Graph Simplification and Matching using Commute Times

Huaijun Qiu and Edwin R. Hancock

Department of Computer Science,University of York

Heslington, York, YO10 5DD, UK

Abstract

This paper exploits the properties of the commute time for the purposes of graph simplification and matching. 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 characterise the random walk using the commute time between nodes, and show how this quantity may be computed from the Laplacian spectrum using the discrete Green's function. In this paper, we explore two different, but essentially dual, simplified graph representations delivered by the commute time. The first representation decomposes graphs into concentric layers. To do this we augment the graph with an auxiliary node which acts as a heat source. We use the pattern of commute times from this node to decompose the graph into a sequence of layers. Our second representation is based on the the minimum spanning tree of the commute time matrix. The spanning trees located using commute time prove to be stable to structural variations. We match the graphs by applying a tree-matching method to the spanning trees. We experiment with the method on synthetic and real-world image data, where it proves to be effective.

Key words: Graph Matching; Graph Simplification; Commute Time; Graph Spectrum

1 Introduction

Spectral graph theory [2] is concerned with characterising 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 arises in the analysis of graphs is that of enumerating the set of paths (and their lengths) between pairs of nodes. This process can be characterised using the heat equation [5]. The heat kernel [5] is the solution of the heat-equation on the graph and is found by exponentiating the normalised Laplacian of the graph (the identity matrix minus the degree normalised adjacency matrix) with time. As a result, the heat kernel can be computed efficiently by exponentiating the Laplacian eigensystem [2]. The heat kernel can be viewed as capturing the way in which information flows with time across the edges of the graph. For large times the heat kernel is dominated by the Fiedler vector, and so it is equivalent to the random walk.

Random walks [17] have found widespread use in information retrieval and structural pattern analysis. For instance, the random walk is the basis of the Page-Rank algorithm which is used by the Googlebot search engine [1]. In computer vision random walks have been used for image segmentation [8] and clustering [15]. More recently both Gori, Maggini and Sarti [4], and, Robles-Kelly and Hancock [14,13] have used random walks to sort the nodes of graphs for the purposes of graph-matching. Most of these methods use a simple approximate characterisation of the random walk based either on the leading eigenvector of the transition probability matrix, or equivalently the Fiedler vector of the Laplacian matrix [6]. In general though, the random walk is not edge-connected on the graph. Hence, it may not preserve the edge structure and can prove to be an ineffective way of capturing the structural properties of the graph. In an attempt to overcome this problem Robles-Kelly and Hancock explore two approaches. The first of these is to use a post-processing step to recover an edge connected path from the components of the leading eigenvector [14]. The second refinement is to pose the recovery of an edge-ordered path as one of graph seriation using a utility function and to recover an approximate solution to this problem using graph-spectral (i.e. eigenvector) methods [13].

This work on path based methods for graph matching is clearly closely akin with graph spectral techniques, since both rely on the eigenvalues and eigenvectors of the Laplacian matrix. There has been a considerable body of work aimed at using graph-spectra for the purposes of matching. Some of the first work was done by Umeyama [19] who showed how graphs with the same number of nodes, but different edge structure. The method computes an approximate node permutation matrix by taking the outer-product of the singular vectors of the adjacency matrices for the graphs being matched. Luo and Hancock [7] have shown how the method can be rendered robust to differences in the numbers of nodes using the apparatus of the EM algorithm. Eigenvalues and eigenvectors of the adjacency matrix or the Laplacian matrix have also been used to cluster [20] and index graphs [16]. An alternative to using graph-spectra as features for the purposes of graph matching is to use eigenvector methods to extract a simplified structure from a graph which is more easily matched than the original graph. Although inexact graph-matching is a problem of potentially exponential complexity, the problem can be simplified by decomposing the graphs to be matched into smaller subgraphs or structures.

However, a single eigenvector can not be used to determine more detailed information concerning the random walk. The aim in this paper is to characterise the properties of the random walk in a finer way using the commute times. The *hitting time* $Q(u, v)$ 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. Both the hitting time and the commute time can be computed from the discrete Green's function or pseudo-inverse of the Laplacian matrix. The idea that is central to this paper is to use the commute time to extract a simplified ordered structure from the graph, and to use the structures for the purposes of graph matching. Here we present two rather different, but essentially dual, simplification methods.

The first graph simplification method is based on the concentric layers that result from repeatedly peeling away the boundary of the graph. Our motivation in adopting this representation is that the pattern of concentric layers is less likely to be disturbed by structural noise than the random walk, which can be diverted. To address this problem using the apparatus of the heat equation, we augment the graph with an auxiliary node. This node is connected to each of the boundary nodes by an edge, and acts as a heat source. Concentric layers are characterised using the commute time from the auxiliary node. We match graphs by separately matching the concentric layers.

The second graph simplification method uses the minimum spanning tree associated with the heat kernel as a way of characterising the graph. However, there is a difficulty with directly using the heat kernel, since the time parameter of the kernel must be set. As we will show later in the paper, the spanning trees evolve in a rather interesting way with time. For small time, they are rooted near the centre of the graph, and the branches connect to terminal nodes that are on the boundary of the graph. As time increases, the tree becomes string like, and winds itself from the centre of the graph to the perimeter. As it does so, the number of terminal nodes decreases, i.e. the large time tree has the appearance of a string to which a small number of short branches or ligatures are attached. Hence, a choice must be made in setting the time parameter.

One way to overcome this problem is to use statistical properties of the random walk. Hence, in this paper we use the minimum spanning tree associated with the minimum commute time as a way of characterising the structure of a graph. We construct an auxiliary fully connected graph in which the weights are the commute times between pairs of nodes in the original graph. We then use Prim's method to locate the spanning tree that minimises the sum of weights. The spanning tree is rooted at the node of minimum weight in the auxiliary graph, and this is located near the centre of the original graph.

The commute time has properties that suggest that it can lead itself to the identification of stable spanning trees. 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.

The outline of this paper is as follows. In Section 2 we review the properties of the commute time and its relationship with the Laplacian eigensystem. Section 3 describes how the multi-layer representations can be extracted from graphs using commute time, and outlines how the representation can be matched. In Section 4 we turn our attention to the tree-based representation and its matching. Section 5 presents experiments with both representations and compares their performance. Finally, Section 6 presents conclusions and outlines directions for future investigation.

2 Heat Kernel, Lazy Random Walks, Green's Function and Commute Time

In this section, we review the theory underpinning the computation of the commute time. We commence by introducing the heat-kernel of a graph, next we show how the heat kernel is related to the lazy random walk on a graph, and then we show how the discrete Green's function can be computed from the Laplacian spectrum. Finally, we show that the commute time is a metric that is obtained from the Green's function.

2.1 Heat kernel

Consider the the weighted graph $\Gamma = (V, E, W)$ where V is the set of nodes, $E \subseteq V \times V$ is the set of edges and W is the $|V| \times |V|$ edge weight matrix. Further let $T = diag(d_v; v \in V)$ be the diagonal weighted degree matrix with elements $T_{u,u} = \sum_{v=1}^{n} W(u,v)$ and A be the adjacency matrix. The un-normalized weighted Laplacian matrix is given by $L = T - W$ 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 $\mathcal{L} = \Phi \Lambda \Phi^T$. Here $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_{|V|})$ is the diagonal matrix with the ordered eigenvalues $0 = \lambda_1 \leq \lambda_2 \ldots \leq \lambda_{|V|}$ as elements. The eigenvector matrix $\Phi = (\phi_1|\phi_2|....|\phi_{|V|})$ has the correspondingly ordered eigenvectors as columns.

In the paper we are interested in the heat equation associated with the graph Laplacian. The heat kernel $\mathcal{H}_t$ satisfies the partial differential equation

$$\frac{\partial \mathcal{H}_t}{\partial t} = -\mathcal{L}\mathcal{H}_t$$

where t is time. The solution of the heat-equation is found by exponentiating the Laplacian eigenspectrum 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

$$\mathcal{H}_t(u,v) = \sum_{i=1}^{|V|} \exp[-\lambda_i t]\phi_i(u)\phi_i(v)$$

2.2 *Lazy random walk*

Let us consider the normalised adjacency matrix matrix $\mathcal{P} = I - \mathcal{L} = T^{1/2}PT^{-1/2}$, where I is the identity matrix. We can re-express the heat kernel by performing the McLaurin expansion,

$$\mathcal{H}_t = e^{-t(I-\mathcal{P})} = e^{-t}\sum_{r=1}^{\infty} \mathcal{P}^r \frac{t^r}{r!}$$

Using the spectral decomposition of the normalized Laplacian, we have $\mathcal{P}^r = (I - \mathcal{L})^r = \Phi(I-\Lambda)^r\Phi^T$ and as a result

$$\mathcal{P}^r(u,v) = \sum_{i=1}^{|V|}(1-\lambda_i)^r\phi_i(u)\phi_i(v) = \sum_{\pi_r}\prod_i \frac{W(u_i,u_{i+1})}{\sqrt{d_{u_i}d_{u_{i+1}}}}$$

The normalized probability matrix $\mathcal{P}^r(u,v)$ is hence the sum of the probabilities of all the random walks π of length r connecting node u and v. As a result the heat kernel is the continuous time limit of the lazy random walk. To show this, consider a lazy random walk with transition matrix

$$R = (1-\alpha)I + \frac{W}{T}\alpha$$

The random walk migrates between different nodes with probability α and remains static at a node with probability $1-\alpha$. Let $\alpha = \alpha_0\Delta t$ where $\Delta t = \frac{1}{N}$. Consider the limit $\Delta t \to 0$

$$\lim_{N\to\infty} R^N = \lim_{N\to\infty}\left(I + (\frac{W}{T} - I)\alpha_0\frac{1}{N}\right)^N = e^{(\frac{W}{T}-I)\alpha_0} \quad (1)$$

while

$$\frac{W}{T} - I = T^{-1}A - I = T^{-1}(T - L) - I = -T^{-1}L \tag{2}$$

Now consider the discrete Laplace operator Δ with the property

$$\mathcal{L} = T^{1/2}\Delta T^{-1/2} = T^{-1/2}LT^{-1/2}$$

which implies $\Delta = LT^{-1}$. As a result, we get $\lim_{N\to\infty} R^N = e^{-\Delta\alpha_0}$, which is just the expression for the heat kernel.

2.3 *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 $vol = \sum_{v\in V(\Gamma)} d_v$ is the volume of the graph. A physical interpretation of the Green's function is the temperature at a node in the graph due to a unit heat source applied to the external node. It is related with the heat kernel $\mathcal{H}_t$ in the following manner

$$G(u, v) = \int_0^\infty d_u^{1/2} \left(\mathcal{H}_t(u, v) - \phi_1(u)\phi_1(v)\right) d_v^{-1/2} dt \tag{3}$$

Here ϕ_1 is the eigenvector associated with the zero eigenvalue 0 and which has k-th element is $\phi_1(k) = \sqrt{d_k/vol}$. Furthermore, the normalized Green's function $\mathcal{G} = T^{-1/2}GT^{1/2}$ is defined as (see [3] page 6),

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

where λ and ϕ are the eigenvalue and eigenvectors of the normalized Laplacian $\mathcal{L}$.

The normalized Green's function is hence the generalized 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

$$(\mathcal{L}\mathcal{G})_{uv} = \delta_{uv} - \frac{\sqrt{d_u d_v}}{vol}$$

From Equation 4, 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 [3] page 4), it follows that $\mathcal{G}$ is a kernel. Finally, it is interesting to note that the Green's function is related to $\mathcal{P}$ by

$$\mathcal{G} = (I - \mathcal{P})^{-1} = I + \mathcal{P} + \mathcal{P}^2 + \cdots = \sum_{r=0}^{\infty} \mathcal{P}^r$$

2.4 Commute time

We note that the *hitting time* $Q(u, v)$ 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 $CT(u, v) = Q(u, v) + Q(v, u)$. The hitting time $Q(u, v)$ is given by [3]

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

where G is the Green's function given in Equation (3). As a result, the commute time is given by

$$CT(u, v) = Q(u, v) + Q(v, u) = \frac{vol}{d_u} G(u, u) + \frac{vol}{d_v} G(v, v) - \frac{vol}{d_u} G(u, v) - \frac{vol}{d_v} G(v, u) \tag{5}$$

As a consequence of Equation (5) 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: $||x_i - x_j||^2 =< x_i - x_j, x_i - x_j >=< x_i, x_i > + < x_j, x_j > - < x_i, x_j > - < x_j, x_i >$.

Substituting the spectral expression for the Green's function into the definition of the commute time, it is straightforward to show that

$$CT(u, v) = vol \sum_{i=2}^{|V|} \frac{1}{\lambda_i} \left(\frac{\phi`_i(u)}{\sqrt{d_u}} - \frac{\phi_i(v)}{\sqrt{d_v}} \right)^2 \tag{6}$$

Hence, the commute time is easily computed from the Laplacian eigensystem. Consider the mapping of the nodes of the graph to the space with co-ordinate matrix

$$Y = \sqrt{vol} \Lambda^{-1/2} \Phi^T T^{-1/2}$$

The matrix can be written in the column form $Y = (y_1 | y_2 | .... | y_u | ... y_{|V|})$ where the column vector y_u is the co-ordinate vector of the u^{th} node. Under this embedding of the nodes, the commute time between the the nodes u and v is simply the Euclidean distance between the corresponding co-ordinate vectors, i.e. $CT(u, v) = (y_u - y_v)^T (y_u - y_v)$.

3 Multilayer Graph Representation and Matching

In this section we provide details of the multilayer graph representation. We commence by describing how the representation is constructed and then detail a simple matching method.

3.1 Graph derivation and representation

We commence by constructing an augmented graph from the original graph by adding an auxiliary external node. We refer to this new graph as the *affixation graph*. It is constructed by connecting the additional node to each of the nodes on the boundary (or perimeter) of the original graph. Our aim in constructing this affixation graph is to simulate heat flow from the external node, which acts like an external heat source. We assign the label τ to the auxiliary node, and the *affixation graph* $\mathcal{A}(V^{'}, E^{'})$ can be defined by $V^{'} = V \cup \{\tau\}$ and $E^{'} = E \cup \{(\tau, u), \forall u \in Boundary(\Gamma)\}$.

By analysing the heat-flow from the auxiliary node on the affixation graph, we can generate a multilayer representation of the original graph. The idea is to characterise the structure of the graph using the pattern of heat-flow from the source node. To embark on this study, let us consider the probability of the random walk with a certain path length $\mathcal{P}^r$. We can make an estimate of the heat flow on the graph by taking the average value of $\mathcal{P}^r$ according to the path length r:

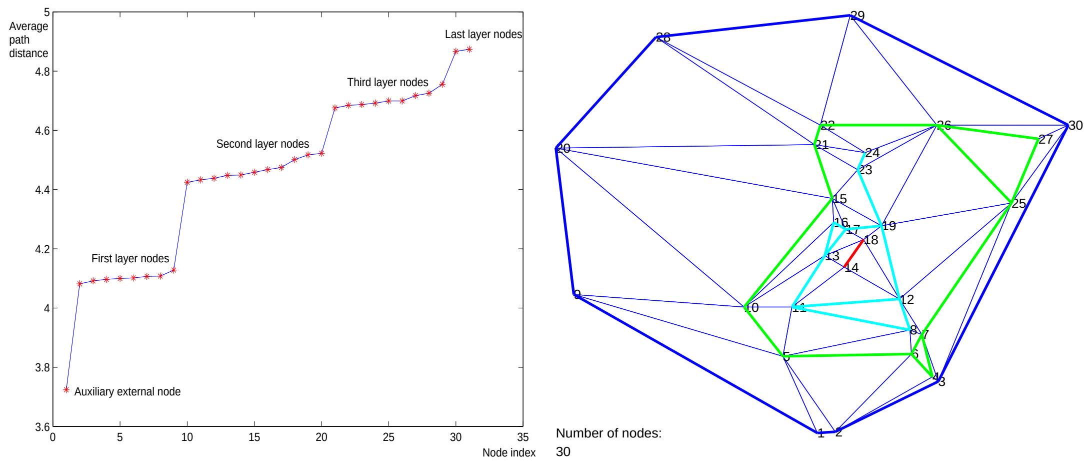
$$\mathcal{D}(u, v) = \frac{\sum_r r\mathcal{P}^r(u, v)}{\sum_r \mathcal{P}^r(u, v)}.$$

We take the external node τ to be the heat source and consider all the random walks starting from the the affixation node τ. The average path distance $\mathcal{D}(\tau, v)$ for all v in V follows a staircase distribution, which we can use to classify nodes into different layers.

Figure 1(a) illustrates this staircase property. The nodes with the same average distance correspond to the same layer of the graph. The corresponding multilayer graph representation is shown in Figure 1(b), where the nodes connected by edges of the same color belong to the same layer.

3.2 Score function and matching process

Our matching process is based on the layers extracted above. To do this, we match the nodes in each layer in one graph to the nodes of the corresponding layer in a



(a) Staircase distribution of the average path distance.

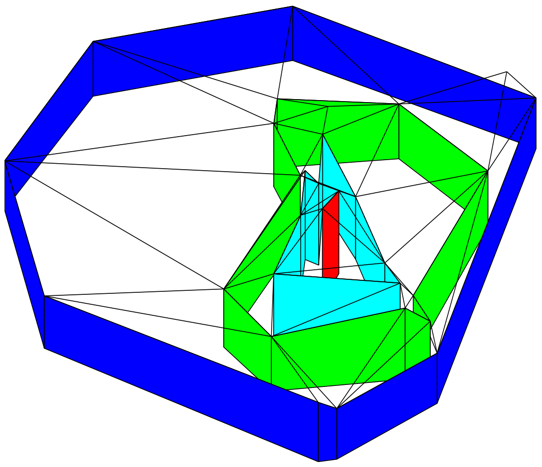
(b) An example of a multilayer graph.

Fig. 1. The staircase distribution and a multilayer graph.

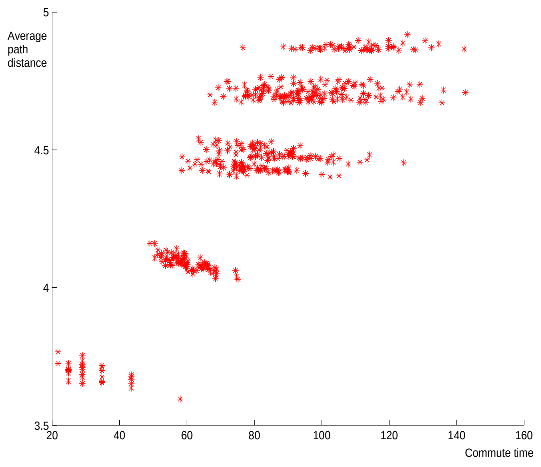
second graph. To do this we need a score-function to distinguish the different nodes in the same layer. Unfortunately, the average path distance can not be used for this purpose, since it is too coarsely quantised and can not be used to differentiate between the nodes in the same layer of a graph. We seek a score function which is related to the heat kernel, and hence the heat-flow from the external source node, but gives more salient values for each individual node.

Here we define the score function S_u for node u as $S_u = CT(\tau, u)$ which is the commute time between node u and the external source node τ. Figure 2(a) shows a visualisation of the score functions for the Delaunay graph in Figure 1(b). The score function is visualised as the height on the edges of the concentric layers of the graph. The scores for the nodes on the same layer are salient enough to distinguish them. In Figure 2(b) we show a scatter plot of commute times $CT(u, v)$ versus the average path length distance $\mathcal{D}(u, v)$. From this plot it is clear that the commute time varies more smoothly and has a longer range than the commute time.

Since we have divided the graph into several separate layers, our graph matching step can proceed on a layer-by-layer basis. To perform the matching process we peel layers of nodes from the boundary inwards. Each layer is a cycle graph where each node is connected to its two adjacent nodes only. In the case when a node has only one neighbour in the layer, the edge between them is duplicated to form a cycle. We match the nodes in the corresponding layers of different graphs by performing a cyclic permutation of the nodes. The cyclic permutation permits possible null-insertions to accommodate missing or extraneous nodes. The cyclic permutation minimises the sum-of-differences in commute times between nodes in the graphs being matched. If C_l denotes the set of nodes in the kth layer of the graph, then the permutation ρ minimises the cost function



(a) 3D visualisation of the scores on the nodes.



(b) Scatter plot of the commute time and the average path distance.

Fig. 2. 3D score visualisation and the scatter plot.

$$\mathcal{E}(\rho) = \sum_{k \in V} \sum_{l \in C_k^M} \sum_{m \in C_k^D} (S_l - S_{\rho(m)})^2$$

4 Minimum Spanning Tree Representation and Matching

In this section we describe our representation based on the minimum spanning tree of the hest kernel. As in the previous section, we commence by describing how the representation is constructed, and then detail our matching algorithm.

4.1 Robust graph representation by trees

Our aim here is to re-cast the inexact graph matching problem as an inexact tree matching problem. The main obstacle here is to locate a tree that is stable to structural variations in the original graph. One way to do this is to extract a minimum spanning trees from the graphs under study. However, unless care is taken, then the structure of the extracted spanning trees will vary in an erratic manner with slight changes in the structure of the original graph. This makes reliable matching impossible. By reducing the graph into a tree, although we obtain a simpler data structure, we also loose information. Hence, we need a means of extracting a stable tree-like graph representation but at the same time preserving as much information from the original graph as possible. Here we argue that commute time provides a solution to this problem.

Given a weighted graph Γ, we generate the commute time matrix CT by computing the commute time between each pair of nodes. From the commute time matrix we construct a complete or fully connected graph Θ. The weights of the edges in this graph are the commute-times. In another word, the weight matrix W of the new graph Θ satisfies: $W_{\Theta}(u, v) = CT(u, v)$. Our representation is based on the minimum spanning tree of the fully connected graph Θ with commute times as weights. The node weights on the spanning tree are found by summing the edge weights. The weight on the node u is

$$\Omega(u) = \sum_{u,v \in V} CT(u, v)$$

The root node of the tree is that having the smallest node-weight and the minimum spanning tree is generated by the Prim's method [11] starting from the root node.

Since commute time is a metric on the original graph and it captures global information rather than the local information, it is likely to be relatively stable to structural modifications. 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. For example, if there is node deletion or edge deletion, then since wherever possible the random walk moves to connect two nodes, the effect of that corruption is small. The stability of the commute time matrix ensures that the weight distribution on the derived fully connected graph is stable. Hence, the minimum spanning tree can also be anticipated to be stable.

Edges of the spanning tree correspond to the path of the most probable random walk. The weights on the nodes of the spanning tree preserve structural information from the original graph. The nodes on the boundary of a graph together with those of small degree are relatively inaccessible to the random walk. The reason for this is that they have a larger average commute time than the remaining nodes. By contrast, the nodes in the interior of the graph and the nodes with large degree are more accessible, and hence have a smaller average commute time. The most frequently visited nodes in the tree is that with the smallest average commute-time, and this is the root node. This node is usually located near the the center of a graph and has a large degree.

Two examples are shown in Figure 3 and Figure 4. In these two figures, we have shown two types of graphs. The first of these is the Delaunay graph and the second is the K-nearest neighbour graph. We have also shown the commute time matrices for the two graphs, the generated complete or fully connected graph and the minimum spanning tree. The main features to note from the plots are as follows. First, the spanning trees are rather different in structure. Second, there is a more defined block structure in the commute time matrix for the K-nearest neighbour graph.

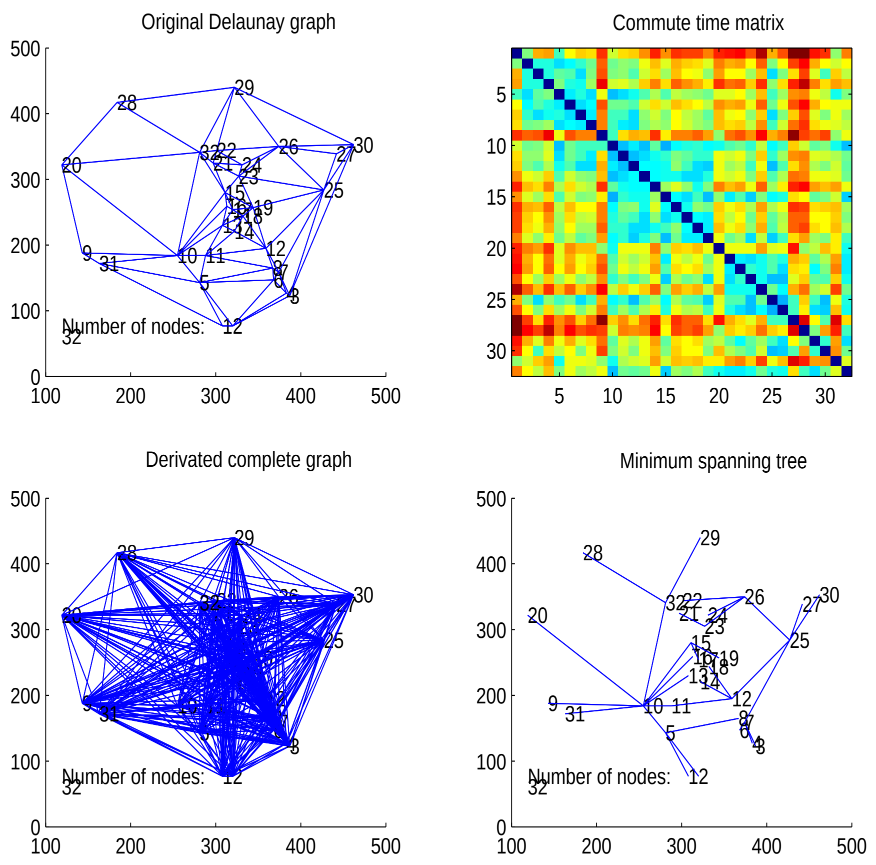


Fig. 3. Delaunay graph example.

To illustrate the problems associated with using the heat-kernel to locate the spanning tree, consider the continuous time random walk on the graph. Let $\vec{p}_t$ be the vector whose element $p_t(i)$ is the probability of visiting node i of the graph under the random walk. The probability vector evolves under the equation

$$\frac{\partial \vec{p}_t}{\partial t} = -\mathcal{L}\vec{p}_t$$

which has the solution

$$\vec{p}_t = \exp[-\mathcal{L}t]\vec{p}_0$$

As a result $\vec{p}_t = \mathcal{H}_t\vec{p}_0$. Consequently the heat kernel determines the random walk. Hence, if we use the heat kernel as the edge weight function of the graph then we can explore how the spanning spanning trees associated with the heat kernel evolve with time.

In Figure 5 for one of the graphs used in our experiments, we illustrate the evolution of the spanning tree with time. The first image in the sequence shows the input graph, and the remaining images show the recovered spanning trees as time elapses. Initially, the tree is rooted near the centre of the graph with terminal nodes on the boundary. The recovered tree has many branches and is very “bushy”. As time evolves, the pattern changes. The tree becomes rather string-like and wraps itself

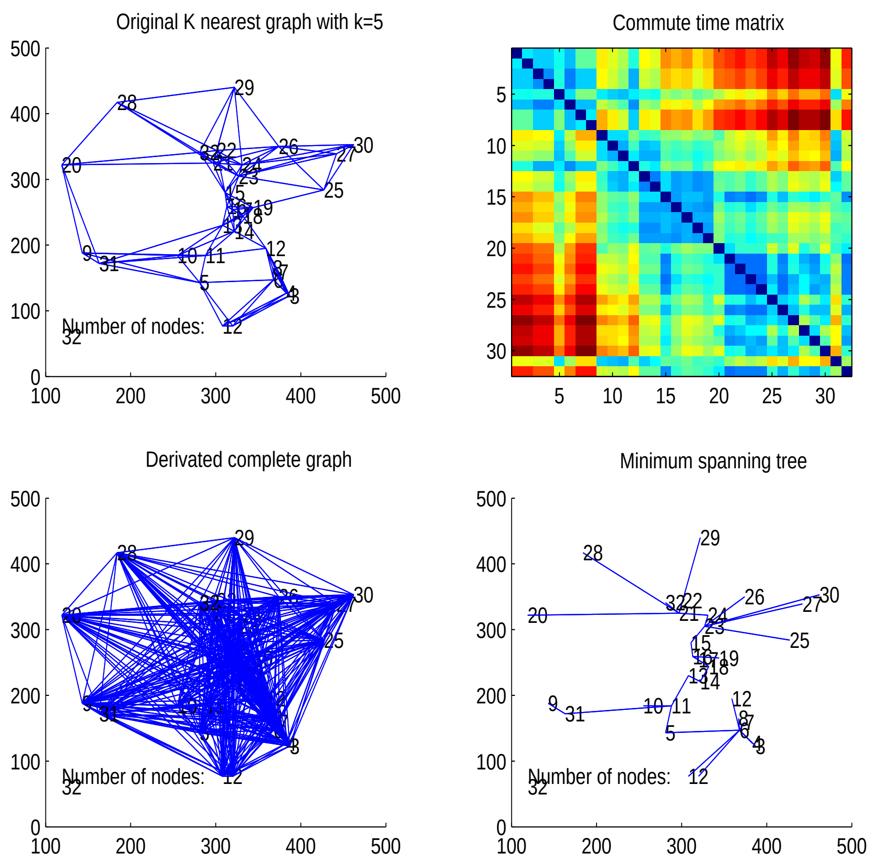


Fig. 4. K nearest neighbour graph example.

around the boundary, with branches extending it to the centre of the original graph. Hence, the structures are unstable and not suitable for matching.

4.2 Tree edit distance and inexact tree matching

With stable minimum spanning trees to hand, then the next step is to match them. Here we use Torsello and Hancock's [18] divide and conquer tree matching method. The method provides a means of computing the tree edit distance, and locates the matches that minimise the distance using relaxation labelling. To compute the tree edit distance, the algorithm exploits the fact that any tree obtained with a sequence of node deletion operations is a subtree of the transitive closure of the original tree. As a result the inexact tree matching problem can be cast as that of locating the maximum common subtree by searching for maximal cliques of the directed association graph. The methods poses the matching problem as a max clique problem, and uses the relaxation labelling method of Pelillo [10,9] to obtain a solution.

The steps of the divide and conquer method are as follows:

(1) Given two trees t and t', calculate their transitive closure TC_t and $TC_{t'}$.

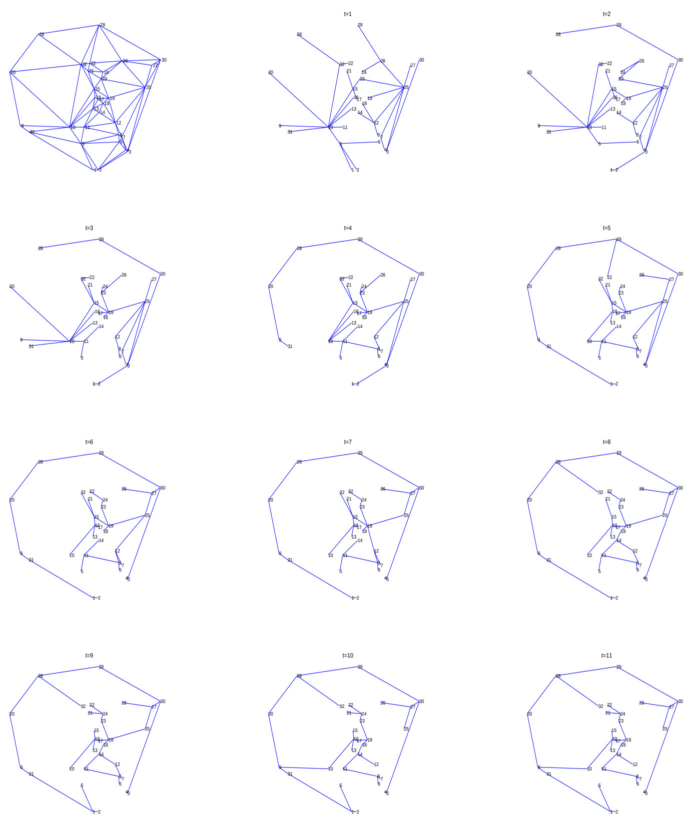


Fig. 5. Minimum spanning tree with varying t.

(2) Construct the directed association graph (DAG) of TC_t and $TC_{t'}$.
(3) The inexact tree matching problem can be solved by finding the common consistent subtree of the two DAGs.
(4) The problem of locating the maximum common subtree can be transformed into that of locating a max-weighted clique. This can be effected using a number of classical methods, including relaxation labelling [18] or quadratic programming [10].

5 Experiments

In this section, we carry out an experimental evaluation of our two graph simplification methods. We test them on both Delaunay graphs and K-nearest neighbour graphs with variable size. We also evaluate the stability of our methods with respect to edge corruption. Finally, we compare the properties of the two simplification methods.

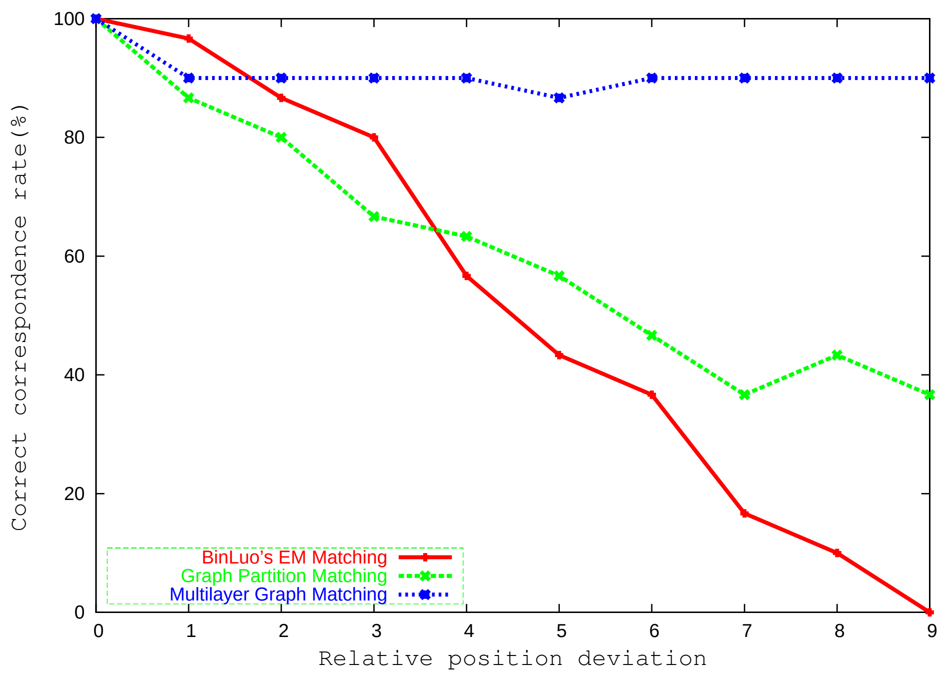


Fig. 6. Comparison of results.

5.1 Multilayer graph matching

The data used in our study is furnished by a sequence of views of a model-house taken from different camera viewing directions. In order to convert the images into abstract graphs for matching, we extract point features using a corner detector. Our graphs are the Delaunay triangulations of the corner-features. Examples of the images are shown in Figure 7.

We have matched the first image to each of the subsequent images in the sequence by using the multilayer matching method outlined earlier in this paper. The results are compared with those obtained using the method of Luo and Hancock [7] and the partition matching method of Qiu and Hancock [12] in Table 1. This table contains the number of detected corners to be matched, the number of correct correspondences, the number of missed corners and the number of miss-matched corners. Figure 6 shows the correct correspondence rate as a function of the difference in view number for the three methods based on the data in Table 1.

From the results, it is clear that our new method out performs both Luo and Hancock's EM method and, Qiu and Hancock's partition matching method for large differences in viewing angles. Figure 8 shows the results for some example image pairs. There are clearly significant structural differences in the images from which the graphs are extracted including rotation, scaling and perspective distortion. Even in the worst case, our method has a correct correspondence rate of 86.7%.

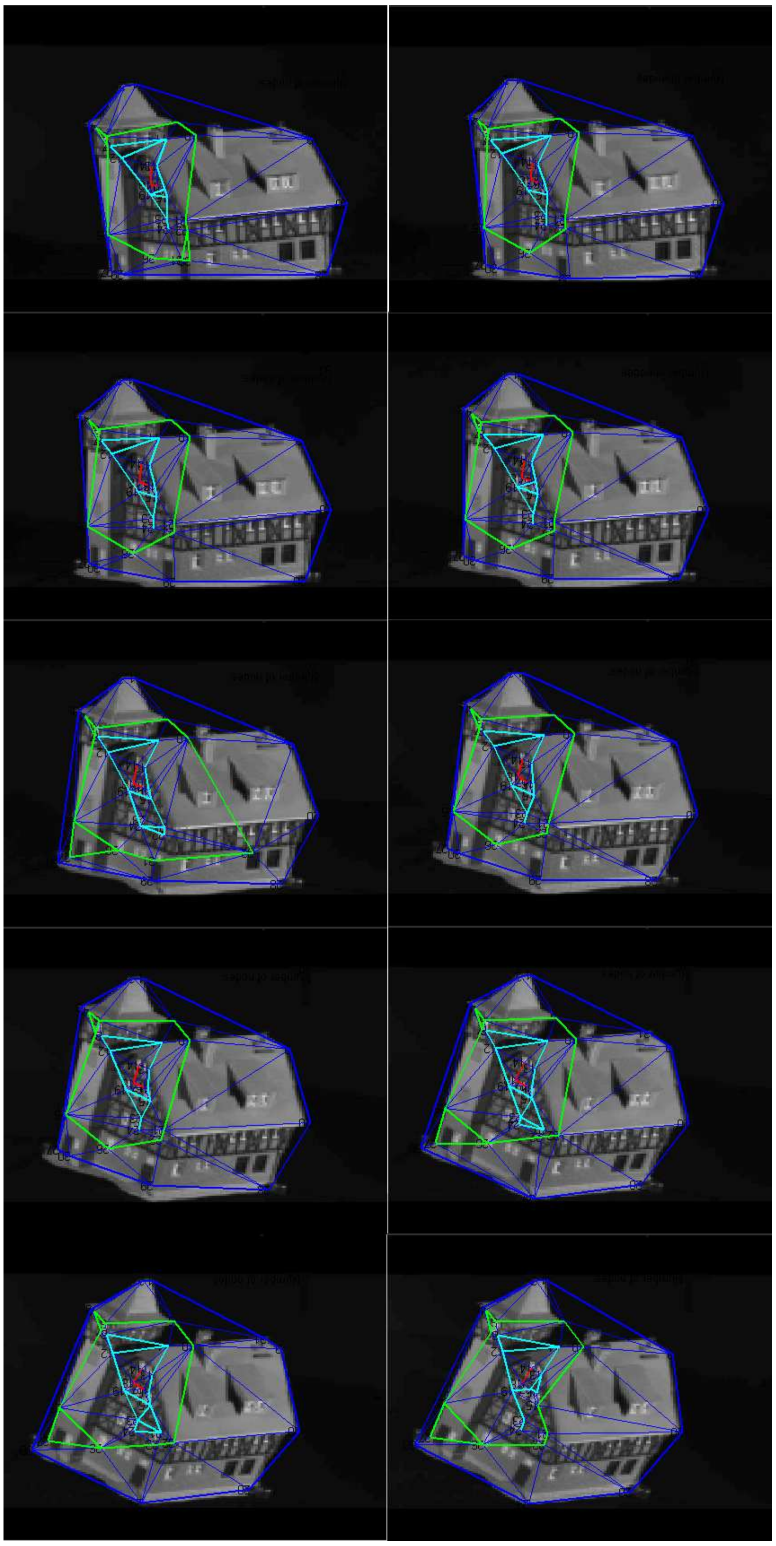
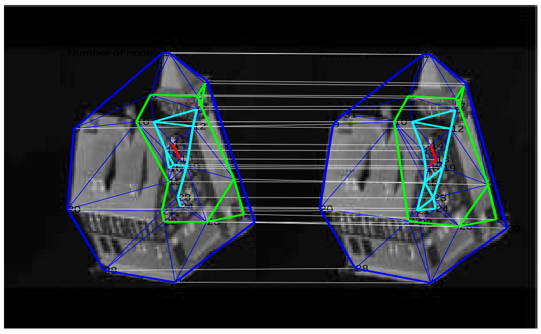
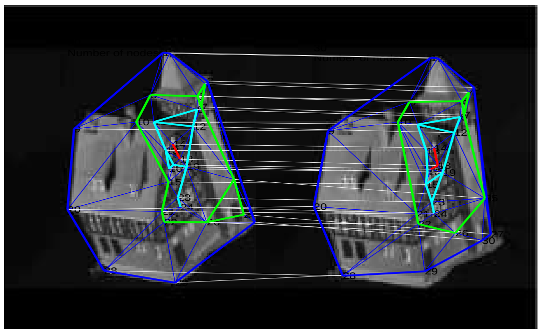


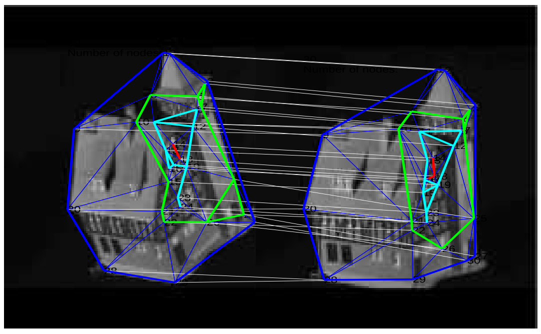
Fig. 7. CMU house sequence.



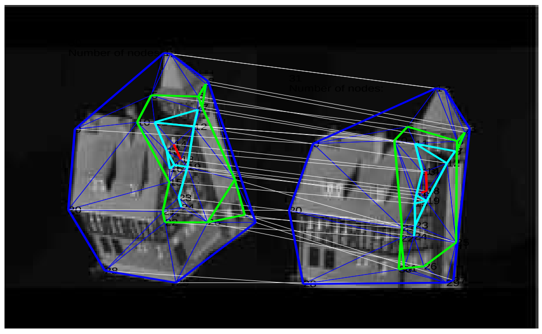
(a) 1st image to 2rd image.



(b) 1st image to 5th image.



(c) 1st image to 7th image.



(d) 1st image to 10th image.
Fig. 8. Matched samples.

Method	House index	0	1	2	3	4	5	6	7	8	9
	Corners	30	32	32	30	30	32	30	30	30	31
EM[7]	Correct	-	29	26	24	17	13	11	5	3	0
	False	-	0	2	3	8	11	12	15	19	24
	Missed	-	1	2	3	5	6	7	10	8	6
Partition matching[12]	Correct	-	26	24	20	19	17	14	11	13	11
	False	-	3	5	8	11	12	16	15	17	19
	Missed	-	1	1	2	0	1	0	4	0	0
Multilayer matching	Correct	-	27	27	27	27	26	27	27	27	27
	False	-	3	3	2	2	3	2	2	2	2
	Missed	-	0	0	1	1	1	1	1	1	1

Table 1
Correspondence results and comparison with the other methods.

5.2 Inexact graph matching by spanning trees

We now turn our attention to illustrating the utility of our spanning tree representation for graph matching. We investigate the robustness of the method under local structural change as well as random edge corruption.

5.2.1 Spanning Tree Robustness

In this section, we aim to compare the stability of the spanning trees delivered by our commute time method with those obtained directly using the Prim's method [11].

The data used here is furnished by the sequences of views of model-houses. The images in the sequence are taken from different camera directions. In order to convert the images into abstract graphs for matching, we extract point features using a corner detector and construct the nearest neighbour graph of the points.

In Figure 9, we show three groups of houses with an increasing complexity in terms of the number of points detected and the image structure. Five examples are shown in each group in a column order. In each group, the top row shows the original images overlaid with their 5 nearest neighbour graph, the second row the spanning trees obtained from Prim's method and the third row the spanning trees obtained using our commute time method.

It is clear from the first group of images in the figure that our method delivers more stable spanning trees. As the view point changes, there is little change in the spanning tree structure. In the second group, the total number of feature points has been approximately doubled and the structure of the extracted 5-nearest neighbour graph is more variable. Our commute time method still delivers very stable spanning trees. Compared with the second row in this group, our spanning trees do not result in erroneous disconnections or connections of the branches and maintain a consistent tree shape. The third group is the most complex one with approximately three times the number of nodes as the first group. Although the trees are quite complex, they are still stable and the local structure are well preserved.

5.2.2 Inexact Graph Matching:

Matching with local structure variance: The data used here is the same as the previous section. However, here we study Delaunay graphs in addition to the k-nearest neighbour graph (with varying k).

In Figure 10, we show five examples from the sequence of 30 views of the house. The top row shows the original image, the second row the Delaunay graphs, the third row the minimum spanning trees obtained from the Delaunay graph commute times, the fourth row the 5 nearest neighbour graphs, and the fifth row the minimum spanning tree obtained from the k-nearest neighbour graph commute times. From the figure it is clear that although the structure of the graphs varies, the spanning trees are quite stable under these changes. This demonstrates that the minimum spanning tree delivered by the commute time can be used as a simple but stable graph representation. It is also interesting to note that the k-nearest neighbour graph gives more stable trees than the Delaunay graph.

Next we aim to investigate whether the spanning trees can be used for the purposes of graph-matching. We have matched the first image in the sequence to each of the subsequent images using the divide and conquer tree matching method outlined earlier in this paper. The results are compared with those obtained using the method of Luo and Hancock [7] and the partition matching method of Qiu and Hancock [12]. Figure 11 shows us the correct correspondence rate as a function of the difference in view number. From the results, it is clear that our new method outperforms both Luo and Hancock's EM method and, Qiu and Hancock's partition matching method for large differences in viewing angles. It also demonstrates that the k-nearest neighbour graph outperforms the Delaunay graph in delivering stable structure. There are clearly significant geometric distortions present in the images including effects due to rotation and perspectivity, and these give rise to significant structural differences in the resulting graphs. Even in the worst case, our method based on the k-nearest neighbour graph has a correct correspondence rate of 80%.

Matching with Random Edge Corruption: We now focus on testing the stability the spanning trees under controlled random noise. To do this we delete a controlled fraction of edges from the initial graphs (either Delaunay or k-nearest neighbour).

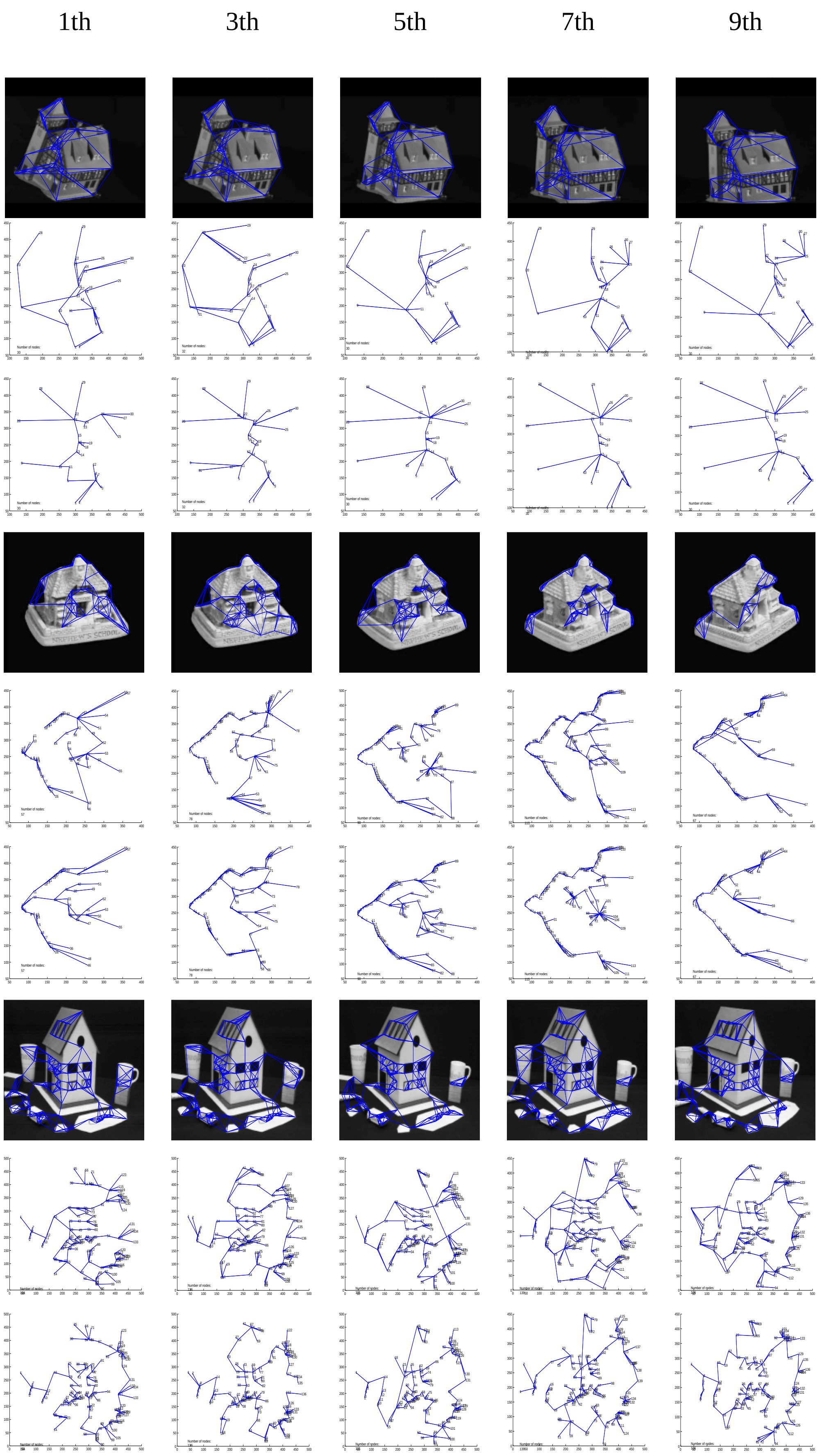


Fig. 9. Three sequences of model houses with their spanning tree representation.

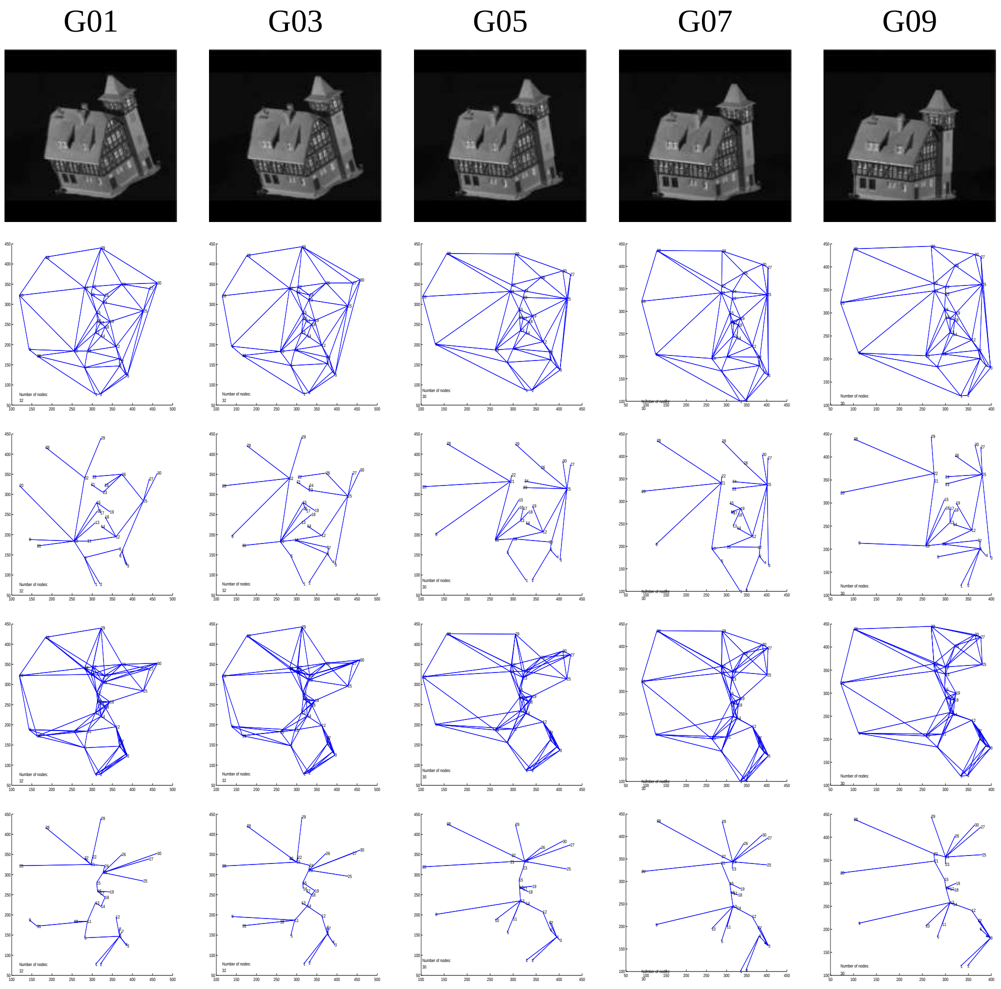


Fig. 10. House images, their graphs and extracted trees.

In Figure 12 we show the effect of this deletion process for the graphs shown earlier. The number at the top of each column is the percentage of edges deleted. The first and the third rows of the figure show the Delaunay graph and the 5-nearest neighbour graph after edge deletion. The second and fourth rows show the corresponding spanning trees. From the figure it is clear that the tree structure is stable under edge corruption, and again the k-neatest graph outperforms the Delaunay graph.

We have matched the edge-corrupted trees to the original trees, and have computed the fraction of correct correspondences. The results are shown in Figure 13. The fraction of correct correspondences decreases in a linear fashion with edge corruption. The different curves in the plot are for the Delaunay graph and the k-nearest neighbour graph. The k-nearest neighbour graph outperforms the Delaunay graph by a margin of about 10% at 50% edge corruption.

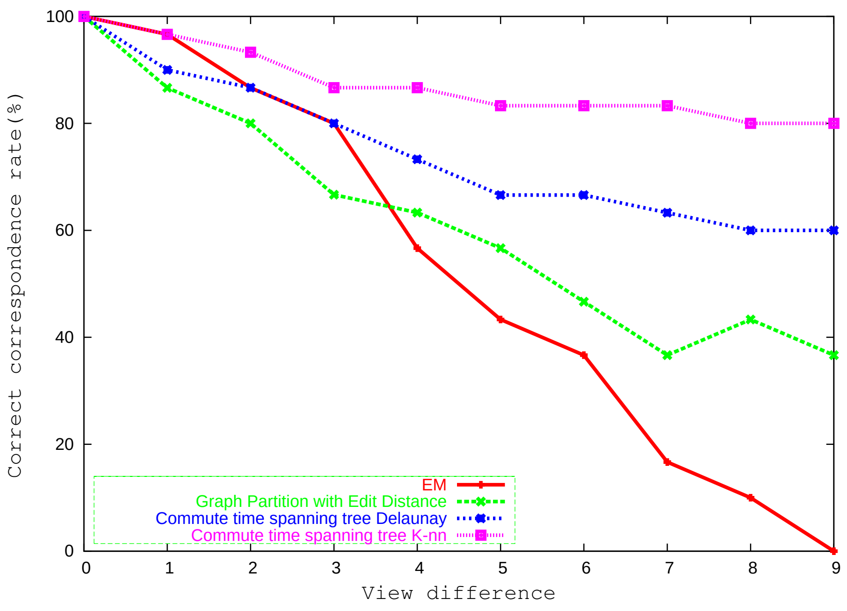


Fig. 11. Comparison of results.

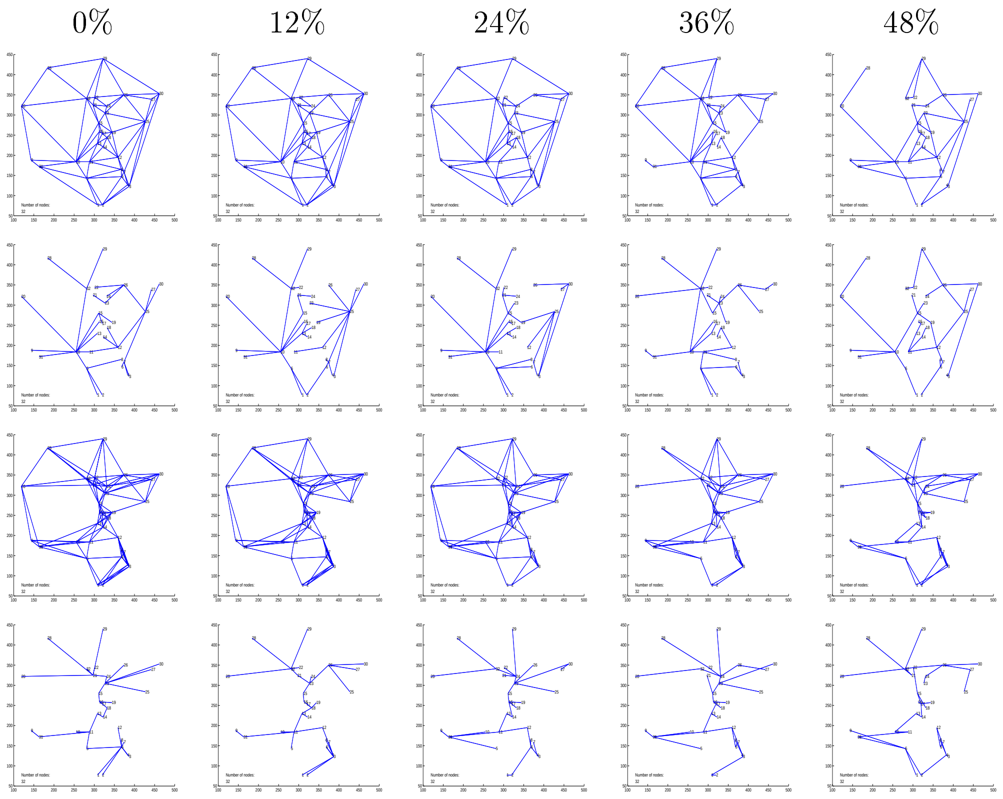


Fig. 12. Random edge deletion.

5.3 *Comparison of the two simplified graph representations*

We now explore the relative merits of the two graph simplification methods presented in this paper. Figure 14 shows the fraction of correct matches for the images

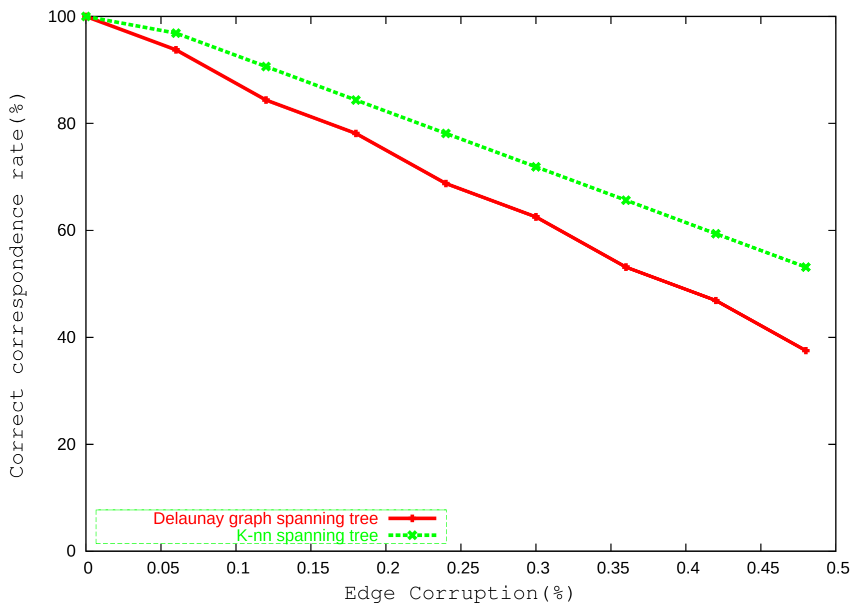


Fig. 13. Graph corruption matching results.

in the CMU house data-set. The curves in the plot are taken from Figures 11 and 13. Additionally, as a pink line we show the result of using the multi-layer simplification method on the 5 nearest neighbour graphs. From the figure it is clear that the multi-layer simplification method delivers the best performance when used with Delaunay graphs (blue line). For the Delaunay graph, the multi-layer representation is stable under graph variation and the composition of each layer is not modified significantly (see Figure 6 for an illustration). However, when this is applied to the 5-nearest neighbour graph, it does not perform well. The reason for this is that the k-nearest neighbour graph does not lend itself to a layer decomposition. Figure15 illustrates the problems. In this example, note how the nodes of the inner green layer are compressed together.

The spanning tree representation gives the best performance when applied to the 5-nearest neighbour graphs and the worst performance for the Delaunay graphs. The reason is that the tree representations for the 5-nearest neighbour graphs are more stable than those for the Delaunay graphs under variations in graph structure (for a comparison see Figure 10).

Overall the stability of the spanning tree representation is better than that for the multi-layer representation. The reason for this is that the structure of layers can be adversely affected by edge corruption. of the edges. An example is shown in Figure 16 with 12.6% of edges randomly pruned. In this example, the connectivity of the layer graph displayed in light blue is destroyed.

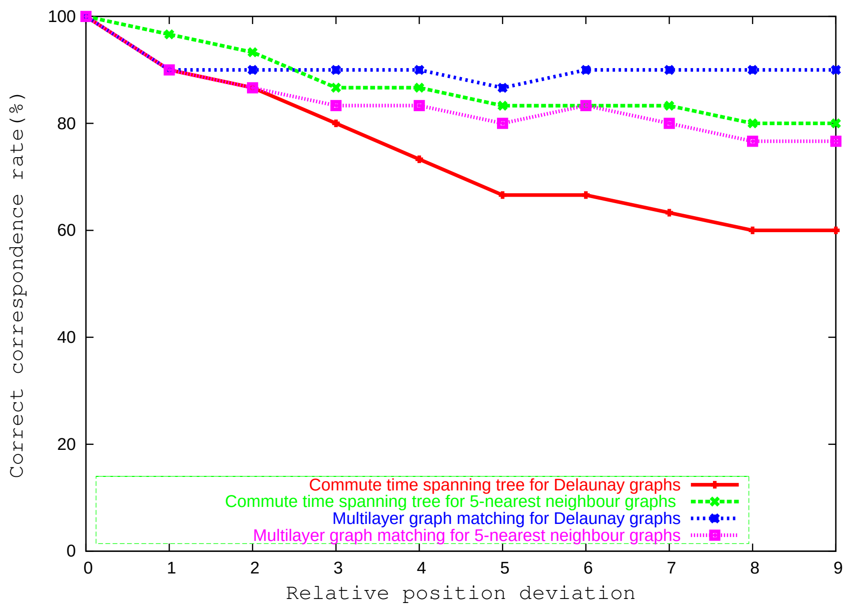


Fig. 14. Comparison of the two methods on graph matching.

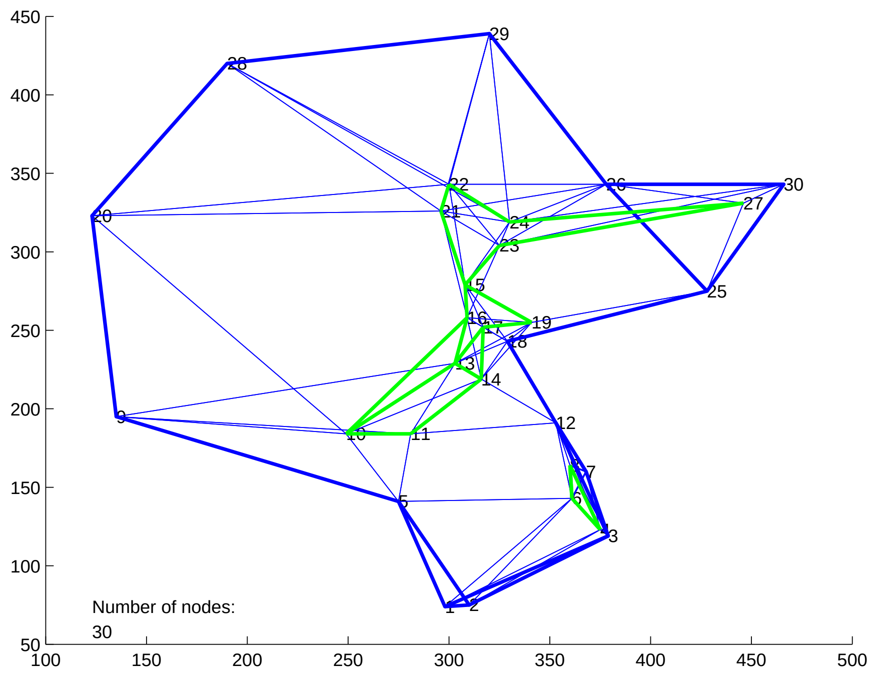


Fig. 15. An example of multi-layer graph of a 5 nearest neighbour graph.

6 Conclusion

In this paper we have described the how commute time can be computed from the Laplacian spectrum. This analysis relies on the discrete Green's function of the graph, and we have reviewed the properties of Green's function. Two of the most important of these are that the Green's function is a kernel and that the commute

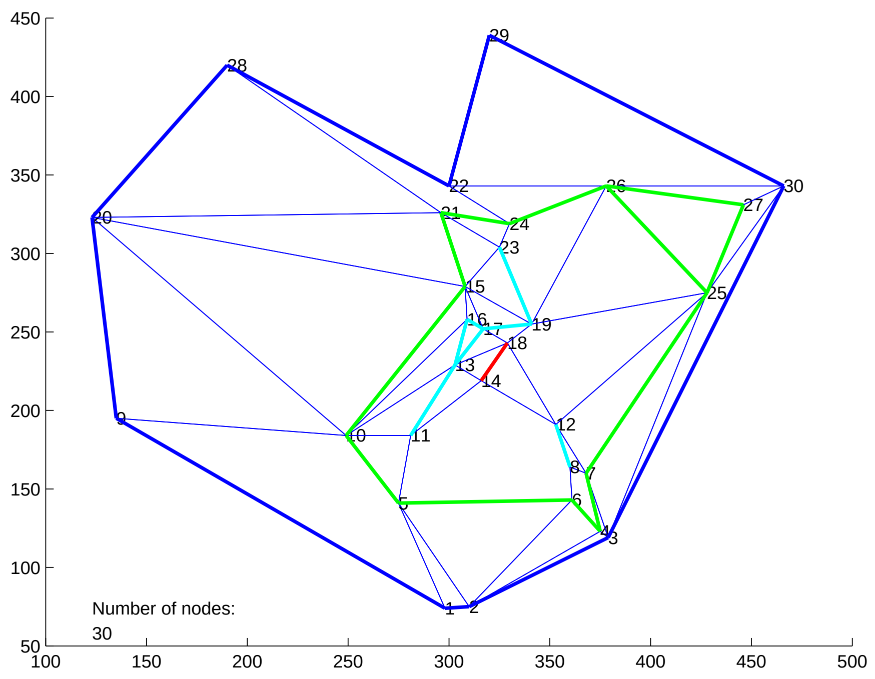


Fig. 16. An example of a multi-layer graph with edge corruption.

time is a metric.

We have shown how to use the commute time to develop two graph simplification algorithms. The first of these uses the commute time to an auxiliary node to extract a multi-layer representation. The second simplification method uses the commute time to extract spanning trees from graphs. Experimentally, we show that our tree representation is not only stable, but also preserves sufficient structural information to be useful for the purposes of graph matching.

References

[1] S. Brin and L.Page. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*, 30(1–7):107–117, 1998.

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

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

[4] M. Gori, M. Maggini, and L. Sarti. Graph matching using random walks. In *ICPR04*, pages III: 394–397, 2004.

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

[6] L. Lovász. Random walks on graphs: A survey. *Combinatorics, Paul Erds is eighty*, 2:353–397, 1996.

[7] B. Luo and E. R. Hancock. Structural graph matching using the em algorithm and singular value decomposition. *IEEE PAMI*, 23(10):1120–1136, 2001.

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

[9] M. Pelillo. Replicator equations, maximal cliques, and graph isomorphism. *Neural Computation*, 11:1933–1955, 1999.

[10] M. Pelillo, K. Siddiqi, and S. W. Zucker. Attributed tree matching and maximum weight cliques. In *ICIAP'99-10th Int. Conf. on Image Analysis and Processing*, pages 1154–1159. IEEE Computer Society Press, 1999.

[11] R.C. Prim. Shortest connection networks and some generalisations. *The Bell System Technical Journal*, 36:1389–1401, 1957.

[12] H. Qiu and E.R. Hancock. Spectral simplification of graphs. In *ECCV*, 2004.

[13] A. Robles-Kelly and E. R. Hancock. Graph edit distance from spectral seriation. *IEEE TPAMI*, 27:365–378, 2005.

[14] A. Robles-Kelly and E. R. Hancock. String edit distance, random walks and graph matching. *IJPRAI*, 18:315–327, 2005.

[15] M. Saerens, F. Fouss, L. Yen, and P. Dupont. The principal components analysis of a graph, and its relationships to spectral clustering. In *ECML*, volume 3201, pages 371–383, 2004.

[16] Ali Shokoufandeh, Diego Macrini, Sven J. Dickinson, Kaleem Siddiqi, and Steven W. Zucker. Indexing hierarchical structures using graph spectra. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(7):1125–1140, 2005.

[17] V. Sood, S. Redner, and D. ben Avraham. First-passage properties of the erdoscrenyi random graph. *J. Phys. A: Math. Gen.*, pages 109–123, 2005.

[18] A. Torsello and E.R. Hancock. Computing approximate tree edit distance using relaxation labelling. *3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition*, pages 125–136, 2001.

[19] S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE PAMI*, 10:695–703, 1988.

[20] R.C. Wilson, B. Luo, and E.R. Hancock. Pattern vectors from algebraic graph theory. *IEEE PAMI*, 27:to appear, 2005.