Clustering and Embedding using Commute Times

Huaijun 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 multi-body 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 (i.e. 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 intra-group coherence while weakening inter-group coherence and is superior to the normalized cut. The second application is to develop a robust multi-body motion tracking method using an embedding based of the eigenvectors of the commute time matrix. Our embedding procedure preserves commute time, and is closely akin to kernel PCA, the Laplacian eigenmap and the diffusion map. We illustrate the results both on synthetic image sequences and real world video sequences, and compare our results with several alternative methods.

I. Introduction

Graph spectral methods have played an important role in the image segmentation and data clustering literature [2], [3], [20], [21], [25], [29]. Spectral graph theory [5] is concerned with characterizing the structural properties of graphs using information conveyed by the eigenvalues

Department of Computer Science, University of York, York, YO1 5DD, UK., jun@cs.york.ac.uk

Department of Computer Science, University of York, UK., erh@cs.york.ac.uk, Tel.+44-1904-433374

and eigenvectors of the Laplacian matrix (the degree matrix minus the adjacency matrix). One of the most important tasks that arises 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 [18]. 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 pseudo inverse 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 [6].

The aim in 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 as follows. A pair of nodes in the graph will have a small commute time value if one of three conditions is satisfied. The first of these is that they are close together, i.e. the length of the path between them is small. The second case is if the sum of the weights on the edges connecting the nodes is small. Finally, the commute time is small if the pair of nodes are connected by many paths. 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 [29]. 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 [33] 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 [12].

A. Related Literature

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

1) Segmentation and Clustering: 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 done by Scott and Longuet-Higgins [28] 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 [26] have a method which 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 [24] have a similar method which uses the second largest eigenvector of the affinity matrix. The method of Shi and Malik [29], 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, i.e. the Fiedler vector. Focusing more on the issue of post-processing, Weiss [33] has shown how this, and other closely related methods, can be improved using a normalized affinity matrix. Shi and Meilă [20] have analyzed the convergence properties of the method using Markov chains. Pavan and Pelillo [22], [23] 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 [7] have show how the diffusion map can be used to accommodate path-length variation and have used the map for scale dependent clustering. Zass and ShaShua [35] show how to provide a probabilistic interpretation for spectral clustering [33] by developing a completely positive factorization scheme.

2) Factorization methods for motion analysis: As a second and more demanding application we consider the multi-body motion tracking problem. Multi-body motion tracking is a challenging problem which 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 [8], [9]. 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 eigen-vector 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 (i.e. feature-point jitter) and outliers are not present. However, real-world image sequences are usually contaminated by the two types of noise. There have been several attempts to overcome this problem. For instance, Ichimura [16] 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 [17], [31], [32] developed a subspace separation method by incorporating dimension correction and model selection. Wu et al [34] 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 [11] and multiple subspace inference analysis [10]. In addition to attempting to improve the behavior of the factorization method under noise, there has been a considerable effort at overcoming problems such as degeneracy, uncertainty and missing data [1], [14], [36].

The factorization method is clearly closely akin to graph-spectral methods used in clustering, since it uses the eigenvector methods to determine the class-affinity of sets of points. In fact Weiss [33] has presented a unifying view of spectral clustering methods, 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 [21] have applied a multi-way 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 methods.

B. Contribution

The aims in this paper are twofold. First, we aim to review the main results from spectral graph theory that relate to the definition of commute time. With these definitions to hand, we explore the properties of the commute time embedding. The embedding co-ordinate 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 which have small commute time are close, and those which have a large commute time are distant. This allows us to separate the objects in the embedded subspace by applying simple k-means clustering. There are of course many graph-spectral embedding algorithms reported in the literature, and recent and powerful additions include kernel PCA [27], the Laplacian eigenmap [4] and the diffusion map [7]. 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 from 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 kernel PCA. 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.

II. 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.

A. 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, satisfying

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

Further let $T = diag(d_u; u \in V)$ be the diagonal weighted degree matrix with elements $d_u = \sum_{v=1}^{|V|} w(u,v)$. The *un-normalized* 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 $\Lambda' = diag(\lambda'_1, \lambda'_2, ..., \lambda'_{|V|})$ is the diagonal matrix with the ordered eigenvalues as elements satisfying: $0 = \lambda'_1 \leq \lambda'_2 \ldots \leq \lambda'_{|V|}$ and $\Phi' = (\phi'_1 | \phi'_2 | .... | \phi'_{|V|})$ is the matrix with the ordered eigenvectors as columns. The corresponding eigen-decomposition of the *un-normalized* Laplacian matrix is $L = \Phi \Lambda \Phi^T$.

The heat equation associated with the graph Laplacian is given by $\frac{\partial \mathcal{H}_t}{\partial t} = -\mathcal{L}\mathcal{H}_t$ where $\mathcal{H}_t$ is the heat kernel and t is time. The solution of the heat-equation is found by exponentiating the Laplacian eigen-spectrum i.e.

$$\mathcal{H}_t = \exp[-t\mathcal{L}] = \Phi' \exp[-t\Lambda'] \Phi'^T$$

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

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

B. 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 identity matrix. The commute time is related to the heat kernel $\mathcal{H}_t$ in the following manner

$$G(u,v) = \int_0^\infty d_u^{1/2} \left(\mathcal{H}_t(u,v) - \phi_1'(u)\phi_1'(v) \right) d_v^{-1/2} dt \tag{1}$$

Here ϕ_1' is the eigenvector associated with the zero eigenvalue, i.e $\lambda_1' = 0$ of the *normalized* Laplacian matrix and which has k-th 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 defined as (see [6] page 6),

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

where λ' and ϕ' are the eigenvalue and eigenvectors of the *normalized* Laplacian $\mathcal{L}$. The corresponding Green's function of the *un-normalized* Laplacian $\bar{G} = T^{-1}G = T^{1/2}\mathcal{G}T^{1/2}$ is given by

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

where λ_i and ϕ_i are the eigenvalue and eigenvectors of the *un-normalized* Laplacian L.

The *normalized* Green's function is hence the pseudo-inverse of the *normalized* Laplacian $\mathcal{L}$. Moreover, it is straightforward to show that $\mathcal{G}\mathcal{L} = \mathcal{L}\mathcal{G} = I - \phi_1'\phi_1'^T$, and as a result $(\mathcal{L}\mathcal{G})(u,v) = \delta(u,v) - \frac{\sqrt{d_u d_v}}{vol}$. From (2), 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 [6] page 4), it follows that $\mathcal{G}$ is a kernel. The same applies to the *un-normalized* Green's function $\bar{G}$.

C. 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 [6]

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

where G is the Green's function given in (1). So, the commute time is given by

$$CT(u,v) = O(u,v) + O(v,u) = \frac{vol}{d_u} G(u,u) + \frac{vol}{d_v} G(v,v) - \frac{vol}{d_u} G(u,v) - \frac{vol}{d_v} G(v,u) \tag{3}$$

or using the *un-normalized* Green's function, as

$$CT(u,v) = vol\left(\bar{G}(u,u) + \bar{G}(v,v) - 2\bar{G}(u,v)\right) \tag{4}$$

As a consequence of (4) 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

$$CT(u,v) = vol\sum_{i=2}^{|V|}\frac{1}{\lambda_i'}\left(\frac{\phi_i'(u)}{\sqrt{d_u}} - \frac{\phi_i'(v)}{\sqrt{d_v}}\right)^2 \tag{5}$$

On the other hand, when the eigenvectors of the *un-normalized* Laplacian are used, we have

$$CT(u,v) = vol\sum_{i=2}^{|V|}\frac{1}{\lambda_i}(\phi_i(u) - \phi_i(v))^2 \tag{6}$$

III. 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 and the diffusion map. 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.

A. Basics

Equation (5) can be re-written in the following form which makes the relationship between the commute time and the Euclidean distance between the components of the eigenvectors explicit

$$CT(u,v) = \sum_{i=2}^{|V|}\left(\sqrt{\frac{vol}{\lambda_i' d_u}}\phi_i'(u) - \sqrt{\frac{vol}{\lambda_i' d_v}}\phi_i'(v)\right)^2 \tag{7}$$

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

$$\Theta = \sqrt{vol}\Lambda'^{-1/2}\Phi'^{T}T^{-1/2} \tag{8}$$

The columns of the matrix are vectors of embedding co-ordinates 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 un-normalized Laplacian, the corresponding matrix of embedding co-ordinates is

$$\Theta = \sqrt{vol}\Lambda^{-1/2}\Phi^T \tag{9}$$

The embedding is nonlinear in the eigenvalues of the Laplacian. This distinguishes it from principle components analysis (PCA) and locality preserving projection (LPP) [15] which are both linear. As we will demonstrate in the next section, the commute time embedding is just kernel PCA [27] on the Green's function. Moreover, it is also related to the Laplacian eigenmap since it minimizes a similar objective functions.

B. The commute time embedding and Kernel PCA

Let us consider the un-normalized case above. Since the Green's function $\bar{G}$ is the pseudo-inverse 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

$$C_f = \Theta\Theta^T = vol\Lambda^{-1/2}\Phi^T\Phi\Lambda^{-1/2} = vol\Lambda^{-1} = vol\Lambda_{\bar{G}} \tag{10}$$

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

$$K = \Theta^T\Theta = vol\Phi\Lambda^{-1/2}\Lambda^{-1/2}\Phi^T = vol\Phi\Lambda^{-1}\Phi^T = vol\bar{G} \tag{11}$$

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

C. The commute time embedding and the Laplacian eigenmap

In the Laplacian eigenmap [3], [4] the aim is to embed a set of points with co-ordinate matrix $\bar{\mathbf{X}} = (\bar{\mathbf{x}}_1|\bar{\mathbf{x}}_2|...|\bar{\mathbf{x}}_n)$ from a R^n space into a lower dimensional subspace R^m with the co-ordinate matrix $\mathbf{Z} = (\mathbf{z}_1|\mathbf{z}_2|...|\mathbf{z}_m)$. 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

$$\epsilon = \sum_{u,v} \|\mathbf{z}_u - \mathbf{z}_v\|^2 \Omega(u,v) = tr(\mathbf{Z}^T L \mathbf{Z}) \tag{12}$$

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

$$\mathbf{Z} = \arg \min_{\mathbf{Z}^{*T} T \mathbf{Z}^* = I} tr(\mathbf{Z}^{*T} L \mathbf{Z}^*) \tag{13}$$

The solution is given by the lowest eigenvectors of the generalized eigen-problem

$$L\mathbf{Z} = \Lambda' T \mathbf{Z} \tag{14}$$

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

$$\epsilon' = \frac{\sum_{u,v} \|\mathbf{z}_u - \mathbf{z}_v\|^2 \Omega(u,v)}{\sum_u \mathbf{z}_u^2 d_u} = tr(\frac{\mathbf{Z}^T L \mathbf{Z}}{\mathbf{Z}^T T \mathbf{Z}}) \tag{15}$$

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

$$\begin{aligned}
\epsilon' &= tr(\frac{\sqrt{vol}\Lambda'^{-1/2}\Phi'^T T^{-1/2} L T^{-1/2}\Phi'\Lambda'^{-1/2}\sqrt{vol}}{\sqrt{vol}\Lambda'^{-1/2}\Phi'^T T^{-1/2} T T^{-1/2}\Phi'\Lambda'^{-1/2}\sqrt{vol}}) \\
&= tr(\frac{\Lambda'^{-1/2}\Phi'^T \mathcal{L} \Phi'\Lambda'^{-1/2}}{\Lambda'^{-1/2}\Phi'^T\Phi'\Lambda'^{-1/2}}) \\
&= tr(\frac{\Lambda'^{-1/2}\Lambda'\Lambda'^{-1/2}}{\Lambda'^{-1}}) \\
&= tr(\Lambda') = \epsilon^*
\end{aligned} \tag{16}$$

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 it also aims to assign large co-ordinate values to nodes (or points) with large degree (i.e. it maximizes $\sum_u \mathbf{z}_u^2 d_u$). Nodes with 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 furthest away from the origin and are hence unlikely to be close to one-another.

Finally, we note that the objective function appearing in (15) is which is identical to that used in the normalized cut. To show this let $\vec{\theta}$ be a dimensional indicator vector. The minimum value

obtained by the normalized cut [29] is

$$\vec{\theta}_1 = \arg \min_{\vec{\theta}^T \mathbf{T}\mathbf{1}=0} \frac{\vec{\theta^T}(\mathbf{T}-\Omega)\vec{\theta}}{\vec{\theta^T}\mathbf{T}\vec{\theta}} \tag{17}$$

Hence comparing with (15) 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 non-zero 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 multi-partition case, the scaling differentiates the importance of different eigenvectors. From (6), it is clear that the eigenvector corresponding to the smallest non-zero 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.

D. The commute time and the diffusion map

Finally, it is interesting to note the relationship with the diffusion map embedding of Lafon *et al* [7]. The method commences from the random walk on a graph which has transition probability matrix $P = T^{-1}A$, where A is the adjacency matrix. Although P is not symmetric, it does have a right eigenvector matrix Ψ, which satisfies the equation

$$P\Psi = \Lambda_P \Psi \tag{18}$$

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

$$\begin{aligned} (I - T^{-1}L)\Psi &= \Lambda_P\Psi \\ T^{-1}L\Psi &= (I-\Lambda_P)\Psi \\ L\Psi &= (I-\Lambda_P)T\Psi \end{aligned} \tag{19}$$

which is identical to (14) if $\mathbf{Z} = \Psi$ and $\Lambda' = I - \Lambda_P$. The embedding co-ordinate matrix for the diffusion map is $\Theta_D = \Lambda^t \Psi^T$, where t is real. For the embedding, the diffusion distance between

a pair of nodes is $D_t^2(u,v) = \sum_{i=1}^{m}(\lambda_P)_i^{2t}\left(\psi_i(u) - \psi_i(v)\right)^2$. Clearly if we take $t = -1/2$ the diffusion map is equivalent to the commute time embedding. Moreover, when this is the case then the diffusion time is equal to the commute time.

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 are 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. The reason for this is that according to the original definition of the diffusion distance for a random walk

$$D_t^2(u,v) = \|p_t(u,\cdot) - p_t(v,\cdot)\|^2$$

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. So, it is a critical to control t carefully in order to obtain useful embeddings.

E. Some embedding examples

Fig. 1 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 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 as follows. 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. The orthogonality is due to the strong block-diagonal structure of the affinity matrix (the commute time matrix in this case) [2].

From (9) we can see that the co-ordinates of the commute time embedding depend on the eigenvalues and eigenvectors of the Laplacian matrix. Hence, the stability of the embedding

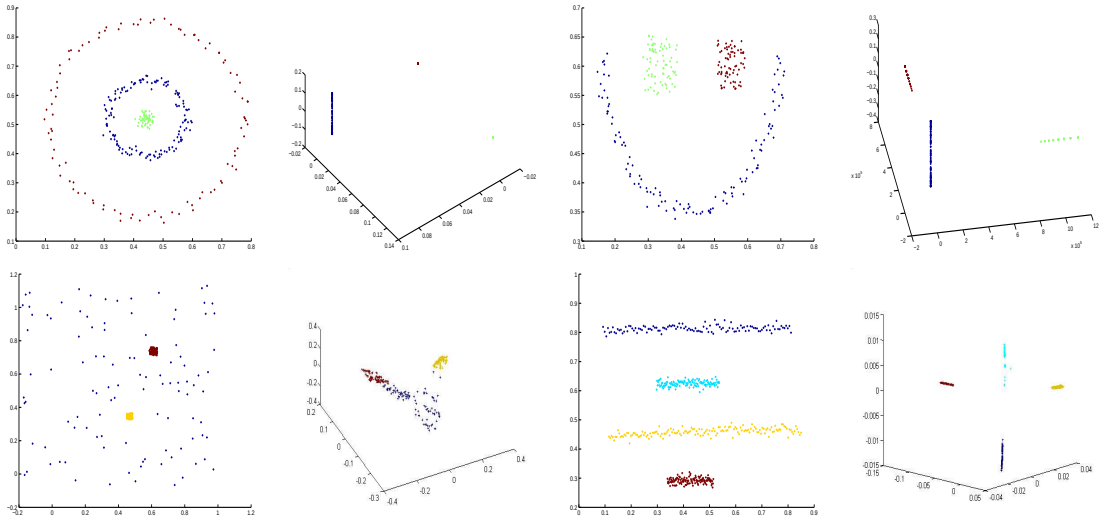


Fig. 1. Commute time embedding examples.

depends on the stability of the eigenvalue and eigenvector matrices. According to Weyl's theorem, the variation of the eigenvalues of a perturbed matrix is bounded by the maximum and the minimum eigenvalues of the perturbing matrix. However, the eigenvectors are less stable under perturbation. Despite this anticipated problem, 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 neither be 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 late, 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.

IV. Applications of Commute Time

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

A. 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. Hence, here we use the eigenvector associated with the smallest eigenvalue of the commute time matrix since it is this eigenvector that contains the most significant partition information.

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:
   $$\Omega(u, v) = \exp\left(\frac{-\left\|\mathbf{F}_u - \mathbf{F}_v\right\|_2}{\sigma_I}\right) * \begin{cases} \exp\left(\frac{-\left\|\mathbf{X}_u - \mathbf{X}_v\right\|_2}{\sigma_X}\right) & \text{if } \left\|\mathbf{X}_u - \mathbf{X}_v\right\|_2 < r \\ 0 & \text{otherwise} \end{cases} \quad (20)$$
   where $\mathbf{F}_u$ is either the intensity value at pixel u for a brightness image or the vector of 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 (2) and the eigen-spectrum of the *normalized* Laplacian $\mathcal{L}$.
4) From (4), 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 $CT(u, v) = vol \sum_{i=2}^{|V|} \frac{1}{\lambda_i'} \left(\frac{\phi_i'(u)}{\sqrt{d_u}} - \frac{\phi_i'(v)}{\sqrt{d_v}}\right)^2$ to bipartition the weighted graph.
6) Decide if the current partition should be sub-divided, and recursively repartition the component parts if necessary.

B. Multi-body Motion Tracking using Commute Time

In this section, we will show how the multi-body 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.

1) Factorization Method Review: Suppose 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 ith point in the fth 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:

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

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). In order to solve the factorization problem, matrix W can be decomposed by SVD:

$$W = U \Sigma R^T$$

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 multi-body 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 [8], [9]

to solve the multi-body separation problem. The shape interaction matrix is

$$Q = RR^T = \begin{bmatrix} S_1^T\Sigma_1^{-1}S_1 & 0 & \cdots & 0 \\ 0 & S_2^T\Sigma_2^{-1}S_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & S_N^T\Sigma_N^{-1}S_N \end{bmatrix} \tag{21}$$

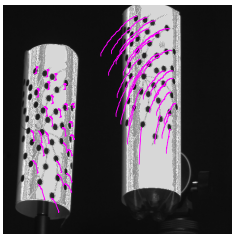
From (21), the shape interaction matrix Q has the convenient properties that $Q(u,v) = 0$, if points u,v belong to different objects and $Q(u,v) \neq 0$, if points u,v belong to the same object. The matrix Q is also invariant to both the object motion and the selection of the object coordinate systems. This leads to a simple scheme for separating multi-object motions by permuting the elements of Q so that it acquires a block diagonal structure. In Costeira and Kanade's method [8], [9] a greedy algorithm is used to permute the Q matrix into block diagonal form. An illustration is shown in Fig. 2. Fig. 2(a) shows the set of original points together with their trails. Fig. 2(b) the Q-matrix for these points, Fig. 2(c) the result of applying Costeira and Kanade's method to the sort the Q-matrix and Fig. 2(d) 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 Figures 2(e) and 2(f) we respectively show the effect of adding Gaussian noise to the Q matrix in 2(b) and the resulting permuted matrix. In this noisy case, the block structure is badly corrupted and object separation is almost impossible.

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 multi-body 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.

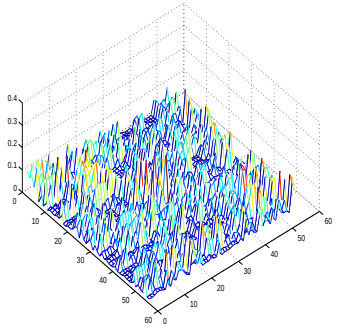
Object Separation Steps:

The algorithm we propose for this purpose has the following steps:

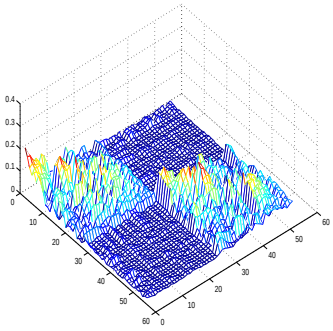
1) Use the shape interaction matrix Q as the weighted adjacency matrix Ω and construct the corresponding graph Γ.
2) Compute the Laplacian matrix of graph Γ using $L = T - Q$.
3) Find the eigenvalue matrix Λ and eigenvector matrix Φ of L using $L = \Phi\Lambda\Phi^T$.



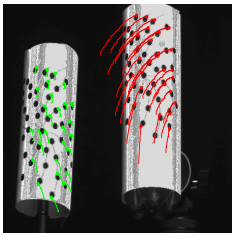
(a) Original picture with trails of the moving feature points.



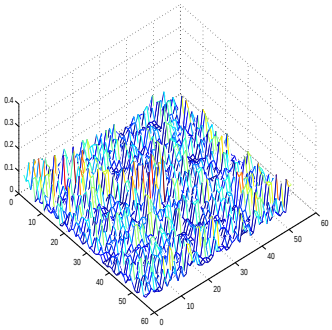
(b) Original Q matrix unsorted.



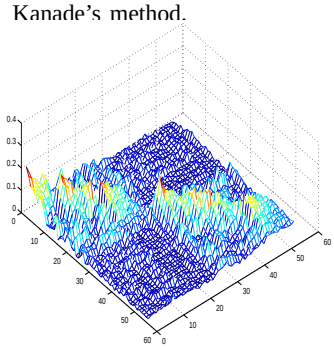
(c) Sorted Q by Costeira and Kanade's method.



(d) Object separation result.



(e) Q matrix with Gaussian noise $\sigma = 0.8$.

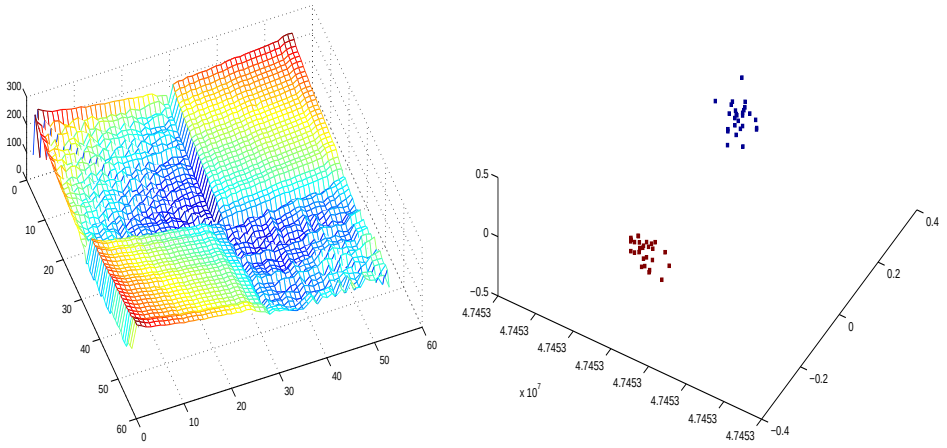


(f) Sorted Q with noise.

Fig. 2. A multi-body motion separation example using Costeira and Kanade's method.

4) Compute the commute time matrix CT using Λ and Φ from (6).
5) Embed the commute time into a subspace of R^n using (8) or (9).
6) Cluster the data points in the subspace using the k-means algorithm [19].

To illustrate the effectiveness of this method, we return to example used earlier in Fig. 2. 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, standard derivation 0.8 Gaussian noise) and the clustering steps listed above we still recover a good set of objects (see Fig. 2(d)). This is illustrated in Fig. 3. Here, sub-figure (a) shows the commute time matrix of graph Γ and sub-figure (b) shows the embedding in a 3D subspace. It is clear that the commute time matrix gives a good block-diagonal structure and the points are well clustered in the embedding space even when significant noise is present.



(a) Sorted commute time matrix.

(b) Clustered points in the commute time subspace for two objects.

Fig. 3. Multi-body motion separation re-cast as a commute time clustering problem.

V. Experiments

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

A. Image segmentation and data clustering

1) Point-set clustering examples: In Fig. 4 and 5 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 sub-figures in both figures are organized as follows. The left-hand column shows the point-sets, the middle column the affinity matrices and the right-most 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 Fig. 4 and 5. From 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 are 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

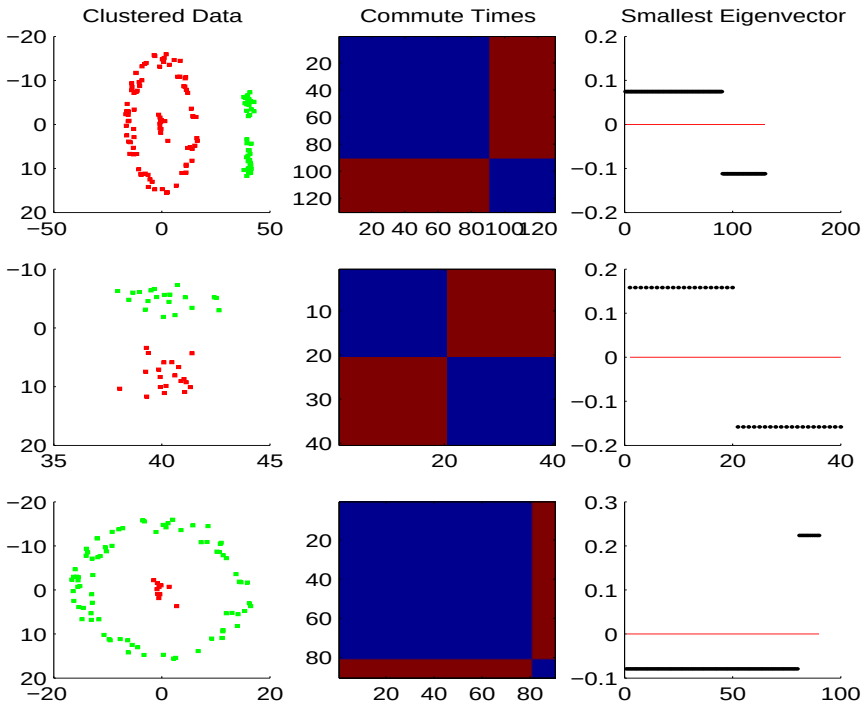


Fig. 4. Data clustering by commute time cut.

the distribution of the components of this eigenvector, the closer of the relaxed solution towards the desired discrete solution. This point is well illustrated in the third column of Fig. 4 compared to the one in Fig. 5. From the figures, it is clear 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. 4 compared to the normalized affinity matrix in Fig. 5.

2) Image segmentation: We have compared our new method with that of Shi and Malik [29] on synthetic images subject to additive Gaussian noise. On the left-hand side of Fig. 6, we show the results of using these two methods for segmenting a synthetic image composed of 3 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. 6 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%.

In Fig. 7, we show some examples of our segmentation results and compare them with those obtained using the normalized cut. The aim here is to investigate the effect of adding and

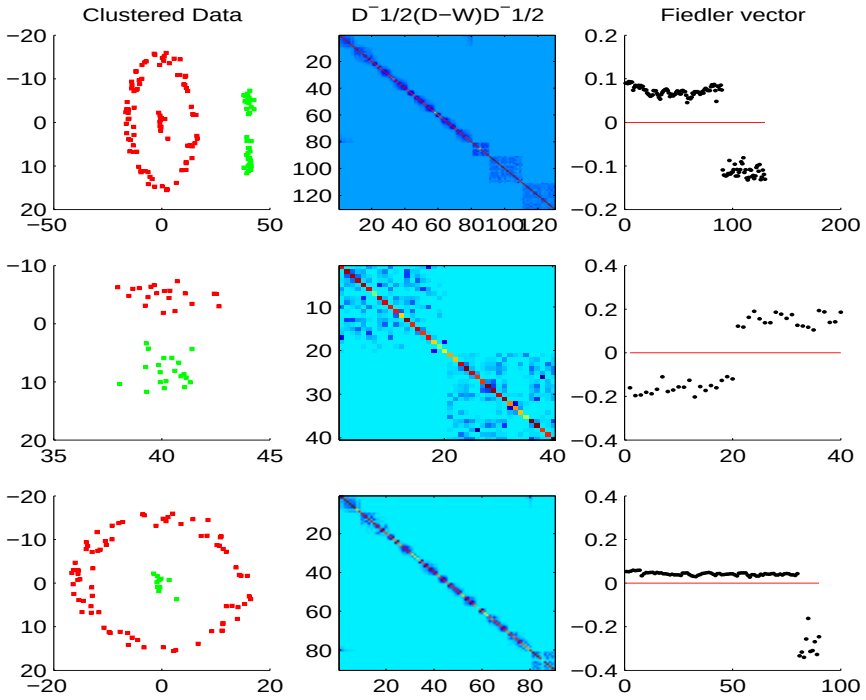


Fig. 5. Data clustering by normalized cut.

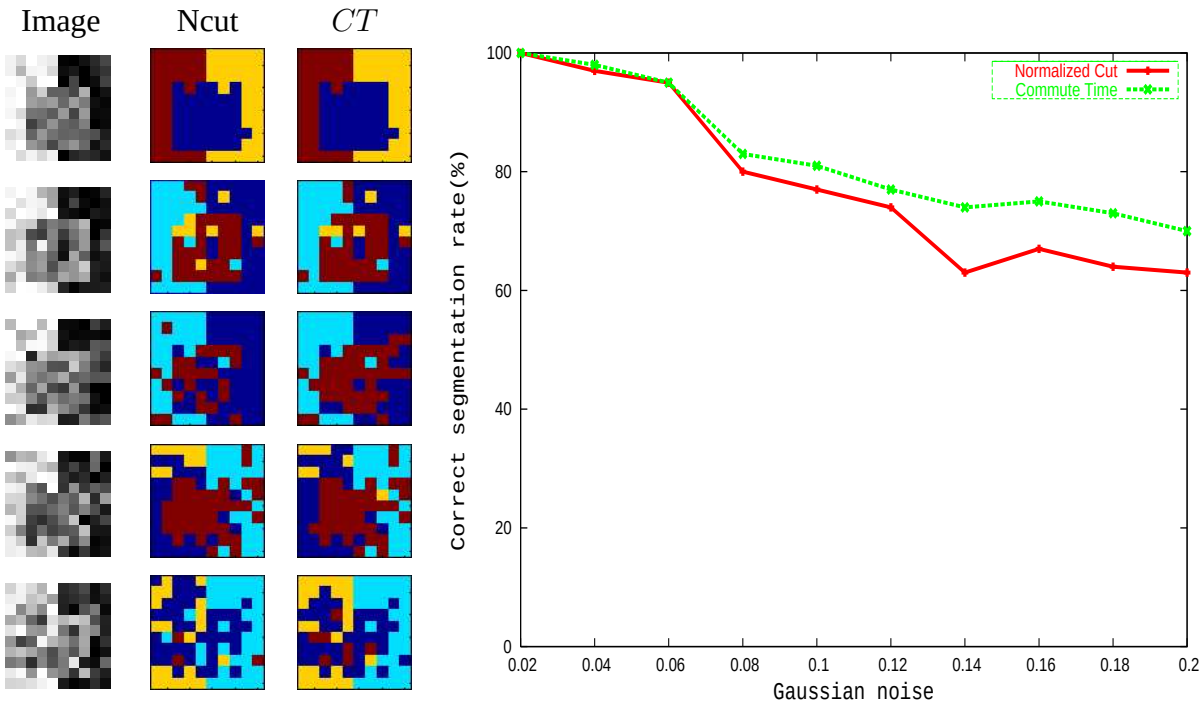


Fig. 6. Method comparison for a synthetic image with increasing Gaussian noise.

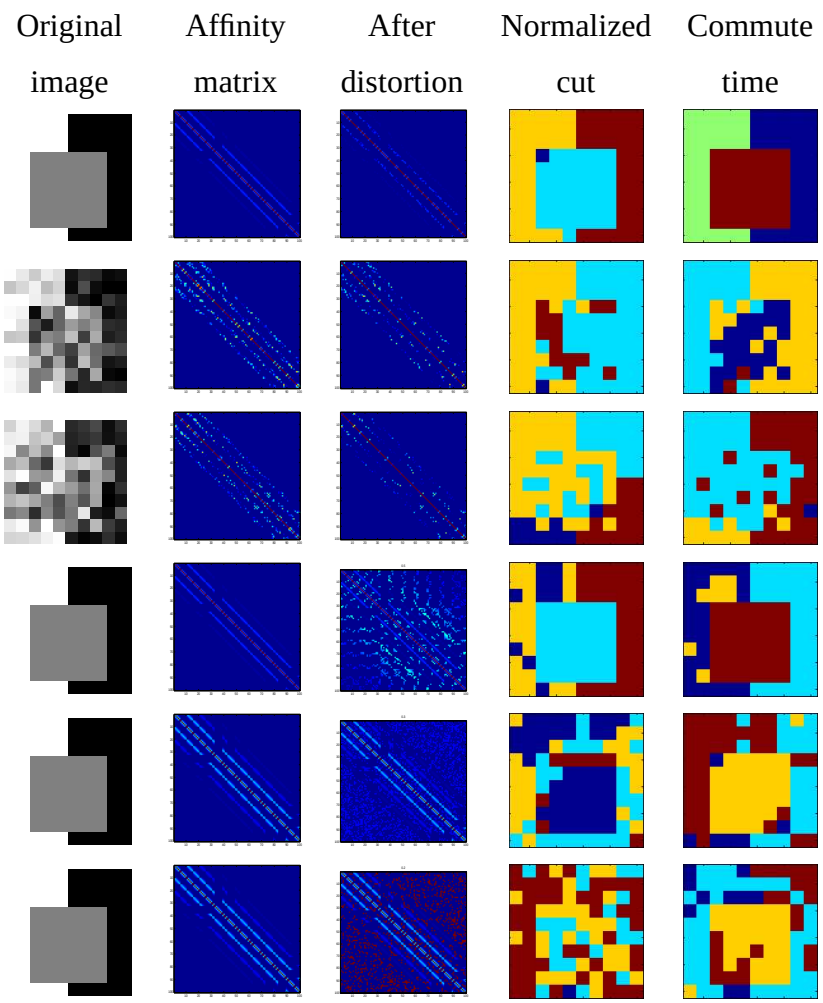


Fig. 7. Examples of segmentation results with different link-weight distortion.

deleting link-weights at random. The first column shows the original image, the second column the original affinity matrix and the third column the affinity matrix after link noise has been added. The first three rows show the effect of random link deletion, and the second three rows the result of link addition. The fourth and fifth columns respectively show the results obtained using the normalized cut and the commute time. For these images, Fig. 8 shows the fraction of correctly assigned pixels as a function of the fraction of links added or deleted. In the figure the red curve shows the effect of link addition on the commute time method, the green curve the effect of link addition on the normalized cut, the blue curve the effect of link deletion on the commute time method and, finally, the pink curve the effect of link deletion on the normalized cut. The main features to note from the plot are as follows. First, the commute time method

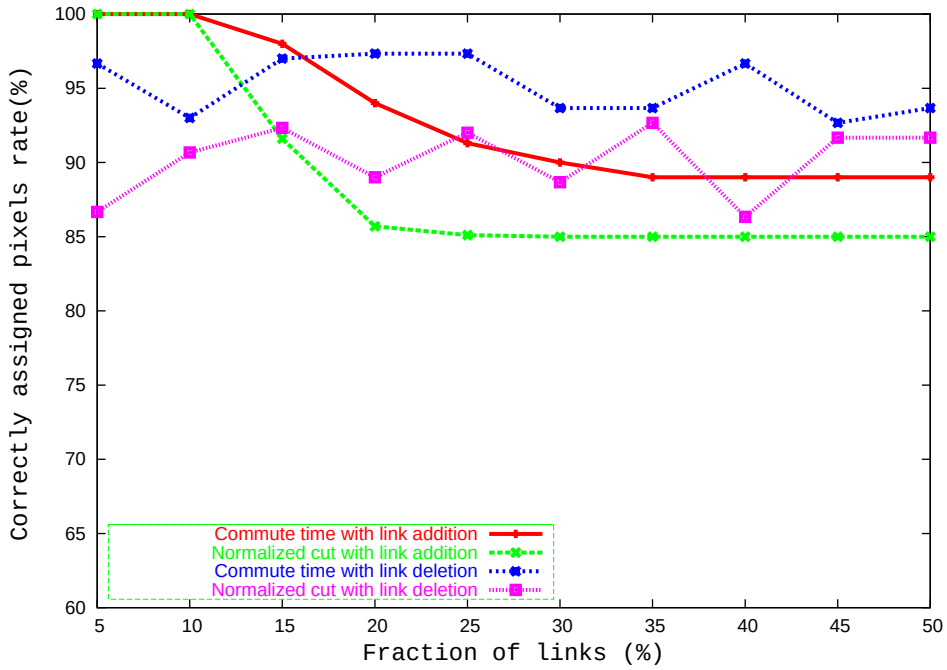


Fig. 8. Method comparison for synthetic images with different link-weight distortion.

is more robust to both link deletion and insertion than the normalized cut. The second feature is that link deletion has a less marked effect on the performance than link insertion. Thirdly, spurious link insertion has a smaller effect on the commute time than the normalized cut.

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 50x50 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 left-most panel shows the original image. The middle and right-hand panels show the results from two successive bi-partitions.

For four of the real images, we compare our method with the normalized cut in Figures 10 and 11. The first column of each sub-figure shows the first, second and third bi-partitions of the images. The second column shows the histogram of the components of the smallest eigenvector, and the right-hand column shows the distribution of the eigenvector components. The blue and red lines in the right-hand column respectively correspond to zero and the eigenvector component threshold.

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

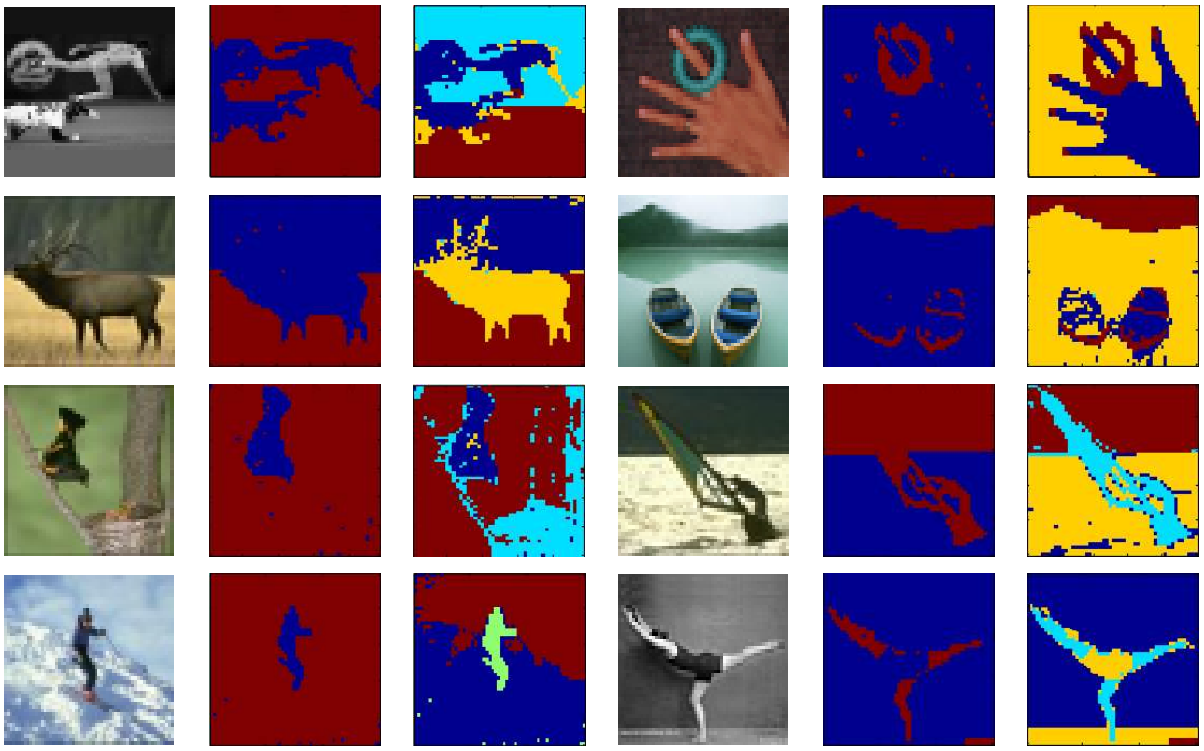


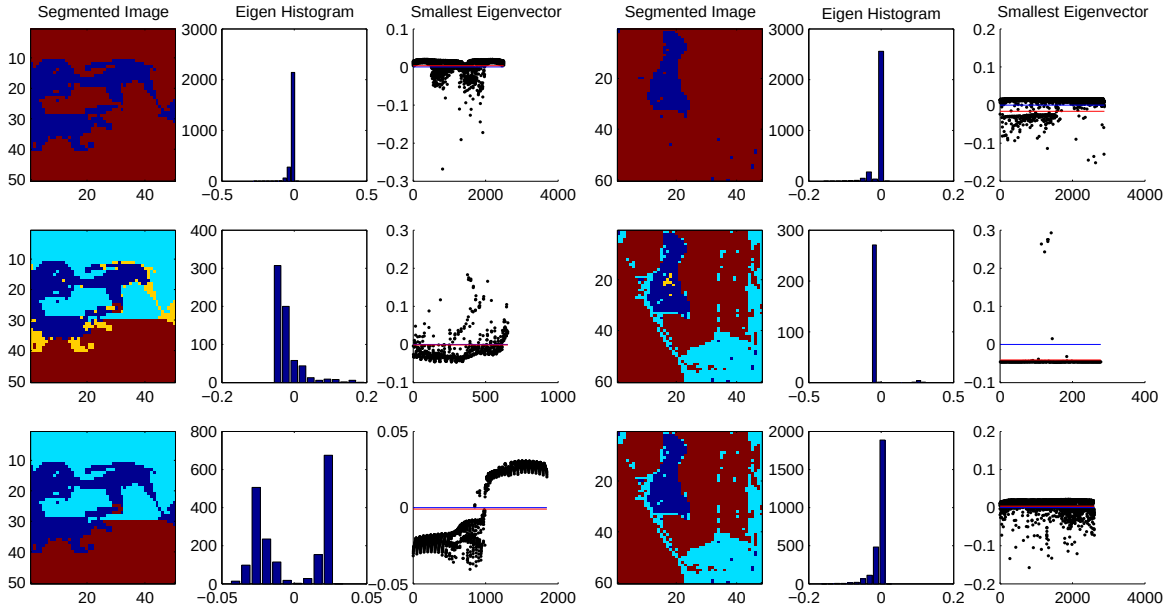
Fig. 9. Real world segmentation examples.

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 Fig. 10 and 11 where the distribution of eigenvector components in the histograms are better separated for the commute time method. Hence, the corresponding cluster indicators give better separation.

B. Multi-body motion tracking problem

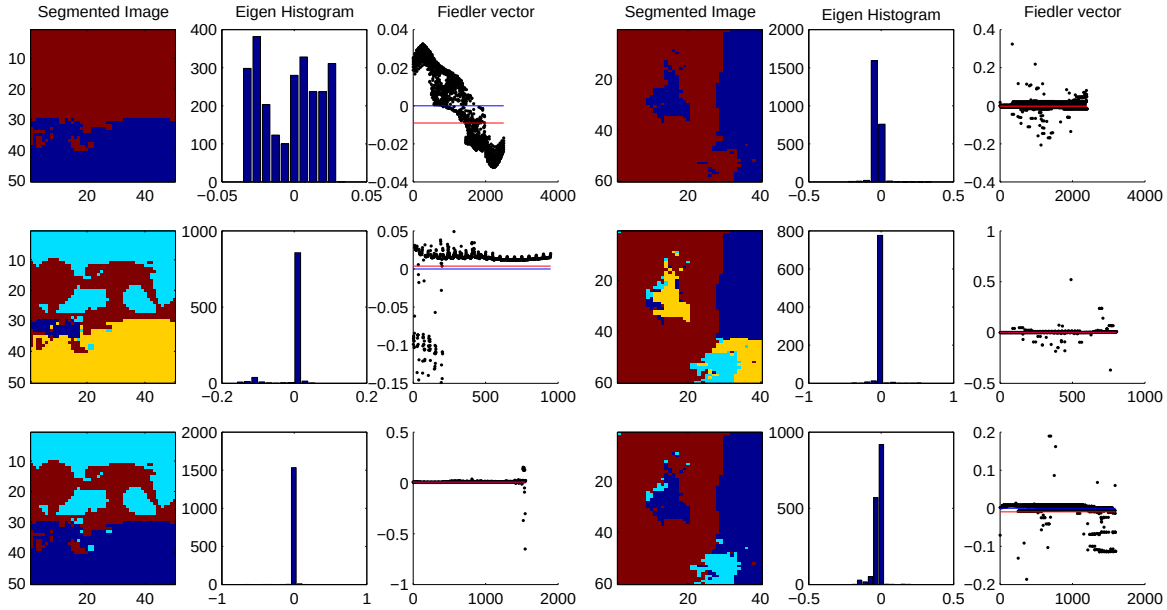
In this section we conduct experiments with the commute time embedding method on both synthetic data and real-world motion tracking problems. To investigate the robustness of the method, we add Gaussian noise to the data sets and compare the results with some classical methods.

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



(a) Commute time for 50x50 image with $r = 8$ $\sigma_X = 0.5$ $\sigma_I = 0.1$

(b) Commute time for 60x40 image with $r = 5$ $\sigma_X = 0.2$ $\sigma_I = 0.02$

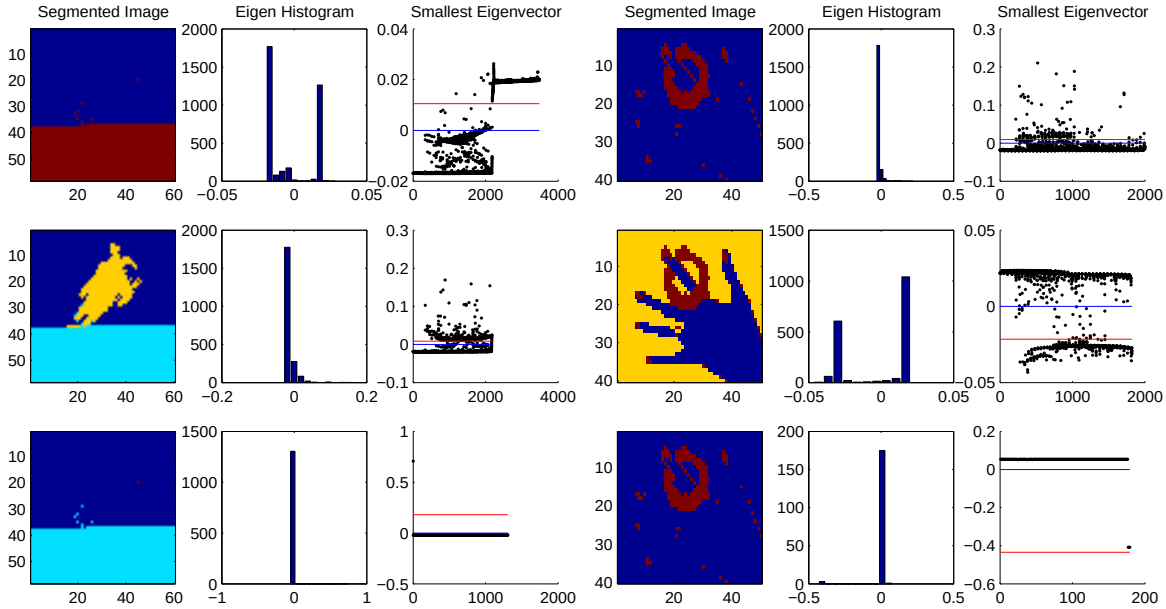


(c) Normalized cut for 50x50 image with $r = 5$ $\sigma_X = 2$ $\sigma_I = 0.05$

(d) Normalized cut for 60x40 image with $r = 5$ $\sigma_X = 0.05$ $\sigma_I = 0.01$

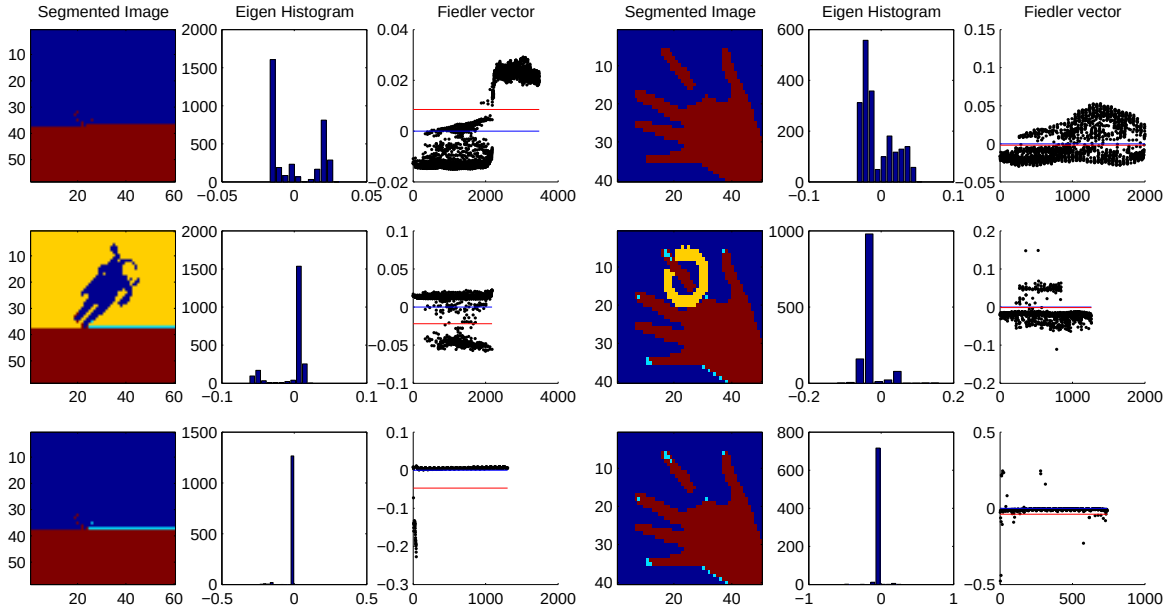
Fig. 10. Detailed segmentation process in comparison.

have added Gaussian noise of zero mean and standard deviation σ to the coordinates of these 29 points, and then cluster them into two groups.



(a) Commute time for 60x58 image with $r = 5$ $\sigma_X = 0.1$ $\sigma_I = 0.03$

(b) Commute time for 50x40 image with $r = 10$ $\sigma_X = 0.1$ $\sigma_I = 0.03$



(c) Normalized cut for 60x58 image with$r = 5$ $\sigma_X = 0.1$ $\sigma_I = 0.03$

(d) Normalized cut for 50x40 image with $r = 5$ $\sigma_X = 5$ $\sigma_I = 0.02$

Fig. 11. Detailed segmentation process comparison.

We have compared our method with Costeira and Kanade's greedy algorithm [8], [9], Ichimura's discrimination criterion method [16] and Kenichi's subspace separation method [17]. In Fig. 13

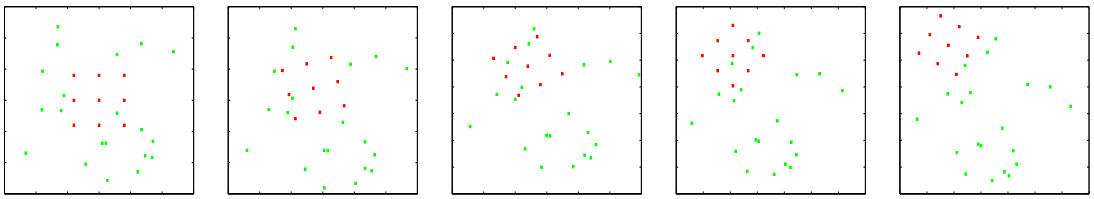
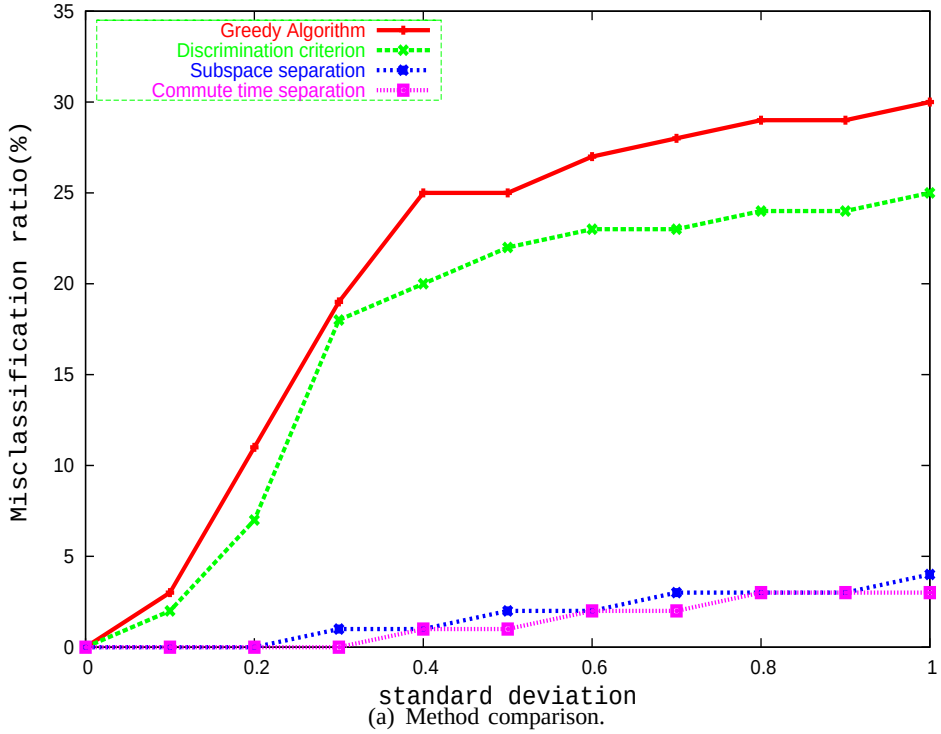
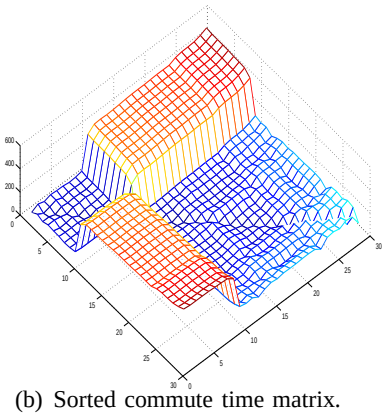


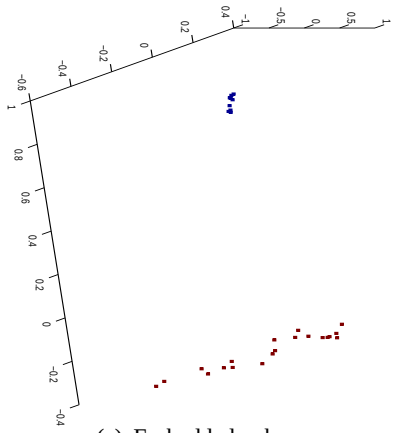
Fig. 12. Synthetic image sequence.



(a) Method comparison.



(b) Sorted commute time matrix.



(c) Embedded subspace.

Fig. 13. Synthetic data.

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. From the figure, it is clear that our method performs significantly better than the greedy method [9] and the discrimination criterion method [16]. It also has a margin of advantage over the subspace separation method [17].

For an example with a Gaussian noise with $\sigma = 0.5$, the commute time matrix and the embedded subspace are shown in Fig. 13(b) and 13(c) respectively. It is clear that even in this heavily noise contaminated case, the commute time matrix still maintains a good block-diagonal structure. Moreover, under the embedding the points are easily separated.

2) Real-world Motion Tracking: In this section we experiment with the commute time method on real-world multi-body motion tracking problems. The columns of 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 [31], [32]. Here there is one moving object and a moving camera. A successful tracking method will separate the moving object from the moving background. The forth and fifth columns in Fig. 14 are two video sequences captured using a Fuji-Film 2.0M camera (320×240 pixels). For each of sequence, we detected feature points using the KLT algorithm [30], 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 ten frames each from the sequences with 117 and 116 feature points respectively. Compared to the data from Sugaya and Kanatani [31], [32], 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 of Fig. 14, our method not only separates the ducks correctly from the moving background, but it also separates the moving ducks from each other. The fifth column of 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.

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

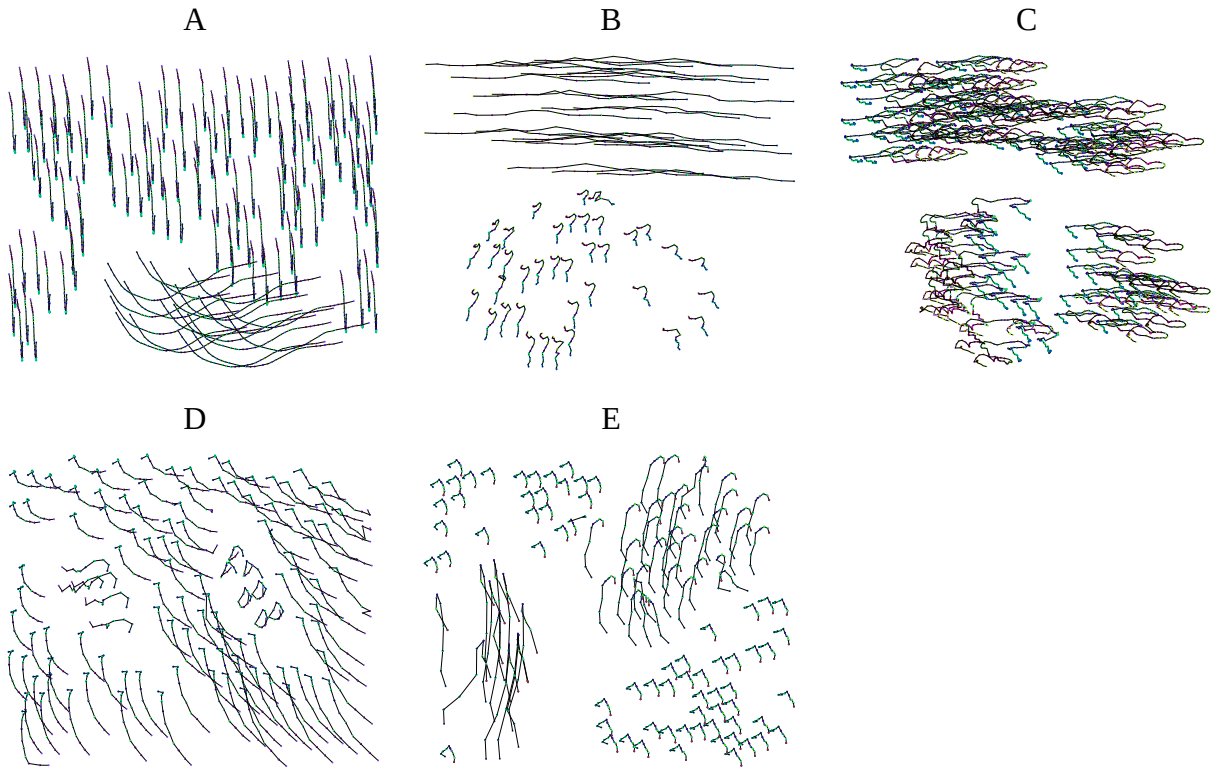


Fig. 15. Feature point trajectories.

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 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 of Fig. 14.

For the same sequences, we compared our results with Costeira and Kanade's greedy algorithm [9], Ichimura's discrimination criterion method [16], Kanatani's subspace separation method [17]

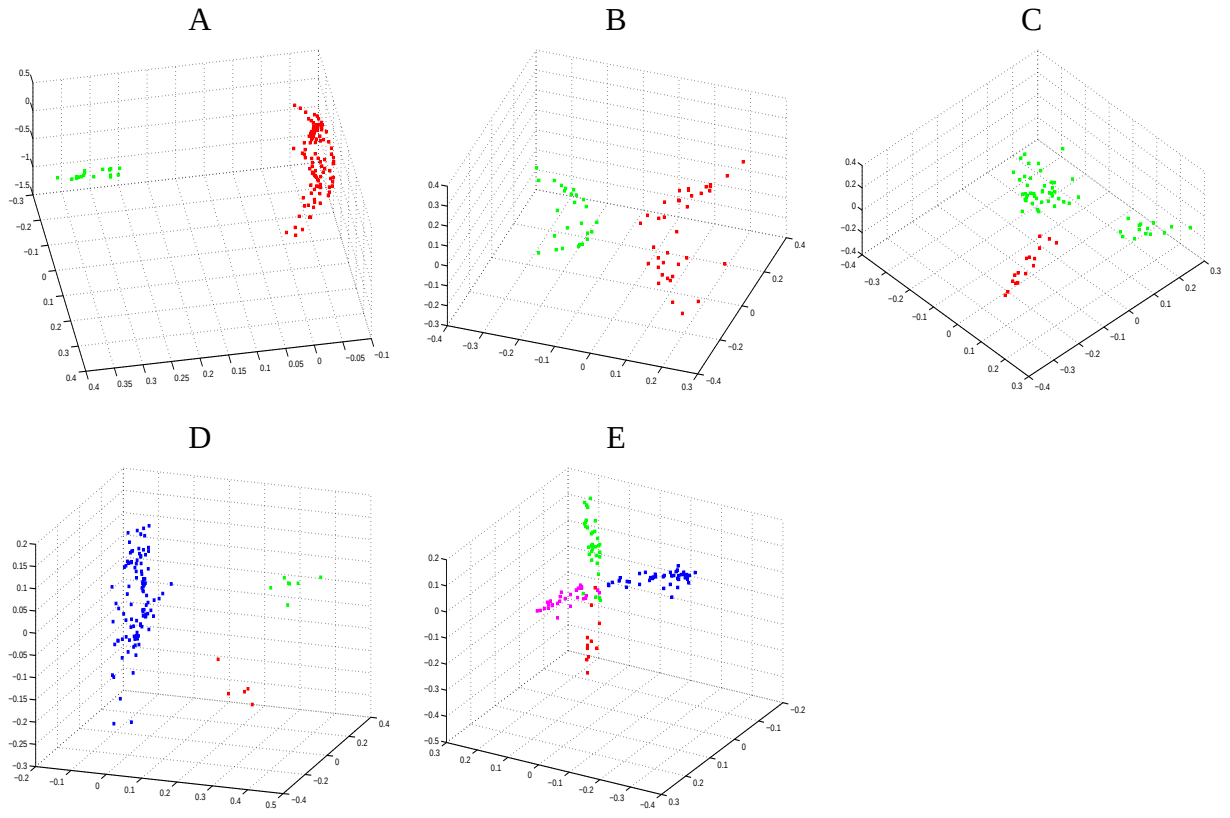


Fig. 16. Sequences embedded by commute time in the subspace.

and Sugaya and Kanatani's multi-stage learning method [32]. The comparison is shown in Table I.

Table I lists the accuracies of the different methods using the ratio of number of correctly classified points to the total number of points. The ratio is averaged over 50 trails for each method. From the table, it is clear that the greedy algorithm [9] 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 [16] and the subspace separation method [17] 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 multi-stage learning method [32] 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% accuracy. In our method, motion jitter or noise disturbance will be correctly recognized and suppressed by the embedding process. Outliers, on the other hand, are automatically rejected in the clustering step by the k-means algorithm.

TABLE I

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**

VI. 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 to 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 multi-body motion tracking. Both methods are proved to outperform alternatives in terms of their ability to separate the input data into cleanly separated clusters.

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.

REFERENCES

[1] P. Anandan and M. Irani. Factorization with uncertainty. *International Journal of Computer Vision*, 49(2-3):101–116, 2002.

[2] A.Ng, M.Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2001.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in Neural Information Processing Systems*, pages 585–591, 2001.

[4] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.

[5] F.R.K. Chung. *Spectral Graph Theory*. CBMS series 92. American Mathmatical Society Ed., 1997.

[6] F.R.K. Chung and S.-T. Yau. Discrete green's functions. In *J. Combin. Theory Ser.*, pages 191–214, 2000.

[7] 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. *National Academy of Sciences*, 102(21):7426–7431, 2005.

[8] J. Costeira and T. Kanade. A multi-body factorization method for motion analysis. In *ICCV*, pages 1071–1076, 1995.

[9] J. Costeira and T. Kanade. A multibody factorization method for independently moving objects. *IJCV*, 29(3):159 – 179, 1997.

[10] Z. Fan, J. Zhou, and Y. Wu. Inference of multiple subspaces from high-dimensional data and application to multibody grouping. In *CVPR*, pages 661–666, 2004.

[11] Z. Fan, J. Zhou, and Y. Wu. Multibody motion segmentation based on simulated annealing. In *CVPR*, pages 776–781, 2004.

[12] I. Fischer and J. Poland. Amplifying the block matrix structure for spectral clustering. In *IDSIA*, 2005.

[13] C.W. Gear. Multibody grouping from motion images. *IJCV*, 29(2):130–150, 1998.

[14] A. Gruber and Y. Weiss. Multibody factorization with uncertainty and missing data using the em algorithm. In *CVPR*, pages 707–714, 2004.

[15] X. He and P. Niyogi. Locality preserving projections. In *NIPS*, pages 585–591, 2003.

[16] N. Ichimura. Motion segmentation based on factorization method and discriminant criterion. In *ICCV*, pages 600–605, 1999.

[17] K. Kanatani. Motion segmentation by subspace separation and model selection. In *ICCV*, pages 301–306, 2001.

[18] R. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. *19th Intl. Conf. on Machine Learning (ICML) [ICM02].*, 2002.

[19] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, pages 281–297, 1967.

[20] M. Meilă and J. Shi. A random walks view of spectral segmentation. In *NIPS*, pages 873–879, 2000.

[21] J. Park, H. Zha, and R. Kasturi. Spectral clustering for robust motion segmentation. In *ECCV*, pages 390–401, 2004.

[22] M. Pavan and M. Pelillo. Dominant sets and hierarchical clustering. In *ICCV*, pages 362–369, 2003.

[23] M. Pavan and M. Pelillo. A new graph-theoretic approach to clustering and segmentation. In *CVPR03*, pages I: 145–152, 2003.

[24] P. Perona and W. T. Freeman. A factorization approach to grouping. In *ECCV*, pages 655–670, 1998.

[25] H. Qiu and E.R. Hancock. Image segmentation using commute times. In *BMVC*, pages 929–938, 2005.

[26] S. Sarkar and K. L. Boyer. Quantitative measures of change based on feature organization: Eigenvalues and eigenvectors. In *CVPR*, page 478, 1996.

[27] B. Sch, A. Smola, and K. Muller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319, 1998.

[28] G. Scott and H. Longuet-Higgins. Feature grouping by relicalisation of eigenvectors of the proximity matrix. In *BMVC.*, pages 103–108, 1990.

[29] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE PAMI*, 22(8):888–905, 2000.

[30] J. Shi and C. Tomasi. Good features to track. In *CVPR*, pages 593–600, 1994.

[31] Y. Sugaya and K. Kanatani. Outlier removal for motion tracking by subspace separation. *IEICE Trans. INF and SYST*, E86-D(6):1095–1102, 2003.

[32] Y. Sugaya and K. Kanatani. Multi-stage unsupervised learning for multi-body motion segmentation. *IEICE Trans. INF and SYST*, E87-D(7):1935–1942, 2004.

[33] Y. Weiss. Segmentatoin using eigenvectors: a unifying view. In *ICCV.*, pages 975–982, 1999.

[34] Y. Wu, Z. Zhang, T. S. Huang, and J. Y. Lin. Multibody grouping via orthogonal subspace decomposition. In *CVPR*, pages 252–257, 2001.

[35] R. Zass and A. Shashua. A unifying approach to hard and probabilistic clustering. *ICCV*, 2005.

[36] L. Zelnik-Manor and M. Irani. Degeneracies, dependencies and their implications in multi-body and multi-sequence factorizations. In *CVPR*, pages 287–293, 2003.