based on the eigenmodes of an affinity matrix to iteratively segment image data. For instance, Sarkar and Boyer [34] have a method that uses the leading eigenvector of the affinity matrix and this locates clusters that maximize the average association. This method is applied to locating line-segment groupings. Perona and Freeman [30] have a similar method that uses the second largest eigenvector of the affinity matrix. The method of Shi and Malik [37], on the other hand, uses the normalized cut, which balances the cut and the association. Clusters are located by performing a recursive bisection using the eigenvector associated with the second smallest eigenvalue of the Laplacian, that is, the Fiedler vector. A random walk view of this method is given by Shi and Meilă [24]. They interpreted the pairwise similarities as edge flows in a Markov random walk and have shown how spectral clustering methods can be explained in a probabilistic way. Focusing more on the issue of postprocessing, Weiss [42] has shown how this and other closely related methods can be improved using a normalized affinity matrix. Pavan and Pelillo [28], [29] have shown how the performance of this method can be improved using a finer measure of cluster cohesion based on dominant sets. More recently, Lafon et al. [22], [25] have shown how the diffusion map can be used to accommodate path-length variation and have used the map for scale-dependent clustering. Zass and Shashua [44] show how to provide a probabilistic interpretation for spectral clustering [42] by developing a completely positive factorization scheme.

Spectral embedding plays an important role in dimensionality reduction literature. It typically commences with an affinity matrix computed from the distances between pairs of data points. This data representation is characterized using the eigenspectrum affinity matrix, often using one or just a few eigenvectors. For example, principal components analysis (PCA) [17] and kernel principal component analysis (KPCA) [35] use the leading eigenvectors of the covariance matrix to determine the projection directions with maximal variance. Multidimensional scaling (MDS) [21] uses the eigenvectors of a pairwise distance matrix to find an embedding of the data that minimizes the stress. As an extension, the isometric feature mapping (Isomap) [41] employs MDS to preserve the geodesic distances between data points located on a manifold. Locally linear embedding (LLE) [32] maps the input data to a lower dimensional space in a manner that preserves the local neighborhood. Similar ideas to those developed later on in this paper are used in the study by Saerens et al. [33]. Here, the commute time is taken as a distance measure for nodes in the graph and embedding of the nodes is performed in a variance preserving manner similar to PCA. The evaluation of the method is confined to visualizing relatively small graphs. The differences between our work and that of Saerens et al. are threefold. First, Saerens et al. introduce the commute time in a traditional way using a Markov random walk model, whereas our approach is posed as a diffusion process on the graph and generalizes the computation of commute time using the normalized Laplacian and the Green's function. Second, Saerens et al. have shown that the commute time-preserving embedding is equivalent to a PCA of the graph, and taking commute time as a kernel matrix, they compared it with five alternative graph kernels [12]. In our work here, we go further to explore and study the advantages of the commute time embedding over existing manifold embedding methods including the Laplacian eigenmap [3] and the diffusion map [6]. Third, most importantly, in our experiments, we show how computer vision problems such as image segmentation and motion tracking can be cast into a commute time embedding framework and solved effectively.

1.1.2 Factorization Methods for Motion Analysis

As a second and more demanding application, we consider the multibody motion tracking problem. Multibody motion tracking is a challenging problem that arises in shape from motion, video coding, the analysis of movement, and surveillance. One of the classical techniques is the *factorization method* of Costeira and Kanade [7], [8]. The basic idea underpinning this method is to use singular value decomposition (SVD) to factorize the feature trajectory matrix into a motion matrix and a shape matrix. The shape interaction matrix is found by taking the self outer product of the right eigenvector matrix and can be used to identify the independently moving objects present. Gear [13] has developed a related method based on the reduced row echelon form of the matrix and object separation is achieved by performing probabilistic analysis on a bipartite graph. Both methods work well in the ideal case when there is no noise (that is, feature-point jitter) and outliers are not present. However, real-world image sequences are usually contaminated by these two types of noise. There have been several attempts to overcome this problem. For instance, Ichimura [18] has improved the *factorization method* by using a discriminant criterion to threshold-out noise and outliers.

Rather than working with an affinity matrix derived from the data, some researchers place the emphasis on the original data. Kanatani et al. [19], [39], [40] developed a subspace separation method by incorporating dimension correction and model selection. Wu et al. [43] argue that the subspaces associated with the different objects are not only distinct but also orthogonal. They hence employ an orthogonal subspace decomposition method to separate objects. This idea is further extended by Fang et al. who use independent subspaces [10] and multiple subspace inference analysis [9]. In addition to attempting to improve the behavior of the factorization method under noise, there has been a considerable effort aimed at overcoming problems such as degeneracy, uncertainty, and missing data [1], [14], [45].

The factorization method is clearly closely akin to graph-spectral methods used in clustering, since it uses eigenvector methods to determine the class affinity of sets of points. In fact, Weiss [42] has presented a unifying view of spectral clustering techniques, and this includes the factorization method. There has been some concerted effort devoted to solving the object separation problem using spectral clustering methods. Park et al. [27] have applied a multiway min-max cut clustering method to the shape interaction matrix. Here, the shape interaction matrix is used as a cluster indicator matrix and noise compensation is effected using a combination of spectral clustering and subspace separation techniques.

1.2 Contribution

The aim of this paper is twofold: First, we aim to review the main results from the spectral graph theory that relate to the

definition of commute time. With these definitions at hand, we explore the properties of commute time for the purposes of spectral clustering and embedding. Although, as we will show later, embedding is more efficient, the spectral clustering process is important since it is closely linked to the normalized cut. The embedding coordinate matrix is found by premultiplying the transpose of the Laplacian eigenvector matrix with the inverse square root of the eigenvalue matrix. Under the embedding, nodes that have small commute time are close and those that have a large commute time are distant. This allows us to separate the objects in the embedded subspace by applying simple techniques such as K-Means clustering. There are of course many graph-spectral embedding algorithms reported in the literature, and recent and powerful additions include KPCA [35], the Laplacian eigenmap [3], and the diffusion map [6]. We explore the relationship of the commute time embedding to these alternatives.

With the mathematical framework in place, we then explore two applications of commute time to problems in computer vision. The first application is that of image segmentation. Here, we suggest the use of commute time as an alternative to the use of edge weight information alone for the purposes of pairwise clustering. The aim in the second application reported in this paper is to explore whether an embedding based on commute time can be used to solve the problem of computing the shape interaction matrix in a robust manner. We use the shape interaction matrix Q as a data-proximity weight matrix and compute the associated Laplacian matrix (the degree matrix minus the weight matrix).

The outline of the paper is as follows: In Section 2, we review the definitions of commute time and its links to the Laplacian spectrum. Section 3 discusses the commute time embedding and explores its links with the diffusion map and KPCA. Section 4 sets up the two applications studied in the paper. Experiments are presented in Section 5. Finally, Section 6 offers some conclusions and offers directions for future investigation.

2 Graph Laplacian, Heat Kernel, Green's Function, and the Commute Time

Commute time is a concept from spectral graph theory that has close links with the graph Laplacian, the heat kernel, and random walks on a graph. In the following sections, we review how to compute commute time and describe the relationships to the graph Laplacian and the heat kernel. The material presented in this section provides the prerequisites for our study and is a summary of results obtained by Chung and Yau [5].

2.1 Graph Laplacian and Heat Kernel

We denote a weighted graph by the triple $\Gamma = (V, E, \Omega)$, where V is the set of nodes, $E \subseteq V \times V$ is the set of edges, and Ω is the weighted adjacency matrix

$$\Omega(u,v) = \begin{cases} w(u,v) & \text{if}(u,v) \in E \\ 0 & \text{otherwise,} \end{cases}$$

where $w(u,v)$ is the weight on the edge $(u,v) \in E$. Further, let $T = diag(d_u; u \in V)$ be the diagonal weighted degree matrix with elements given by the degrees of the nodes $d_u = \sum_{v=1}^{|V|} w(u,v)$. The *unnormalized* weighted Laplacian matrix is given by $L = T - \Omega$, and the normalized weighted Laplacian matrix is defined to be $\mathcal{L} = T^{-1/2} L T^{-1/2}$ and has elements

$$\mathcal{L}_\Gamma(u,v) = \begin{cases} 1 & \text{if } u = v \\ -\frac{w(u,v)}{\sqrt{d_u d_v}} & \text{if } u \neq v \text{ and } (u,v) \in E \\ 0 & \text{otherwise.} \end{cases}$$

The spectral decomposition of the *normalized* Laplacian is $\mathcal{L} = \Phi' \Lambda' \Phi'^T$, where $\Lambda' = diag(\lambda'_1, \lambda'_2, \ldots, \lambda'_{|V|})$ is the diagonal matrix with the ordered eigenvalues as the elements satisfying the condition $0 = \lambda'_1 \leq \lambda'_2 \ldots \leq \lambda'_{|V|}$, and $\Phi' = (\phi'_1 | \phi'_2 | \ldots | \phi'_{|V|})$ is the matrix with the ordered eigenvectors as columns. The corresponding eigendecomposition of the *unnormalized* Laplacian matrix is $L = \Phi \Lambda \Phi^T$.

In spectral clustering, both the *normalized* Laplacian and the *unnormalized* Laplacian have been used for partitioning data. For instance, the ratio cut [15] uses the *unnormalized* Laplacian, and the normalized cut [37] uses the *normalized* Laplacian. The difference lies in the cutting criterion used. Both methods aim at minimizing the weighted edge cut between clusters. The ratio cut only balances the number of vertices, whereas the normalized cut balances the volume of each class. The latter criterion has been demonstrated to yield a better performance.

Commute time is a property of a diffusion process on a graph. Diffusion is governed by the heat equation—the partial differential equation $\frac{\partial \mathcal{H}_t}{\partial t} = -\mathcal{L}\mathcal{H}_t$, where $\mathcal{H}_t$ is the heat kernel, and t is time. The solution of the heat equation is found by exponentiating the Laplacian eigenspectrum, that is,

$$\mathcal{H}_t = \exp[-t\mathcal{L}] = \exp\left[-t\Phi'\Lambda'\Phi'^T\right] = \exp\left[\Phi'(-t\Lambda')\Phi'^T\right]. \quad (1)$$

Let A be a symmetric $n \times n$ diagonizable matrix with eigendecomposition $A = MEM^T$, where E is the diagonal matrix whose elements are the ordered eigenvalues of A and M is the matrix with the ordered unit eigenvectors of A as columns satisfying the condition $M^T M = I$. From the MacLaurin expansion $\exp[A] = I + A + (1/2)A^2 + \ldots + (1/N!)A^N + \ldots$ and using the fact that $A^N = ME^N M^T$, it follows that $\exp[A] = M \exp[E] M^T$. Therefore, we have $\mathcal{H}_t = \Phi' \exp[-t\Lambda']\Phi'^T$. As a result, the heat kernel is a $|V| \times |V|$ matrix, and for the nodes u and v of the graph Γ, the element of the matrix is

$$\mathcal{H}_t(u,v) = \sum_{i=1}^{|V|} \exp[-\lambda'_i t]\phi'_i(u)\phi'_i(v).$$

2.2 Green's Function

Now, consider the discrete Laplace operator $\Delta = T^{-1/2}\mathcal{L}T^{1/2}$. The Green's function is the left inverse operator of the Laplace operator Δ, defined by $G\Delta(u,v) = I(u,v) - \frac{d_v}{vol}$, where $vol = \sum_{v \in V} d_v$ is the volume of the graph and I is the $|V| \times |V|$ identity matrix. The Green's function of the graph is related to the heat kernel H_t and has elements given by

$$G(u,v) = \int_0^\infty d_u^{1/2}\left(\mathcal{H}_t(u,v) - \phi'_1(u)\phi'_1(v)\right)d_v^{-1/2}dt, \quad (2)$$

where ϕ'_1 is the eigenvector associated with the zero eigenvalue, that is, $\lambda'_1 = 0$ of the *normalized* Laplacian matrix, and whose kth element is $\phi'_1(k) = \sqrt{d_k/vol}$. Furthermore, the *normalized* Green's function $\mathcal{G} = T^{1/2} G T^{-1/2}$ is

given in terms of the normalized Laplacian spectrum (see [5, p. 6]) as

$$\mathcal{G}(u,v) = \sum_{i=2}^{|V|} \frac{1}{\lambda'_i} \phi'_i(u)\phi'_i(v), \tag{3}$$

where λ' and ϕ' are the eigenvalue and eigenvectors of the *normalized* Laplacian $\mathcal{L}$. The corresponding Green's function of the *unnormalized* Laplacian $\bar{G}$ is given by

$$\bar{G}(u,v) = \sum_{i=2}^{|V|} \frac{1}{\lambda_i} \phi_i(u)\phi_i(v), \tag{4}$$

where λ_i and ϕ_i are the eigenvalue and eigenvectors of the *unnormalized* Laplacian L.

The *normalized* Green's function is hence the pseudoinverse of the *normalized* Laplacian $\mathcal{L}$. Moreover, it is straightforward to show that $\mathcal{G}\mathcal{L} = \mathcal{L}\mathcal{G} = I - \phi'_1\phi'^T_1$, and, as a result, $(\mathcal{L}\mathcal{G})(u,v) = \delta(u,v) - \frac{\sqrt{d_u d_v}}{vol}$. From (3), the eigenvalues of $\mathcal{L}$ and $\mathcal{G}$ have the same sign, and $\mathcal{L}$ is positive semidefinite and, so, $\mathcal{G}$ is also positive semidefinite. Since $\mathcal{G}$ is also symmetric (see [5, p. 4]), it follows that $\mathcal{G}$ is a kernel. The same applies to the *unnormalized* Green's function $\bar{G}$.

The relationship between G, $\bar{G}$, and $\mathcal{G}$ can be obtained if we consider an induced subgraph Γ_S of the original graph Γ. If Γ_S is connected, Δ, L, and $\mathcal{L}$ are nonsingular (see [4]) and we have the relationship $G\Delta = \bar{G}L = \mathcal{G}\mathcal{L} = I$ between the normalized and unnormalized Laplacians and their corresponding Green's functions. From the fact that $\Delta = T^{-1/2}\mathcal{L}T^{1/2}$ and $\mathcal{L} = T^{-1/2}LT^{-1/2}$, then $\Delta = T^{-1}L$. As a result, we have $G\Delta = GT^{-1}L = \bar{G}L$ and, as a consequence, $\bar{G} = GT^{-1}$. Making use of the fact that $\mathcal{G} = T^{1/2}GT^{-1/2}$, we have that $G = T^{-1/2}\mathcal{G}T^{1/2}$ and we then obtain

$$\bar{G} = GT^{-1} = T^{-1/2}\mathcal{G}T^{1/2}T^{-1} = T^{-1/2}\mathcal{G}T^{-1/2}. \tag{5}$$

2.3 Commute Time

We note that the *hitting time* $O(u,v)$ of a random walk on a graph is defined as the expected number of steps before node v is visited, commencing from node u. The *commute time* $CT(u,v)$, on the other hand, is the expected time for the random walk to travel from node u to reach node v and then return. As a result, $CT(u,v) = O(u,v) + O(v,u)$. The hitting time $O(u,v)$ is given by [5]

$$O(u,v) = \frac{vol}{d_v} G(v,v) - \frac{vol}{d_u} G(u,v),$$

where G is the Green's function given in (2). Therefore, the commute time is given by

$$\begin{aligned} CT(u,v) &= O(u,v) + O(v,u) \\ &= \frac{vol}{d_u} G(u,u) + \frac{vol}{d_v} G(v,v) - \frac{vol}{d_u} G(u,v) - \frac{vol}{d_v} G(v,u) \end{aligned} \tag{6}$$

or, using the *unnormalized* Green's function, as

$$CT(u,v) = vol\big(\bar{G}(u,u) + \bar{G}(v,v) - 2\bar{G}(u,v)\big). \tag{7}$$

As a consequence of (7), the commute time is a metric on the graph. The reason for this is that, if we take the elements of G as inner products defined in a euclidean space, CT will become the norm satisfying $\|x_u - x_v\|^2 = < x_u - x_v, x_u - x_v > = < x_u, x_u > + < x_v, x_v > - < x_u, x_v > - < x_v, x_u >$.

Substituting the spectral expression for the Green's function into the definition of the commute time, it is straightforward to show that in terms of the eigenvectors of the *normalized* Laplacian

$$CT(u,v) = vol \sum_{i=2}^{|V|} \frac{1}{\lambda'_i} \left(\frac{\phi'_i(u)}{\sqrt{d_u}} - \frac{\phi'_i(v)}{\sqrt{d_v}} \right)^2. \tag{8}$$

On the other hand, taking (4) into (7), the commute time can then be expressed using the eigensystem of the *unnormalized* Laplacian as

$$CT(u,v) = vol \sum_{i=2}^{|V|} \frac{1}{\lambda_i} (\phi_i(u) - \phi_i(v))^2. \tag{9}$$

3 Commute Time Embedding

The commute time embedding is a mapping from the data space into a Hilbert subspace, which preserves the original commute times. It has some properties similar to existing embedding methods including PCA, the Laplacian eigenmap [2], [3] and the diffusion map [22], [25]. In this section, we will first introduce the commute time embedding, and then, we compare it with alternative embedding methods. Some embedding examples are illustrated and the robustness of the embedding is also discussed.

3.1 Basics

Equation (8) can be rewritten in the following form, which makes the relationship between the commute time and the euclidean distance between the components of the eigenvectors explicit:

$$CT(u,v) = \sum_{i=2}^{|V|} \left(\sqrt{\frac{vol}{\lambda'_i d_u}} \phi'_i(u) - \sqrt{\frac{vol}{\lambda'_i d_v}} \phi'_i(v) \right)^2. \tag{10}$$

Hence, the embedding of the nodes of the graph into a vector space that preserves commute time has the coordinate matrix

$$\Theta = \sqrt{vol}\Lambda'^{-1/2}\Phi'^T T^{-1/2}. \tag{11}$$

The columns of the matrix are vectors of embedding coordinates for the nodes of the graph. The term $T^{-1/2}$ arises from the normalization of the Laplacian. If the commute time is computed from the unnormalized Laplacian, the corresponding matrix of embedding coordinates is

$$\Theta = \sqrt{vol}\Lambda^{-1/2}\Phi^T. \tag{12}$$

The embedding is nonlinear in the eigenvalues of the Laplacian. This distinguishes it from PCA and locality preserving projection (LPP) [16], which are both linear. As we will demonstrate in the next section, the commute time embedding is just KPCA [35] on the Green's function. Moreover, it is also related to the Laplacian eigenmap since it minimizes similar objective functions.

3.2 The Commute Time Embedding and Kernel PCA

Let us consider the unnormalized case above. Since the Green's function $\bar{G}$ is the pseudoinverse of the Laplacian, it discards the zero eigenvalue and the corresponding

eigenvector $\vec{1}$ of the Laplacian. The columns of the eigenvector matrix are orthogonal, which means that the eigenvector matrix Φ of $\bar{G}$ satisfies $\Phi^T\vec{1} = \vec{0}$. Hence, $\sqrt{vol}\Lambda^{-1/2}\Phi^T\vec{1} = \vec{0}$, and this means that the data is centered. As a result, the covariance matrix for the centered data is

$$C_f = \Theta\Theta^T = vol\Lambda^{-1/2}\Phi^T\Phi\Lambda^{-1/2} = vol\Lambda^{-1} = vol\Lambda_{\bar{G}}, \quad (13)$$

where $\Lambda_{\bar{G}}$ is the eigenvalue matrix of the *unnormalized* Green's function with eigenvalues ordered according to decreasing magnitude down the diagonal. The kernel or Gram matrix of the embedding is given by

$$K = \Theta^T\Theta = vol\Phi\Lambda^{-1/2}\Lambda^{-1/2}\Phi^T = vol\Phi\Lambda^{-1}\Phi^T = vol\bar{G}, \quad (14)$$

which is just the Green's function multiplied by a constant. Hence, we can view the embedding as performing KPCA on the Green's function for the Laplacian.

3.3 The Commute Time Embedding and the Laplacian Eigenmap

In the Laplacian eigenmap [2], [3], the aim is to embed a set of points $\bar{\mathbf{X}} = \{\bar{\mathbf{x}}_1, \bar{\mathbf{x}}_2, \ldots, \bar{\mathbf{x}}_n\}$ from an R^l space into a lower dimensional subspace R^m with the corresponding embedded coordinate matrix $\mathbf{Z}_{nxm} = [\mathbf{z}_1|\mathbf{z}_2|\ldots|\mathbf{z}_m]$. The original data points have a proximity weight matrix Ω with elements $\Omega(u,v) = \exp[-\|\bar{\mathbf{x}}_u - \bar{\mathbf{x}}_v\|^2]$. The aim is to find the embedding that minimizes the objective function (see [2, p. 4])

$$\epsilon = \sum_{u,v}\|\mathbf{Z}(u) - \mathbf{Z}(v)\|^2\Omega(u,v) = tr(\mathbf{Z}^T L\mathbf{Z}), \quad (15)$$

where Ω is the edge weight matrix of the original data $\bar{\mathbf{X}}$.

To remove the arbitrary scaling factor and to avoid the embedding undergoing dimensionality collapse, the constraint $\mathbf{Z}^T T\mathbf{Z} = I$ is applied. The embedding problem becomes

$$\mathbf{Z} = \arg\min_{\mathbf{Z}^{*T}T\mathbf{Z}^* = I} tr(\mathbf{Z}^{*T}L\mathbf{Z}^*). \quad (16)$$

The solution is given by the lowest eigenvectors of the generalized eigenproblem

$$L\mathbf{Z} = \Lambda' T\mathbf{Z} \quad (17)$$

and the value of the objective function corresponding to the solution is $\epsilon^* = tr(\Lambda')$.

For the commute time embedding, the objective function minimized is

$$\epsilon' = \frac{\sum_{u,v}\|\mathbf{Z}(u) - \mathbf{Z}(v)\|^2\Omega(u,v)}{\sum_{v=1}^m\sum_{u=1}^n \mathbf{Z}(u,v)^2 d_u} = tr\left(\frac{\mathbf{Z}^T L\mathbf{Z}}{\mathbf{Z}^T T\mathbf{Z}}\right). \quad (18)$$

To show that we achieve the same minimum, let $\mathbf{Z} = \Theta^T = (\sqrt{vol}\Lambda'^{-1/2}\Phi'^T T^{-1/2})^T$, we have

$$\begin{aligned}\epsilon' &= tr\left(\frac{\sqrt{vol}\Lambda'^{-1/2}\Phi'^T T^{-1/2}LT^{-1/2}\Phi'\Lambda'^{-1/2}\sqrt{vol}}{\sqrt{vol}\Lambda'^{-1/2}\Phi'^T T^{-1/2}TT^{-1/2}\Phi'\Lambda'^{-1/2}\sqrt{vol}}\right)\\ &= tr\left(\frac{\Lambda'^{-1/2}\Phi'^T\mathcal{L}\Phi'\Lambda'^{-1/2}}{\Lambda'^{-1/2}\Phi'^T\Phi'\Lambda'^{-1/2}}\right)\\ &= tr\left(\frac{\Lambda'^{-1/2}\Lambda'\Lambda'^{-1/2}}{\Lambda'^{-1}}\right)\\ &= tr(\Lambda') = \epsilon^*.\end{aligned} \quad (19)$$

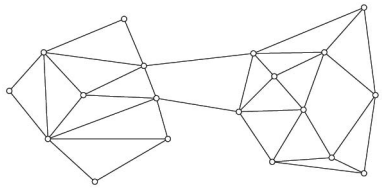

Fig. 1. Original planar graph.

Hence, the commute time embedding not only aims to maintain proximity relationships by minimizing $\sum_{u,v}\|\mathbf{Z}(u) - \mathbf{Z}(v)\|^2\Omega(u,v)$ but also aims to assign large coordinate values to nodes (or points) with a large degree (that is, it maximizes $\sum_{v=1}^m\sum_{u=1}^n \mathbf{Z}(u,v)^2 d_u$). Nodes with a large degree are the most significant in a graph since they have the largest number or total weight of connecting edges. In the commute time embedding, these nodes are farthest away from the origin and are hence unlikely to be close to one-another.

Finally, we note that the objective function appearing in (18) is identical to that used in the normalized cut. To show this, let $\vec{\theta}$ be an $N = |V|$ dimensional binary indicator vector, which determines to which component of the bipartition a node belongs. The minimum value obtained by the normalized cut [37] is

$$\vec{\theta}_1 = \arg\min_{\vec{\theta}^T\mathbf{T}\mathbf{1}=0} \frac{\vec{\theta^T}(\mathbf{T} - \Omega)\vec{\theta}}{\vec{\theta^T}\mathbf{T}\vec{\theta}}. \quad (20)$$

Hence, comparing with (18), it is clear that the objective function minimized by the commute time embedding is exactly the same as that minimized by the normalized cut, provided that the eigenvectors are scaled by the inverse of the corresponding nonzero eigenvalues. In the bipartition case, this does not make any difference since scaling will not change the distribution of the eigenvector components. However, in the multipartition case, the scaling differentiates the importance of different eigenvectors. From (9), it is clear that the eigenvector corresponding to the smallest nonzero eigenvalue contributes the greatest amount to the sum. Moreover, it is this eigenvector or Fiedler vector that is used in the normalized cut to bipartition the graphs recursively. In the case of the commute time embedding, the scaled eigenvectors are used as projection axes for the data. As a result, if we project the data into the commute time embedding subspace, the normalized cut bipartition can be realized by simply dividing the projected data into two along the axis spanned by the Fiedler vector. Further partitions can be realized by projecting and dividing along the axes corresponding to the different scaled eigenvectors.

In Fig. 2, we compare the result of embedding using the Laplacian eigenmap (Fig. 2a) and the commute time embedding (Fig. 2b) on the planar graph shown in Fig. 1. The original graph is constructed by connecting two randomly generated planar graphs with two edges. The graph is unweighted. We project the nodes of the graph onto the plane spanned by the two principal eigenvectors of the mapping. In the figure, it is clear that both embeddings maintain the original graph structure and that the two original graphs are well separated. However, compared to the Laplacian embedding, the points in the two original graphs are more densely distributed by the commute time embedding. In fact, in the case of the commute

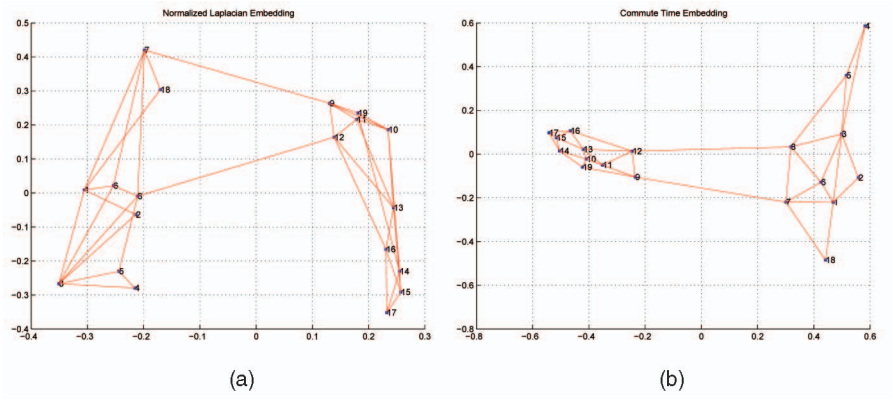

Fig. 2. Graph embedding comparison. (a) Normalized Laplacian embedding. (b) Commute time embedding.

time embedding, the two original graphs are embedded in two orthogonal planes.

3.4 The Commute Time and the Diffusion Map

Finally, it is interesting to note the relationship with the diffusion map embedding of Lafon et al. [22], [25]. The method commences from the random walk on a graph that has transition probability matrix $P = T^{-1}\Omega$, where Ω is the adjacency matrix. Although P is not symmetric, it does have a right eigenvector matrix Ψ, which satisfies the equation

$$P\Psi = \Lambda_P \Psi. \tag{21}$$

Since $P = T^{-1}\Omega = T^{-1}(T - L) = I - T^{-1}L$, as a result,

$$\begin{aligned} (I - T^{-1}L)\Psi &= \Lambda_P \Psi, \\ T^{-1}L\Psi &= (I - \Lambda_P)\Psi, \\ L\Psi &= (I - \Lambda_P)T\Psi, \end{aligned} \tag{22}$$

which is identical to (17) if $\mathbf{Z} = \Psi$ and $\Lambda' = I - \Lambda_P$. The embedding coordinate matrix for the diffusion map is $\Theta_D = \Lambda_P^t \Psi^T$, where t is real. For the embedding, the diffusion distance between a pair of nodes is $D_t^2(u,v) = \sum_{i=1}^{m} (\lambda_P)_i^{2t} (\psi_i(u) - \psi_i(v))^2$. Summing the distance over the possible discrete time steps on the graph, we have

$$\sum_{t=0}^{\infty} D_t^2(u,v) = \sum_{t=0}^{\infty} \sum_{i=1}^{m} (\lambda_P)_i^{2t} (\psi_i(u) - \psi_i(v))^2. \tag{23}$$

Making use of the property of the power series that $\sum_{t=0}^{\infty} (\lambda_P)_i^{2t} = \frac{1}{1-(\lambda_P)_i}$, we have

$$\begin{aligned} \sum_{t=0}^{\infty} D_t^2(u,v) &= \sum_{i=1}^{m} \frac{1}{1-(\lambda_P)_i} (\psi_i(u) - \psi_i(v))^2 \\ &= \sum_{i=1}^{m} \frac{1}{\lambda_i'} (\psi_i(u) - \psi_i(v))^2, \end{aligned} \tag{24}$$

which is identical to (9) up to a constant. As a result, commute time is an integral of the diffusion map over all time. This can be understood using random walks. When t is small, the number of steps a random walk can take is limited by t. Pairs of nodes on the graph can therefore only be linked by relatively short paths, and diffusion takes place only over a very local neighborhood. As t becomes large, diffusion occurs over a larger area, and the random walk takes more steps. As a result, the pairs of nodes become linked by longer paths. In other words, as t goes from zero to infinity, the diffusion map measures the connectivity of a pair of nodes with a specific path length. Commute time, on the other hand, is the sum of the diffusion distance over all possible paths connecting a pair of nodes. An alternative explanation can be provided using the expression for the commute time in (6). Here, the commute time is computed using the sum of the Green's functions for the graph. From (2), the Green's function is an integral of the heat kernel and can be used to compute the diffusion distance.

The diffusion map is designed to give a distance function that reflects the connectivity of the original graph or point set. The distance should be small if a pair of points is connected by many short paths and this is also the behavior of the commute time. The advantage of the diffusion map (or distance) is that it has a free parameter t and this may be varied to alter the properties of the map. The disadvantage is that when t is small, the diffusion distance is ill posed. This can be explained by the original definition of the diffusion distance for a random walk as $D_t^2(u,v) = \|p_t(u,\cdot) - p_t(v,\cdot)\|_{\omega}^2$, where ω is the weight. As a result, the distance between a pair of nodes depends not only on the transition probability between the nodes under consideration but also upon all of the remaining nodes in the graph. Hence, if t is small, then the random walk will not have propagated significantly and the distance will depend only on very local information. There are also problems when t is large. When this is the case, the random walk converges to its stationary state with $P^t = T/vol$ (a diagonal matrix) and this gives zero diffusion distance for all pairs of distinct nodes. Therefore, it is critical to control t carefully in order to obtain useful embeddings.

To illustrate the difficulty of selecting the correct diffusion time t for clustering, we explore a simple example. Fig. 3a shows a nonuniformly distributed set of points on a circle. Here, the density function over the circle is as shown in Fig. 3b and is a bimodal Gaussian mixture with respect to the angle subtended at the center of the circle. From this data set, we construct a graph Γ_P with Gaussian weights $\omega(u,v) = \exp[-\|\mathbf{X}_u - \mathbf{X}_v\|^2/\sigma]$, where $\mathbf{X}$ is the coordinate matrix, and σ is a scale parameter. We embed this graph into a vector space (the diffusion space) using the diffusion map $\Theta_D = \Lambda_P^t \Psi^T$ with diffusion times $t = 1, 16$, and 64. We then partition the embedded points so that the distance between any two points $\Theta_t(u)$ and $\Theta_t(v)$ (where $\Theta_t(u)$ is the uth column of the matrix of embedding coordinates Θ_t for the diffusion map with parameter t) in a given partition satisfies the condition $\|\Theta_t(u) - \Theta_t(v)\| = D_t(u,v) \leq \rho$. The results of embedding

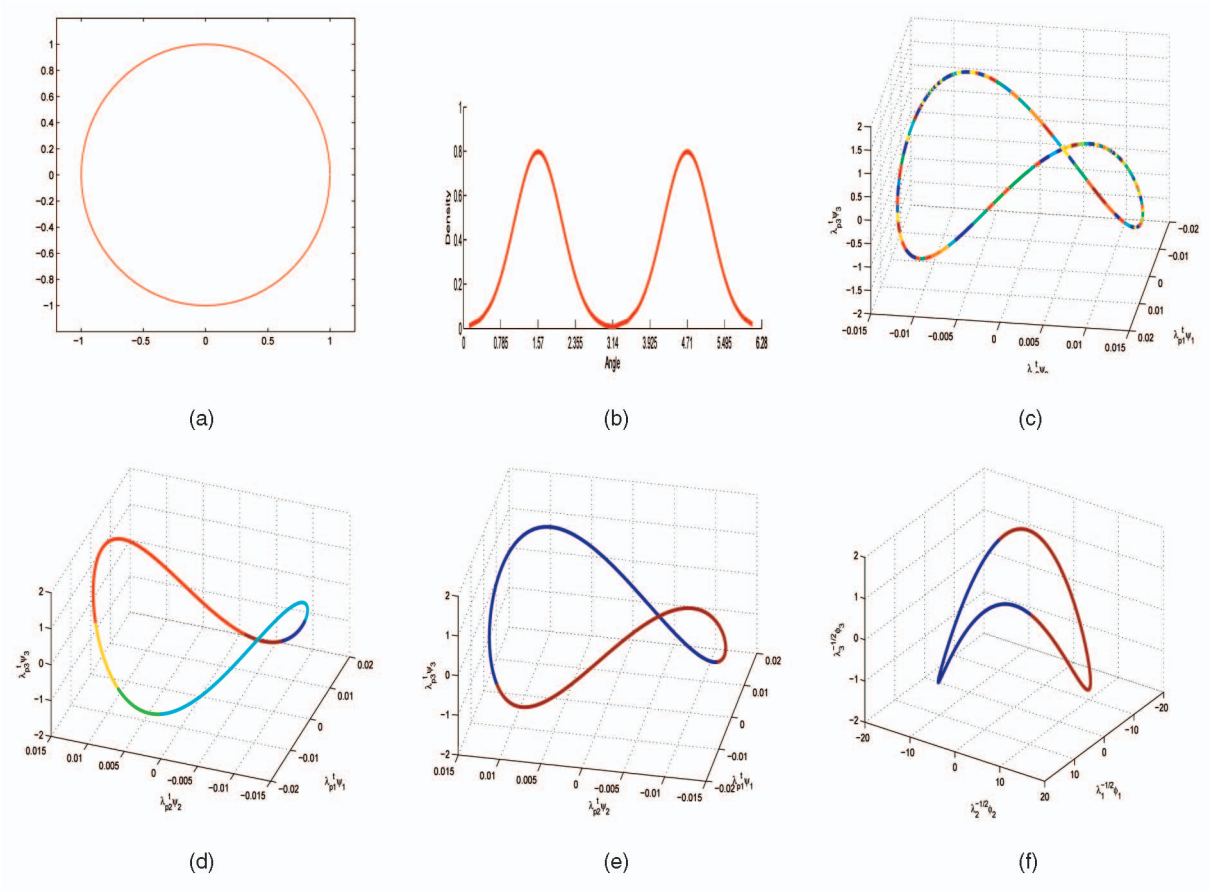

Fig. 3. An embedding comparison between diffusion map and commute time. (a) Original circle. (b) Density of points on the circle. (c) Embedding using diffusion map when $t = 1$. (d) Embedding using diffusion map when $t = 16$. (e) Embedding using diffusion map when $t = 64$. (f) Embedding using commute time.

with different diffusion times are shown in Figs. 3c, 3d, and 3e, respectively. Here, the different colors indicate the different partitions. When the diffusion time is small, as shown in Fig. 3c, the diffusion distance between distinct pairs of points gives rise to many partitions. As time increases, these partitions begin to merge. The largest partitions correspond to the regions of highest point density, as shown in Fig. 3d. When the diffusion time $t = 64$, two large partitions are formed, corresponding to the two ideal components of the Gaussian mixture density. However, if we continue to increase t, these two partitions will merge into a single one once $t > 80$. If we embed the same graph Γ_P using the commute time embedding, we obtain the result shown in Fig. 3f. Without parameter tuning, the commute time embedding gives the required bipartition of the data.

3.5 Some Embedding Examples

In Fig. 4, we show some examples of point configurations and their commute time embeddings. The figure shows four examples. In the left-hand panel for each example, we show the original configuration of the points and in the right-hand panel, we show the embedding. Here, we have computed the proximity weight matrix Ω by exponentiating the euclidean distance between points. The main features to note are the following: First, the embedded points corresponding to the same point clusters are cohesive, being scattered around approximately straight lines in the subspace. Second, the clusters corresponding to different objects give rise to straight lines that are nearly orthogonal. This is due to the strong block-diagonal structure of the affinity matrix (the commute time matrix in this case). Ng et al. [26] have proposed an embedding method using the row-normalized eigenvectors of the affinity matrix. They have proved that in an ideal case, all embedded points will reside on a k-dimensional unit sphere, where k is equal to the number of eigenvectors selected. Points belonging to the same cluster will be located at the same position after normalization. However, if the eigenvectors are not normalized (as in our case), points in the same cluster will be distributed along a radius of the sphere, hence maintaining orthogonality in the remaining clusters. In an ideal case, where all the points in different clusters are infinitely far apart, this gives a strong block-diagonal structure of the affinity matrix.

From (12), we can see that the coordinates of the commute time embedding depend on the eigenvalues and eigenvectors of the Laplacian matrix. Hence, the stability of the embedded coordinates depends on the stability of the eigenvalue and eigenvector matrices. Although the variation of the eigenvalues can be bounded by the maximum and the minimum eigenvalues of the perturbing matrix using Weyl's theorem, the eigenvectors are less stable under perturbation or even differences in the implementation of the eigensolvers. Fortunately, like all other spectral embedding methods, we do not consider the (unstable) individual coordinates but, instead, the subspace spanned by the scaled eigenvectors,

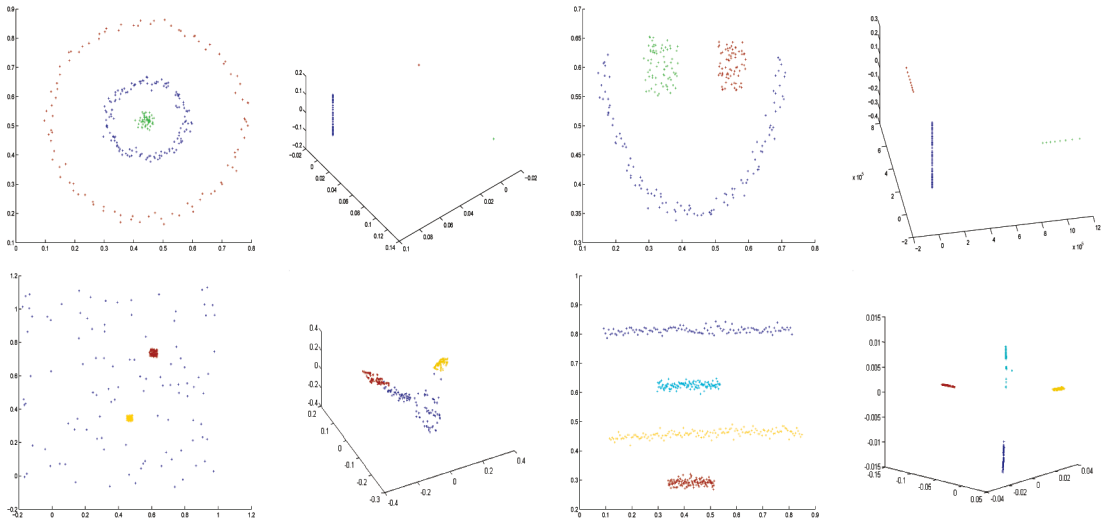

Fig. 4. Commute time embedding examples.

which can be considerably more stable [26]. However, the commute time matrix is likely to be relatively stable under perturbations in graph structure. According to Rayleigh's Principle in the theory of electrical networks, commute time can be neither increased by adding an edge or a node nor decreased by deleting a single edge or a node. In fact, the impact of deleting or adding an edge or a node to the commute time between a pair of nodes is negligible if they are well connected. As we will see later, in the application to motion tracking, this property reduces the impact of outliers, since once embedded, outliers will be excluded from the object point clusters.

4 Applications of Commute Time

We explore two applications of commute time, namely, for image segmentation and multibody motion tracking. In this section, we set up the model ingredients needed for these two applications.

4.1 Commute Time for Grouping

The idea underpinning our segmentation algorithm is to use the spectrum of the commute time matrix for the purposes of grouping. In the normalized cut method, the eigenvector corresponding to the second smallest eigenvalue of the Laplacian matrix is utilized to bipartition data. The method exploits the relatively uniform distribution of the components in the smallest eigenvector.

For the purposes of comparison, here, we use the eigenvector associated with the smallest eigenvalue of the commute time matrix. As we have explained in Section 3.3, a bipartition using commute time embedding does not outperform the normalized cut. However, the eigenvector corresponding to the smallest eigenvalue of the commute time matrix contains more information concerning cluster structure since the complete Laplacian eigenspectrum is taken into account in computing the commute time matrix. Our commute time algorithm consists of the following steps:

1. Given an image, or a point set, set up a weighted graph $\Gamma = (V, E, \Omega)$ where each pixel, or point, is taken as a node, and each pair of nodes is connected by an edge. The weight on the edge is assigned according to the similarity between the two node as follows:
   a. For a point set, the weight between nodes u and v is set to be $\Omega(u,v) = \exp(-d(u,v)/\sigma_x)$, where $d(u,v)$ is the euclidean distance between two points, and σ_x controls the scale of the spatial proximity of the points.
   b. For an image, the weight is
   $$\Omega(u,v) = \exp\left(\frac{-\|\mathbf{F}_u - \mathbf{F}_v\|_2}{\sigma_I}\right) * \begin{cases} \exp\left(\frac{-\|\mathbf{X}_u - \mathbf{X}_v\|_2}{\sigma_X}\right) & \text{if } \|\mathbf{X}_u - \mathbf{X}_v\|_2 < r \\ 0 & \text{otherwise,} \end{cases} \quad (25)$$
   where $\mathbf{F}_u$ is either the intensity value at pixel u for a brightness image or the vector of the red, green, blue (RGB) value for a color image.
2. From the weight matrix Ω, we compute the Laplacian $L = T - \Omega$.
3. Then, we compute the *normalized* Green's function using (3) and the eigenspectrum of the *normalized* Laplacian $\mathcal{L}$.
4. From (7), we compute the commute time matrix CT whose elements are the commute times between each pair of nodes in the graph Γ.
5. Use the eigenvector corresponding to the smallest eigenvalue of the commute time matrix $CT(u,v) = vol \sum_{i=2}^{|V|} \frac{1}{\lambda_i} (\frac{\phi_i'(u)}{\sqrt{d_u}} - \frac{\phi_i'(v)}{\sqrt{d_v}})^2$ to bipartition the weighted graph.
6. Decide if the current partition should be subdivided, and recursively repartition the component parts if necessary.

The major computational overhead in our method arises from Step 3, that is, the computation of the *normalized* Green's function and Step 5, that is, computation of the eigenvector corresponding to the smallest eigenvalue. In (3), the computation of the *normalized* Green's function is realized using the full eigendecomposition of the Laplacian matrix.

The most reliable algorithm for computing the eigenspectrum of an $n \times n$ matrix is by reducing the matrix to a tridiagonal form and then solving the eigenproblem for the tridiagonal matrix. The Householder method is normally used for tridiagonal matrix reduction. The method takes $\frac{4}{3}n^3 + \mathcal{O}(n^2)$ operations (see SSBTRD in **LAPACK**). Typical eigensolvers such as QR have a complexity of $\mathcal{O}(n^3)$ for a tridiagonal eigenproblem. However, this can be reduced to $\mathcal{O}(n^2)$ using the divide-and-conquer (D&C) method or multiple relatively robust representations (MR3). Even so, the overall complexity of solving a full eigenproblem is still $\mathcal{O}(n^3)$. Fortunately, the matrices that concern us here are sparse, since from (25), it is clear that the graphs are only locally connected. In this case, more efficient methods such as the Lanczos method (**ARPACK**) can be used. In the Lanczos method, the eigenspectrum of a $\mathrm{k} \times \mathrm{k}$ symmetric tridiagonal matrix TD is used to approximate the eigenspectrum of the original matrix A. The most expensive part of this method is the construction of the matrix TD, which requires k matrix-vector multiplications with A. If A is sparse, as in our case, the matrix-vector computation is only $\mathcal{O}(n)$. As a result, the overall complexity of the Lanczos method is $\mathcal{O}(kn)$. Since we compute the full eigenspectrum, then $k = n$ and the corresponding complexity rises to $\mathcal{O}(n^2)$. The computations required for Step 5 are similar. Although the commute time matrix becomes dense, we only need to compute the eigenvector corresponding to the smallest eigenvalue. In this case, an iterative eigensolver such as the Lanczos method is much more efficient than a fully diagonalized one (**LAPACK**) and has a complexity $\mathcal{O}(n^2)$.

Compared to the normalized cut method, our commute time *clustering* method is less efficient. This is because the matrix underpinning the normalized cut is sparse and only a few eigenvectors need to be computed. The normalized cut method uses the Lanczos method to solve the eigenproblem in $\mathcal{O}(n^{3/2})$ time. Commute time *embedding*, on the other hand, is similar to the normalized cut and has comparable efficiency. From (11) or (12), the commute time embedding requires the first d eigenvalues and eigenvectors of either the *normalized* or *unnormalized* Laplacian matrix to be computed. Here, d is the dimension of embedded subspace. In all our experiments, $d \leq 3$. Since both the *normalized* and *unnormalized* Laplacian matrices are real, symmetric, and sparse, their first few eigenvalues and eigenvectors can be computed using the Lanczos method in $\mathcal{O}(n^{3/2})$ time.

4.2 Multibody Motion Tracking Using Commute Time

In this section, we will show how the multibody motion tracking problem can be posed as one of commute time embedding using the Q matrix. The method is motivated by the intuition that since the eigenvectors associated with the different objects span different subspaces, they can be embedded using a spectral method and separated using a simple clustering method.

4.2.1 Factorization Method Review

Suppose that there are N objects moving independently in a scene and the movement is acquired by an affine camera as F frames. In each frame, P feature points are tracked and the coordinate of the $i\mathrm{th}$ point in the $f\mathrm{th}$ frame is given by (x_i^f, y_i^f). Let X and Y denote two $F \times P$ matrices constructed from the image coordinates of all the points across all of the frames

$$X = \begin{bmatrix} x_1^1 & x_2^1 & \cdots & x_P^1 \\ x_1^2 & x_2^2 & \cdots & x_P^2 \\ \vdots & \vdots & \ddots & \vdots \\ x_1^F & x_2^F & \cdots & x_P^F \end{bmatrix} \quad Y = \begin{bmatrix} y_1^1 & y_2^1 & \cdots & y_P^1 \\ y_1^2 & y_2^2 & \cdots & y_P^2 \\ \vdots & \vdots & \ddots & \vdots \\ y_1^F & y_2^F & \cdots & y_P^F \end{bmatrix}.$$

Each row in the two matrices above corresponds to a single frame, and each column corresponds to a single point. The two coordinate matrices can be stacked to form the matrix $W = \left[\frac{X}{Y}\right]_{2F \times P}$.

The W matrix can be factorized into a motion matrix M and a shape matrix S; thus, $W_{2F \times P} = M_{2F \times r} \times S_{r \times P}$, where r is the rank of W ($r = 4$ in the case of W without noise and outliers). The intuition underpinning the factorization method is that using shape information alone, image features can be segmented or grouped into individual objects based on their shape properties. In order to solve the factorization problem, matrix W can be decomposed using SVD as $W = U\Sigma R^T$. If the features from the same object are grouped together, then U, Σ, and R will have a block-diagonal structure

$$W = [U_1 \cdots U_N] \begin{bmatrix} \Sigma_1 & & \\ & \ddots & \\ & & \Sigma_N \end{bmatrix} \begin{bmatrix} R_1^T & & \\ & \ddots & \\ & & R_N^T \end{bmatrix}$$

and the shape matrix for object k can be approximated by $S_k = B^{-1}\Sigma_k R_k^T$, where B is an invertible matrix that can be found from M.

In a real multibody tracking problem, the coordinates of the different objects are potentially permuted into a random order. As a result, it is impossible to correctly recover the shape matrix S_k without knowledge of the correspondence order. Since the eigenvector matrix V is related to the shape matrix, the shape interaction matrix was introduced by Costeira and Kanade [7], [8] to solve the multibody separation problem. The shape interaction matrix is

$$Q = RR^T = \begin{bmatrix} S_1^T\Sigma_1^{-1}S_1 & 0 & \cdots & 0 \\ 0 & S_2^T\Sigma_2^{-1}S_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & S_N^T\Sigma_N^{-1}S_N \end{bmatrix}. \quad (26)$$

From (26), the shape interaction matrix Q has the convenient properties that $Q(u, v) = 0$ if points u and v belong to different objects and $Q(u, v) \neq 0$ if points u and v belong to the same object. The matrix Q is also invariant to both the object motion and the selection of the object coordinate systems. This leads to a simple scheme for separating multiobject motions by permuting the elements of Q so that it acquires a block-diagonal structure. In Costeira and Kanade's method [7], [8], a greedy algorithm is used to permute the Q matrix into block-diagonal form. An illustration is shown in Fig. 5. Fig. 5a shows the set of original points together with their trails, Fig. 5b the Q matrix for these points, Fig. 5c the result of applying Costeira and Kanade's method to sort the Q matrix, and Fig. 5d the result of separating the points into moving objects. This method works well only for the ideal case, where there is no noise, and outliers are not present. In Figs. 5e and 5f, we, respectively, show the effect of adding Gaussian noise to the Q matrix in 5b and the resulting permuted matrix. In this noisy case, the block structure is badly corrupted and object separation is almost impossible.

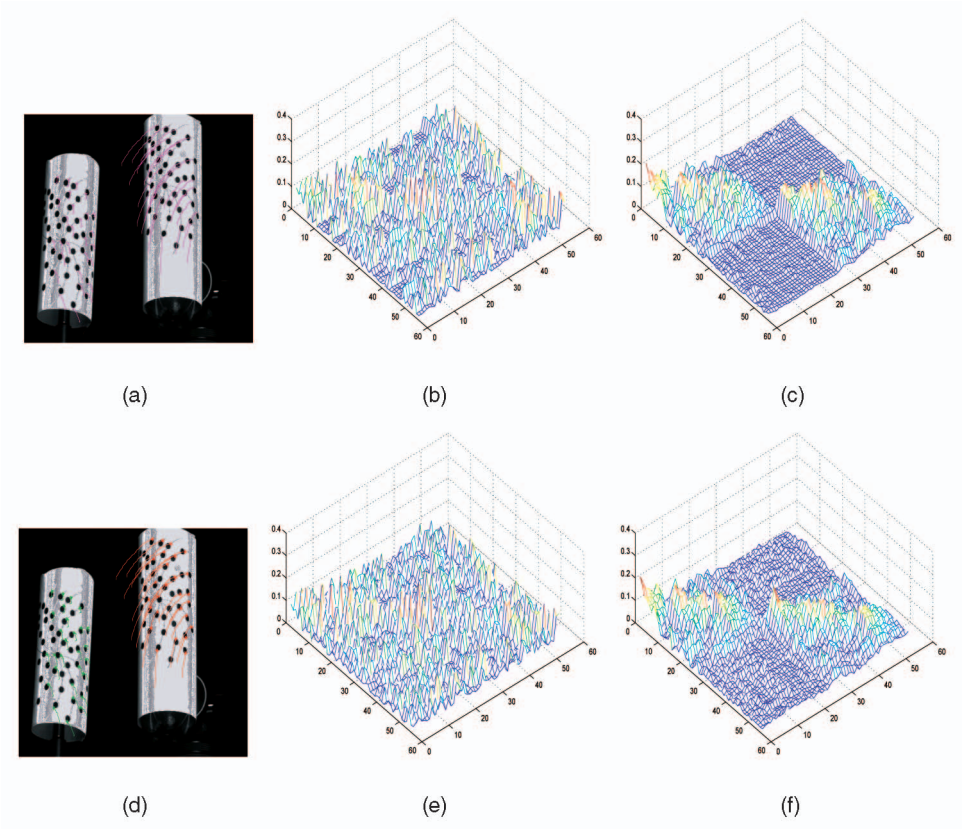

Fig. 5. A multibody motion separation example using Costeira and Kanade's method. (a) Original picture with trails of the moving feature points. (b) Original Q matrix unsorted. (c) Sorted Q by Costeira and Kanade's method. (d) Object separation result. (e) Q matrix with Gaussian noise $\sigma = 0.8$. (f) Sorted Q with noise.

4.2.2 Commute Time Formulation

Having discussed some of the properties of the commute time embedding, in this section, we will show how it may be used for multibody motion analysis.

As we have already seen, the shape interaction matrix Q introduced in the factorization method is invariably contaminated by noise and this limits its effectiveness. Our aim is to use commute time as a shape separation measure. Specifically, we use the commute time to refine the block structure of the Q matrix and group the feature points into objects.

The idea is to embed the feature points into a subspace in such a way that feature points belonging to the same object will remain in close proximity. This can be realized by commute time embedding. When commute time is preserved, points belonging to the same object will be in close proximity since they have a smaller commute time. On the other hand, noise and outliers will be spread sparsely and can be eliminated by a simple clustering method such as K-Means. An alternative approach to embedding feature points is to use the eigenvectors of the commute time matrix. As we have seen in Section 4.1, the eigenvectors of the commute time matrix contain cluster information and can be used for grouping points into partitions. The reason why we use the commute time embedding here are twofold: First, the euclidean distance in the embedding subspace approximates the commute time. Second, the commute time embedding is more efficient since it requires that only the Laplacian eigenspectrum be computed and not the additional eigendecomposition of the commute time matrix. In principle, embedding using eigenvectors of the commute time matrix will give finer clusters. This is because the commute time matrix is more block diagonal than the Laplacian.

Object Separation Steps. The algorithm we propose for this purpose has the following steps:

1. Use the shape interaction matrix Q as the weighted adjacency matrix Ω, and construct the corresponding graph Γ.
2. Compute the Laplacian matrix of graph Γ using $L = T - Q$.
3. Find the eigenvalue matrix Λ and eigenvector matrix Φ of L using $L = \Phi\Lambda\Phi^T$.
4. Embed the commute time into a subspace of R^n using (11) or (12).
5. Cluster the data points in the subspace using the K-Means algorithm [23].

To illustrate the effectiveness of this method, we return to the example used earlier in Fig. 5. First, in the ideal case, the Q matrix will have a zero value for the feature points belonging to different objects. As a result, the graph Γ, constructed from Q, will have disjoint subgraphs corresponding to the nodes belonging to different objects. The partitions give rise to infinite commute times and are hence unreachable by the random walk. However, when we add noise (Q with zero mean and standard derivation 0.8 Gaussian noise) and the clustering steps listed above, we still recover a good set of objects (see Fig. 5d). This is illustrated in Fig. 6. Here, Fig. 6a shows the commute time matrix of graph Γ and Fig. 6b shows the embedding in a 3D subspace. It is clear that the commute

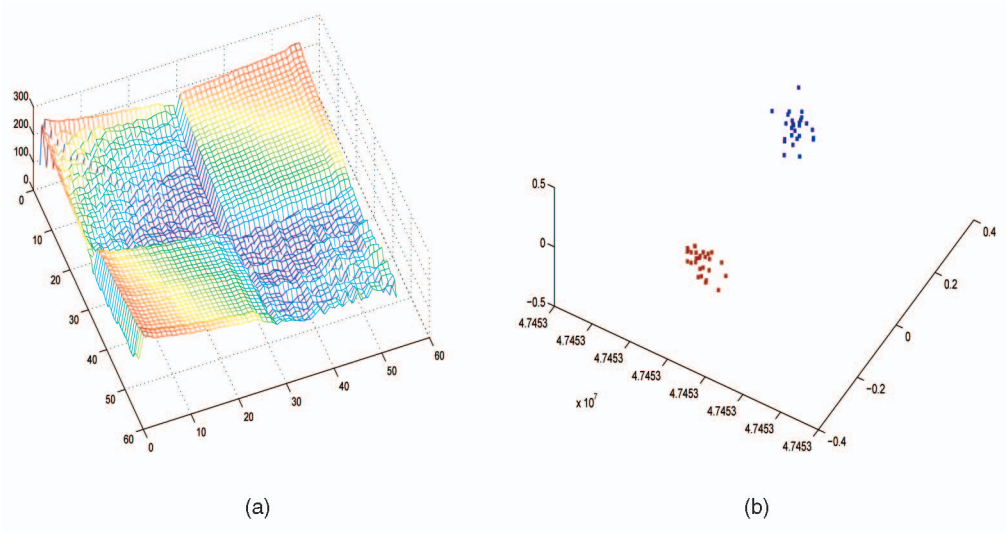

Fig. 6. Multibody motion separation recast as a commute time clustering problem. (a) Sorted commute time matrix. (b) Clustered points in the commute time subspace for two objects.

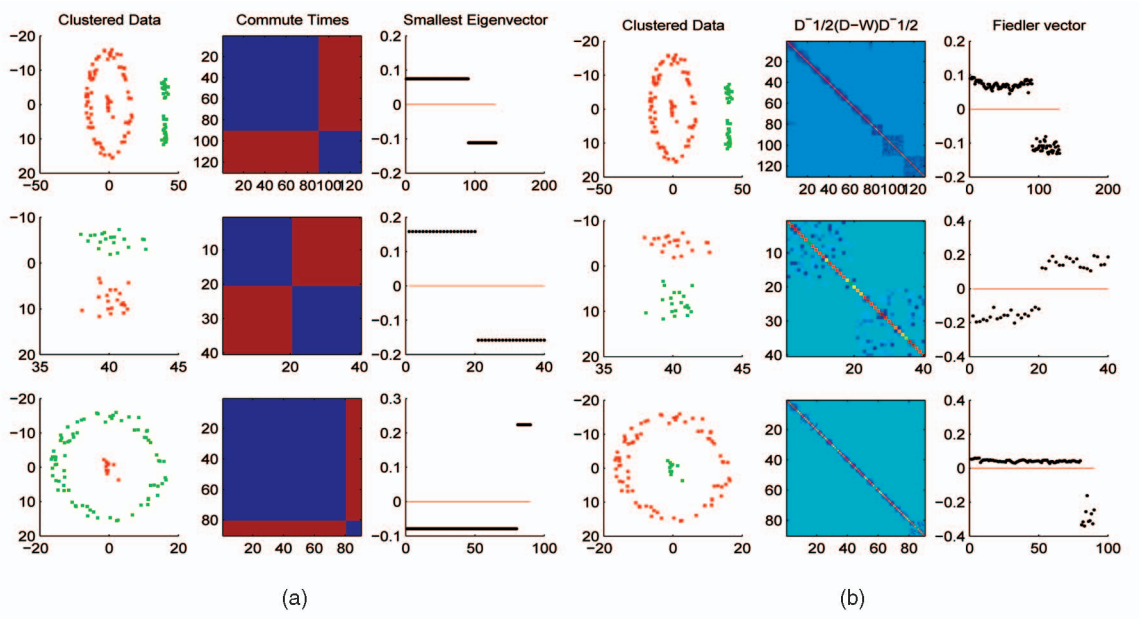

Fig. 7. Clustering examples. (a) Data clustering by commute time. (b) Data clustering by normalized cut.

time matrix gives a good block-diagonal structure and the points are well clustered in the embedding space even when significant noise is present.

5 Experiments

In this section, we describe our experiments with the two applications of commute time. First, we show results on clustering and image segmentation, and then, we show motion tracking results on synthetic and real-world videos sequences.

5.1 Image Segmentation and Data Clustering

5.1.1 Point-Set Clustering Examples

In Figs. 7a and 7b, we, respectively, show and compare the results obtained for point-set clustering using commute times and the normalized cut. Here, we set $\sigma = 1.5$. The subfigures in both figures are organized as follows: The leftmost column shows the point sets, the middle column the affinity matrices, and the rightmost column the components of the smallest eigenvector. The first row shows the first bipartition on the original data. From this bipartition, we obtain two separate clusters, and using each of them, we perform a second bipartition. The second bipartition results are shown in the second and third rows of Figs. 7a and 7b. In the figures, it is clear that both methods succeeded in grouping the data. However, the commute time method outperforms the normalized cut since its affinity matrix has a stronger block structure and the distribution of the smallest eigenvector components is more stable. Moreover, its jumps, corresponding to the different clusters in the data, are larger. Since the eigenvector is taken as an indicator for the membership of the cluster, the more differentiated the distribution of the components of this eigenvector, the closer the relaxed solution is toward the desired discrete solution. This point

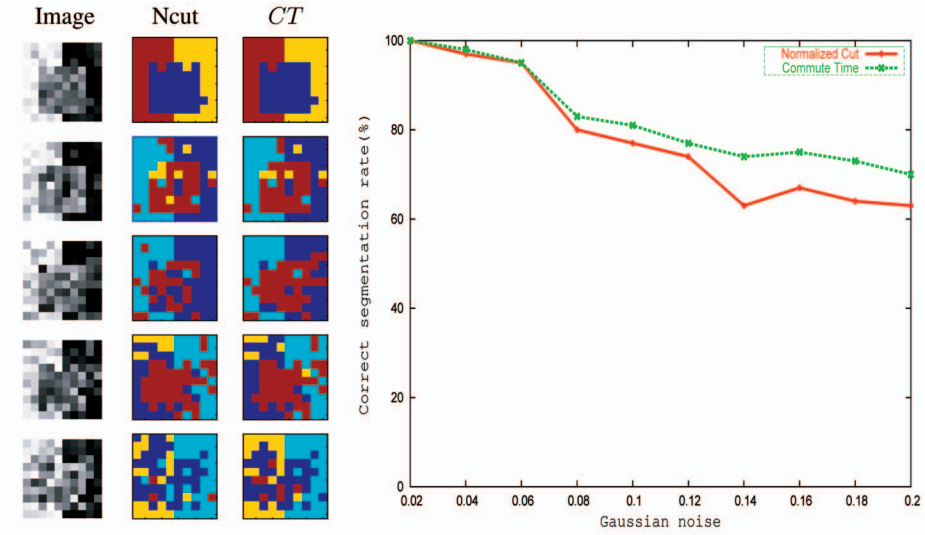

Fig. 8. Method comparison for a synthetic image with increasing Gaussian noise.

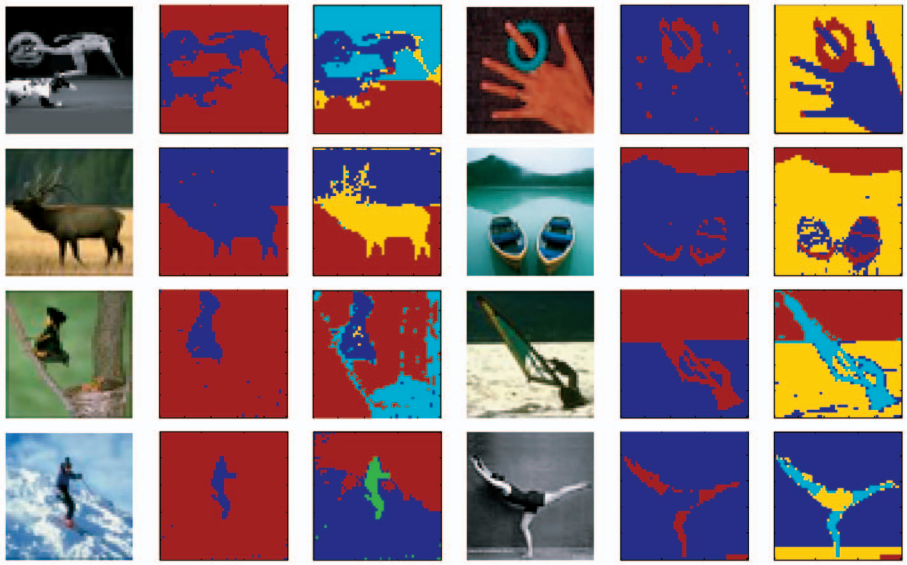

Fig. 9. Real-world segmentation examples.

is well illustrated in the third column of Fig. 7a compared to the one in Fig. 7b. In the figures, it is clear that the eigenvector delivered by our commute time matrix is strongly bimodal. This is due to the strong block structure of the commute time matrix as illustrated in the middle of Fig. 7a compared to the normalized affinity matrix in Fig. 7b.

5.1.2 Image Segmentation

We have compared our new method with that of Shi and Malik [37] on synthetic images subject to additive Gaussian noise. On the left-hand side of Fig. 8, we show the results of using these two methods for segmenting a synthetic image composed of three rectangular regions with additive (zero mean and standard derivation increasing evenly from 0.04 to 0.20) random Gaussian noise. On the right-hand side of Fig. 8, we show the fraction of pixels correctly assigned as a function of the noise standard derivation. At the highest noise levels, our method outperforms the Shi and Malik method by about 10 percent.

In Fig. 9, we show eight real-world images (from the Berkeley image database) with the corresponding segmentation results. The images are scaled to be 50×50 in size, and the parameters used for producing the results are $r = 5$, $\sigma_I = 0.02$, and $\sigma_X = 0.2$. In each set of the images, the leftmost panel shows the original image. The middle and rightmost panels show the results from two successive bipartitions.

For four of the real images, we compare our method with the normalized cut in Figs. 10 and 11. The first column of each subfigure shows the first, second, and third bipartitions of the images. The second column shows the histogram of the components of the smallest eigenvector, and the third column shows the distribution of the eigenvector components. The blue and red lines in the third column, respectively, correspond to zero and the eigenvector component threshold.

Comparing the segmentation results in the first column, it is clear that commute time outperforms the normalized cut in maintaining both region integrity and continuity. For

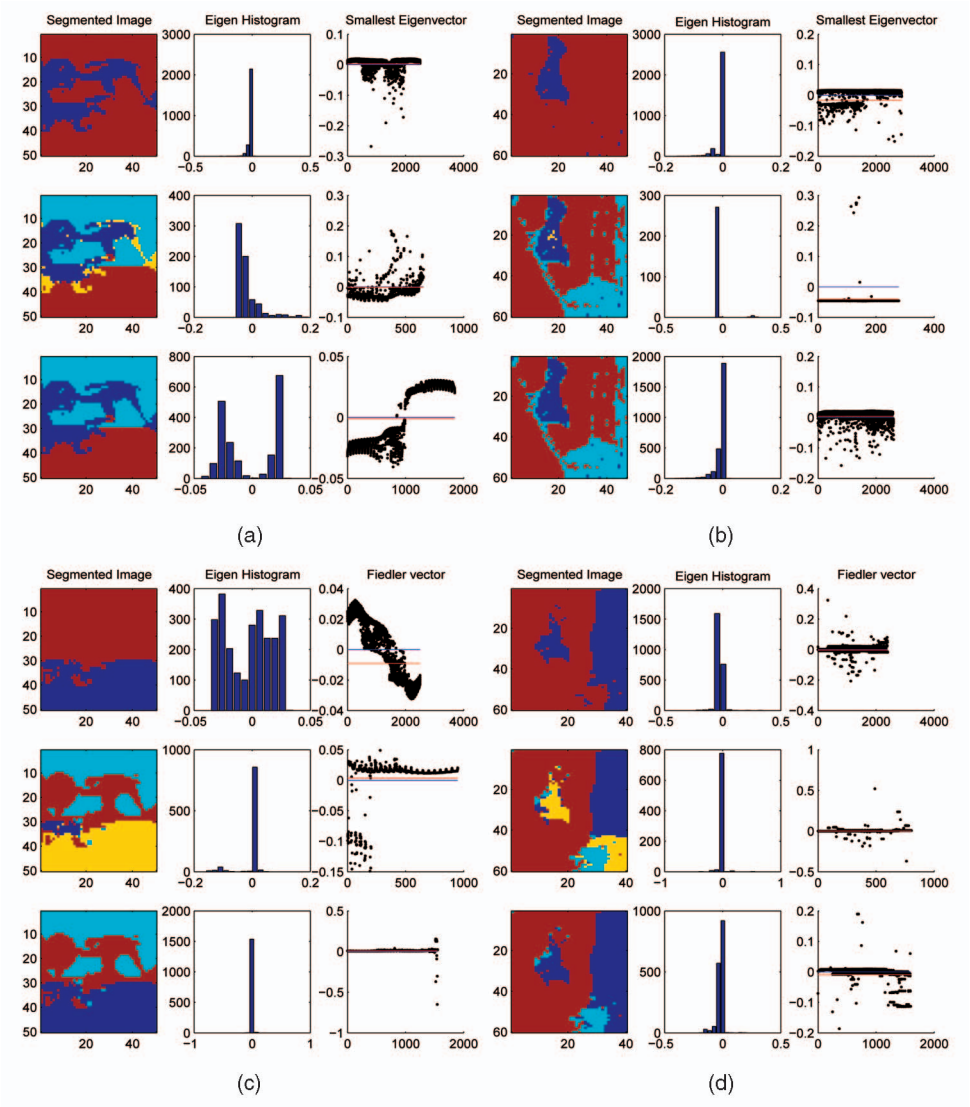

Fig. 10. Detailed segmentation process comparison. (a) Commute time for a 50×50 image with $r = 8$, $\sigma_X = 0.5$ and $\sigma_I = 0.1$. (b) Commute time for a 60×40 image with $r = 5$, $\sigma_X = 0.2$ and $\sigma_I = 0.02$. (c) Normalized cut for a 50×50 image with $r = 5$, $\sigma_X = 2$ and $\sigma_I = 0.05$. (d) Normalized cut for a 60×40 image with $r = 5$, $\sigma_X = 0.05$ and $\sigma_I = 0.01$.

instance, in the case of the baseball player, the background trademark and the limbs of the players are well segmented. In the case of the bird, the thin tree branch is detected. For the astronaut, the boundary between space and the earth is detected. Finally, for the hand, the finger nails and ring are correctly segmented by the commute time method. Another important feature is that, once again, the eigenvector distribution is more stable and discriminates more strongly between clusters. This is illustrated in the second and third columns of Figs. 10 and 11, where the distribution of eigenvector components in the histograms is better separated for the commute time method. Hence, the corresponding cluster indicators give better separation.

5.2 Multibody Motion Tracking Problem

In this section, we conduct experiments with the commute time embedding method on both synthetic data and real-world motion tracking problems. To investigate the robustness of the method, we add Gaussian noise to the data sets and compare the results with some classical methods.

5.2.1 *Synthetic Data*

Fig. 12 shows a sequence of five consecutive synthetic images with 20 background points (green dots) and 20 foreground points (red dots) moving independently. We have added a Gaussian noise of zero mean and standard deviation σ to the coordinates of these 29 points and then clustered them into two groups.

We have compared our method with Costeira and Kanade's greedy algorithm [7], [8], Ichimura's discrimination criterion method [18], and Kanatani's subspace separation method [19]. In Fig. 13, we plot the average misclassification ratio as a function of σ for different algorithms. The results are based on the averages of 50 trials for each method. In the figure, it is clear that our method performs significantly better than the greedy method [8] and the discrimination criterion method [18]. It also has a margin of advantage over the subspace separation method [19].

For an example with a Gaussian noise with $\sigma = 0.5$, the commute time matrix and the embedded subspace are shown in Figs. 13b and 13c, respectively. It is clear that even in this

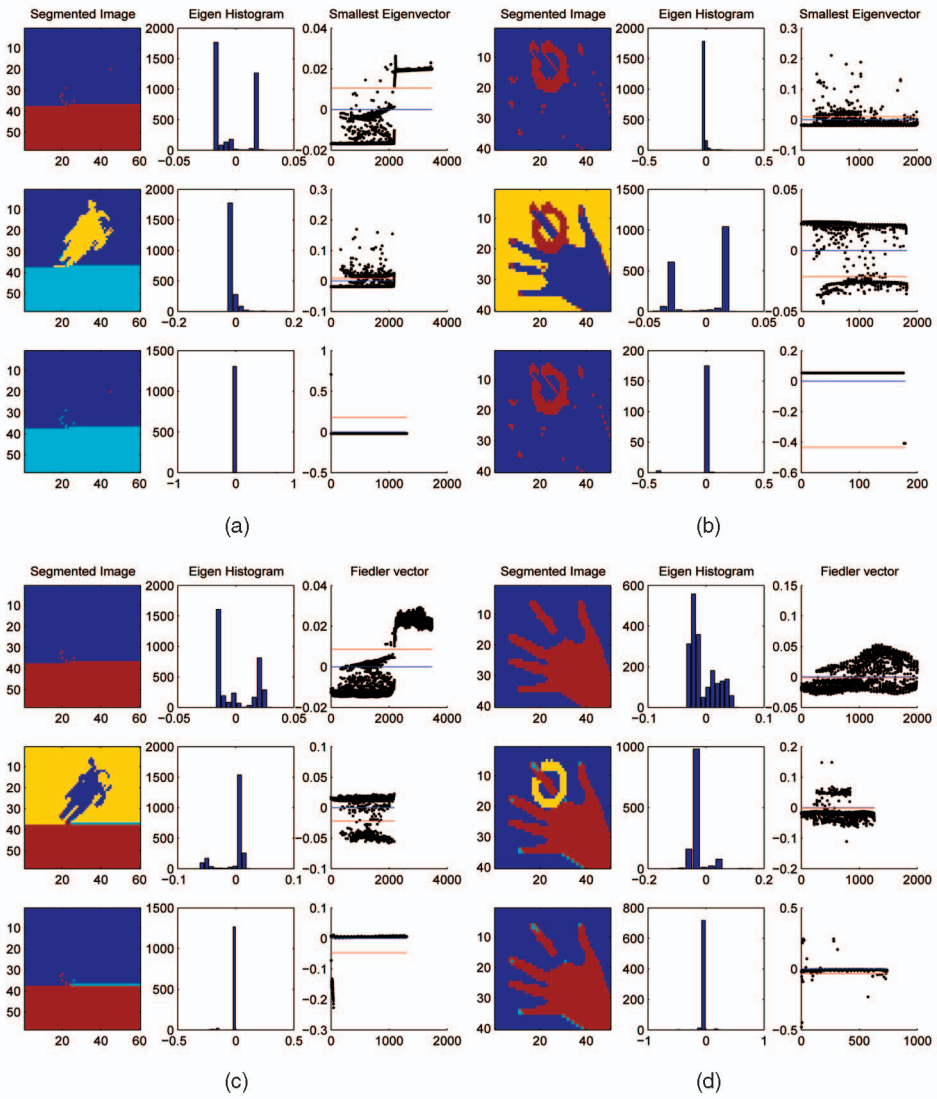

Fig. 11. Detailed segmentation process comparison. (a) Commute time for a 60×58 image with $r = 5$, $\sigma_X = 0.1$ and $\sigma_I = 0.03$. (b) Commute time for a 50×40 image with $r = 10$, $\sigma_X = 0.1$ and $\sigma_I = 0.03$. (c) Normalized cut for a 60×58 image with$r = 5$, $\sigma_X = 0.1$ and $\sigma_I = 0.03$. (d) Normalized cut for a 50×40 image with $r = 5$, $\sigma_X = 5$ and $\sigma_I = 0.02$.

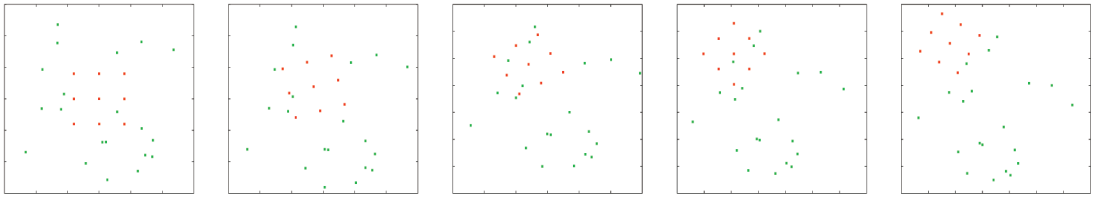

Fig. 12. Synthetic image sequence.

heavily noise-contaminated case, the commute time matrix still maintains a good block-diagonal structure. Moreover, under the embedding, the points are easily separated.

5.2.2 Real-World Motion Tracking

In this section, we experiment with the commute time method on real-world multibody motion tracking problems. The columns in Fig. 14 show five real-world video sequences overlaid with the successfully tracked feature points using the commute time method. The full sequences can be found in the supplementary material Web site.

The first three columns are for the data used by Sugaya and Kanatani in [39], [40]. Here, there is one moving object and a moving camera. A successful tracking method will separate the moving object from the moving background. The fourth and fifth columns in Fig. 14 are two video sequences captured using a FujiFilm 2.0M camera (320×240 pixels). For each of sequence, we detected feature points using the Kanade-Lucas-Tomasi (KLT) algorithm [38] and tracked the feature points using the commute time method. Due to the continuous loss of the feature points in the successive frames by the KLT algorithm, we use only 10 frames each from the sequences with 117 and 116 feature points, respectively.

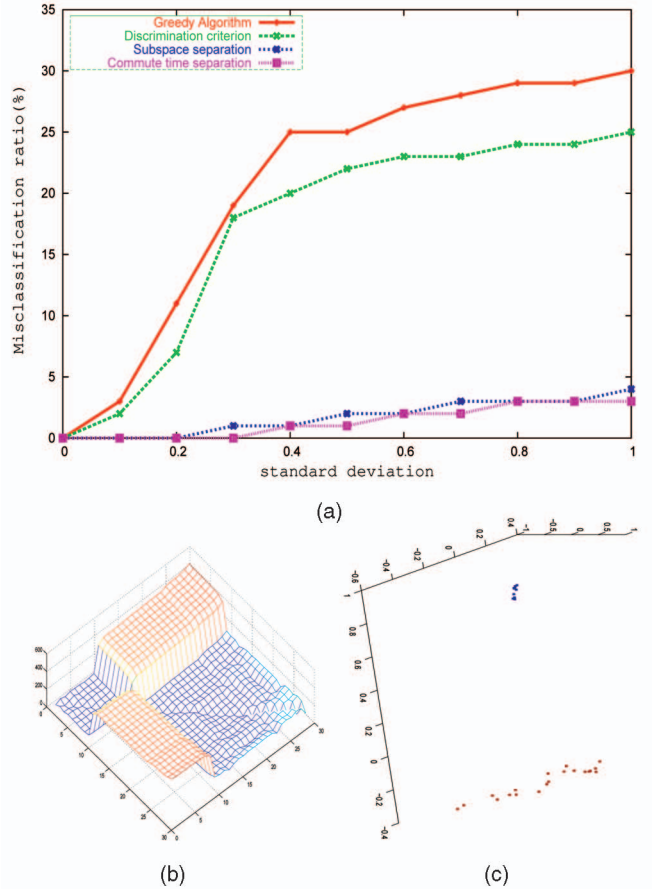

Fig. 13. Synthetic data. (a) Method comparison. (b) Sorted commute time matrix. (c) Embedded subspace.

Compared to the data from Sugaya and Kanatani [39], [40], we increase the number of detected moving objects from one to two, which makes the separation more difficult.

In the case of the forth column in Fig. 14, our method not only separates the ducks correctly from the moving background but also separates the moving ducks from each other. The fifth column in Fig. 14 is the most difficult one with two independently moving hands and a moving background. It also separates the wall from the floor correctly.

In Fig. 15, we show the trajectories for the tracked points in each of the video sequences. Here, the outliers are successfully removed. The different sequences offer tasks of increasing difficulty. The easiest sequence is the one labeled **A**, where the background has a uniform and almost linear relative movement, and the foreground car follows a curved trajectory. There is a similar pattern in the sequence labeled **B**, but, here, the background movement is more significant. In sequence **C**, there is both camera pan and abrupt object movement. Sequence **D** has camera pan and three independently moving objects. Finally, in sequence **E**, there is background jitter (due to camera shake) and two objects exhibiting independent overall movements together with articulations. In Fig. 16, we show the embeddings of the tracked points for the sequences. The feature to note is that the different moving objects form distinct clusters and are well separated from the background. The color coding scheme used in the plot is the same as that used in the rows in Fig. 14.

For the same sequences, we compared our results with Costeira and Kanade's greedy algorithm [8], Ichimura's discrimination criterion method [18], Kanatani's subspace separation method [19], and Sugaya and Kanatani's multistage learning [40]. The comparison is shown in Table 1.

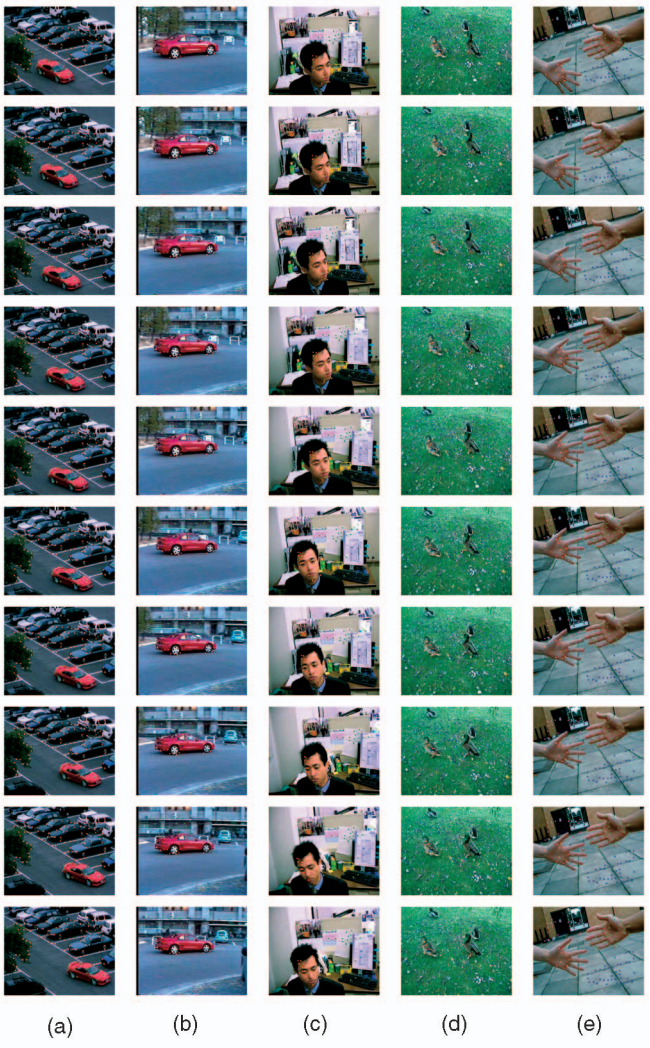

Fig. 14. Real-world video sequences and successfully tracked feature points.

Table 1 lists the accuracies of the different methods using the ratio of the number of correctly classified points to the total number of points. The ratio is averaged over 50 trails for each method. In the table, it is clear that the greedy algorithm [8] gives the worst results. This is because the greedy algorithm simply sorts according to the magnitude of elements of the Q matrix and this matrix is susceptible to noise. The discrimination criterion method [18] and the subspace separation method [19] perform better due to their robustness to the noise. The discrimination criterion method effectively rejects noise and outliers by selecting the most reliable features. The subspace separation method removes outliers by fitting a subspace only to consistent trajectories.

The multistage learning method [40] delivers significantly better results due to its adaptive capabilities but failed on our data. The failures are most pronounced when there are several moving objects and an inconsistent moving background. Our method gives the best performance and achieves 100 percent accuracy. In our method, motion jitter or noise disturbance will be correctly recognized and suppressed by

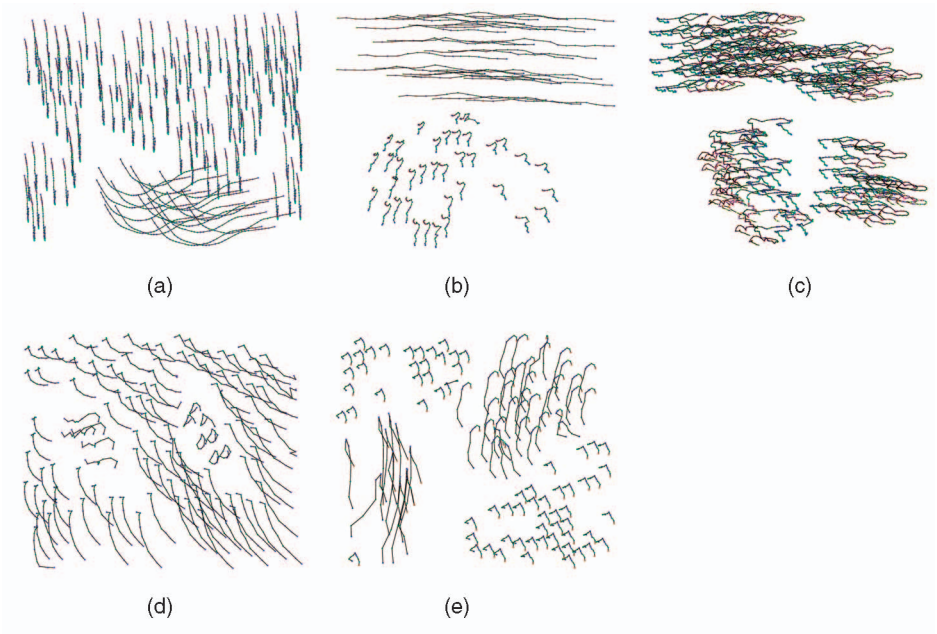

Fig. 15. Feature point trajectories.

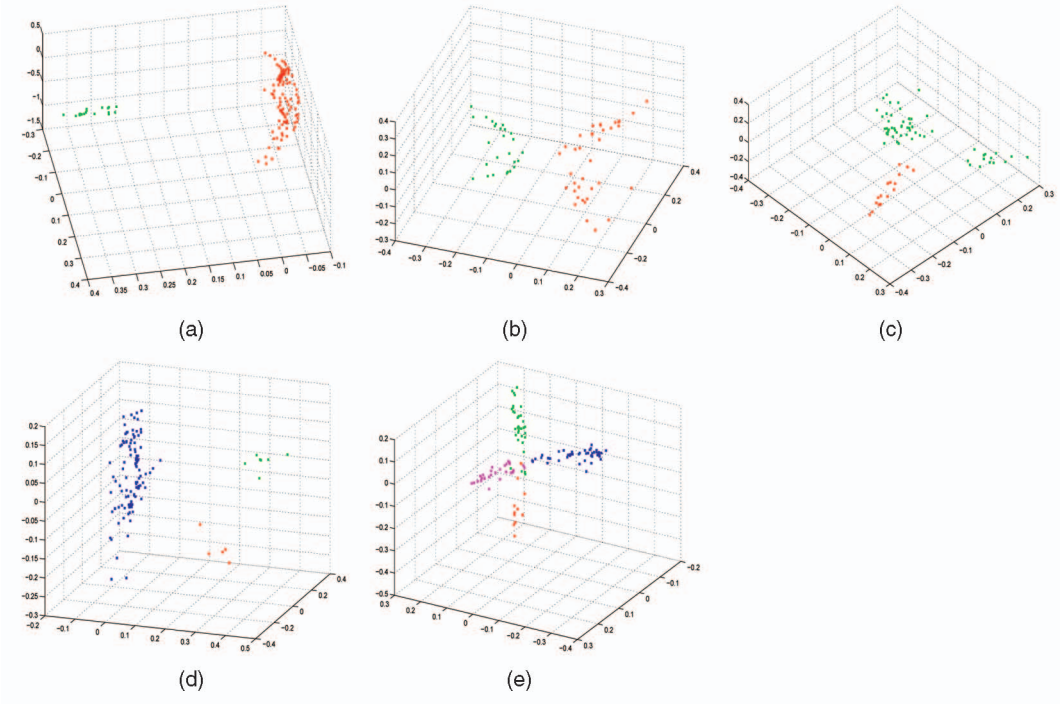

Fig. 16. Sequences embedded by commute time in the subspace.

the embedding process. Outliers, on the other hand, are automatically rejected in the clustering step by the K-Means algorithm.

6 Conclusion

In this paper, we have explored the use of commute time for image segmentation problems in computer vision. We commenced by reviewing some of the properties of commute time and its relationship with the Laplacian spectrum. This analysis relied on the discrete Green's function of the graph. Two of the most important properties are that the Green's function is a kernel and that the commute time is a metric.

With the mathematical definitions of commute time at hand, we have analyzed the properties of the commute time embedding. This allows us to understand the links between the commute time and alternative methods such as the normalized cut and the diffusion map.

We have explored two applications of the commute time. The first of these is image segmentation and the second is multibody motion tracking. Both methods are proven to outperform alternatives in terms of their ability to separate the input data into cleanly separated clusters.

TABLE 1
Separation Accuracy for the Sequences in Fig. 14

	A	B	C	D	E
Costeira-Kanade	60.3	71.3	58.8	45.5	30.0
Ichimura	92.6	80.1	68.3	55.4	47.2
Subspace Separation	59.3	99.5	98.9	80.6	67.2
Multi-stage Learning	100.0	100.0	100.0	93.7	81.5
Commute Time Separation	**100.0**	**100.0**	**100.0**	**100.0**	**100.0**

There are a number of ways in which the work described in this paper can be extended. First, we would like to perform a detailed matrix perturbation analysis to better understand the stability properties of commute time. Second, we would like to extend the methods reported here to discrete graph structures and to see if they lend themselves to higher level image analysis tasks such as object recognition and shape indexing.

ACKNOWLEDGMENTS

The work reported in this paper was undertaken while Huaijun Qiu was a PhD student at the University of York.

REFERENCES

[1] P. Anandan and M. Irani, "Factorization with Uncertainty," *Int'l J. Computer Vision,* vol. 49, no. 2-3, pp. 101-116, 2002.
[2] M. Belkin and P. Niyogi, "Laplacian Eigenmaps and Spectral Techniques for Embedding and Clustering," *Advances in Neural Information Processing Systems,* pp. 585-591, 2001.
[3] M. Belkin and P. Niyogi, "Laplacian Eigenmaps for Dimensionality Reduction and Data Representation," *Neural Computation,* vol. 15, no. 6, pp. 1373-1396, 2003.
[4] F.R.K. Chung, *Spectral Graph Theory,* Am. Math. Soc. ed., CBMS 92, 1997.
[5] F.R.K. Chung and S.T. Yau, "Discrete Green's Functions," *J. Combinatorial Theory Series,* pp. 191-214, 2000.
[6] R.R. Coifman, S. Lafon, A.B. Lee, M. Maggioni, B. Nadler, F. Warner, and S.W. Zucker, "Geometric Diffusions as a Tool for Harmonic Analysis and Structure Definition of Data: Diffusion Maps," *Proc. Nat'l Academy of Sciences,* vol. 102, no. 21, pp. 7426-7431, 2005.
[7] J. Costeira and T. Kanade, "A Multi-Body Factorization Method for Motion Analysis," *Proc. Fifth Int'l Conf. Computer Vision,* pp. 1071-1076, 1995.
[8] J. Costeira and T. Kanade, "A Multibody Factorization Method for Independently Moving Objects," *Int'l J. Computer Vision,* vol. 29, no. 3, pp. 159-179, 1997.
[9] Z. Fan, J. Zhou, and Y. Wu, "Inference of Multiple Subspaces from High-Dimensional Data and Application to Multibody Grouping," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 661-666, 2004.
[10] Z. Fan, J. Zhou, and Y. Wu, "Multibody Motion Segmentation Based on Simulated Annealing," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 776-781, 2004.
[11] I. Fischer and J. Poland, "Amplifying the Block Matrix Structure for Spectral Clustering," technical report, Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), 2005.
[12] F. Fouss, L. Yen, A. Pirotte, and M. Saerens, "An Experimental Investigation of Graph Kernels on Two Collaborative Recommendation Tasks," *Proc. Sixth IEEE Int'l Conf. Data Mining,* 2006.
[13] C.W. Gear, "Multibody Grouping from Motion Images," *Int'l J. Computer Vision,* vol. 29, no. 2, pp. 130-150, 1998.
[14] A. Gruber and Y. Weiss, "Multibody Factorization with Uncertainty and Missing Data Using the EM Algorithm," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 707-714, 2004.
[15] L.W. Hagen and A.B. Kahng, "New Spectral Methods for Ratio Cut Partitioning and Clustering," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems,* vol. 11, no. 9, pp. 1074-1085, 1992.
[16] X. He and P. Niyogi, "Locality Preserving Projections," *Proc. Neural Information Processing System Conf.,* pp. 585-591, 2003.
[17] H. Hotelling, "Analysis of Complex Statistical Variables in Principal Components," *J. Educational Psychology,* vol. 24, pp. 417-441, 498-520, 1933.
[18] N. Ichimura, "Motion Segmentation Based on Factorization Method and Discriminant Criterion," *Proc. Seventh IEEE Int'l Conf. Computer Vision,* pp. 600-605, 1999.
[19] K. Kanatani, "Motion Segmentation by Subspace Separation and Model Selection," *Proc. Eighth IEEE Int'l Conf. Computer Vision,* pp. 301-306, 2001.
[20] R. Kondor and J. Lafferty, "Diffusion Kernels on Graphs and Other Discrete Structures," *Proc. 19th Int'l Conf. Machine Learning,* 2002.
[21] J.B. Kruskal and M. Wish, *Multidimensional Scaling.* Sage Publications, 1978.
[22] S. Lafon and A.B. Lee, "Diffusion Maps and Coarse-Graining: A Unified Framework for Dimensionality Reduction, Graph Partitioning, and Data Set Parameterization," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 28, no. 9, pp. 1393-1403, Sept. 2006.
[23] J.B. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," *Proc. Fifth Berkeley Symp. Math. Statistics and Probability,* pp. 281-297, 1967.
[24] M. Meilă and J. Shi, "A Random Walks View of Spectral Segmentation," *Proc. Neural Information Processing Systems Conf.,* pp. 873-879, 2000.
[25] B. Nadler, S. Lafon, R.R. Coifman, and I.G. Kevrekidis, "Diffusion Maps, Spectral Clustering and Eigenfunctions of Fokker-Planck Operators," *Proc. Neural Information Processing Systems Conf.,* 2005.
[26] A. Ng, M. Jordan, and Y. Weiss, "On Spectral Clustering: Analysis and an Algorithm," *Proc. Neural Information Processing Systems Conf.,* 2001.
[27] J. Park, H. Zha, and R. Kasturi, "Spectral Clustering for Robust Motion Segmentation," *Proc. European Conf. Computer Vision,* pp. 390-401, 2004.
[28] M. Pavan and M. Pelillo, "Dominant Sets and Hierarchical Clustering," *Proc. Ninth IEEE Int'l Conf. Computer Vision,* pp. 362-369, 2003.
[29] M. Pavan and M. Pelillo, "A New Graph-Theoretic Approach to Clustering and Segmentation," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition ,* vol. I, pp. 145-152, 2003.
[30] P. Perona and W.T. Freeman, "A Factorization Approach to Grouping," *Proc. European Conf. Computer Vision,* pp. 655-670, 1998.
[31] H. Qiu and E.R. Hancock, "Image Segmentation Using Commute Times," *Proc. British Machine Vision Conf.,* pp. 929-938, 2005.
[32] S. Roweis and L. Saul, "Nonlinear Dimensionality Reduction by Locally Linear Embedding," *Science,* vol. 290, no. 5500, pp. 2323-2326, 2000.
[33] M. Saerens, F. Fouss, L. Yen, and P. Dupont, "The Principal Components Analysis of a Graph, and Its Relationships to Spectral Clustering," *Proc. 15th European Conf. Machine Learning,* vol. 3201, pp. 371-383, 2004.
[34] S. Sarkar and K.L. Boyer, "Quantitative Measures of Change Based on Feature Organization: Eigenvalues and Eigenvectors," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* p. 478, 1996.
[35] B. Scholkopf, A. Smola, and K. Muller, "Nonlinear Component Analysis as a Kernel Eigenvalue Problem," *Neural Computation,* vol. 10, pp. 1299-1319, 1998.
[36] G. Scott and H. Longuet-Higgins, "Feature Grouping by Relicalisation of Eigenvectors of the Proximity Matrix," *Proc. British Machine Vision Conf.,* pp. 103-108, 1990.
[37] 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.
[38] J. Shi and C. Tomasi, "Good Features to Track," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 593-600, 1994.
[39] Y. Sugaya and K. Kanatani, "Outlier Removal for Motion Tracking by Subspace Separation," *IEICE Trans. Information and Systems,* vol. E86-D, no. 6, pp. 1095-1102, 2003.

[40] Y. Sugaya and K. Kanatani, "Multi-Stage Unsupervised Learning for Multi-Body Motion Segmentation," *IEICE Trans. Information and Systems,* vol. E87-D, no. 7, pp. 1935-1942, 2004.
[41] J.B. Tenenbaum, V. de Silva, and J.C. Langford, "A Global Geometric Framework for Nonlinear Dimensionality Reduction," *Science,* vol. 290, no. 5500, pp. 2319-2323, 2000.
[42] Y. Weiss, "Segmentation Using Eigenvectors: A Unifying View," *Proc. Seventh IEEE Int'l Conf. Computer Vision,* pp. 975-982, 1999.
[43] Y. Wu, Z. Zhang, T.S. Huang, and J.Y. Lin, "Multibody Grouping via Orthogonal Subspace Decomposition," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 252-257, 2001.
[44] R. Zass and A. Shashua, "A Unifying Approach to Hard and Probabilistic Clustering," *Proc. 10th IEEE Int'l Conf. Computer Vision,* 2005.
[45] L. Zelnik-Manor and M. Irani, "Degeneracies, Dependencies and Their Implications in Multi-Body and Multi-Sequence Factorizations," *Proc. IEEE CS Conf. Computer Vision and Pattern Recognition,* pp. 287-293, 2003.

Huaijun (John) Qiu received the BS degree in physics and computer science from East China University of Science and Technology in 2001 and the PhD degree in computer science from the University of York, United Kingdom, in 2006. Since February 2006, he has been a postdoctoral Fellow at Queen Mary Vision Laboratory, Queen Mary University of London, United Kingdom. His research interests are in pattern recognition, image processing, and video surveillance.

Edwin R. Hancock received the BSc degree in physics, with honors, in 1977 and the PhD degree in the area of high-energy physics in 1981 from the University of Durham. Following this, he worked for 10 years as researcher in the fields of high-energy nuclear physics and pattern recognition at the Rutherford-Appleton Laboratory (now the Central Research Laboratory of the Research Councils). During this period, he also held adjunct teaching posts at the University of Surrey and the Open University. In 1991, he moved to the University of York as a lecturer in the Department of Computer Science. He was promoted to senior lecturer in 1997 and to reader in 1998. In 1998, he was appointed to as chair of Computer Vision. He now leads a group of some 20 faculty, research staff, and PhD students working in the areas of computer vision and pattern recognition. His main research interests are in the use of optimization and probabilistic methods for high and intermediate level vision. He is also interested in the methodology of structural and statistical pattern recognition. He is currently working on graph matching, shape-from-X, image databases, and statistical learning theory. His work has found applications in areas such as radar terrain analysis, seismic section analysis, remote sensing, and medical imaging. He has published about 120 journal papers and 450 refereed conference publications. He was awarded the Pattern Recognition Society medal in 1991 and an outstanding paper award in 1997 by the journal *Pattern Recognition*. In 1998, he became a fellow of the International Association for Pattern Recognition. He is also a fellow of the Institute of Physics, the Institute of Engineering and Technology, and the British Computer Society. He has been a member of the editorial boards of the journals *IEEE Transactions on Pattern Analysis and Machine Intelligence* and *Pattern Recognition*. He has also been a guest editor for special editions of the journals *Image and Vision Computing* and *Pattern Recognition*. In 2006, he was appointed as the founding editor-in-chief of the *IET Computer Vision Journal*. He has been on the program committees for numerous national and international meetings. In 1997, with Marcello Pelillo, he established a new series of international meetings on energy minimization methods in computer vision and pattern recognition.