Spectral Methods for Computer Vision Problems

Huaijun Qiu

Submitted for the degree of Philosophy Doctor

Department of Computer Science

THE UNIVERSITY of York

October 15, 2006

Abstract

Graph spectral methods are concerned with using the eigenvalues and eigenvectors of the adjacency or Laplacian matrices to characterise graph structure. Applications in computer vision include object recognition, image segmentation and data analysis. Although widely used, most graph spectral algorithms are relatively simple. Most of the current applications are limited to use only one or just a few eigenvalues and eigenvectors of the affinity matrix. Although elegant and concise many valuable properties are also neglected. In this thesis, we focus on exploring more complex uses of the Laplacian spectrum.

Our starting point is the Fiedler vector, i.e. the second smallest eigenvector of the Laplacian matrix. Although it has been intensively applied in graph bipartition and image segmentation, its usage is still quite simple and restricted. We aim to further extend its utility to decompose graphs into non-overlapping partitions. By doing so, we will be able to cast inexact graph matching problem into the matching of these subunits and the whole matching process can be realized in a hierarchical framework. Further, the pattern of partitions can be stabilised by incorporating a diffusion process to smooth away the effects of structural errors. The matching criteria is given by two comparable methods: one is dictionary-padding based discrete relaxation and the other one is an edit distance measure. To test our method, we have applied it to both synthetic and real-world images and the results show that it is robust under severe structural corruption and variation.

Our second contribution in this thesis is to develop spectral methods which are capable of utilising the full Laplacian eigenspectrum effectively. We turn to the commute time (the expected time a random walk takes from node u to node v and return). A theoretical analysis of the commute time demonstrates how it can be used for embedding and clustering.

The first application of commute time is to apply it as an energy dissipation measure on nodes of the graph. To simulate a diffusion process, we introduce an extra node as heat source. Then a graph can be divided into several concentric layers based on the heat distribution and graph matching is realized by matching these layers with commute times as attributes on the nodes.

The second application of commute time relies on the robustness of the commute time matrix to structural noise. Commute time is a more robust graph representation than the adjacency matrix. As a result, the minimum spanning tree for the commute time of the graph is more stable under structural variation and can be used as a stable structure for inexact graph matching.

Our third application of commute time exploits its grouping properties. We propose an image segmentation method based on the recursive bipartition of the smallest eigenvector of the commute time matrix.

Finally, a commute time preserving embedding is used to solve the multi-body motion tracking problem. We extend the traditional *factorisation method* of Costeira and Kanade (Costeira and Kanade, 1997; Costeira and Kanade, 1995) by embedding the shape interaction matrix into a subspace. Object points in this space are easily separated by a k-means algorithm.

Contents

List of Figures

List of Tables

Glossary of Symbols

Γ	Graph
A	Adjacency matrix
T	Degree matrix
Δ	Laplace operator
L	*Un-normalised* Laplacian matrix
$\mathcal{L}$	*Normalised* Laplacian matrix
Λ	Eigenvalue matrix of *un-normalised* Laplacian matrix L
Φ	Eigenvector matrix of *un-normalised* Laplacian matrix L
Λ'	Eigenvalue matrix of *normalised* Laplacian matrix $\mathcal{L}$
Φ'	Eigenvector matrix of *normalised* Laplacian matrix $\mathcal{L}$
$\mathcal{H}_t$	Heat kernel
G	Green's function
$\mathcal{G}$	*normalised* Green's function
O	Hitting time
CT	Commute time
D_t	Diffusion distance
Q	Shape index matrix
W	Motion co-ordinates matrix
M	Motion matrix
S	Shape matrix
r	Rank
P	Transition probability matrix
Θ	Co-ordinates matrix in the emededed space
C	Covariance matrix
N_u	Neighbour nodes set of node u
K	Kernel
ϵ	Objective function
$\mathcal{P}$	*Normalised* adjacency matrix
l	Path length of random walk on the graph
W_p	Path-weighted matrix
$\mathcal{T}$	Edit transformation

Acknowledgement

I would like to express my deep and sincere gratitude to my supervisor Professor Edwin R. Hancock for his support and advises throughout my PhD. Without his insightful guidance, it would not have been possible for me to complete this thesis.

I would also like to thank my assessor Dr Richard C. Wilson for his excellent suggestions and encouragement.

My former colleagues Dr A. Torsello, Dr A. Al-Shaher, Mr N. Nasios, Mr A. Doshi et al. supported me in my research work. I want to thank them for all their help, support, interest and valuable hints.

Finally and especially, I would like to give my special thanks to my parents for their love and support.

Declaration

I declare that the work in this thesis is solely my own except where attributed and cited to another author. Most of the material in this thesis has been previously published by the author. For a complete list of publications, please refer to the next page.

List of Publications

The following is a list of publications that has been produced during the course of my research.

2006

- Huaijun Qiu and Edwin R. Hancock, ”Graph Matching and Clustering using Spectral Partitions”, Pattern Recognition, Vol. 39, part 1, pages 22-34, 2006.

- Huaijun Qiu and Edwin R. Hancock, ”Robust Multi-body Motion Tracking using Commute Time Clustering”, 9th European Conference on Computer Vision (ECCV 2006), to appear.

2005

- Huaijun Qiu and Edwin R. Hancock, ”Image Segmentation using Commute Times”, 16th British Machine Vision Conference (BMVC 2005), UK, pages 929-938, 2005.

- Huaijun Qiu and Edwin R. Hancock, ”Commute Times, Discrete Green’s Functions and graph Matching”, 13th International Conference on Image Analysis and Processing (ICIAP 2005), Italy, 2005.

- Fan, Zhang, Huaijun Qiu and Edwin R. Hancock, ”Evolving Spanning Trees Using the Heat Equation”, 11th International Conference on Computer Analysis of Images and Patterns (CAIP 2005), France, pages 272-279, 2005.

- Huaijun Qiu and Edwin R. Hancock, ”Commute Times for Graph Spectral Clustering”, 11^{th} International Conference on Computer Analysis of Images and Patterns (CAIP 2005), France, pages 128-136, 2005.

- Huaijun Qiu and Edwin R. Hancock, ”A Robust Graph Partition Method from the Path-weighted Adjacency Matrix”, Graph Based Representations in Pattern Recognition (GBRPR 2005), France, pages 362-372, 2005.

2004

- Huaijun Qiu and Edwin R. Hancock, ”Grey Scale Skeletonisation with Curvature Sensitive Noise Damping”, Syntactical and Structural Pattern Recognition (SSPR 2004), Portugal, pages 461-469, 2004.

- Huaijun Qiu and Edwin R. Hancock, ”Grey Scale Image Skeletonisation from Noise-Damped Vector Potential”, 17^{th} International Conference on Pattern Recognition (ICPR 2004), UK, pages 839-842, 2004.

- Huaijun Qiu and Edwin R. Hancock, ”Spectral Simplification of Graphs”, 8^{th} European Conference on Computer Vision (ECCV 2004), Prague, pages 114-126, 2004.

2003

- Huaijun Qiu and Edwin R. Hancock, ”Graph Partition for Matching”, Graph Based Representations in Pattern Recognition (GBRPR 2003), UK, pages 178-189, 2003.

Submitted Papers

- Huaijun Qiu and Edwin R. Hancock, ”Robust Graph Representation from the Heat Kernel Path-weight Distribution”, Submitted to Pattern Recognition.

- Huaijun Qiu and Edwin R. Hancock, ”Spectral Graph Matching and Simplification Method”, Submitted to Pattern Recognition.

- Huaijun Qiu and Edwin R. Hancock, ”Image Segmentation and Multi-body Motion Tracking from Commute Times”, Submitted to IEEE Transactions on Pattern Analysis and Machine Intelligence.

Chapter 1

Introduction

1.1 The Problem

This thesis is concerned with using graph spectral methods to solve problems from computer vision. Although graph spectral methods have been successfully applied to many computer vision problems, including graph matching, graph embedding and clustering, the principle behind the technique is still relatively under-developed and many questions remain unanswered.

Graph spectral methods are concerned with using the eigenvalues and the corresponding eigenvectors of the data proximity matrix for partitioning, embedding and clustering. Most of the existing methods focus on using only one or a few of the eigenvectors. For example, random-walk based graph matching methods (Robles-Kelly and Hancock, 2005a; Caelli and Kosinov, 2004) use the largest eigenvector of the adjacency matrix to convert a graph into a string. In data clustering and image segmentation, a well known technique (Shi and Malik, 2000) is to use an eigenvector as a cluster indicator. The components of the leading eigenvector can be used to recursively bipartition the nodes of the graph into clusters. Furthermore, most of the existing data embedding methods such as the principle component analysis(PCA) (Hotelling, 1933) and multi-dimension scaling(MDS) (Kruskal and Wish, 1978) use just a few eigenvectors of the affinity matrix to embed data into a low dimensional subspace.

Although elegant to use, these methods suffer from unstable eigenvectors and are vulnerable to noise. Even small disturbances in graph structure will result in a considerable variation in the graph spectrum. This creates a significant obstacle for matching graphs with different structure or different size. Furthermore, by using only one or a few eigenvectors, much useful information contained in the remainder of the spectrum is also discarded.

One way to overcome these problems is to use the whole Laplacian eigenspectrum. As pointed out in (Alpert and Yao, 1995), utilising more eigenvectors always gives better clustering results. Also in (Wilson et al., 2005), a graph representation based on the full adjacency eigenspectrum gives far richer structural information. Motivated by the need to improve existing spectral methods, our aim in this thesis is to develop more sophisticated methods using the full Laplacian spectrum.

1.2 Goals

The overall goal of this thesis is to develop more sophisticated graph spectral methods, and to apply them to a variety of applications from compute vision. The applications considered are as follows:

- Robust graph representation: Simpler graph representations need to be constructed based on spectral analysis of the graph for the purposes of efficient graph matching and clustering.

- Inexact graph matching: With stable graph representations in hand, we shall develop reliable graph matching methods which are robust to structural corruption and noise.

- Embedding and clustering: A spectral embedding algorithm for the purpose of clustering is developed that improves data coherence.

- Multi-body motion tracking: We aim to overcome the effects of noise and outliers, which render the classical factorisation method impractical, by casting the motion tracking problem into a spectral clustering framework.

1.3 Thesis Overview

At the beginning of this chapter, we have discussed the difficulties spectral methods have confronted when they are applied to computer vision problems. Besides spectral methods, there have been many alternative methods for solving these problems in the computer vision literature and we will briefly review them in Chapter 2.

Based on a spectral partition of a graph, Chapter 3 presents a graph matching method which simplifies the inexact graph matching problem by matching the non-overlapping partitions. The new graph representations constructed from these subgraphs are stable under structural corruptions and have been used for category based clustering.

In Chapter 4, we focus on the theory of our clustering method and solve data clustering problems by enhancing data coherence. We will show how the method is based in spectral graph theory and why it is superior to the normalised cut. Closely related to this, an embedding method for data-clustering is also presented and its advantages over other embedding methods are explained.

Chapter 5 is concerned with the real world application of the theory developed in Chapter 4. In this chapter, we make four contributions. The first of these is to develop a graph matching method based on a simulation of a diffusion process on a graph. The graph is decomposed into several concentric layers using the commute time and graph matching is performed in a layer to layer scheme. Secondly, taking account of the robustness of the commute time to structural error and noise, we propose a graph simplification method based on minimum spanning trees. Thirdly, as an application in spectral clustering, an image segmentation method is developed by performing bipartition using the smallest eigenvector of the commute time matrix. Finally, to solve the noise contaminated multi-body motion tracking problem, we embed the shape interaction matrix into a commute time preserving subspace and group objects using a simple k-means method.

Chapter 6 gives conclusions and focuses on the successes and shortcomings of the thesis. We also discuss directions for future research.

Chapter 2

Literature Review

The overall aim of the thesis is to apply spectral methods to solve computer vision problems. In this chapter, we will review the relevant literature. First, spectral graph theory is introduced. We then focus in detail on the computer vision problems studied. These include graph matching, graph simplification and seriation, graph embedding and clustering, image segmentation and motion tracking. Finally, the graph spectral algorithms for solving these problems are reviewed.

2.1 Graph Spectrum

Graph spectral methods aim to utilise the eigenvalues and eigenvectors of the Laplacian matrix to characterise graph structure.

The earliest literature on algebraic graph theory can be traced back to that of Collatz and Sinogowitz (Collatz and Sinogowitz, 1957). This work focused on the cospectrality of graphs as well as the fundamental inequalities for bounding the eigenvalues. Since then, a large body of literature has emerged aimed at exploiting the relationship between the spectral and structural properties of a graph. This literature is well documented in several surveys including Biggs

(Biggs, 1974), Cvetković, Doob and Sachs (Doob et al., 1995), Chung (Chung, 1997) and Mohar (Mohar, 1997).

One of the most important matrices in spectral graph theory is the adjacency matrix. By representing graphs in terms of their adjacency matrices, we open up the possibility of using tools from linear algebra to study the properties of graphs. For example, the trace of the adjacency matrix is equal to twice the number of loops on the graph. The number of walks with length l joining two vertices is the lth power of the adjacency matrix. The number of edges and triangles in a graph corresponds to different coefficients in the characteristic polynomial of the adjacency matrix.

The graph spectrum refers to the set of eigenvalues of the adjacency or Laplacian matrix of a graph (Biggs, 1974). The spectrum is useful because it can be computed quickly and it conveys many important properties of a graph. For example, the component number, chromatic number, diameter, circuit number and the clique number (Dulmage and Mendensol, 1967), (Collatz and Maas, 1987), (Marcus and Minc, 1964), (Cvetkovic' and Rowlinson, 1990), (Doob et al., 1988) are all given by the spectrum. Furthermore, the isomorphism of two graphs can also be determined by their spectra. If the eigenvalues of the adjacency matrices of the two graphs are not equal, then the graphs will not be isomorphic (although the converse does not apply due to co-spectrality). The two most important components in the graph spectrum are the largest and the second largest eigenvalues of the adjacency matrix together with the corresponding eigenvectors. The largest eigenvalue is also called the *index* of the graph and it is useful for estimating the connectivity of the graph. The second largest eigenvalue on the other hand is closely related to the properties of rapidly mixing Markov chains. For example, it can be used for estimating the convergence rate of a Markov chain on a graph (Desai and Rao, 1993), (Sinclair, 1991), (Diaconis and Stroock, 1991).

Although the adjacency matrix and its spectrum have been studied for understanding the structure of graphs, their properties are mostly understood for specific graphs (such as regular graphs, symmetric graphs, random graphs and line graphs). In order to bring spectral methods to a more general family of graphs, many researchers seek answers from the link between spectral graph theory and differential geometry (Fiedler, 1993),(Chung, 1997). This resulted in the study of the Laplacian matrix (degree matrix minus adjacency matrix) as well as its eigenspectrum (Chung, 1997), (Merris and Grone, 1994), (Grone, 1991), (Merris, 1994), (Merris, 1995) , (Mohar, 1991), (Mohar, 1992).

The Laplacian matrix is a discrete analogy of the Laplacian operator on a Riemannian manifold (Chavel, 1984). It is important in the study of energy minimisation (Chung, 1997) and network communication (Kirchhoff, 1847). In a useful review, Mohar (Mohar, 1997) has summarised some important applications of the Laplacian eigenvalues. These include the max-cut problem, semidefinite programming and steady state random walks on Markov chains. Among the eigenvalues of the Laplacian matrix, the second smallest eigenvalue plays a special role. This is due to its connection with graph invariants, including connectivity (Merris and Grone, 1987), (Fiedler, 1989), (Merris and Grone, 1990), the isoperimetric number (minimal possible ratio between the size of edges connecting a pair of subgraphs and the smallest volume of them), the maximum cut (a bipartition of graph so that the sum of the weights of the edges going from one subset to the other is maximised) and the independent number (the size of a maximum independent set), etc. In this thesis, we are interested in the connectivity information supplied by this eigenvalue together with its corresponding eigenvector, namely the Fiedler vector. The Fiedler vector (Fiedler, 1973),(Fiedler, 1975) has been extensively used for the purpose of image segmentation (Shi and Malik, 2000), data clustering (Weiss, 1999) and graph labelling (Diaz et al., 2000), (Juvan and Mohar, 1992).

Another important concept that we are concerned with in this thesis is the random walk on the graph. Particularly, we are interested in random walks on undirected graphs, which can be viewed as time-reversible Markov chains (Aldous and Fill, 2003). Many properties of random walks (or Markov chains), such as the hitting time, the commute time and the cover time are determined by the graph spectrum. For a pair of nodes u and v, the hitting time is defined as the expected number of steps before node v is visited, commencing from node u. In other words, the commute time is the expected time for the random walk to travel from node u to node v and then return. The cover time is the expected number of steps to reach each node on the graph. For a good review, see Lovász's survey (Lovász, 1996) and Chung's book (Chung, 1997). Recently, random walks (Sood et al., 2005) 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 (Brin and Page, 1998). In computer vision, random walks have been used for image segmentation (Meil$\breve{a}$ and Shi, 2000) and clustering (Saerens et al., 2004). More recently both Gori, Maggini and Sarti (Gori et al., 2004) and Robles-Kelly and Hancock (Robles-Kelly and Hancock, 2005b; Robles-Kelly and Hancock, 2005a) have used random walks to sort the nodes of graphs into a string order for the purpose 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 (Lovász, 1996).

A lazy random walk is a random walk with a probability of, or remaining, static. The behaviour of a lazy random walk on a graph is linked to the information flowing (with time) across the edges connecting the nodes. This process can be characterised using the heat equation (Kondor and Lafferty, 2002). The solution of the heat equation, or heat kernel, can be found by exponentiating the Laplacian eigensystem over time (Chung, 1997). The heat kernel, or diffusion kernel, contains a considerable amount of information concerning the distribution of paths on a graph (Chung and Yau, 1997). As mentioned by Chung (Chung, 1997), the definition for the heat kernel on graphs is analogous to the heat kernel on Riemannian manifolds (Yau and Schoen, 1988), This field of study is sometimes referred as spectral geometry (Auscher et al., 2003), (Grigor'yan, 2000), (Lafferty and Lebanon, 2005). It has been applied to solve graph embedding and clustering problems (Chung and Yau, 1999), (Bai and Hancock, 2004). One of the most valuable properties of the heat kernel for charactering graphs is the presence of the time variable t. An alternative, but closely related, characterisation of the graph is the discrete Green's function, which captures the distribution of sources in the heat flow process. The Green's function is the pseudo-inverse of the Laplacian (Chung and Yau, 2000). As a result, it shares the same spectrum with the Laplacian matrix, except that the eigenvalues are reciprocated. It turns out that the Green's function can be used in conjunction with diffusion-like problems on graphs (such as electric potential distribution and random walks). Some examples of the Green's function on regular graphs can be found in Ellis's review (Ellis, 2002). Not surprisingly, there is a direct link between commute times and the Green's function (Chung and Yau, 2000).

2.2 Computer Vision and Pattern Recognition Problems and The Spectral Solutions

In this section, we review several problems in the computer vision and pattern recognition field, and show how spectral methods can be applied to solve these problems.

2.2.1 Graph Matching

The pioneering work done by Barrow and Burstall (Barrow and Popplestone, 1971) and by Fischler and Enschlager (Fischler and Elschlager, 1973) in the 1970's shows how to realize the recognition of abstract pictorial descriptions by matching graph structures. Since then, graph matching has been a sustained research activity. In this section, we review the literature on graph matching.

Since one of the perennial difficulties associated with the effective matching of relational descriptions is the need to accommodate inexactness caused by inevitable noise and clutter, early work concentrated on measuring relational similarities. To overcome this problem graph matching can be posed as maximising a measure of relational similarity or minimising as a distance function. For instance, Shapiro and Haralick (Shapiro and Haralick, 1985) showed how to realize inexact graph matching by counting the consistent subgraphs. Later on, Fu and his co-workers (Eschera and Fu, 1986), (Sanfeliu and Fu, 1983) showed how string edit distance could be extended to relational structures. Here edit distances are computed using separate costs for relabelling, insertion and removal of nodes. This idea was further extended by Bunke and his co-workers (Bunke, 1999), (Messmer and Bunke, 1998). They showed that the edit distance is related to the size of the maximum common subgraph.

Most of the work above adopted a heuristic or goal directed approach to measure graph similarity. A more principled approach is to adopt a probabilistic framework. For instance, Wong and You (Wong and You, 1985) have defined an entropy measure for structural graph matching; Christmas, Kittler and Petrou (Christmas et al., 1995) developed an evidence combining method. They use probability distribution functions to model the pairwise attribute relations and cast the graph matching problem into a Bayesian framework. Wilson and Hancock (Wilson and Hancock, 1997) have shown how to construct a mixture model over a dictionary of structure-preserving mappings between two graphs. An alternative to the exhaustive compilation of dictionaries is the edit distance method of Myers, Wilson and Hancock (Myers et al., 2000) which showed that the Levenshtein distance can be used to model the probability distribution for structural errors. Luo and Hancock (Luo and Hancock, 2001) have posed the structural matching problem as maximum likelihood estimation and solved this problem using the apparatus of the EM algorithm.

Continuous and discrete optimisation methods can also be used for structural graph matching. The methods used include genetic search (Cross et al., 1997), (Myers and Hancock, 1997), simulated annealing (Williams et al., 1999), tabu search (Williams et al., 1999) and hybrid method (Magyar et al., 2000). Cross, Wilson and Hancock (Cross et al., 1997) have cast the genetic search into a Bayesian framework using the global consistency measure. Rather than performing random crossover they realized the process at the level of subgraphs. Furthermore, they employed a hill-climbing process to locate the nearest local optimum. Another difficulty of applying genetic search is the setting of parameters. This problem was intensively discussed by Myers and Hancock when they applied the method to the graph labelling problem (Myers and Hancock, 1997).

Recently, there has been increased interest in the use of spectral graph theory for characterising the global structural properties of graphs. There are several examples of the application of spectral matching methods for grouping and matching in the computer vision literature. For instance, Umeyama has shown how graphs of the same size can be matched by performing singular value decomposition on adjacency matrices (Umeyama, 1988). The permutation matrix that brings the nodes of the graphs into correspondence is found by taking the outer product of the matrices of left eigenvectors for the two graphs. In related work, Shapiro and Brady (Shapiro and Brady, 1992) have shown how to locate feature correspondence using the eigenvectors of a point-proximity weight matrix. However, these two methods fail when the graphs being matched contain different numbers of nodes. A number of works have shown that this problem can be overcame by using the apparatus of the EM algorithm (Luo and Hancock, 2001; Wilson and Hancock, 1997). Shokoufandeh, Dickinson, Siddiqi and Zucker (Shokoufandeh et al., 1999) have shown that graphs can be efficiently retrieved using an indexing mechanism that maps the topological structure of shock-trees to a low-dimensional vector space. Here the topological structure is encoded by exploiting the interleaving property of the eigenvalues. Based on the spectral analysis of point-sets, Carcassoni and Hancock (Carcassoni and Hancock, 2000), (Carcassoni and Hancock, 2003) have shown that the modal structure of point-sets can be embedded within an EM framework and the probabilities of point correspondence can be computed using a proximity matrix. To overcome difficulties in node correspondence for different sized graphs, Wilson, Luo and Hancock (Wilson et al., 2005), (Luo et al., 2004) have proposed construction of permutation invariant polynomials and have characterised graphs using the coefficients of these polynomials. They have shown how to embed vectors of permutation invariants into a low-dimensional space. Bai, Yu and Hancock (Bai et al., 2004a), (Bai et al., 2004b) have gone one step further and realized graph matching by recovering the correspondence of nodes embedded in the low-dimensional space. They commence by using Isomap to embed the nodes of a graph into a metric space and align the points in this space using a variant of the Scott and Longuet-Higgins algorithm (Scott and Longuet-Higgins, 1990).

2.2.2 Graph Seriation and Simplification

An alternative of using graph-spectra for the purpose of graph matching is to use eigenvector methods to extract a simplified structure from a graph. This simplified structure is more easily matched than the original graph. Although inexact graph-matching is a problem of potentially exponential complexity, error-tolerant graph matching can be simplified using decomposition methods (as demonstrated by Messmer and Bunke (Messmer and Bunke, 1998)). This reduces the problem to one of subgraph indexing.

The earliest work on graph seriation and simplification can be traced back to the graph layout problem (Harper, 1964), (Harper, 1966). Graph layout problems are concerned with re-arranging the input graph so that an objective function is optimised. A large number of applications can be posed as graph layout problems. These include optimisation of networks for parallel computer architectures, VLSI circuit design, information retrieval, numerical analysis, computational biology and scheduling. One of the simplest (and the area of our interest) is the minimum linear arrangement problem. This is also referred as optimal linear ordering, minimum-1-sum or graph seriation. The problem involves placing the nodes of a graph in a serial order which is suitable for the purposes of visualisation (Diaz et al., 2000), job scheduling (Adolphson, 1977) and graph drawing (Shahrokhi et al., 2001). The MinLA problem is NP-complete (Garey et al., 1976) but optimal solutions can be obtained for trees (Chung, 1988) and some special graphs (Muradyan and Piliposyan, 1980). In order to obtain feasible solutions for the MinLA problem for general graphs, several approximation methods have been proposed. These include metric techniques (Rao and Richa, 1998), simulated annealing (Petit, 2000) and spectral methods (Juvan and Mohar, 1992), (Atkins et al., 1998). Juvan and Mohar's (Juvan and Mohar, 1992) spectral sequencing (also known as the path method) first computes the Fiedler vector of the Laplacian matrix of the input graph and then orders the result by ranking the components. The lower bound is determined by the second smallest eigenvalue, which is the eigenvalue corresponding to the Fiedler vector.

An extension of the MinLA problem is the consecutive ones problem. This involves finding the serial ordering of nodes, which maximally preserves edge connectivity. This is a complex problem, and to simplify it, approximate solution methods have been employed. These involve casting the problem in an optimisation setting. Hence techniques such as simulated annealing and mean field annealing have been applied to the problem. However, recently, a graph-spectral solution to the problem has been found. Atkins, Boman and Hendrikson (Atkins et al., 1998) have shown how to use the Fiedler eigenvector of the Laplacian matrix to sequence relational data. The method has been successfully applied to the consecutive ones problem and a number of DNA sequencing tasks. There is an obvious parallel between this method and steady state random walks on graphs, which can be located using the leading eigenvector of the Markov chain transition probability matrix. However, in the case of a random walk the path is not guaranteed to encourage edge connectivity. The spectral seriation method of Robles-Kelly and Hancock (Robles-Kelly and Hancock, 2005a), on the other hand, does impose edge connectivity constraints on the recovered path. They have shown how to use eigenvector methods to reduce graphs to strings, and have then applied string matching methods to the resulting structures. In related work, Yu and Hancock (Yu and Hancock, 2005a; Yu and Hancock, 2005b) have shown how to cast the graph seriation problem into a matrix setting so that it can be solved using semi-definite programming(SDP). SDP is a technique related to spectral graph theory since it also relies on matrix representation.

2.2.3 Embedding and Clustering

The low dimensional representation of high dimensional data and clustering is an important topic in pattern recognition. The fundamental problem of dimensionality reduction is how to embed the data in a compact space for the purposes of analysis and visualisation. Although a variety of methods exist, they share the same principle of using one or more eigenvectors of an affinity matrix or similarity matrix for the embedding. For example, principle component analysis (PCA) (Hotelling, 1933) and kernel principle component analysis (KPCA) (Scholkopf et al., 1998; Aizerman et al., 1964) use the leading eigenvectors of the covariance matrix to determine the projection directions with maximal variance. Linear discriminant analysis (LDA) (Fisher, 1936) and KDA (kernel version of LDA) search for the directions that are maximally discriminating. Although different from PCA and KPCA, the solution of LDA and KDA is obtained using the eigenvectors of the projection matrix. This is the ratio of the trace of the between-class scatter matrix and within-class scatter matrix. Multi-dimensional scaling (MDS) (Kruskal and Wish, 1978) uses the eigenvectors of a pairwise distance matrix to find an embedding that minimises the distance of the data. As an extension, isometric feature mapping (Isomap) (Tenenbaum et al., 2000) employs MDS to preserve the geodesic distances of the data pairs located in the manifold. Locally linear embedding (LLE) (Roweis and Saul, 2000) maps the input data to a lower dimensional space in a manner that preserves the local neighbourhood. It uses a matrix containing the correlations of the data in barycentric coordinates. The coordinates in the lower dimensional space are the corresponding components of the smallest eigenvectors of this matrix. The Laplacian eigenmap (Belkin and Niyogi, 2003; Belkin and Niyogi, 2001), and its linear version, locality preserving projection (LPP), (He and Niyogi, 2003), use the Fiedler vector of a Laplacian matrix to preserve the similarities of the neighbouring points. Finally, the recently developed diffusion map (Lafon and Lee, 2005; Coifman et al., 2005) also uses the eigenvalue-scaled eigenvectors of the transition matrix as coordinates of the embedded points as a simulation of a heat diffusion process. The embedded structure may be varied by varying a time parameter t.

The eigenspectrum of an affinity matrix plays an important role in dimensionality reduction methods. This is because by ranking the eigenvectors with respect to the magnitude of the corresponding eigenvalues, the significance of the correlations within the data are accordingly evaluated. The orthogonality of the eigenvectors also offers a natural bases for the embedded data. It is important to note that the dimensionality reduction methods discussed above are concerned with recovering the lower dimensional structure of the data, rather than a way of pre-grouping. Here we refer pre-grouping as a process performed before clustering. It is aimed at reorganising data in a way that makes clustering easier. Although some of the methods can be used as a pre-grouping process, such as the Laplacian eigenmap and KPCA, their utility is restricted. For example, the Laplacian eigenmap embeds similar data so that it is close in the embedded space. On the other hand, KPCA maintains the maximum variance of the embedded data in the vector space in such a way that it can be separated using a kernel function.

Clustering is the un-supervised classification of patterns based on their similarities (Jain et al., 1999). As clustering plays such a central role in pattern analysis, a large number of alternative approaches to the problem have been developed over the last four decades. The two main approaches are statistical method and graph-theoretic methods. Details of both will be presented in the next two paragraphs.

Parametric models assume that patterns are drawn from a mixture of several distributions, such as the Gaussian, and the goal is to estimate the parameters of the distribution. This approach encompasses the maximum likelihood estimation (MLE) (Dempster et al., 1977), expectation maximisation algorithm (EM) (Zhang et al., 2003) and K-means (MacQueen, 1967). MLE estimates the parameters of a mixture by maximising the logarithmic function of the underlying probability distribution of a given data set. EM is an iterative optimisation method to estimate some unknown parameters defined in the model. K-means aims to cluster data into k partitions by minimising the total intra-cluster variance. Nonparametric techniques such as histogram based estimation (Silverman, 1986), kernel density estimation (Elgammal et al., 2003) and mean shift (Comaniciu, 2003) are also employed for density based clustering. The basic idea is to view the clusters as regions of the pattern space in which the patterns are dense, separated by regions of low pattern density. Then the clusters can be identified by searching for regions of high density. Histogram based estimation divides the pattern space into a number of non-overlapping regions based on the constructed histograms. Mean shift (Comaniciu and Meer, 2002) is a recursive kernel density estimation method that shifts each data point to the average of data points in its neighbourhood.

Graph-theoretic methods define clusters in terms of a weighted data proximity matrix. The earliest method is based on searching for structures in the similarity graph such as the minimal spanning tree (MST) (Zahn, 1971). Using the idea of the maximal cliques of a graph, Pavan and Pelillo (Pavan and Pelillo, 2003a; Pavan and Pelillo, 2003b) improve the similarity measure by introducing the concept of a dominant set. The resulting utility measure is optimised using a relaxation scheme. The earliest spectral clustering method is that of Donath and Hoffman (Donath and Hoffman, 1972). They suggested to use the eigenvectors of an adjacency matrix to find partitions. Later, Fiedler (Fiedler, 1973) proposed splitting the partitions by using the second smallest eigenvector of the Laplacian matrix. Since then, the spectral clustering method has proved to be successful and has been the focus of much research. Scott and Longuet-Higgins (Scott and Longuet-Higgins, 1990) have 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 (Sarkar and Boyer, 1996) have a method which uses the leading eigenvector of the affinity matrix, and this locates clusters that maximise the average association. This method is applied to locate line-segment groupings. Perona and Freeman (Perona and Freeman, 1998) have a similar method which uses the second largest eigenvector of the affinity matrix. The method of Shi and Malik (Shi and Malik, 2000), on the other hand, uses the normalised 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 (Weiss, 1999) has shown how this, and other closely related methods, can be improved using a normalised affinity matrix. Shi and Meilă (Meilă and Shi, 2000) have analysed the convergence properties of the method using Markov chains. Ng et al's (Ng et al., 2001) method first embeds the graph into a space and then clusters the embedded points using a K-means algorithm. There are good reviews of spectral clustering methods in the literature. Spielman and Teng (Spielman and Teng, 1996) investigated why spectral partitioning works on planar graphs and meshes. Kannan et al (Kannan et al., 2000) have proposed a new bi-criteria measure for assessing the quality of a spectral clustering. They argued that a good clustering method should be able to maximise intra-cluster association and minimise inter-cluster edge linkage simultaneously. Alpert and Yao (Alpert and Yao, 1995) analyse the number of eigenvectors that should be used in spectral clustering and suggest that it is best to use as many eigenvectors as possible. Finally, a further unifying view regarding spectral embedding and clustering is given by Brand and Huang (Brand and Huang, 2003). In their work, they have used the angles between the eigenvectors to explain the functionality of the spectral clustering methods.

2.2.4 Motion Tracking

Multi-body motion tracking is a challenging problem which arises in shape from motion, video coding, surveillance and the analysis of movement. One of the classic techniques is the *factorisation method* of Costeira and Kanade (Costeira and Kanade, 1997; Costeira and Kanade, 1995). The basic idea underpinning this method is to use singular value decomposition (SVD) to factorise the feature trajectory matrix into a motion matrix and a shape matrix. The shape interaction matrix is found by taking the outer product of the right eigenvector matrix, and can be used to identify any independently moving objects present. Gear (Gear, 1998) has developed a related method based on the reduced row echelon form of the matrix where object separation is achieved using 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 usually are contaminated by noise. There have been several attempts to overcome this problem. For instance, Ichimura (Ichimura, 1999) has improved the *factorisation method* by using a discriminant criterion to threshold-out noise and outliers.

Rather than working with a matrix derived from the data, some researchers place the emphasis on the original data. Kanatani (Kanatani, 2001; Sugaya and Kanatani, 2004; Sugaya and Kanatani, 2003) developed a subspace separation method by incorporating dimension correction and model selection. Wu et al (Wu et al., 2001) argue that the subspaces associated with the different objects are not only distinct, but also orthogonal. Hence they employ an orthogonal subspace decomposition method to separate objects. This idea is further extended by Fang et al who use independent subspace (Fan et al., 2004b) and multiple subspace inference analysis (Fan et al., 2004a). In addition to attempting to improve the behaviour of the factorisation method under noise, there has been a considerable effort to overcome problems such as degeneracy, uncertainty and missing data (Gruber and Weiss, 2004; Zelnik-Manor and Irani, 2003; Anandan and Irani, 2002).

The factorisation method is 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 (Weiss, 1999) has presented a unifying view of spectral clustering methods, and this includes the factorisation method. There has been some dedicated effort devoted to solving the object separation problem using spectral clustering methods. Park et al (Park et al., 2004) 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.

2.3 Motivation and Contributions

In the previous section, we have reviewed not only the related literature on spectral graph theory, but also methods developed based on these theories for solving various computer vision problems. In particular, we have observed how the Fielder vector has been employed for solving graph bipartition and seriation problems. Although there has been some effort aimed at extending its utility to graph matching, the results have not been optimal due to the methods instability and the loss of information it causes. Focusing on these two obstacles, in this thesis, we are interested in developing new methods by using the properties of the Fiedler vector for robust graph matching.

More specifically, our aim is to develop an inexact graph matching method based on the robust decomposition of a graph into partitions. Here we use the Fiedler vector to find a non-overlapping partitions of a graph. These partitions are subgraphs comprised of a centre node together with its immediate neighbours. In order to improve the robustness of the partitions to structural corruption and noise, we incorporate a diffusion process on the graph to smooth away the effects of errors. The implementation is carried out by using the heat-kernel to construct a path-weighted matrix, which is more robust to noise than the original adjacency matrix and can be used for regulating graphs. With the partitions in hand, we will be able to realize graph matching by comparing their sub-structures. The matching process is cast into a hierarchical framework by first locating the correspondence between partitions and then individual correspondences between nodes are obtained by comparing the partitions in detail.

To test if the partitions we obtained can be used for graph simplification, we use them for category-based clustering. First, we decompose the original graph into non-overlapping partitions. Then we construct a simpler graph representation whose nodes are the centre nodes of the partitions and the edges are constructed according to the adjacency relations between the partitions. Finally, we take random images from different groups and examine whether clusterings provided by the simplified graph representations can deliver the correct grouping result.

Our observation is that 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. The commute time can lead to a finer measure of cluster cohesion than the simple use of edge-weight which underpins algorithms such as the normalised cut (Shi and Malik, 2000). Furthermore, it has been shown (Weiss, 1999) that the reason certain 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 (Fischer and Poland, 2005). Hence, commute time maybe used for solving clustering problems.

In Chapter 4 of this thesis we will first review the spectral basic of the commute time and then in Chapter 5, we will present its applications. We will show how commute time is related to the heat kernel and how it can be computed using the full Laplacian eigenspectrum. A link between the commute time and the Green's function shows that the commute time is a metric. To extend this distance measure one step further, a commute time preserved embedding method has been proposed and its relation to alternative embedding methods is also examined. As we have already observed, commute time can be applied to solve clustering problems. To further understand its properties, a comparison with the normalised cut (Shi and Malik, 2000) will be carried out.

Chapter 3

Graph Matching and Simplification using Spectral Partitions

The aims in this chapter are twofold. First, we consider whether the partitions delivered by the Fiedler vector can be used to simplify the graph-matching problem. Secondly, we investigate whether the information conveyed by the heat kernel can be used for stabilising the spectral partitioning of graphs. We seek a more global graph representation for the purpose of graph matching, graph simplification and graph clustering than can be achieved using the adjacency matrix alone.

For our first goal, we focus on two problems. The first of these is to use the Fiedler vector to decompose graphs by partitioning them into super-cliques. Our aim is to explore whether the partitions are stable under structural error, and in particular whether they can be used for the purposes of graph-matching.white The second problem studied is whether the partitions can be used to simplify the graphs in a hierarchical manner. Here we construct a graph in which the nodes are the partitions and the edges indicate whether the partitions are connected by edges in the original graph. This spectral construction can be applied recursively to provide a hierarchy of simplified graphs. We show that the simplified graphs can be used for efficient and reliable clustering.

To achieve the second goal, we use the heat-kernel to construct a path-weighted adjacency matrix to represent the graph structure. The weighting process aims to smooth away the effects of structural error due to node or edge deletions. We explore whether this representation can be used to characterise the graph globally and whether it is stable under structural error. In particular, we explore its use in conjunction with our graph partition method proposed earlier for the problem of graph matching.

3.1 Laplacian Matrix

We denote a weighted graph by $\Gamma = (V, E)$ where V is the set of nodes and $E \subseteq V \times V$ is the set of edges. Let Ω be 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 $T(\Gamma) = diag(d_v; v \in V)$ be the diagonal weighted degree matrix with $T_u = \sum_{v=1}^{|V|} w(u,v)$. The *un-normalised* weighted Laplacian matrix is given by $L = T - \Omega$, and has elements

$$L_\Gamma(u,v) = \begin{cases} \sum_{(u,k)\in E} w(u,k) & \text{if } u = v \\ -w(u,v) & \text{if } u \neq v \text{ and } (u,v) \in E \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

The *normalized* weighted Laplacian matrix is defined to be $\mathcal{L} = T^{-1/2}LT^{-1/2}$, and has elements

$$\mathcal{L}_\Gamma(u,v) = \begin{cases} 1 & \text{if } u = v \\ -\frac{w(u,v)}{\sqrt{d_u d_v}} & \text{if } u \neq v \text{ and } (u,v) \in E \\ 0 & \text{otherwise} \end{cases} \tag{3.2}$$

The *normalized* Laplacian $\mathcal{L}$ can also be viewed as a harmonic operator that acts on the function $f : V(\Gamma) \mapsto \Re$ with the result that $\mathcal{L}f(x) = \sum_{x'} \mathcal{L}_{x,x'} f(x')$. The spectral decomposition of the *un-normalised* Laplacian matrix is

$$L = \Phi\Lambda\Phi^T = \sum_{i=1}^{|V|} \lambda_i \phi_i \phi_i^T$$

where $\Lambda = diag(\lambda_1, \lambda_2, ..., \lambda_{|V|})$ is the diagonal matrix with the ordered eigenvalues as elements and $\Phi = (\phi_1|\phi_2|....|\phi_{|V|})$ is the matrix with the ordered eigenvectors as columns. The corresponding eigen-decomposition of the *normalized* Laplacian matrix is $\mathcal{L} = \Phi'\Lambda'\Phi'^T$ with Λ' and Φ' be the eigenvalue matrix and eigenvector matrix respectively.

The Laplacian matrix has a number of important properties. It is symmetric and positive semidefinite. The eigenvector $\vec{e} = (1, 1, \ldots, 1)^T$ corresponds to the trivial zero eigenvalue. If the graph is connected then all other eigenvalues are positive and the smallest eigenvalue is a simple (i.e. un-repeated) one, which means that the number of connected components of the graph is equal to the multiplicity of the smallest eigenvalue. If we arrange all the eigenvalues from the smallest to the largest i.e. $0 = \lambda_1 \leq \lambda_2 \ldots \leq \lambda_n$, the most important are the largest eigenvalue λ_{max} and the second smallest eigenvalue λ_2, whose corresponding eigenvector is referred to as the *Fiedler Vector* (Fiedler, 1975).

3.2 Graph Partition

The aim in this section is to use the Fiedler vector to partition graphs into non-overlapping super-cliques and to use the super-cliques generated by this decomposition for the purposes of graph-matching and graph-simplification.

Fiedler vector has the property of grouping similar nodes of a graph and has been used in applications such as graph seriation (Robles-Kelly and Hancock, 2005a) and image segmentation (Shi and Malik, 2000). Given a graph $\Gamma(V, E)$, the components of its Fiedler vector indicate a permutation π of the nodes. If the weighted adjacency matrix has $\Omega(u, v) > \Omega(u, k)$ and $\Omega(v, k) > \Omega(u, k)$, the permutation satisfies $\pi(u) < \pi(v) < \pi(k)$.

The super-clique of the node u consists of its center node, together with its immediate neighbours connected by edges in the graph, i.e., $\hat{N}_u = \{u\} \cup \{v; (u, v) \in E\}$. Here, centre nodes of the super-cliques are not shared but the exterior nodes can appear in more than one unit as illustrated in Figure 3.1. Figure 3.1 shows a graph with its three super-cliques highlighted. Centre nodes are marked with red color and shared exterior nodes are marked with blue. Hence, each super-clique consists of a *center node* and *immediate neighbours* of the center node, i.e. $N_u = \hat{N}_u \setminus \{u\}$.

The problem addressed here is how to partition the graph into a set of non-overlapping super-cliques using the node order defined by the Fiedler vector. Our idea is to assign to each node a measure of significance as the centre of a super-clique. We then traverse the path defined by the Fielder vector selecting the centre-nodes on the basis of this measure.

We commence by assigning weights to the nodes on the basis of the rank-order of their component in the Fiedler vector. Let $\Upsilon =< v_1, v_2, v_3, ..., v_{|V|} >$ be the rank-order of the nodes as defined by the Fiedler vector so that the permutation satisfies the condition $\pi(v_1) < \pi(v_2) < \pi(v_3) < ..... < \pi(v_{|V|})$ and the

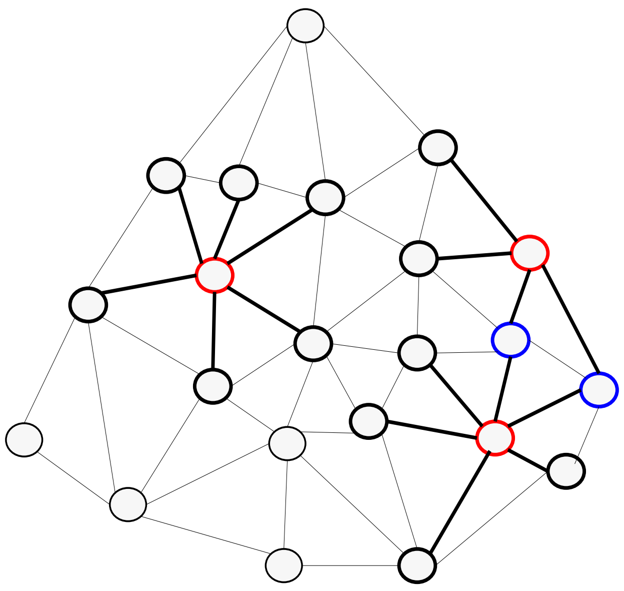


Figure 3.1: Super-cliques.

components of the Fiedler vector follow the condition $x_{v_1} > x_{v_2} > .. > x_{v_{|V|}}$. We assign weights to the nodes based on their rank order in the permutation. The weight assigned to the node $u \in V$ is $w_u = Rank(u)$. With this weighted graph in hand, we can gauge the significance of each node using the following score function:

$$\mathcal{F}_u = \alpha \left(deg(u) + |N_u \cap \mathcal{B}| \right) + \frac{\beta}{w_u} \tag{3.3}$$

where $\mathcal{B}$ is the set of nodes on the perimeter of the graph, and α and β are heuristically set thresholds (we set $\alpha = 0.015$ and $\beta = 5.0$ in our experiments). The first term depends on the degree of the node and its proximity to the perimeter. Hence, it will sort nodes according to their distance from the perimeter. This will allow us to partition nodes from the outer layer first and then work inwards. The second term ensures that the first ranked nodes in the Fielder vector are visited first.

We use the score function to locate the non-overlapping super-cliques of the graph Γ. We traverse this list until we find a node k_1 which is neither in the

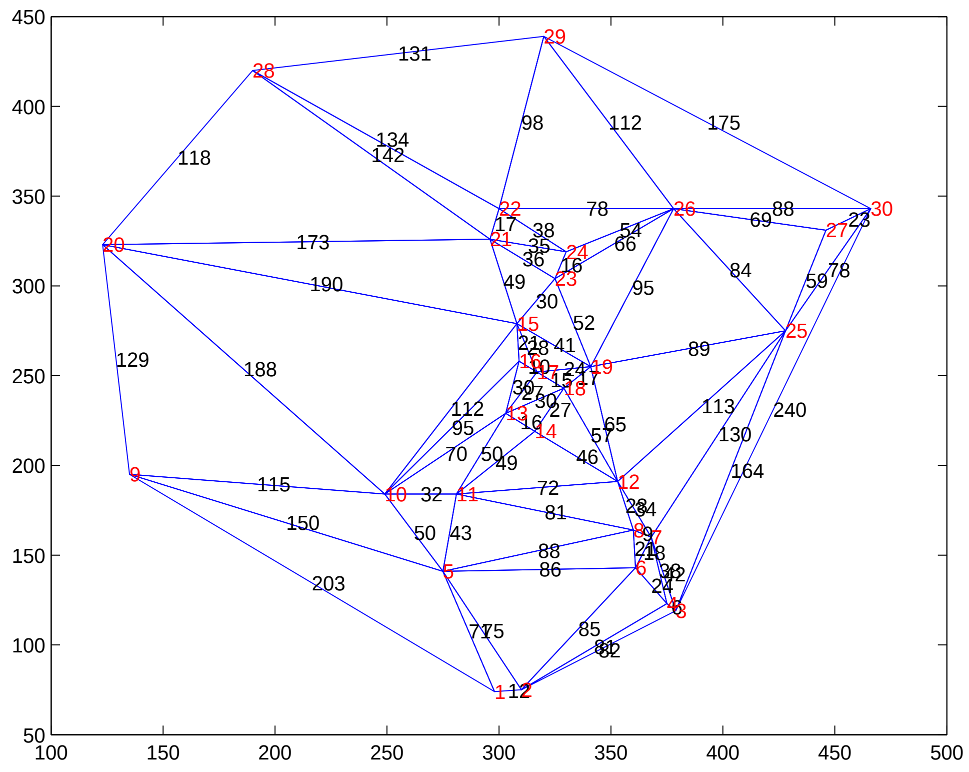


Figure 3.2: Delaunay graph of a set of points.

perimeter, i.e. $k_1 \notin \mathcal{B}$ nor whose score is exceeded by those of its neighbours, i.e. $\mathcal{F}_{k_1} = \arg\max_{u \in k_1 \cup N_{k_1}} \mathcal{F}_u$. When this condition is satisfied, then the node k_1 together with its neighbours N_{k_1} represent the first super-clique. The set of nodes $\hat{N}_{k_1} = k_1 \cup N_{k_1}$ are appended to a list $\hat{L}$ that tracks the set of nodes assigned to the super-cliques. This process is repeated for all the nodes which have not yet been assigned to a super-clique i.e. $R = \Upsilon - \hat{L}$. The procedure terminates when all the nodes of the graph have been assigned to non-overlapping super-clique. An example performed on Figure 3.2 is shown at Figure 3.3. The original graph in Figure 3.2 contains 30 nodes and 78 edges. In Figure 3.3 the edges are labelled to indicate the partition to which they belong.

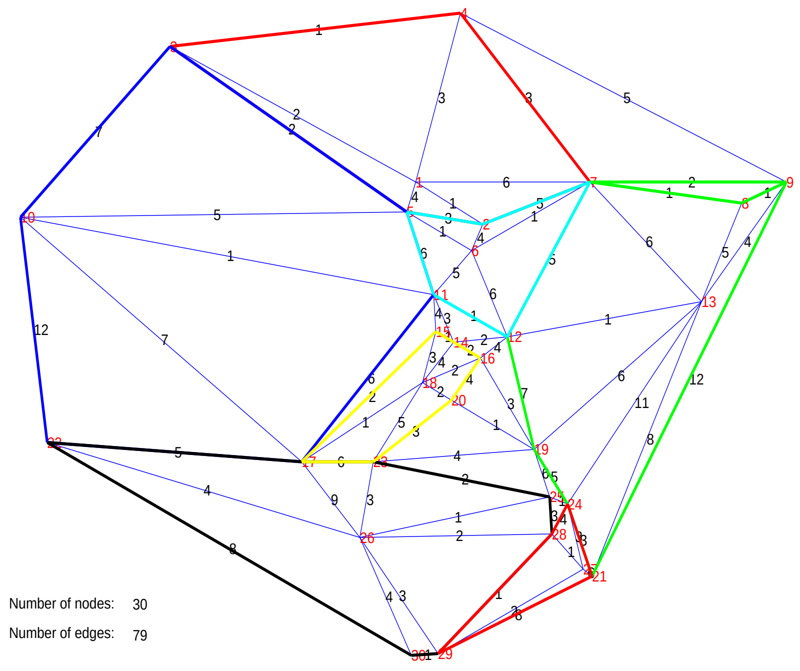


Figure 3.3: Graph Partition.

3.3 Partition Stabilisation

Unfortunately, the process of partitioning can prove unstable when the graph undergoes changes in node or edge structure. To overcome this problem, in the next section we demonstrate how the heat-kernel can be used to stabilise the partition structure.

3.3.1 Heat Kernel

Kernel-based methods have been widely used for pattern recognition and have lead to the development of a number of methods including support vector machines (Cristianini and Shawe-Taylor, 2000) and kernel PCA (Scholkopf et al., 1998). Heat kernel, which is found by solving the diffusion equation for the discrete structure in-hand is one of the most important Kernel-based methods. Heat kernel is also an important analytical tool for physics and has been used in many other areas including spectral graph theory (Chung, 1997). Recent work by Smola and Kondor (Smola and Kondor, 2003) has shown how kernels can be used to smooth or regularise graphs. A number of alternatives has been suggested and compared, and these include the heat kernel.

We are interested in the heat equation associated with the *normalised* Laplacian, i.e.

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

where $\mathcal{H}_t$ is the heat kernel and t is time. The heat kernel can hence be viewed as describing the flow of information across the edges of the graph with time. The rate of flow is determined by the *normalized* Laplacian. The solution is found by exponentiating the Laplacian eigenspectrum i.e.

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

where Λ' and Φ' are the eigenvalue and eigenvector matrices of $\mathcal{L}$ respectively. The heat kernel is a $|V| \times |V|$ matrix, and for the nodes u and v of the graph Γ the resulting component is

$$\mathcal{H}_t(u,v) = \sum_{i=1}^{|V|} \exp[-\lambda'_i t]\phi'_i(u)\phi'_i(v) \tag{3.4}$$

When t tends to zero, then $\mathcal{H}_t \approx I - \mathcal{L}t$, i.e. the kernel depends on the local connectivity structure or topology of the graph. If, on the other hand, t is large, then $\mathcal{H}_t \approx \exp[-t\lambda'_2]\phi'_2\phi'^T_2$, where λ'_2 is the smallest non-zero eigenvalue and ϕ'_2 is the Fiedler vector. Hence, the large time behaviour is governed by the global structure of the graph.

3.3.2 Path Length Distribution

Consider the *normalised* adjacency matrix $\mathcal{P} = T^{-\frac{1}{2}}\Omega T^{-\frac{1}{2}} = I - \mathcal{L}$, where I is the identity matrix. The heat kernel can be rewritten as $\mathcal{H}_t = e^{-t(I-\mathcal{P})}$. We can perform a McLaurin expansion on the heat-kernel to re-express it as a polynomial in t. The result of this expansion is

$$\begin{aligned}\mathcal{H}_t &= e^{-t(I-\mathcal{P})}\\ &= e^{-t}\left(I + t\mathcal{P} + \frac{(t\mathcal{P})^2}{2!} + \frac{(t\mathcal{P})^3}{3!} + \cdots\right)\\ &= e^{-t}\sum_{l=0}^{\infty}\mathcal{P}^l\frac{t^l}{l!}\end{aligned}$$

We can find a simplified expression for the matrix $\mathcal{P}^l$ using the eigen-decomposition of the *normalised* Laplacian. The result is

$$\mathcal{P}^l = (I-\mathcal{L})^l = \Phi'(I-\Lambda')^l\Phi'^T \tag{3.5}$$

and as a result the element

$$\mathcal{P}^l(u,v) = \sum_{i=0}^{|V|}(1-\lambda'_i)^l\phi'_i(u)\phi'_i(v) \tag{3.6}$$

If, on the other hand, we consider the element-wise definition of $\mathcal{P}$

$$\mathcal{P}(u,v) = \begin{cases} 1 & \text{if } u = v\\ \frac{w(u,v)}{\sqrt{d_u d_v}} & \text{if } u \neq v \text{ and } (u,v) \in E\\ 0 & \text{otherwise}\end{cases} \tag{3.7}$$

Exponent of matrix $\mathcal{P}$ supplies a connectivity measure between each pair of nodes. For example, $\mathcal{P}^2(u,v)$ measures the sum of weights of all paths with length two connecting node u and v. To show this, let us assume there are two nodes m and n connecting both u and v, i.e. $(u,m) \in E, (m,v) \in E$ and $(u,n) \in E, (n,v) \in E$. The computation of $\mathcal{P}^2(u,v)$ can be realized by $\mathcal{P}^2(u,v) = \mathcal{P}(u,m)\mathcal{P}(m,v) + \mathcal{P}(u,n)\mathcal{P}(n,v) = \sum_{s_2=m,n} \mathcal{P}(u,s_2)\mathcal{P}(s_2,v)$. If we represent the sequence of u, s_2, v by u_0, u_1, u_2 and use i for indexing, we have $\mathcal{P}^2(u,v) = \sum_{s_2=m,n} \prod_{i=0,1} \mathcal{P}(u_i, u_{i+1}) = \sum_{s_2=m,n} \prod_{i=0,1} \frac{w(u_i,u_{i+1})}{\sqrt{d_{u_i} d_{u_{i+1}}}}$. In a general case when the path length equals l ,we have that

$$\mathcal{P}^l(u,v) = \sum_{s_l} \prod_i \frac{w(u_i, u_{i+1})}{\sqrt{d_{u_i} d_{u_{i+1}}}} \tag{3.8}$$

Here, $\mathcal{P}^l$ is interpreted as the sum of weights of all walks of length l joining nodes u and v. A walk S_l is a sequence of vertices $u_0, \cdots, u_l$ such that $u_i = u_{i+1}$ or $(u_i, u_{i+1}) \in E$. By defining $\mathcal{P}(u,u) = 1$, we create a self-loop for each node on the walk. So the walk can pause on any node for a number of steps before the next move. This gives us better behaved distribution of P^l over the path length l. Here the definition of $\mathcal{P}(u,u) = 1$ is important because it allows self-loops in the adjacency matrix. To this end, we aim to exploit the fact that the matrix $\mathcal{P}^l$ contains information concerning the inter-node distance distribution to construct a measure that can be used to partition graphs.

3.3.3 Proximity Weights

Our idea is to use the distribution of distances to compute the average path-length between pairs of nodes in the graph. For the nodes u and v the average path-length is given by

$$\hat{d}(u,v) = \frac{\sum_l l\mathcal{P}^l(u,v)}{\sum_l \mathcal{P}^l(u,v)} \tag{3.9}$$

This average distance measure can be used to compute a Gaussian weighted node proximity matrix. For the nodes u and v the proximity weight is given by *path-*

weighted matrix.

$$\Omega_p(u,v) = \exp\left[-\frac{\hat{d}^2(u,v)}{2\sigma^2(u,v)}\right] \tag{3.10}$$

where

$$\sigma^2(u,v) = \frac{\sum_l (l-\hat{d}(u,v))^2 \mathcal{P}^l(u,v)}{\sum_l \mathcal{P}^l(u,v)} \tag{3.11}$$

is the variance of the path-length distribution for nodes u and v.

3.3.4 Properties of the Proximity Matrix

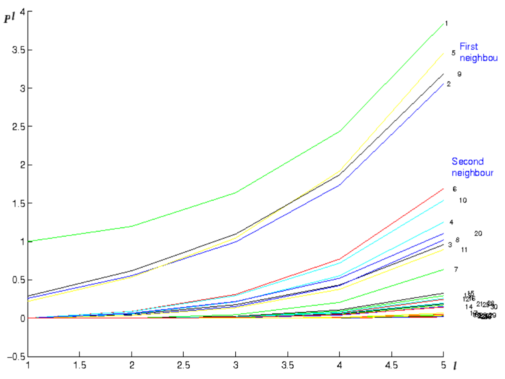
The proximity matrix Ω_p defined in the previous section has some interesting properties that distinguish it from the raw adjacency matrix. Here we focus on some of these in detail.

Firstly, although the adjacency matrix may contain a significant number of zero off-diagonal entries, provided that the graph under study is connected, then the path-length proximity matrix will not have zero off-diagonal entries since a path of finite length can always be located between a pair of nodes. The consequence of this is that the path-length proximity matrix will be less likely to be singular or to have a zero determinant.

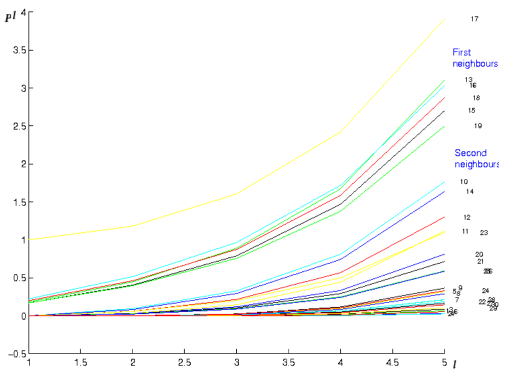
Second, nodes which have similar locations with respect to the boundary of the graph will have similar path-weight values. Here graphs are restrict to planar ones so graph boundary indicates the set of nodes and edges located on graph's perimeter. Since the path-length proximity matrix is constructed using node distance, the nodes on the boundary will have different values to those near the centre of the graph. This means that the measure could be useful for the purposes of assigning node affinity in the problem of graph-matching.

As an illustration of the points mentioned above, for the Delaunay graph shown in Figure 3.2, Figure 3.4(a) and Figure 3.4(b) show $\mathcal{P}^l(u,v)$ as a function of l for the nodes labelled 1 and 17. The different curves are obtained when v

runs over the remaining nodes of the graph, and are labelled with node number. From the figure we can see that $\mathcal{P}^l(u,u)$ always takes on the largest value, irrespective of l, since it counts the number of loops of length l to node u. The remaining curves are ordered in descending order according to whether nodes are first, second or third etc. neighbours. The most distant nodes are associated with the smallest values of $\mathcal{P}^l(u,v)$. Another important property is that the nodes in the interior of the graph always have larger values of $\mathcal{P}^l(u,v)$ than those on or near the boundary.



(a) Path length distribution of node01

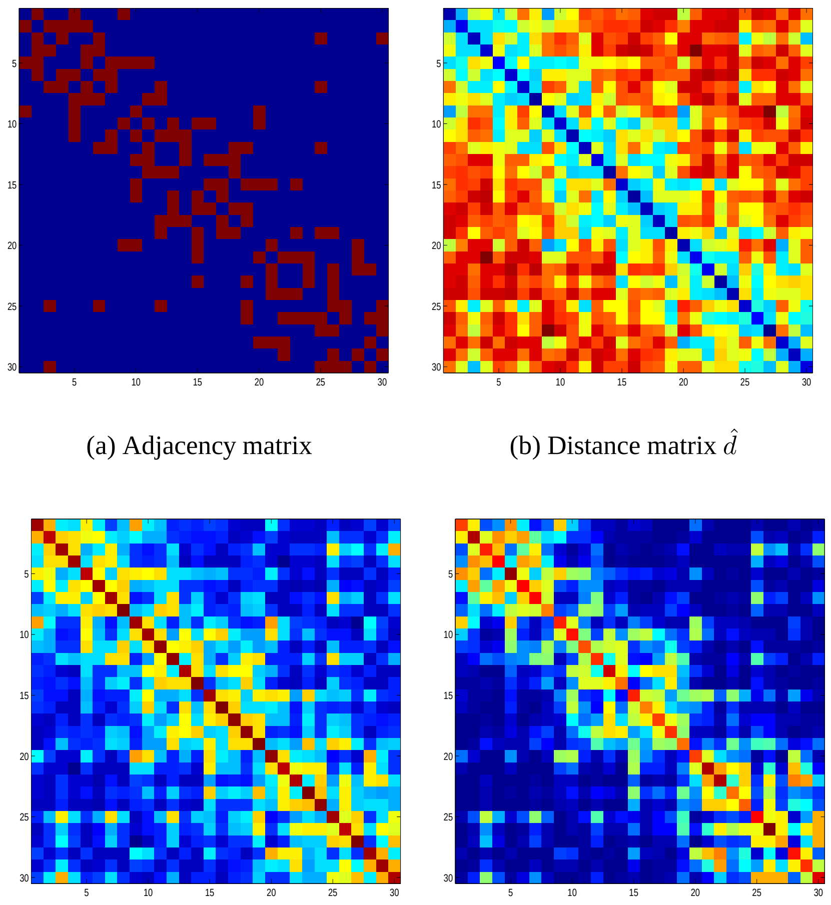


(b) Path length distribution of node17

Figure 3.4: Distribution of $\mathcal{P}^l$ based on the path step l.

Finally, we note that when compared to the binary adjacency matrix, the path-weighted proximity matrix is more robust to changes in graph structure. To illustrate this point consider the deletion of an edge. In the case of the adjacency matrix, two symmetrically placed elements flip from one to zero. Hence, all memory of the edge is lost. However, in the case of the path-weighted proximity matrix the mean distance between the nodes is increased. Our aim is to use this path-weighted proximity matrix Ω_p to represent graphs rather than using the binary adjacency matrix Ω. Graphs represented by Ω_p should be more robust to structural error and noise.

For the graph shown above, in Figure 3.4, the four panels in Figure 3.5 show the adjacency matrix, the matrix of path weighted distances $\hat{d}(u,v)$, the path length variance $\sigma(u,v)$ and the path weighted proximity matrix $\Omega_p(u,v)$. The entries in the adjacency matrix correspond to maxima in the weight matrix.



(a) Adjacency matrix

(b) Distance matrix $\hat{d}$

(c) Sigma matrix σ

(d) Path-weighted matrix Ω_p

Figure 3.5: Similarity matrices.

3.4 Matching

Our aim here is to match the graphs using the non-overlapping super-cliques delivered by the Fiedler vector. With these super-cliques in hand, our partition matching is realized by two consecutive steps. Given two graphs to be matched, the first step is to look for the correspondences between super-cliques. We try each possible pair of super-cliques in an exhaustive way and determine the correspondences by maximum a posteriori (MAP). Then the next step is to find the mappings of each pair of nodes in the matched super-cliques. To perform the matching we use both the discrete relaxation method from Wilson and Hancock (Wilson and Hancock, 1997) and the edit-distance method of Myers, Wilson and Hancock (Myers et al., 2000). In this section for completeness, we review the elements of their methods and explain how they are extended to our graph partition matching frame work.

3.4.1 Matching Probabilities

Given a data graph $\Gamma_D = (V_D, E_D)$ to be matched onto a model graph $\Gamma_M = (V_M, E_M)$, we first compute their super-cliques using the partition method presented in Section 3.2.

Let the obtained super-cliques be $\mathcal{S}_D = (\hat{N}_1^D, \hat{N}_2^D, \cdots, \hat{N}_u^D, \cdots, \hat{N}_m^D)$ and $\mathcal{S}_M = (\hat{N}_1^M, \hat{N}_2^M, \cdots, \hat{N}_v^M, \cdots, \hat{N}_n^M)$ for graph Γ_D and Γ_M respectively. Here, m and n are the total number of super-cliques in each of the graphs. The state of correspondence match can be represented by the function $f : V_D \mapsto V_M \cup \{\epsilon\}$ from the node-set of the data graph onto the node-set of the model graph, where the node-set of the model graph is augmented by adding a NULL label, ϵ, to allow for unmatchable nodes in the data graph.

Our objective function for the match is the matching probabilities for the set of super-cliques of the graphs and given by

$$\mathcal{M}_{D,M}(f) = \sum_{\hat{N}_u^D \in \mathcal{S}_D} \sum_{\hat{N}_v^M \in \mathcal{S}_M} P(\hat{N}_u^D, \hat{N}_v^M) \tag{3.12}$$

Here, $P(\hat{N}_u^D, \hat{N}_v^M)$ denotes the matching probability of a pair of super-cliques $\hat{N}_u^D$ and $\hat{N}_v^M$ under the matching function f. The idea is to find the correspondence between each pair of nodes by optimising Equ. 3.12 in a super-clique to super-clique way.

3.4.2 Discrete Relaxation

If we take the matching function f as a memoryless error process, super-clique matching probability $P(\hat{N}_u^D, \hat{N}_v^M)$ in Equ. 3.12 can then be factorized into the matching probability of nodes in each pair of the super-cliques

$$P(\hat{N}_u^D, \hat{N}_v^M) = \prod_{i \in \hat{N}_u^D, j \in \hat{N}_v^M} P\left(f\left(i\right)|j\right) \tag{3.13}$$

Furthermore, node matching probability $P(f(i)|j)$ can be given as a function of the node error matching probability P_e by

$$P\left(f\left(i\right)|j\right) = \begin{cases} (1 - P_e) & \text{if } \left(f\left(i\right), j\right) \text{ is a correct correspondence} \\ P_e & \text{otherwise} \end{cases} \tag{3.14}$$

Here, in Fig. 3.6, we show an example of matching two un-equal sized super-cliques using discrete relaxation. The super-clique on the left with a degree of three is to be matched onto the super-clique on the right with a degree of five. In order to preceed node matching, we have to pad the less degree super-clique with some dummy nodes. The total number of added dummy nodes is equal to the difference in their degrees. Moreover, all possible ways of paddings have to be considered. In Fig. 3.6, we show six sample patterns of dictionary paddings in the middle of the figure and the dummy nodes are marked with symbol d in red color.

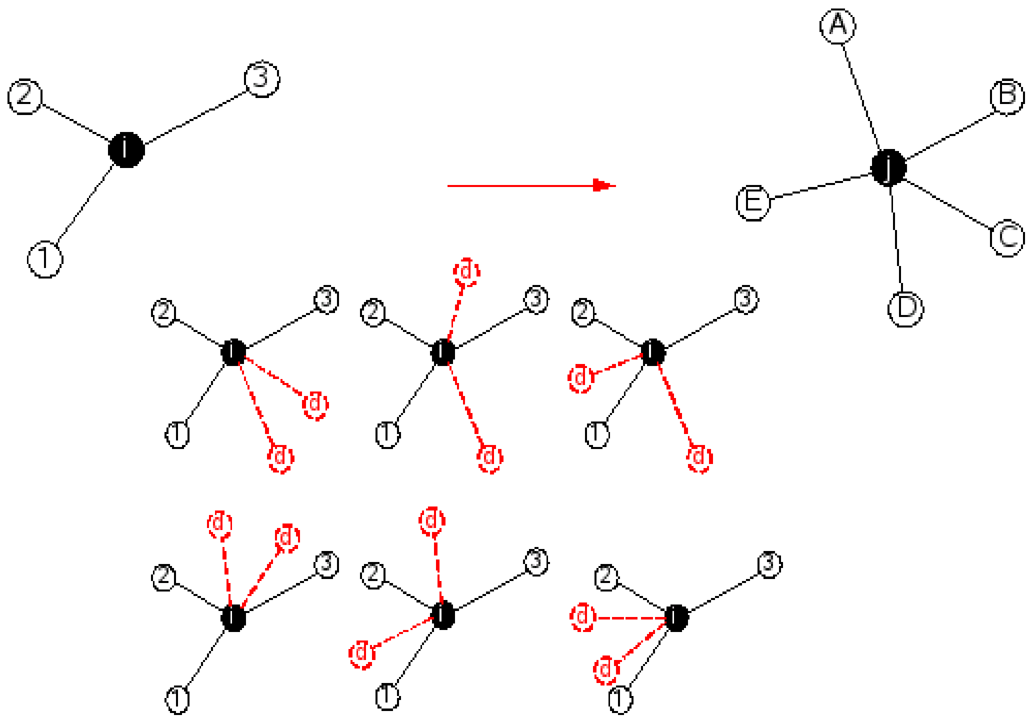


Figure 3.6: Dictionary padding with two dummy nodes.

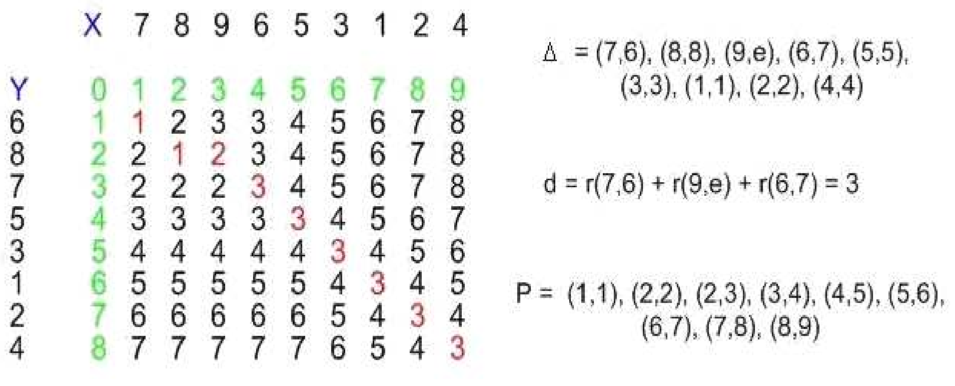
3.4.3 Edit Distance

Dictionary padding is an explicit way of characterising the structural differences in the super-cliques but it is also computationally un-efficient. This is due to the increasing number of padding patterns we have to consider when the degree difference in two super-cliques grows large. In this section, we will introduce an efficient alternative to measure the structural differences in the super-cliques, the Levenshtein or string edit distance (Levenshtein, 1966; Wagner and Fischer, 1974; Myers et al., 2000).

Let X and Y be two strings of symbols drawn from an alphabet Σ. We wish to convert X to Y via an ordered sequence of operations such that the cost associated with the sequence is minimal. The original string to string correction algorithm defined *elementary edit operations*, $(a, b) \neq (\epsilon, \epsilon)$ where a and b are symbols from the two strings or the NULL symbol, ϵ. Thus, changing symbol x to y is denoted by (x, y), inserting y is denoted (ϵ, y), and deleting x is denoted

(x, ϵ). A sequence of such operations which transforms X into Y is known as an *edit transformation* and denoted $\mathcal{T} =< \delta_1, ..., \delta_{|\mathcal{T}|} >$. Elementary costs are assigned by an elementary weighting function $\gamma : \Sigma \cup \{\epsilon\} \times \Sigma \cup \{\epsilon\} \mapsto \Re$; the cost of an edit transformation, $C(\mathcal{T})$, is the sum of its elementary costs. The edit distance between X and Y is defined as

$$\mathbf{d}(X, Y) = \min\{C(\mathcal{T}) | \mathcal{T} \text{ transforms } X \text{ to } Y\} \tag{3.15}$$



	X	7	8	9	6	5	3	1	2	4
Y	0	1	2	3	4	5	6	7	8	9
6	1	1	2	3	3	4	5	6	7	8
8	2	2	1	2	3	4	5	6	7	8
7	3	2	2	2	3	4	5	6	7	8
5	4	3	3	3	3	3	4	5	6	7
3	5	4	4	4	4	4	3	4	5	6
1	6	5	5	5	5	5	4	3	4	5
2	7	6	6	6	6	6	5	4	3	4
4	8	7	7	7	7	7	6	5	4	3

Δ = (7,6), (8,8), (9,e), (6,7), (5,5), (3,3), (1,1), (2,2), (4,4)

d = r(7,6) + r(9,e) + r(6,7) = 3

P = (1,1), (2,2), (2,3), (3,4), (4,5), (5,6), (6,7), (7,8), (8,9)

Figure 3.7: Edit distance and *edit path* for two strings

In (Marzal and Vidal, 1995), Marzal and Vidal introduced the notion of an *edit path* which is a sequence of ordered pairs of positions in X and Y such that the path monotonically traverses the edit matrix of x and y from $(0, 0)$ to $(|X|, |Y|)$. An example of *edit path* between string X and Y is shown in Fig. 3.7. From the figure, it is clear that their edit distance is 3, as listed in the right-bottom corner of the edit matrix. The corresponding *edit path* is shown in red color. Essentially, the transition from one point in the path to the next is equivalent to an elementary edit operation: $(a, b) \rightarrow (a + 1, b)$ corresponds to deletion of the symbol in X at position a. Similarly, $(a, b) \rightarrow (a, b + 1)$ corresponds to insertion of the symbol at position b in Y. The transition $(a, b) \rightarrow (a + 1, b + 1)$ corresponds to a change from $X(a)$ to $Y(b)$. Thus, the cost of an edit path can be determined by summing the elementary weights of the edit operations implied by the path.

As a result, we can replace X and Y by N_u^D and N_v^M, which are the neighbour nodes of two super-cliques $\hat{N}_u^D$ and $\hat{N}_v^M$ to be matched. The node matching probability $P(f(i)|j)$ in Equ. 3.13 can then be computed using the corresponding edit operations

$$P\left(f\left(i\right)|j\right) = \begin{cases} (1-P_e) & \text{if } \left(f\left(i\right),j\right) \text{ is an identity} \\ P_e & \text{otherwise} \end{cases} \tag{3.16}$$

3.5 Hierarchical Simplification

The super-cliques extracted using the Fiedler vector may also be used to perform hierarchical graph simplification.

3.5.1 Partition Arrangements

Our simplification process proceeds as follows. We create a new graph in which each super-clique $\hat{N}_u = \{u\} \cup \{v; (u,v) \in E\}$ is represented by a node. In practice this is done by eliminating those nodes, which are not the center nodes of the super-cliques $N_u = \hat{N}_u \setminus \{u\}$. In other words, we select the center node of each super-clique to be the node-set for the next level representation. The node set is given by $\hat{V} = \left\{\hat{N}_1 \setminus N_1, \hat{N}_2 \setminus N_2, \ldots, \hat{N}_n \setminus N_n\right\}$. Our next step is to construct the edge-set for the simplified graph. We construct an edge between two nodes if there is a common edge contained within their associated super-cliques. The condition for the nodes $u \in \hat{V}$ and $v \in \hat{V}$ to form an edge in the simplified graph $\hat{\Gamma} = (\hat{V}, \hat{E})$ is $(u,v) \in \hat{E} \Rightarrow |\hat{N}_u \cap \hat{N}_v| \geq 2$.

3.5.2 Clustering

To provide an illustration of the usefulness of the simplifications provided by the Fiedler vector, we focus on the problem of graph clustering. The aim here is to investigate whether the simplified graphs preserve the pattern space distribution of the original graphs. There are a number of ways in which we could undertake this study. However, in order to keep with the overall philosophy of this chapter, here we use a simple graph-spectral method by Wilson et al. (Wilson et al., 2005).

Suppose that we aim to cluster the set of M graphs $\{\Gamma_1, ...\Gamma_k, ...\Gamma_M\}$. We commence by performing the spectral decomposition $L_k = \Phi_k \Lambda_k \Phi_k^T$ on the Laplacian matrix L_k for the graph indexed k, where $\Lambda_k = diag(\lambda_k^1, \lambda_k^2, ...)$ is the diagonal matrix of eigenvalues and Φ_k is a matrix with eigenvectors as columns. For the graph Γ_k, we construct a vector $B_k = (\lambda_k^1, \lambda_k^2, \dots, \lambda_k^m)^T$ from the leading m eigenvalues. We can visualise the distribution of graphs by performing multidimensional scaling (MDS) on the matrix of distances $d_{k1,k2}$ between graphs. This distribution can be computed using either the edit distance technique used in the previous section where $d_{k1,k2} = -\ln d(k1, k2)$ or by using the spectral features where $d_{k1,k2} = (B_{k1} - B_{k2})^T (B_{k1} - B_{k2})$.

Multidimensional scaling (MDS) is a procedure which allows data specified in terms of a matrix of pairwise distances to be embedded in a Euclidean space. Here we intend to use the method to embed the graphs extracted from different viewpoints in a low-dimensional space. The pairwise distances d_{k_1,k_2} are used as the elements of an $N \times N$ dissimilarity matrix $\mathbf{R}$, whose elements are defined as follows

$$R_{k_1,k_2} = \begin{cases} d_{k_1,k_2} & \text{if } k_1 \neq k_2 \\ 0 & \text{if } k_1 = k_2 \end{cases} \tag{3.17}$$

Here, we use the classical multidimensional scaling method to embed the graphs in a Euclidean space using the matrix of pairwise dissimilarities $\mathbf{R}$. The first step of MDS is to calculate a matrix $\mathcal{R}$ whose element with row r and column c is given by $\mathcal{R}_{rc} = -\frac{1}{2}[d_{rc}^2 - \hat{d}_{r.}^2 - \hat{d}_{.c}^2 + \hat{d}_{..}^2]$, where $\hat{d}_{r.} = \frac{1}{N}\sum_{c=1}^{N} d_{rc}$ is the average dissimilarity value over the r^{th} row, $\hat{d}_{.c}$ is the dissimilarity average value over the c^{th} column and $\hat{d}_{..} = \frac{1}{N^2}\sum_{r=1}^{N}\sum_{c=1}^{N} d_{r,c}$ is the average dissimilarity value over all rows and columns of the dissimilarity matrix $\mathbf{R}$.

We subject the matrix $\mathcal{R}$ to an eigenvector analysis to obtain a matrix of embedding coordinates $\mathbf{Z}$. If the rank of $\mathcal{R}$ is $k, k \leq N$, then we will have k non-zero eigenvalues. We arrange these k non-zero eigenvalues in descending order, i.e. $l_1 \geq l_2 \geq \cdots \geq l_k > 0$. The corresponding ordered eigenvectors are denoted by $\vec{v}_i$ where l_i is the ith eigenvalue. The embedding coordinate system for the graphs is $\mathbf{Z} = [\sqrt{l_1}\vec{v}_1, \sqrt{l_2}\vec{v}_2, \ldots, \sqrt{l_s}\vec{v}_s]$, For the graph indexed j, the embedded vector of coordinates is a row of matrix $\mathbf{Z}$, so $\mathbf{Z}_j = (Z_{j,1}, Z_{j,2}, ..., Z_{j,s})^T$.

3.6 Experiments

The aims in this section are threefold. First, we perform a sensitivity study to illustrate that the super-cliques delivered by the Fiedler vector are stable for computing edit distance. Second, we show that the graph partition scheme leads to accurate matches on real world data. Third, we aim to illustrate that the simplification procedure results in a stable distribution of graphs in pattern-space.

3.6.1 Sensitivity Study

The aim in this part of the experiments is to measure the sensitivity of our new partition-based matching method to structural error. The synthetic graphs we used here are the Delaunay triangulations of randomly generated point-sets with various sizes. In Figure 3.8, we show three synthetic graphs on the left column and their corresponding partitions on the right column. Graphs on the first, second and third row have 20, 40 and 60 nodes respectively. The effects of structural errors are simulated by randomly deleting nodes and re-triangulating the remaining nodes. An example is illustrated in Figure 3.9, which shows the sequence with one node deleted at a time for the graph with 40 nodes. Coded in different colours are the different super-cliques which result from the partitioning of nodes. These remain relatively stable as the nodes are deleted.

To asset the stability of graph partitions on different size of graphs, we have matched the three sets of corrupted graphs to their original ones. We used the edit distance matching method presented in Section 3.4.3 and averaged the results with 50 trials for each graph set. Figure 3.10 shows the fraction of correct correspondences as a function of the fraction of nodes deleted. From the figure it is clear that graph with node size 40 gives the best result. This is because the number of partitions in this set of graphs is quite moderate. As a result, it didn't turn out to have matching errors in the partition level, i.e. miss-matching of partitions. However larger graphs such as the one with 60 nodes in this experiment do have this problem and it can be seen in the figure that even when there is no structural corruption, successful partition matching only achieves 90%. Another interesting thing to notice is that larger graphs (60 nodes) outperformed the middle size ones (40 nodes) when they are under severe corruption. This is because larger graphs still have a considerable amount of nodes left to form the super-cliques in that condition. This also explains why small size graphs (20 nodes) performed badly.

For the best performing graphs, i.e. the graphs with original size of 40, we also compare the results with four alternative algorithms. These are the original discrete relaxation method of Wilson and Hancock (Wilson and Hancock, 1997) which is applied to overlapping super-cliques, the quadratic assignment method

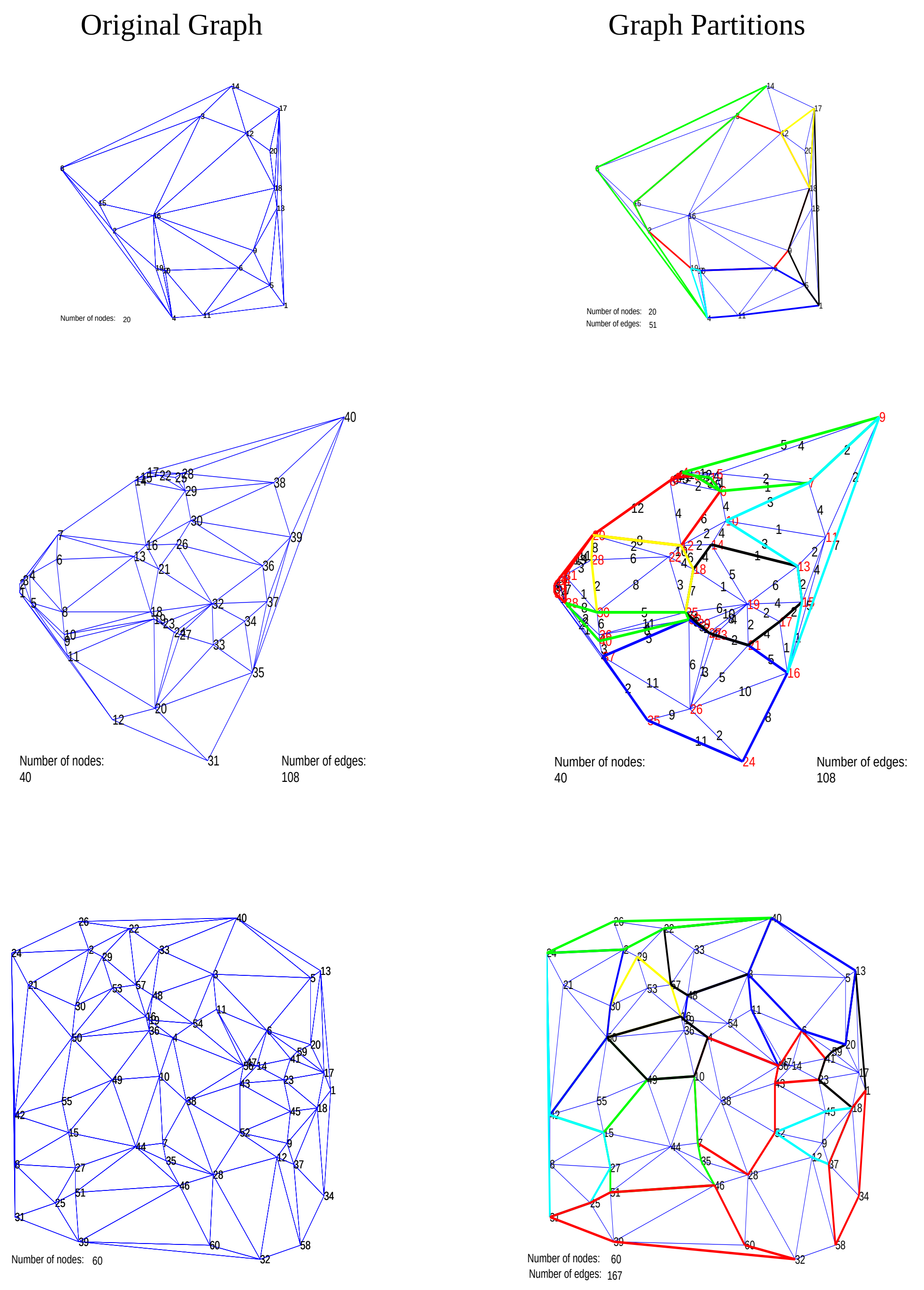


Figure 3.8: Synthetic graphs and their partitions.

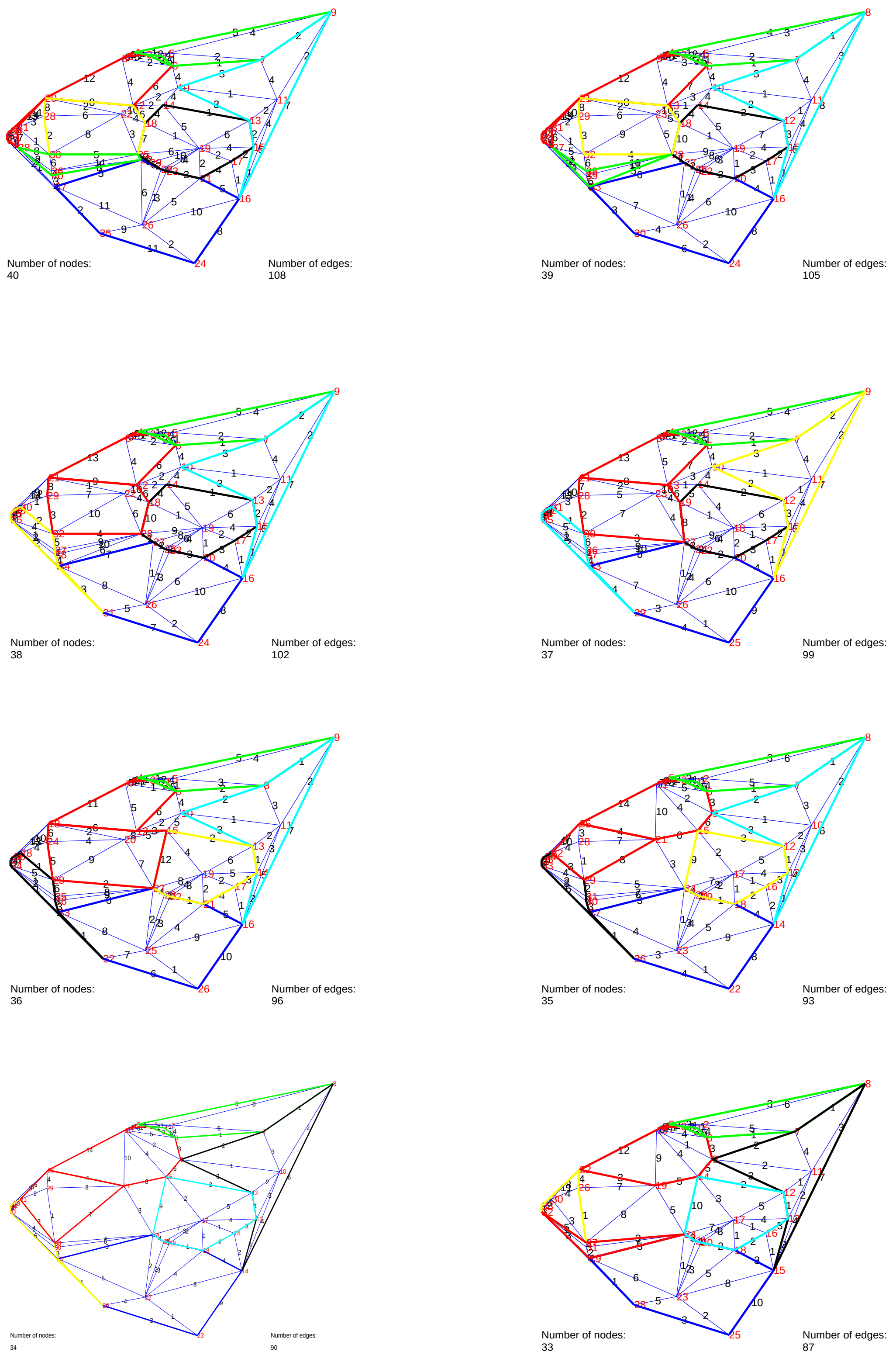


Figure 3.9: A sequence of synthetic graphs showing the effect of controlled node deletion on the stability of the super-clique.

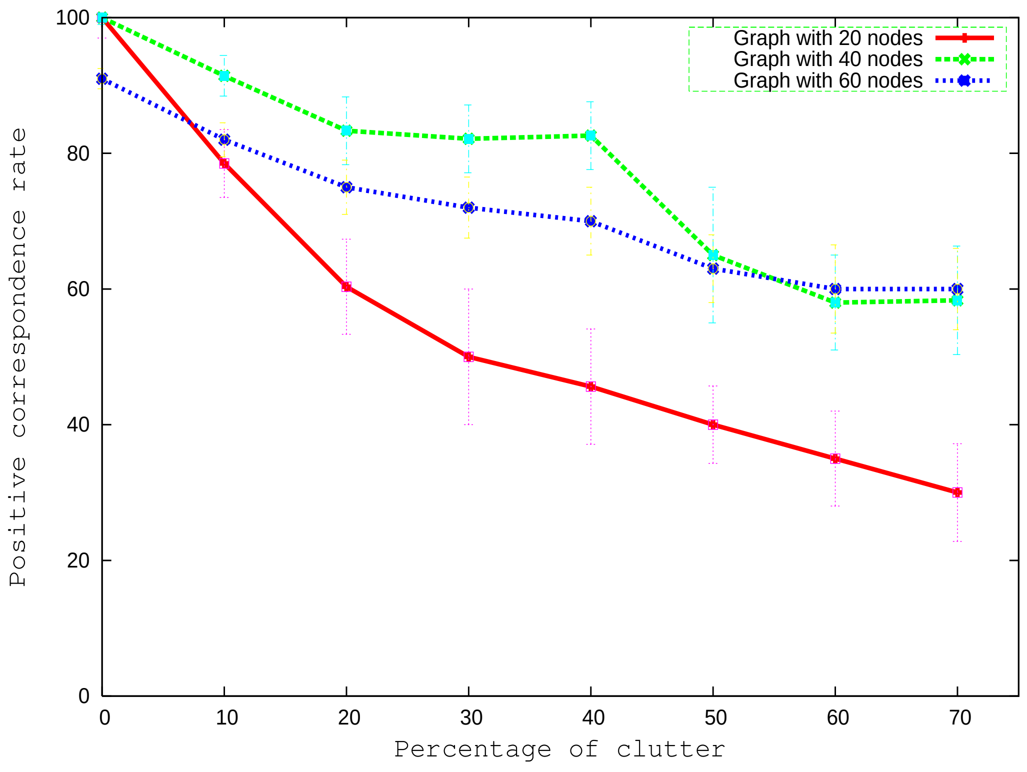


Figure 3.10: Sensitivity study for graphs of different size.

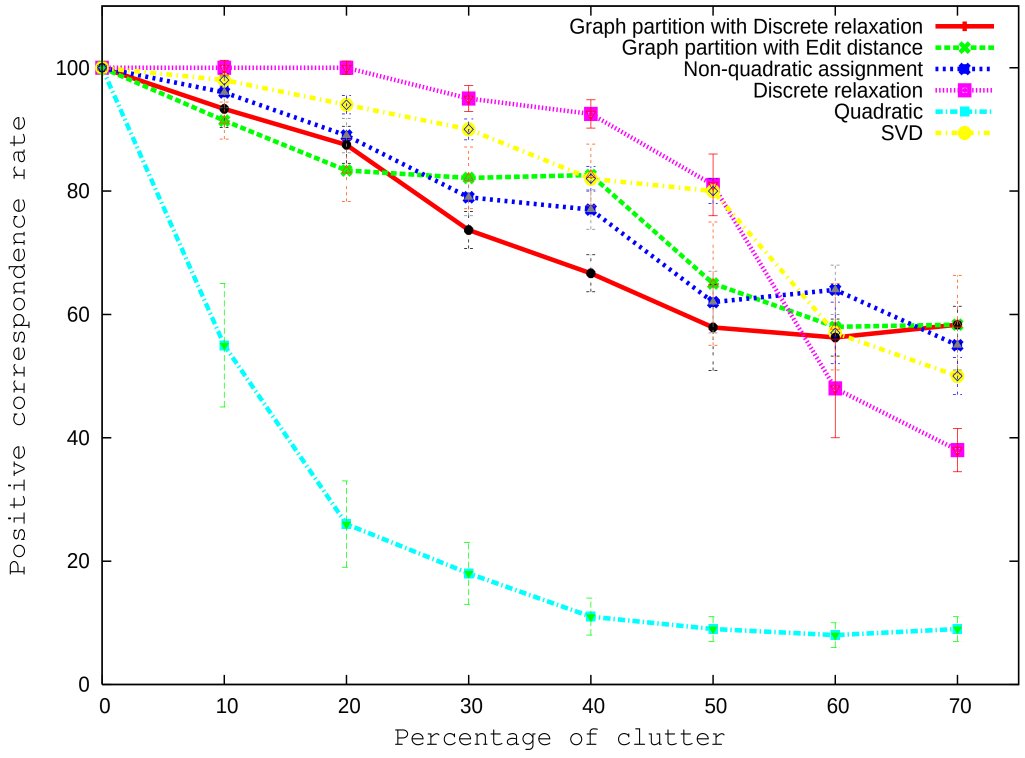


Figure 3.11: Sensitivity comparison for original graph with 40 nodes.

of Gold and Rangarajan (Gold and Rangarajan, 1996), the non-quadratic graduated assignment method of Finch, Wilson and Hancock (Finch et al., 1998), and, the singular value decomposition method of Luo and Hancock (Luo and Hancock, 2001). In Figure 3.11 we show the fraction of correct correspondences as a function of the fraction of nodes deleted.

From this comparison it is clear that our method is robust to structural error. However, it does not perform as well as the original Wilson and Hancock method. One reason for this is that the super-cliques delivered by our partitioning method do become unstable under significant corruption.

When used in conjunction with the edit-distance method, the partitions lead to better results than when used with the dictionary-based discrete relaxation method. This is important since the former method is more computationally effi-

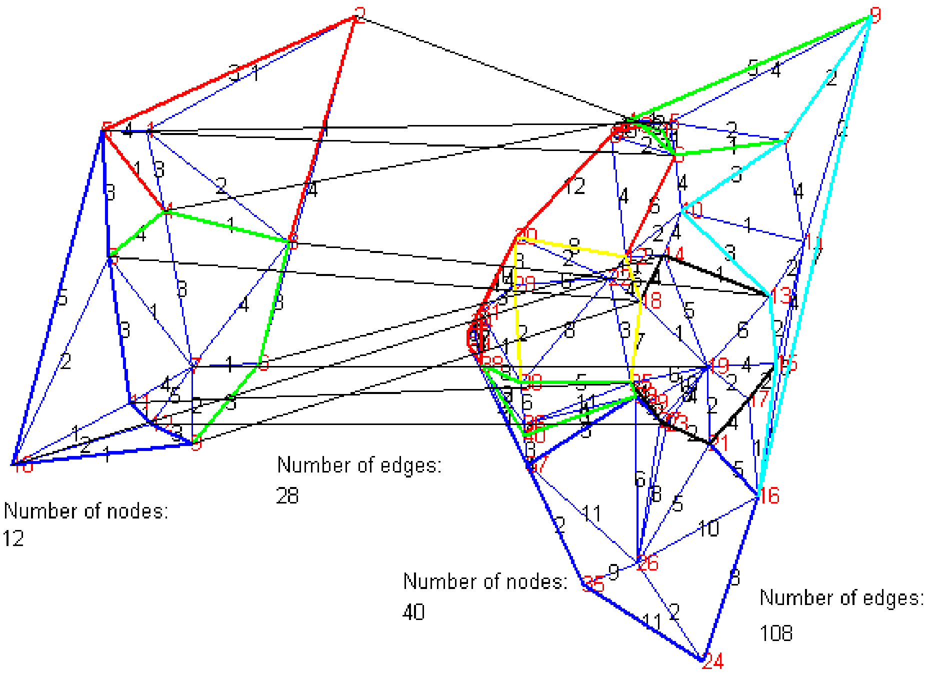


Figure 3.12: An example in inexact graph matching.

cient than the latter, since the overheads associated with dictionary construction can grow exponentially if dummy nodes need to be inserted.

An example of the set of matches used in this experiment is shown in Figure 3.12. Here the different colours in the two graphs again encode the super-cliques. The thin black lines between the two graphs show the correspondence matches. Here the results were obtained using edit-distance method described earlier. The graphs are of very different size. The set of roughly parallel lines correspond to the correct correspondences, and the remaining lines are the correspondence errors.

3.6.2 Real-Word Data

The real-world data used here is comprised of two house sequences. One of them is taken from the CMU model-house sequence and the other is from the MOVI

Table 3.1: Correspondence results for the three methods.

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	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
Discrete Relaxation	Correct	-	26	23	18	16	15	15	11	14	9
	False	-	4	6	9	12	14	13	17	16	20
	Missed	-	0	1	3	2	1	2	2	0	1
Edit Distance	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

model-house sequence. Those two sequences are made up of a series of images which have been captured from different viewpoints. In order to convert the images into abstract graphs for matching, we extract point features using corner detector by Luo, Cross and Hancock (Luo et al., 1998; Luo et al., 1999). Our graphs are the Delaunay triangulations of the corner-features. Two examples from both sequences are shown in Figure 3.13. First row of the figure shows the original images and the second row shows their corresponding partitions. To illustrate the structural variation of the Delaunay graphs w.r.t. the view point change, we show the CMU house sequence overlayed by their Delaunay graphs in Figure 3.14. The super-cliques obtained by graph partition are also shown and coded in different colors.

We have matched the first image to each of the subsequent images in CMU sequence by using discrete relaxation and edit distance. The results of those two methods are compared with those obtained using the method of Luo and Hancock (Luo and Hancock, 2001) in Table 3.1. This table contains the number of detected corners to be matched, the number of correct correspondence, the number of missed corners and the number of miss-matched corners.

Figure 3.15 shows us the correct correspondence rate as a function of view

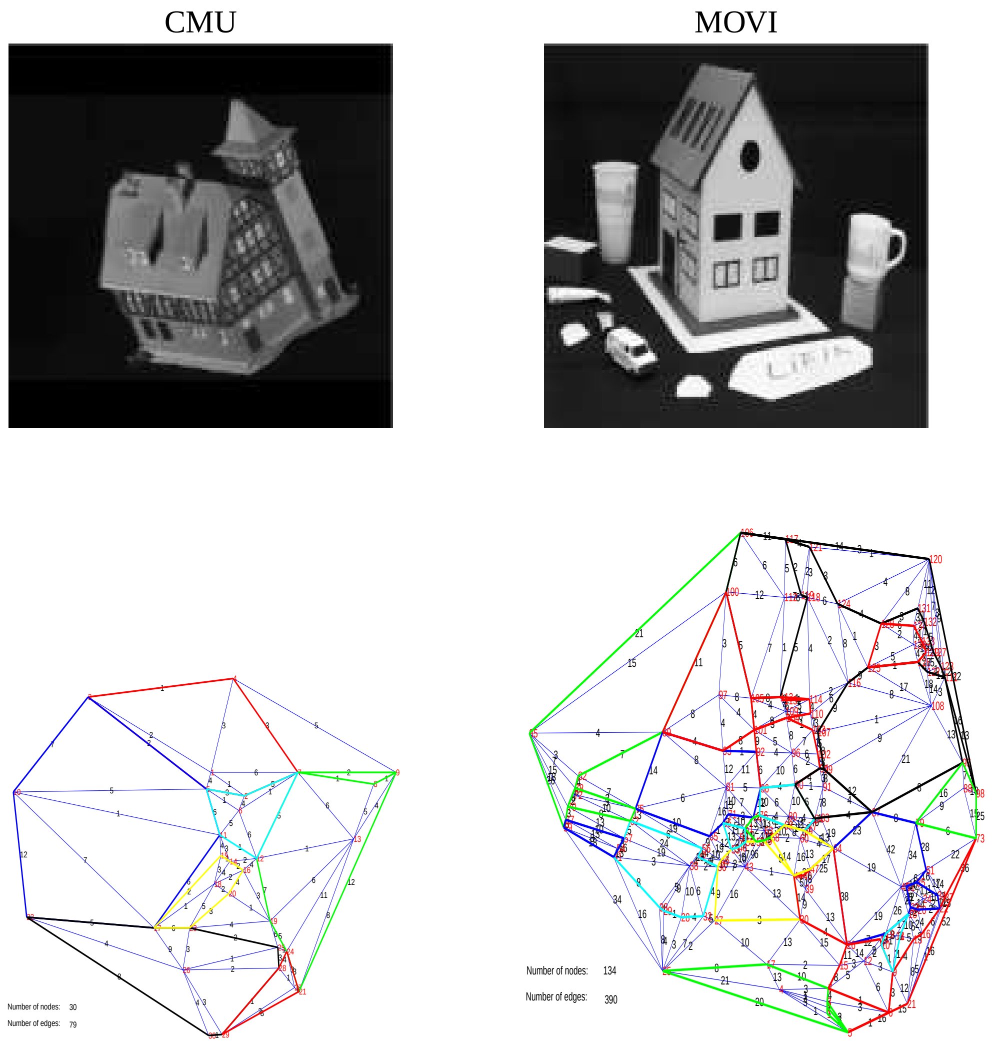


Figure 3.13: Delaunay graphs with their partitions from real-world data.

difference for the two methods based on the data in Table 3.1. It also shows the result of edit distance based partition matching method on MOVI house sequence. From the results, it is clear that our new method degrades gradually and out performs the Luo and Hancock's EM method when the difference in viewing angle is large. As we have seen in the previous subsection, our method did not perform well for larger size graphs (MOVI house sequence in this case). Again, this is due to the miss-matching of the partitions. When graph grows, the number of its partitions increases as well. The result of it is to have many similar

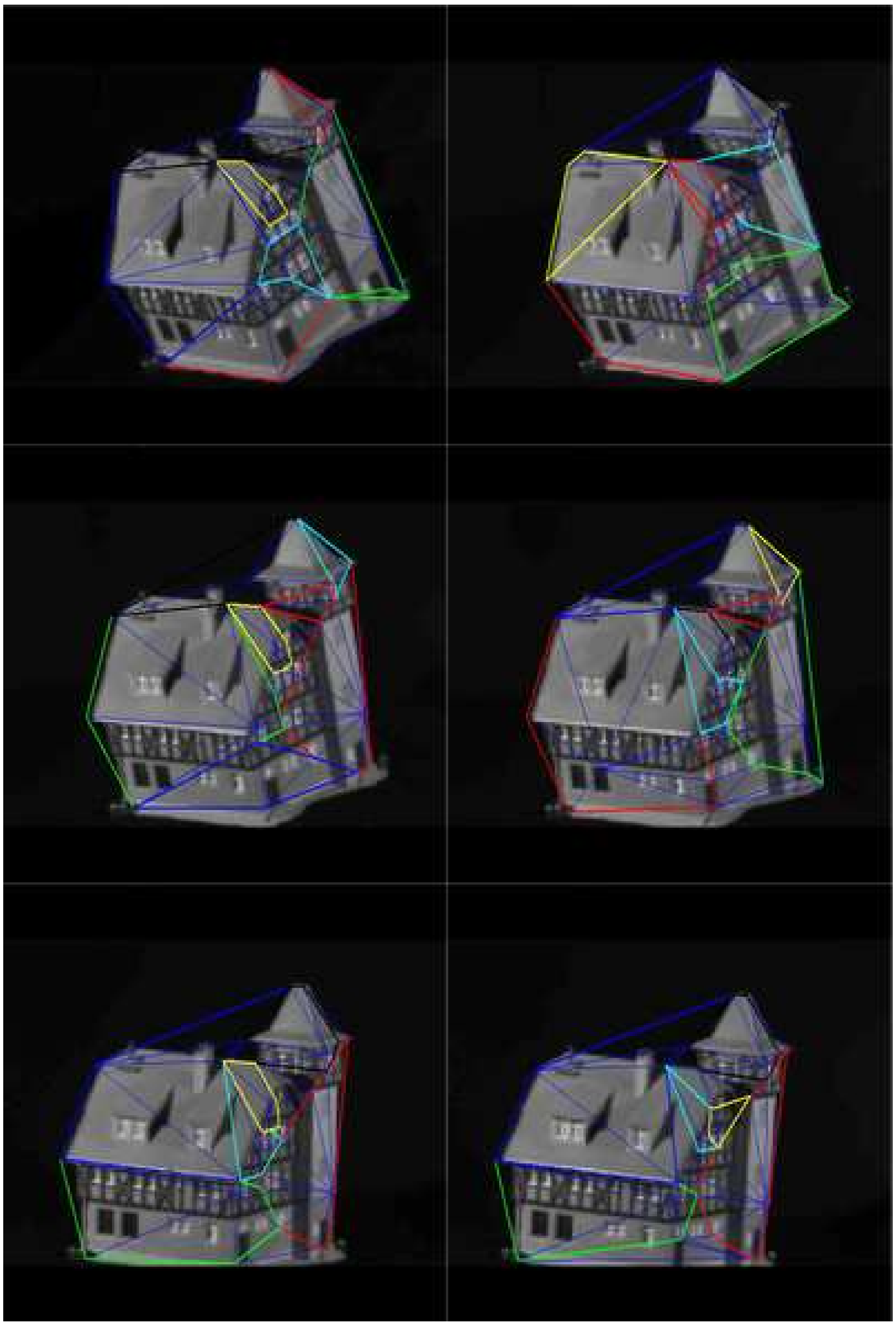


Figure 3.14: Graph partition on Delaunay triangulations.

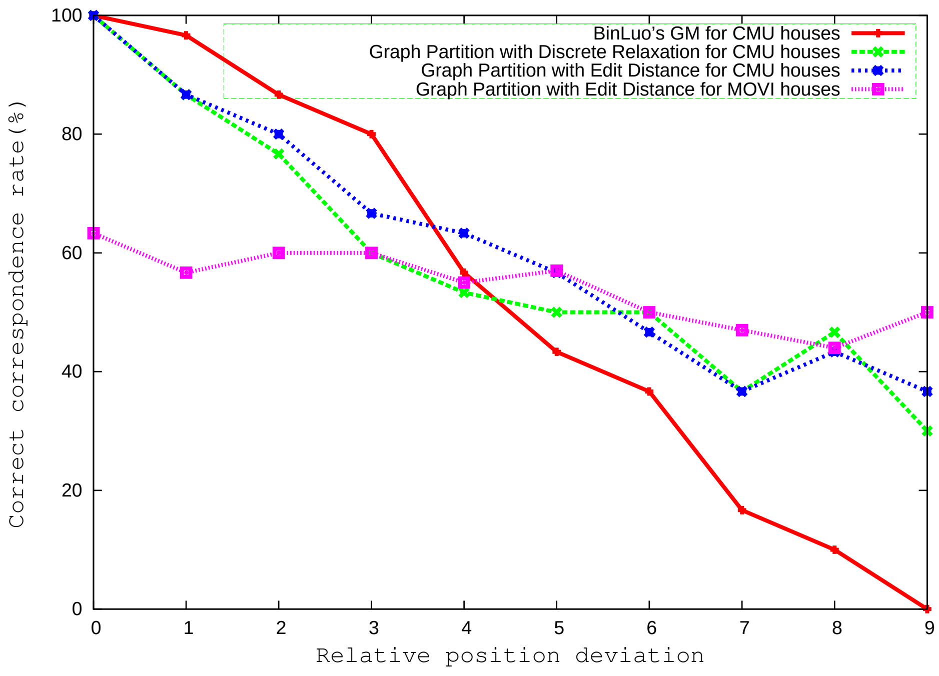


Figure 3.15: Comparison of results.

partitions, which downgrade the matching accuracy significantly. To illustrate the matching correspondence, Figures 3.16 to 3.19 show the results of each pair of graph matching for CMU houses. There are clearly significant structural differences in the graphs including rotation, scaling and perspective distortion. But even in the worst case, our method has a correct correspondence rate of 36%.

3.6.3 Partition Structure Stabilization

Our aim in this section is to explore how the path-weighted proximity matrix can be used for the purposes of graph partition, and to determine whether it can render the process more robust to structural error.

There are two aspects to our study. We commence by investigating the difference in the partitions obtained with the adjacency matrix and the path-weighted proximity matrix. Second, we perform a sensitivity study to compare the robustness of the partitions under node and edge deletions.

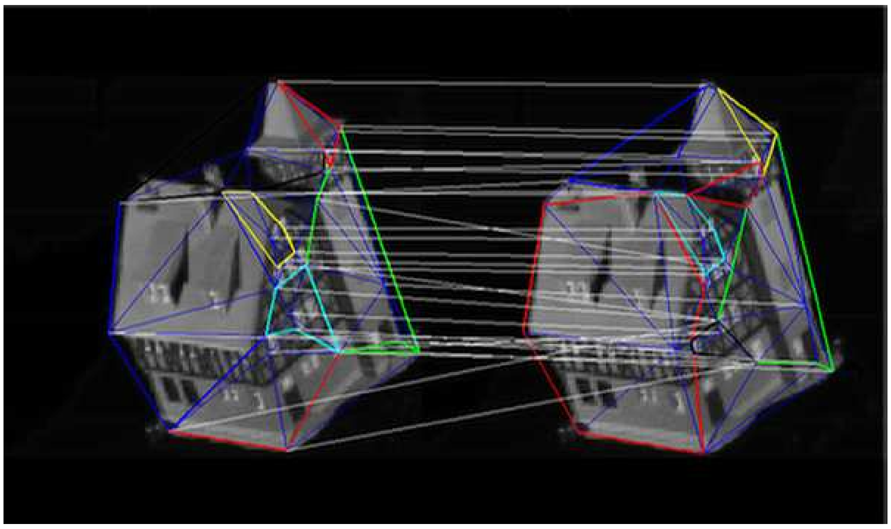


Figure 3.16: Correspondences between the first and the third images.

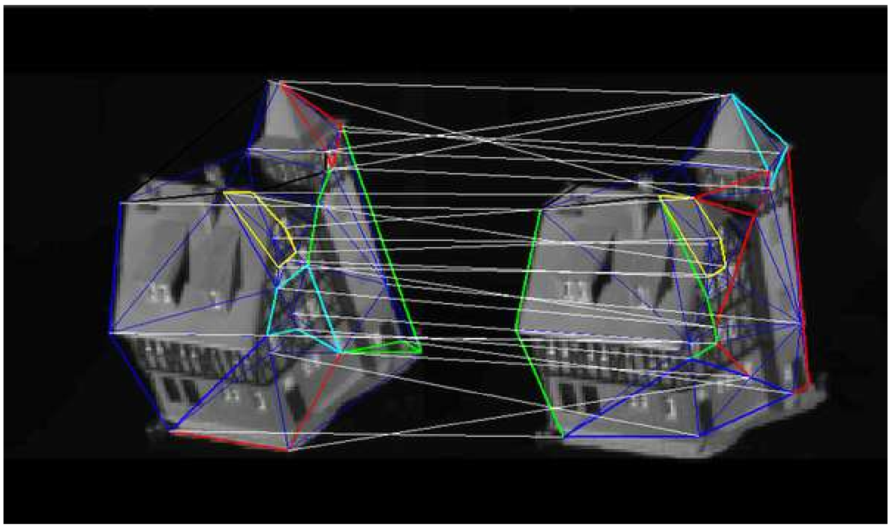


Figure 3.17: Correspondences between the first and the fifth images.

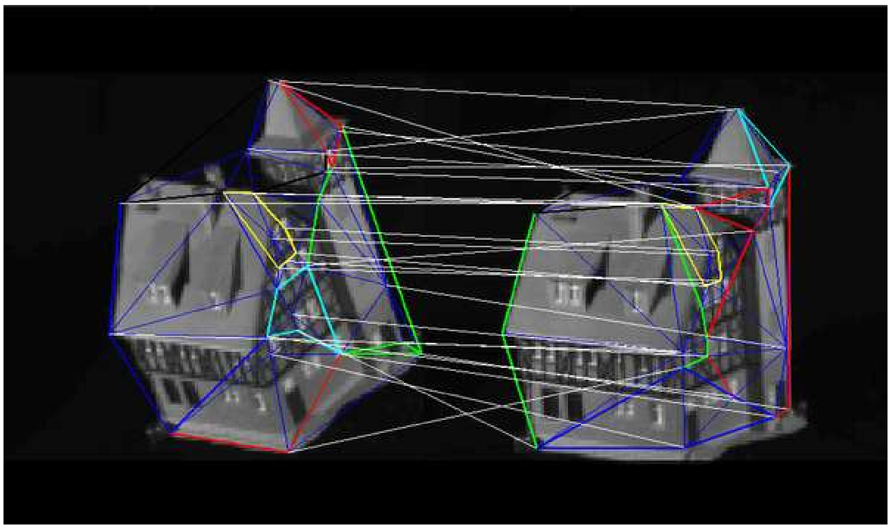


Figure 3.18: Correspondences between the first and the seventh images.

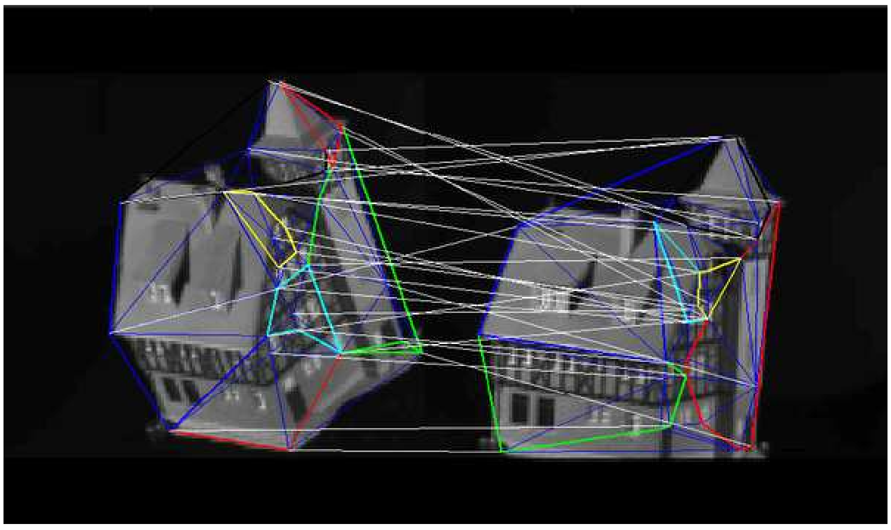


Figure 3.19: Correspondences between the first and the tenth images.

To test the performance of our new graph representation on the real-world images, the graphs furnished here are the same as the previous section.

Partition and Matching Consistency Analysis

Since our graphs represent a series with similar structures, they should share a similar partition arrangements. This is important since if we are to use the partitions for graph-matching, then they must be stable. The more similar two partitions, the better the matching result will be. Our aim here is to check which matrix-representation preserves the partition consistency better. We use the partition of the first graph as the model pattern and compare with the partitions of the remaining graphs in the sequence.

In Figure 3.20(a) we show the fraction of edges that remain in the same partition as a function of the difference in view number. The green curve shows the result obtained using the path-weighted proximity matrix, while the red curve shows the result obtained with the adjacency matrix. For large difference in view number, i.e. when the structural differences are greatest, then the path weighted proximity matrix seems to be more stable than the adjacency matrix.

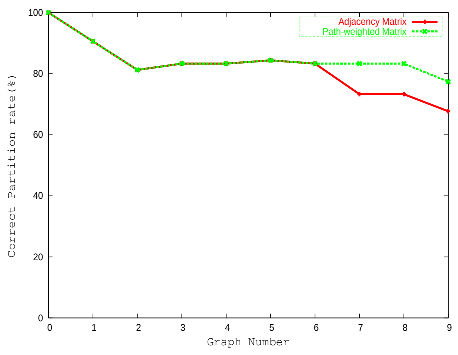
In Figure 3.20(b) we show the fraction of correct matches as a function of difference in view number. The blue curve, which represents the path-weighted proximity matrix, outperforms the green one from the adjacency matrix and the red one which is the result of the EM graph matching method described (Luo and Hancock, 2001).

Partition and Matching Stability Analysis

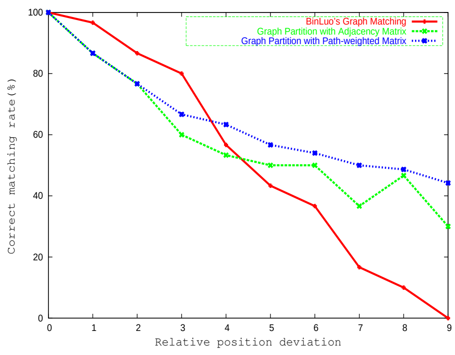
In this subsection we aim to measure the sensitivity of our graph partition method to structural error, and compare the results obtained with the path-weighted proximity and the adjacency matrix.

The effects of structural error are simulated by randomly deleting nodes or edges from the graphs under study. Figure 3.20(c) shows fraction of nodes that remain in the same partition as the graph shown in Figure 3.3 is subjected to increasing corruption. The graph corruption rate is defined to be the number of deleted edges divided by the total number of original edges. As the level of corruption is increased, then the path-weighted proximity matrix outperforms the adjacency matrix in terms of partition stability. This means that the path-weighted proximity matrix better preserves the partition structure and is more stable under structural error. This stability property has knock-on effects for the performance of the graph-matching method. In Figure 3.20(d) we show the performance of the matching process as the fraction of corruption is increased. Here the red curve is the result of the original discrete relaxation scheme, the blue curve is that obtained when we apply spectral partitioning to the adjacency matrix, and the blue curve the result when we apply spectral partitioning to the path-weighted adjacency matrix. For large levels of corruption, the results obtained using the path-weighted adjacency matrix outperform those obtained using the alternative methods.

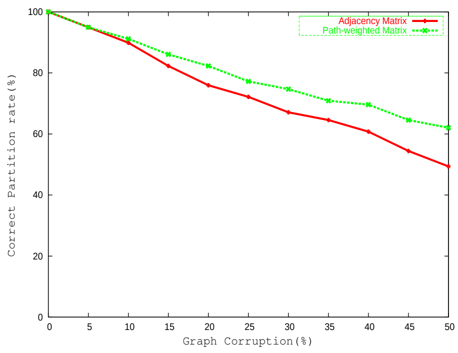
Finally, we provide some examples to illustrate the stability of the partitions obtained. In the left-hand column, we show the partitions obtained using the adjacency matrix while the right-hand column shows the partitions from the path-weighted adjacency matrix. The differently coloured edges of the graph indicate the different partitions obtained by the two methods. In the top row of Figure 3.21 we show the partitions of the graph shown in Figure 3.2, and here the result obtained by the path-weighted adjacency matrix is closer to the original than that delivered by the adjacency matrix. The remaining rows in Figure 3.21 show the effect of graph-corruption on the partitions. Rows 2 and 3 show the effect of different levels of edge corruption, and Row 4 the effect of node corruption. In all case the path-weighted adjacency matrix is more stable than the adjacency matrix.



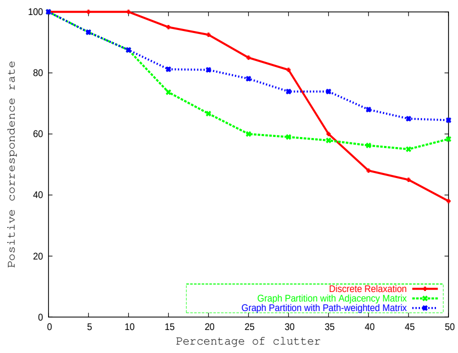
(a) Partition Consistency Analysis



(b) Matching Consistency Analysis



(c) Partition Stability Analysis



(d) Matching Stability Analysis

Figure 3.20: Partition and matching analysis.

3.6.4 Graph clustering

We have collected sequences of views for three toy houses. For each object the image sequences are obtained under slowly varying changes in viewer direction. From each image in each view sequence, we extract corner features. We use the extracted corner points to construct Delaunay graphs. In our experiments we use three different sequences. Each sequence contains images with equally spaced viewing directions. In Figure 3.22 we show examples of the raw image data and the associated graphs for the three toy houses, which we refer to as CMU/VASC,

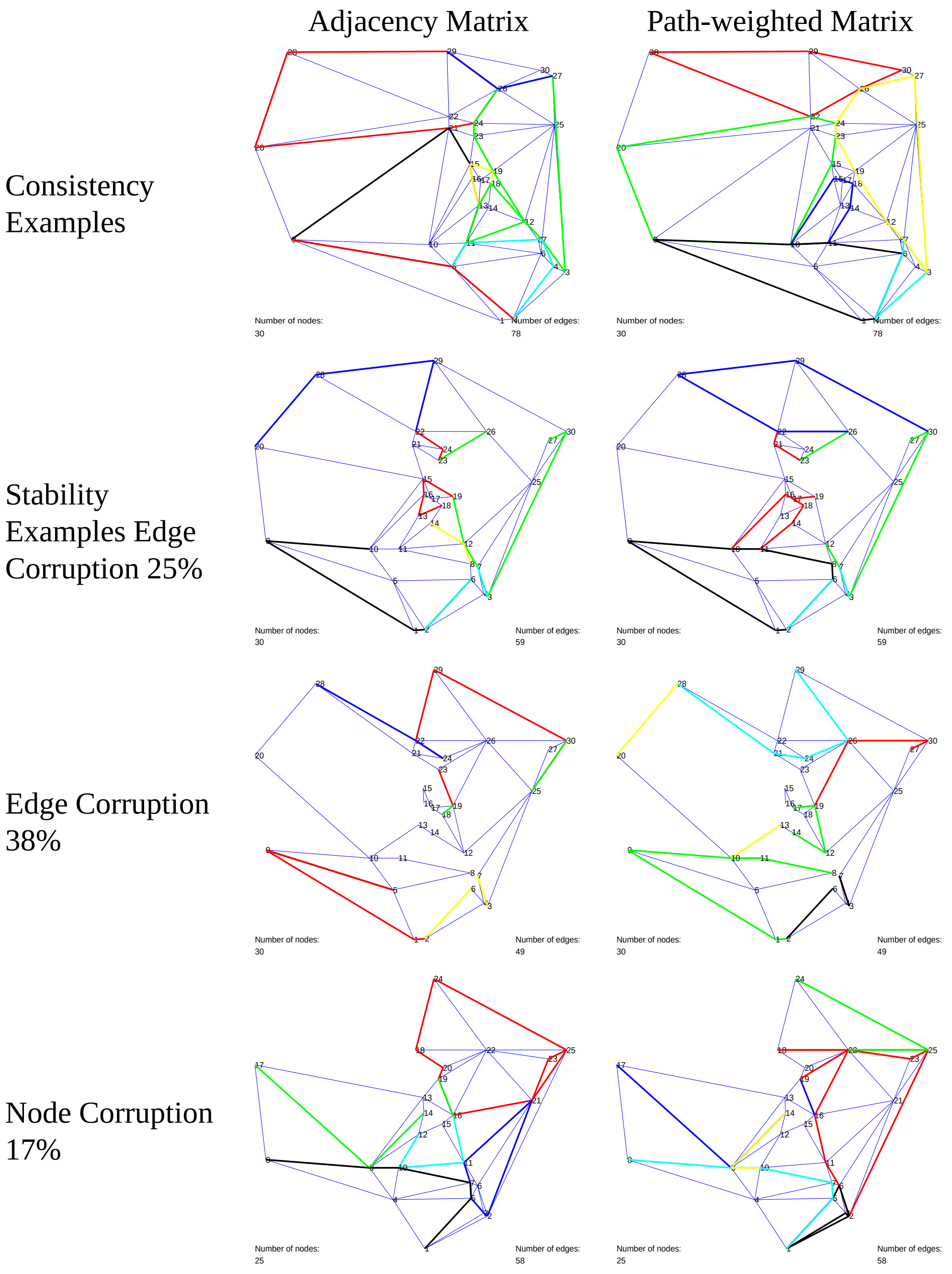


Figure 3.21: Examples of the partitions.

MOVI and Swiss Chalet. CMU and MOVI house sequences are obtained from CMU database [1] and INRIA database [2] respectively. The Chalet house sequence was captured at York.

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

[2]http://www.irisa.fr/texmex/baseimages/

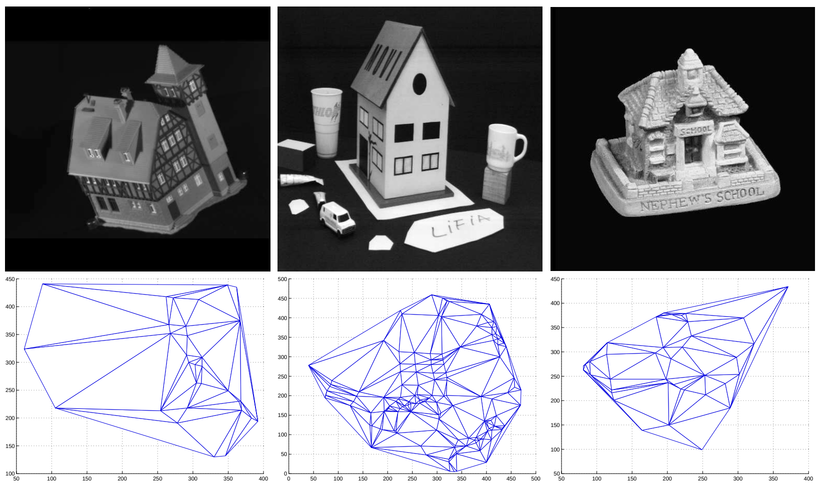


Figure 3.22: Example images from the CMU, MOVI and chalet sequences and their corresponding graphs.

In Figure 3.23 the two panels show the distances $d(k_1, k_2) = (B_{k_1} - B_{k_2})^T (B_{k_1} - B_{k_2})$ between the vectors of eigenvalues for the graphs indexed k_1 and k_2. The left panel is for the original graph and the right panel is for the simplified graph. It is clear that the simplification process has preserved much of the structure in the distance plot. For instance, the three sequences are clearly visible as blocks in the panels. Figure 3.24 shows a scatter plot of the distance between the simplified graphs (y-axis) as a function of the distance between the original graphs. Although there is considerable dispersion, there is an underlying linear trend.

Figures 3.25 and 3.26 repeat the distance matrices and the scatter plot using edit distance rather than the L2 norm for the spectral feature vectors. Again, there is a clear block structure. However, the dispersion in the scatter plot is greater. To take this study one step further, in Figures 3.27 and 3.28 we show the result of performing MDS on the distances for both the edit distance and the spectral feature vector. Here the images from which the graphs are extracted are shown as thumbnails embedded in the space spanned by the leading eigenvectors of the MDS analysis. In both cases the views of the different houses fall into distinct

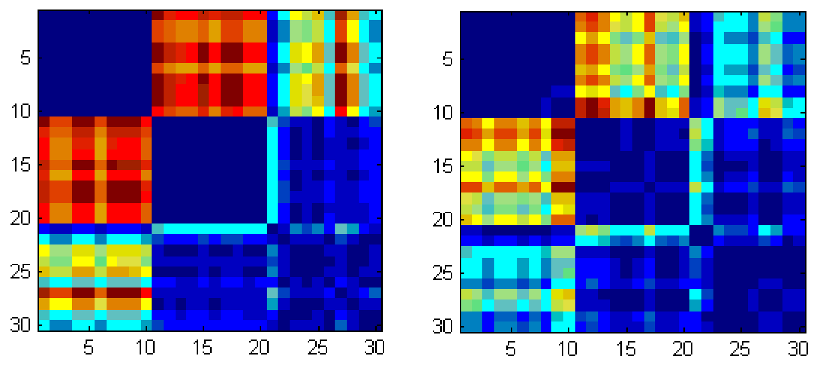


Figure 3.23: Pairwise spectral graph distance; (left) original graph, (right) reduced graph.

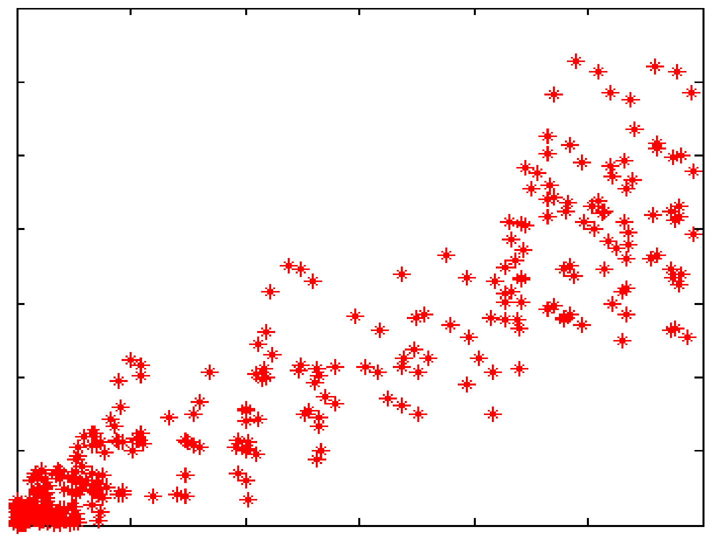


Figure 3.24: Scatter plot for the original graph and reduced graph pairwise distance.

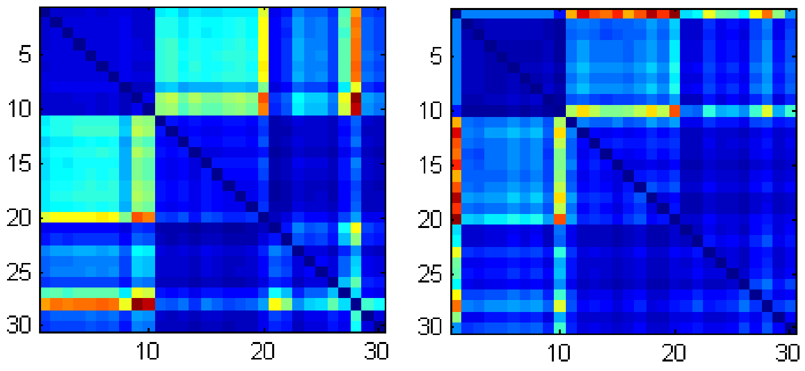


Figure 3.25: Graph edit distance; (left) original graph, (right) reduced graph

regions of the plot. Moreover, the hierarchical simplification of the graphs does not destroy the cluster structure.

3.7 Conclusions

In this chapter, we have used the Fiedler vector of the Laplacian matrix to partition the nodes of a graph into super-cliques for the purposes of matching. This allows us to decompose the problem of matching the graphs into that of matching structural subunits, the super-cliques. We investigate the matching of the structural subunits using a edit distance method. The partitioning method is sufficiently stable under structural error that accuracy of match is not sacrificed. Our motivation in undertaking this study is to use the partitions to develop a hierarchical matching method. The aim is to construct a graph that represents the arrangement of the partitions. By first matching the partition arrangement graphs, we provide constraints on the matching of the individual partitions.

Focusing the aim of developing a more robust graph representation, we have shown how ideas from the spectral theory of the heat kernel can be used to construct a path-weighted proximity matrix. We show how the heat-kernel can be

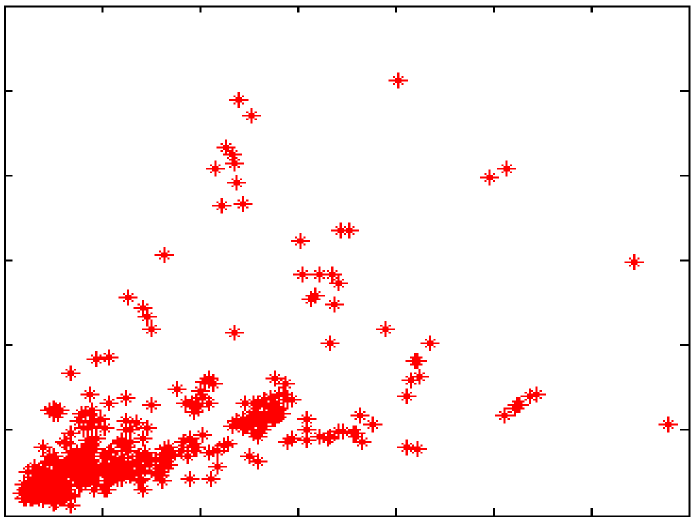


Figure 3.26: Scatter plots for the original graph and reduced graph edit distance

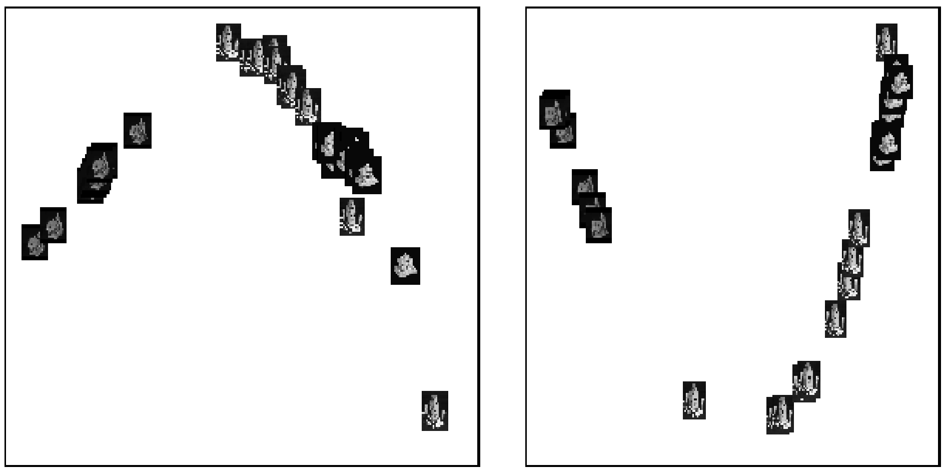


Figure 3.27: MDS for the original graph (left) edit distance, (right)spectral feature vector

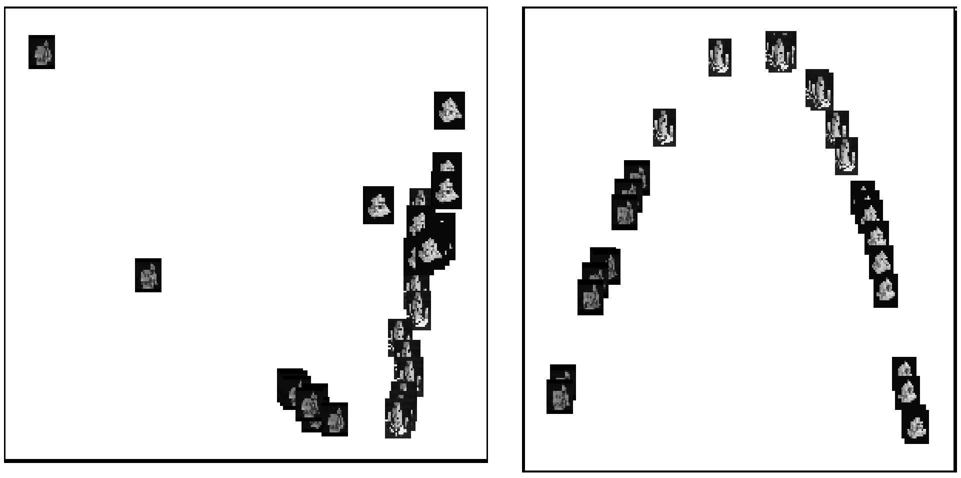


Figure 3.28: MDS for the reduced graph (left) edit distance, (right)spectral feature vector

used to compute the path weight distribution on the graph. The distribution is used to compute the mean and variance of the path length between pairs of nodes. Our path weighted proximity matrix is computed by exponentiating the squared mean-distance. We have studied the properties of the path weighted proximity matrix. This study shows that it gives us more stable representation of graph-structure under structural error.

Chapter 4

Commute Time

Commute time is a concept first introduced to study random walks in the graph (Desai and Rao, 1993; Aldous and Fill, 2003). The quantity measures the time taken for random walk from one node to another and back again. Commute time has a close relationship with spectral graph theory (Chung and Yau, 2000). We first show how commute time is related to other important concepts in spectral graph theory and how it can be computed in a spectral manner. Then we focus on the commute time preserving embedding, which embeds a graph into a subspace where the Euclidean distance between a pair of points is equal to the commute time value of the corresponding nodes in the original graph. Commute time embedding is also akin to some other classic embedding methods such as PCA, the Laplacian map and the diffusion map. We will show that our commute time embedding is related to these methods. Finally, based on an analysis of the clustering properties of commute time, we will show how it can be effectively applied to the clustering problem and why it could be superior to the *normalised cut*.

4.1 Spectral Affinity

In this section, we review the theory underpinning the computation of commute time. We commence by showing what is the Green's function and how it is related to the heat kernel and how it can be computed from the Laplacian spectrum. Then, we will show that the commute time is a metric that is obtained from the Green's function.

4.1.1 Green's Function

Now consider the discrete Laplace operator $\Delta = T^{-1/2}\mathcal{L}T^{1/2}$. The Green's function is the left inverse operator of the Laplace operator Δ, defined by

$$G\Delta(u,v) = I(u,v) - \frac{d_v}{vol}$$

Where $vol = \sum_{v\in V} d_v$ is the volume of the graph and I is the $|V| \times |V|$ identity matrix. 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. While the external node is connected by edges with the nodes on the boundary of the graph. The Green's function of the graph is related to the heat kernel $\mathcal{H}_t$ and has element given by

$$G(u,v) = \int_0^\infty d_u^{1/2} \left(\mathcal{H}_t(u,v) - \phi_1'(u)\phi_1'(v)\right) d_v^{-1/2} dt \tag{4.1}$$

where 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 $\mathcal{G} = T^{-1/2}GT^{1/2}$ is given in terms of the normalised Laplacian spectrum (see (Chung and Yau, 2000) page 6) as

$$\mathcal{G}(u,v) = \sum_{i=2}^{|V|} \frac{1}{\lambda'_i}\phi'_i(u)\phi'_i(v) \tag{4.2}$$

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 $\bar{G}$ 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 $\mathcal{G}\mathcal{L} = \mathcal{L}\mathcal{G} = I - \phi'_1\phi'^T_1$, and as a result $(\mathcal{L}\mathcal{G})(u,v) = \delta(u,v) - \frac{\sqrt{d_u d_v}}{vol}$. From (4.2), 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 (Chung and Yau, 2000) page 4), it follows that $\mathcal{G}$ is a kernel. The same applies to the *un-normalized* Green's function $\bar{G}$.

The relationship between G, $\bar{G}$ and $\mathcal{G}$ can be obtained if we consider an induced subgraph Γ_S of the original graph Γ. If Γ_S is connected, Δ, L and $\mathcal{L}$ are nonsingular (see (Chung, 1997)) and we have $G\Delta = \bar{G}L = \mathcal{G}\mathcal{L} = I$. From the fact that $\Delta = T^{-1/2}\mathcal{L}T^{1/2}$ and $\mathcal{L} = T^{-1/2}LT^{-1/2}$, then $\Delta = T^{-1}L$. As a result we have $GT^{-1}L = \bar{G}L$ and as a consequence $\bar{G} = GT^{-1}$. Making use of the fact that $\mathcal{G} = T^{-1/2}GT^{1/2}$, we then obtain

$$\bar{G} = T^{-1/2}\mathcal{G}T^{-1/2} \tag{4.3}$$

4.1.2 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 (Chung and Yau, 2000)

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

where G is the Green's function given in equation 4.1. So, the commute time is given by

$$\begin{aligned} CT(u,v) &= O(u,v) + O(v,u) \\ &= \frac{vol}{d_u} G(u,u) + \frac{vol}{d_v} G(v,v) - \frac{vol}{d_u} G(u,v) - \frac{vol}{d_v} G(v,u) \end{aligned} \tag{4.4}$$

or using *un-normalised* Green's function, as

$$CT(u,v) = vol\left(\bar{G}(u,u) + \bar{G}(v,v) - 2\bar{G}(u,v)\right) \tag{4.5}$$

As a consequence of Equation 4.4 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 *normalised* Laplacian

$$CT(u,v) = vol \sum_{i=2}^{|V|} \frac{1}{\lambda'_i} \left(\frac{\phi'_i(u)}{\sqrt{d_u}} - \frac{\phi'_i(v)}{\sqrt{d_v}} \right)^2 \tag{4.6}$$

On the other hand, performing an eigen-decomposition on both sides of Eq. (4.3):

$$\begin{aligned}\Phi\Lambda^{-1}\Phi^{T} &= T^{-1/2}\Phi'\Lambda'^{-1}\Phi'^{T}T^{-1/2} \\ &= (T^{-1/2}\Phi')\Lambda'^{-1}(T^{-1/2}\Phi')^{T}\end{aligned} \tag{4.7}$$

It follows that $\Lambda^{-1} = \Lambda'^{-1}$ and $\Phi = T^{-1/2}\Phi'$. Substituting these relationships between the eigensystems into Eq. (4.6), the commute time can be expressed in terms of the eigen-system of the *un-normalized* Laplacian.

$$CT(u,v) = vol\sum_{i=2}^{|V|}\frac{1}{\lambda_i}(\phi_i(u)-\phi_i(v))^2 \tag{4.8}$$

4.2 Commute Time Embedding

Commute time embedding is a mapping from the data space into a Hilbert subspace that keeps the original commute time value. It has some properties similar to alternative embedding methods such as PCA, the Laplacian eigenmap and the diffusion map. In this section, we will first introduce the principles of commute time embedding and then we will compare it to alternative embedding methods. Some embedding examples are illustrated and the robustness of embedding is also discussed.

4.2.1 Basics

Equation 4.6, can be re-written in the following form which makes the relationship between the commute time and the Euclidean distance more explicit

$$CT(u,v) = \sum_{i=2}^{|V|} \left(\sqrt{\frac{vol}{\lambda'_i d_u}} \phi'_i(u) - \sqrt{\frac{vol}{\lambda'_i d_v}} \phi'_i(v) \right)^2 \tag{4.9}$$

Given two points $\mathbf{x}_u$ and $\mathbf{x}_v$ in a R^n space, their squared Euclidean distance can be computed as $\sum_{i=1}^{n} \left(x_u(i) - x_v(i)\right)^2$, where $x_u(i)$ is the cor-ordinate of $\mathbf{x}_u$ on the i-th axis. As a result, from Equation 4.9, $\sqrt{\frac{vol}{\lambda'_i d_u}} \phi'_i(u)$ can be taken as the i-th co-ordinate of node u in the commute time embedded subspace. Therefore, the embedding of the nodes of the graph into a vector space that preserves commute time has the co-ordinate matrix

$$\Theta = \sqrt{vol} \Lambda'^{-1/2} \Phi'^T T^{-1/2} \tag{4.10}$$

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 normalisation of the Laplacian. If the commute time is computed from the un-normalised Laplacian, the corresponding matrix of embedding co-ordinates is

$$\Theta = \sqrt{vol} \Lambda^{-1/2} \Phi^T \tag{4.11}$$

The embedding is nonlinear in the eigenvalues of the Laplacian. This distinguishes it from principle components analysis (PCA) and locality preserving projection (LPP) (He and Niyogi, 2003) which are both linear. As we will demonstrate in the next section, the commute time embedding is just kernel PCA (Scholkopf et al., 1998) on the Green's function. Moreover, it can be viewed as Laplacian eigenmap since it minimises the same objective function.

4.2.2 The Commute Time Embedding and Kernel PCA

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

$$C_f = \Theta\Theta^T = vol\Lambda^{-1/2}\Phi^T\Phi\Lambda^{-1/2} = vol\Lambda^{-1} = vol\Lambda_{\bar{G}} \tag{4.12}$$

where $\Lambda_{\bar{G}}$ is the eigenvalue matrix of *un-normalised* Green's function with decreasingly ordered eigenvalues. The kernel or Gram matrix is given by the inner product of the co-ordinates matrix with itself

$$K = \Theta^T\Theta = vol\Phi\Lambda^{-1/2}\Lambda^{-1/2}\Phi^T = vol\Phi\Lambda^{-1}\Phi^T = vol\bar{G} \tag{4.13}$$

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. Actually, K being a kernel is inevitable since we have defined the commute time as an equivalent distance measure to Euclidean distance in Equation 4.9.

4.2.3 The Commute Time Embedding and the Laplacian Eigenmap

In the Laplacian eigenmap (Belkin and Niyogi, 2003; Belkin and Niyogi, 2001) 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 minimises the objective function

$$\epsilon = \sum_{u,v} \|\mathbf{z}_u - \mathbf{z}_v\|^2 \, \Omega(u,v) = tr(\mathbf{Z}^T L \mathbf{Z}) \tag{4.14}$$

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

$$\mathbf{Z} = \arg \min_{\mathbf{Z}^{*T} T \mathbf{Z}^* = I} tr(\mathbf{Z}^{*T} L \mathbf{Z}^*) \tag{4.15}$$

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

$$L\mathbf{Z} = \Lambda' T \mathbf{Z} \tag{4.16}$$

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

As we will show later on in Equation 4.21, the objective function minimized by the normalized cut can also be given by

$$\epsilon' = \frac{\sum_{u,v} \|\mathbf{z}_u - \mathbf{z}_v\|^2 \, \Omega(u,v)}{\sum_u \mathbf{z}_u^2 d_u} = tr(\frac{\mathbf{Z}^T L \mathbf{Z}}{\mathbf{Z}^T T \mathbf{Z}}) \tag{4.17}$$

Here we argue that although the objective function that the commute time embedding optimises is still unknown, we can achieve the same minimized score ϵ^* as the Laplacian eigenmap using Equ. 4.17. To show this, let $\mathbf{Z} = \Theta^T =$

$(\sqrt{vol}\Lambda'^{-1/2}\Phi'^{T}T^{-1/2})^{T}$, then we have

$$\begin{aligned}\epsilon' &= tr(\frac{\sqrt{vol}\Lambda'^{-1/2}\Phi'^{T}T^{-1/2}LT^{-1/2}\Phi'\Lambda'^{-1/2}\sqrt{vol}}{\sqrt{vol}\Lambda'^{-1/2}\Phi'^{T}T^{-1/2}TT^{-1/2}\Phi'\Lambda'^{-1/2}\sqrt{vol}}) \\ &= tr(\frac{\Lambda'^{-1/2}\Phi'^{T}\mathcal{L}\Phi'\Lambda'^{-1/2}}{\Lambda'^{-1/2}\Phi'^{T}\Phi'\Lambda'^{-1/2}}) \\ &= tr(\frac{\Lambda'^{-1/2}\Lambda'\Lambda'^{-1/2}}{\Lambda'^{-1}}) \\ &= tr(\Lambda') = \epsilon^{*}\end{aligned} \tag{4.18}$$

Hence, the commute time embedding not only aims to maintain proximity relationships by minimising $\sum_{u,v}\|\mathbf{z}_u - \mathbf{z}_v\|^2\,\Omega_{uv}$, but it also aims to assign large co-ordinate values to nodes (or points) with large degree (i.e. it maximises $\sum_u \mathbf{z}_u^2 d_u$). Nodes with large degree are the most significant in a graph since they have the largest number 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.

4.2.4 The Commute Time and the Diffusion Map

Finally, it is interesting to note the relationship with the diffusion map embedding of Coifman *et al* (Coifman et al., 2005). The method commences from the random walk on a graph which has transition probability matrix $P = T^{-1}\Omega$, where Ω is the adjacency matrix. Although P is not symmetric, it does have a right eigenvector matrix Ψ, which satisfies the equation

$$P\Psi = \Lambda_P\Psi \tag{4.19}$$

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

$$\begin{aligned} (I - T^{-1}L)\Psi &= \Lambda_P \Psi \\ T^{-1}L\Psi &= (I - \Lambda_P)\Psi \\ L\Psi &= (I - \Lambda_P)T\Psi \end{aligned} \tag{4.20}$$

which is identical to Equation (4.16) if **Z** $= \Psi$ and $\Lambda' = I - q\Lambda_P$. 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, 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 behaviour 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 the original definition of the diffusion distance for a random walk can be given by

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

As a result, the distance between a pair of nodes depends on the transition probability between the nodes under consideration and 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 embedding.

4.2.5 Some Embedding Examples

Figure 4.1 shows four synthetic examples of point-configurations. These points are located in the original Euclidean space and color coded to indicate which cluster they belong to. We then computed the proximity weight matrix Ω by exponentiating the Euclidean distance between points. Their corresponding embeddings in the commute time embedded space is shown in Figure 4.2. Here the co-ordinates in the commute time embedded space is computed by Equ. 4.10 and we take the first three columns as axes.

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) and a full explanation can be found in Ng's paper (Ng et al., 2001).

4.2.6 Robustness of the Commute Time Embedding

From Equation (4.11) 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 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

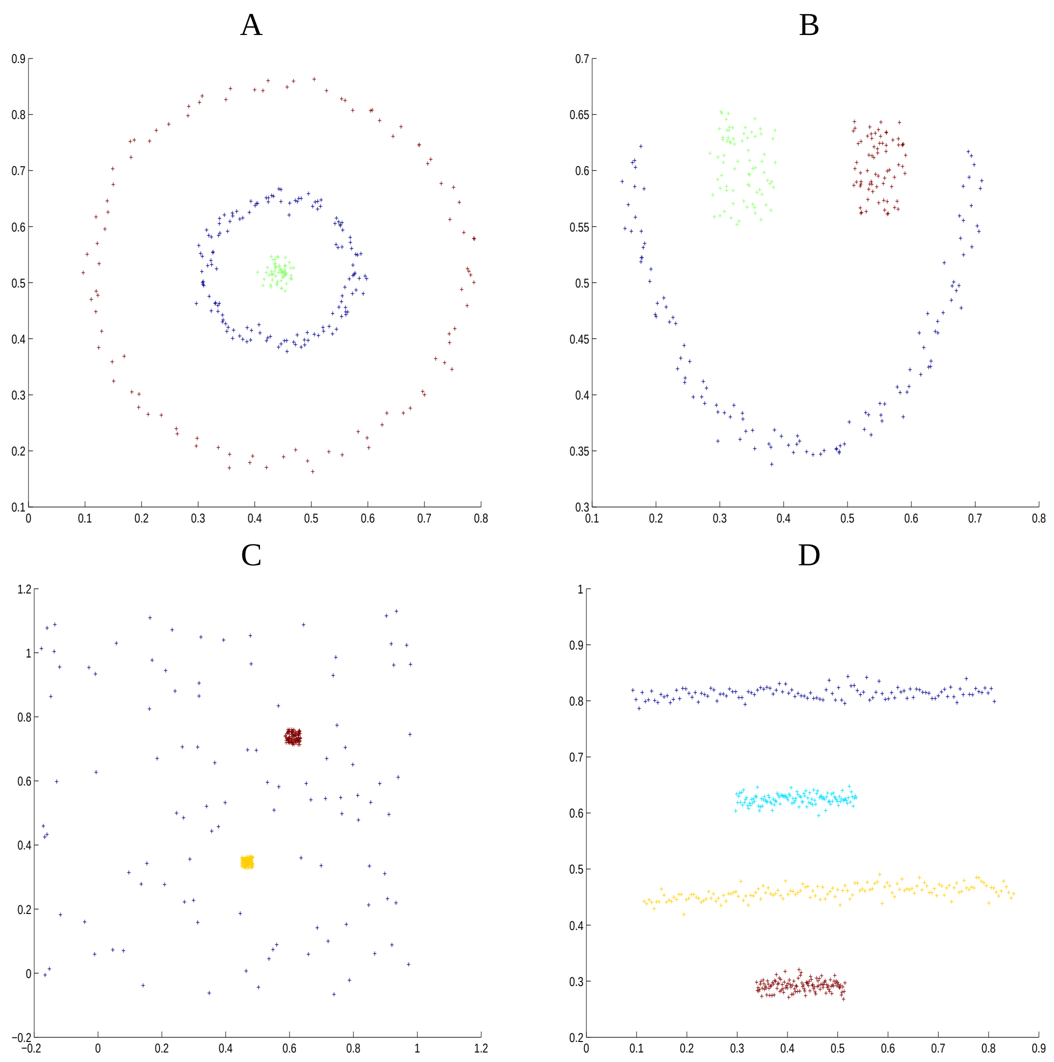


Figure 4.1: Four sets of data points in their original space. Here in each set, points belonging to the same cluster are coded with the same color.

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. Particularly, in the application of motion tracking, this property

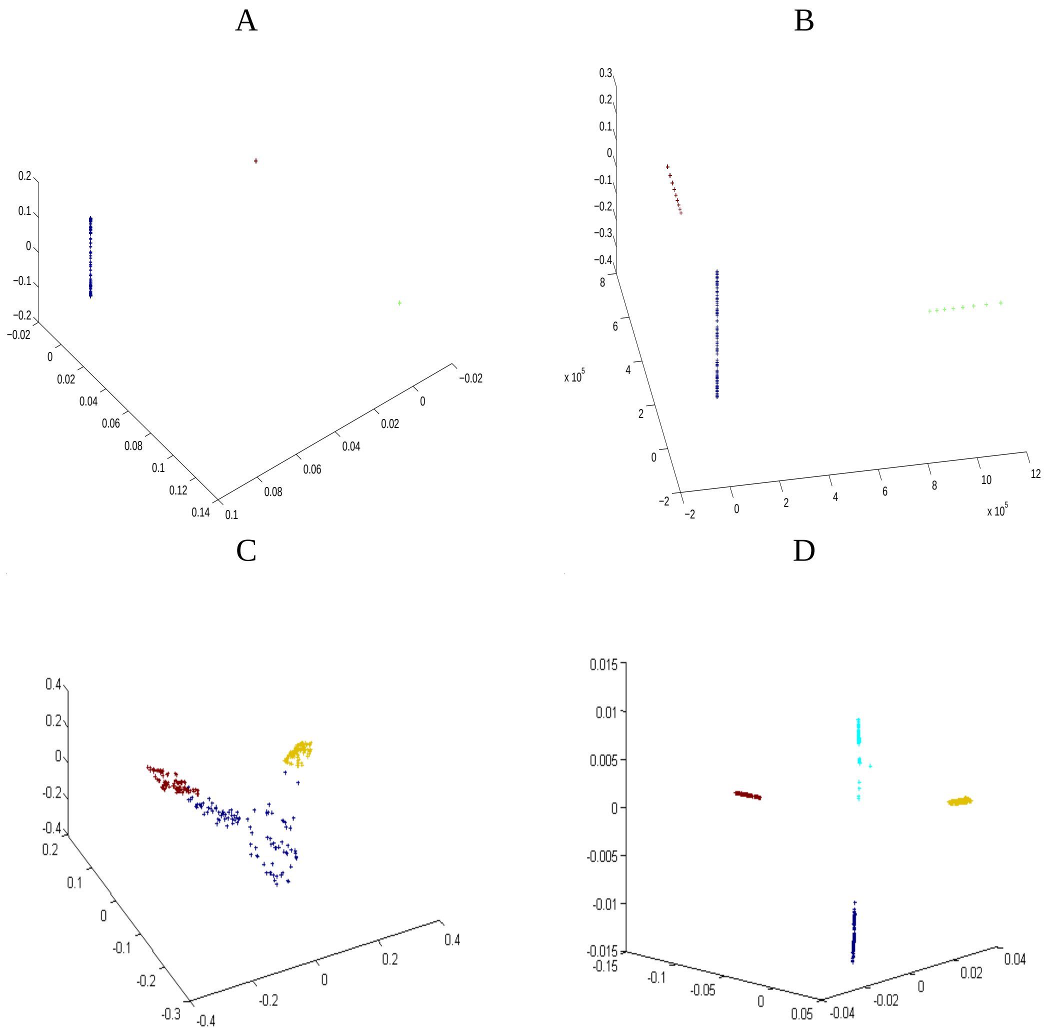


Figure 4.2: The corresponding four sets of data points in the commute time embedded space. Color pattern is the same as Figure 4.1.

reduces the impact of outliers, since once embedded, outliers will be excluded from the object point-clusters.

4.3 Commute Time Properties for Grouping

In this section, we will compare commute time embedding with the normalised cut.

4.3.1 Commute Time Properties

Commute time has the following properties:

- The points embedded in the subspace are allocated along its principle axes;
- The close or similar points are embedded close to each other;
- Large degree points are allocated far from the origin;
- The original commute time distance is preserved. This means that the Euclidean distance in the embedded subspace preserves the properties from the original commute time distance. This means a pair of nodes will be close in the embedded subspace if they are connected and satisfy the following:
  - They are close together, i.e. the length of the path between them is small;
  - The paths connecting them have a small sum of weights;
  - They are connected by many paths;

4.3.2 Comparison with the Normalised Cut

Here we argue that the normalised cut is the separation of the axis projection of the points in commute time embedded subspace.

From the previous section on commute time embedding, we make a number of observations. First, we observe the objective function(Equation 4.17) minimised is exactly that minimised by the normalised cut in (Shi and Malik, 2000)(see page 9(10)). To show this let $\vec{\theta}$ be an $N = |V|$ dimensional binary indicator vector, which determines to which component of the bi-partition a node belongs. The minimum value obtained by the normalized cut (Shi and Malik,

2000) is

$$\vec{\theta}_1 = \arg \min_{\vec{\theta}^T \mathbf{T} \mathbf{1} = 0} \frac{\vec{\theta}^T (\mathbf{T} - \Omega) \vec{\theta}}{\vec{\theta}^T \mathbf{T} \vec{\theta}} \tag{4.21}$$

From Equation 4.17 it is clear that the both methods achieve the same minimisation and use the same eigenvectors as solutions. The only difference is that the eigenvectors used in the commute time embedding are scaled by the reciprocal 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 Equation 4.8, 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 normalised cut to bipartition the graphs recursively.

Turning our attention to the commute time embedding, here the scaled eigenvectors are used as the projection axes for the data. As a result, if we project the data into the commute time embedding subspace, the normalised 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.

4.3.3 Why Commute Time Clustering is Successful

The *normalised cut* method (Shi and Malik, 2000) seeks the bi-partition that simultaneously maximises intra-cluster association and minimises inter-cluster edge linkage. However, this problem is NP-hard and only a relaxed approximation can be found, and this is given by the Fiedler vector. As a result, the more discrete the distribution of the components in Fiedler vector, the closer the relaxed solution to the exact one. In a bipartition, if the components in the Fiedler vector take on only two distinct values, the Fiedler vector will become the exact solution and two partitions are well separated. Meilă and Shi (Meilă and Shi, 2000) extend the bipartition normalised cut to the multi-partitions case. and they called this nearly discrete eigenvector pair-wise constant. Turning our attention to the commute time, from Equation 4.8, it is clear that if all the eigenvectors are pair-wise constant, the points belonging to the same cluster will have a zero commute time and those belonging to different clusters will have a large value. This further proves that commute time can be taken as a measure of data cohesion.

The only way to obtain pair-wise constant eigenvectors is to have a block diagonal affinity matrix. This has been discussed extensively in the literature (Meilă and Shi, 2000; Ng et al., 2001; Weiss, 1999). Ng et al (Ng et al., 2001) use tools from matrix perturbation theory to analyse spectral clustering methods. The ”ideal” case in their model is to have a pure block diagonal affinity matrix. Weiss (Weiss, 1999) has shown the data must be normalised in order to obtain a more block-diagonal affinity matrix, if the original matrix has no constant blocks. If this is not the case methods such as Perona and Freeman’s algorithm (Perona and Freeman, 1998), Shi and Malik’s normalised cut method (Shi and Malik, 2000) and Scott and Longuet-Higgins algorithm (Scott and Longuet-Higgins, 1990) will not succeed. Hence, what determines the quality of the clustering is not a better cut- criteria, but an improved block structure in the affinity matrix. The block structure can be enhanced by the commute time as shown by Fischer and Poland (Fischer and Poland, 2005). Here, a new affinity measure based on graph conductivity is introduced so as to quantify cluster memberships. This graph conductivity measure is equivalent to the commute time.

Since commute time can amplify the block structure of an affinity matrix that have a better pair-wise constant eigenvectors and hence give better clustering performance.

4.4 Conclusions

The focus of this chapter is commute time. 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 analysed the properties of the commute time embedding. This allows us to understand the links between the commute time embedding and alternative embedding methods such as Kernel PCA, The Laplacian eigenmap and the diffusion map. An interesting feature of the commute time embedding is that it maintains the maximum variance of data and at the same time groups data together. Furthermore, the commute time matrix gives us a more block-like affinity matrix and a finer data cohesion measure. A comparison with the *normalised cut* method sheds light on its properties which are used for data clustering.

Chapter 5

Commute Time Applications

In the previous chapter we have summarised the properties of commute time and explored its relationship with the Laplacian spectrum. There are four properties of the commute time that are important to us. First, as a time measurement for a random walk, the commute time is closely related to the heat equation or heat kernel. This allows us to use commute time to simulate the heat diffusion process on graphs. Secondly, commute time is also a distance metric that measures the connectivity of pairs of nodes. Its robustness to structural corruption means that it could provide a very reliable graph representation. Thirdly, based on the analysis of its grouping properties and comparing it with the normalised cut, commute time offers finer data cohesion. This means that it can be applied to data clustering problems. Finally, since commute time embedding possesses the properties of preserving the maximum data variance and proximity, it is suitable for applications requiring simultaneous dimensionality reduction and data separation.

These four properties of commute time allow us to develop four corresponding methods that can be used in computer vision. The first of these is a graph simplification method, based on a simulation of the heat diffusion process on a graph. We use them to develop two ways of representation of graphs for matching. The first of these is based on concentric layers of its graph. The second is based on the commute time minimum spanning tree. Comparing our results with those from the normalised cut, we explore the use of commute time for the image segmentation problem. Finally, we have applied our commute time embedding method to the multi-body motion tracking problem. This is realized by embedding the matrix containing object shape information into a lower dimensional space and clustering using a K-means algorithm.

The remainder of this chapter is organised as follows: In Section 5.1, we present our graph matching method based on the decomposition of graphs into concentric layers. In Section 5.2, we discuss the problem of generating stable spanning trees of graphs and elaborate on our robust tree representation using commute times. Section 5.3 compares the previous two graph matching methods on both Delaunay and K-NN graphs. In Section 5.4, we show how to use the eigenvector of the commute time matrix to recursively bipartition graphs and provide a comparison with *the normalised cut* method. In Section 5.5, we cast the multi-body objects tracking problem into our commute time embedding framework and show how objects can be separated using a simple K-means method. Finally, we provide our conclusions in Section 5.6.

5.1 Multilayer Graph Representation and Matching

The first graph simplification method is based on the concentric layers that result from repeatedly peeling away the boundary of the graph. Here, graph is restricted to planar ones. 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.

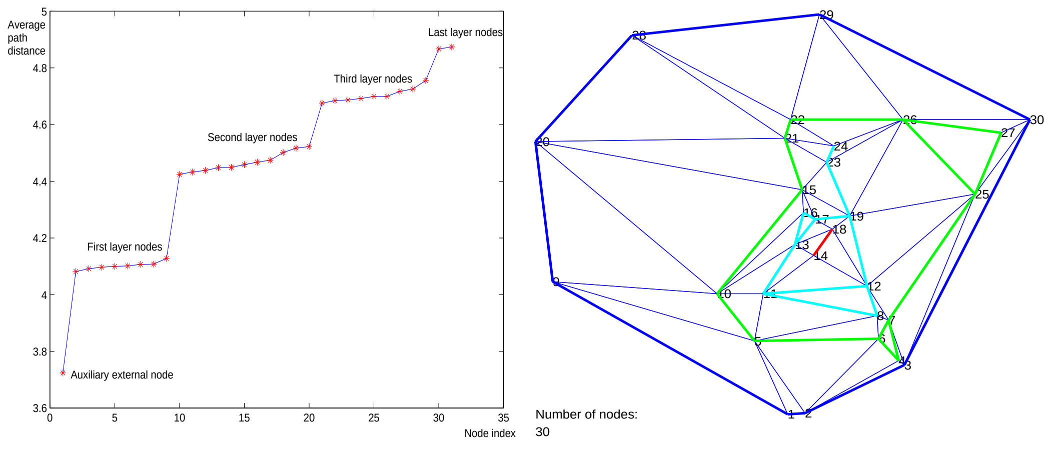
5.1.1 Graph Derivation and Representation

We commence by constructing an augmented graph from the original graph $\Gamma(V, E, \Omega)$ 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 reconsider the probability of the random walk $\mathcal{P}^l$ with a certain path length l, introduced in the last chapter. We can make an estimate of the heat flow on the graph by taking the average value of $\mathcal{P}^l$ according to the path length l:

$$\hat{d}(u, v) = \frac{\sum_l l\mathcal{P}^l(u, v)}{\sum_l \mathcal{P}^l(u, v)}.$$

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 $\hat{d}(\tau, v)$ for all v in V follows a staircase distribution, which we can use to classify nodes



(a) Staircase distribution of the average path distance.

(b) An example of a multilayer graph.

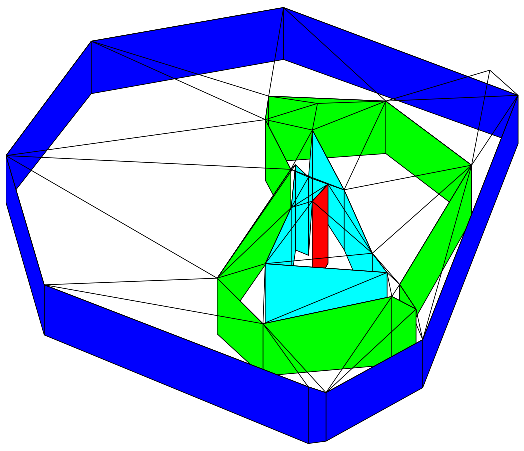
Figure 5.1: The staircase distribution and a multilayer graph.

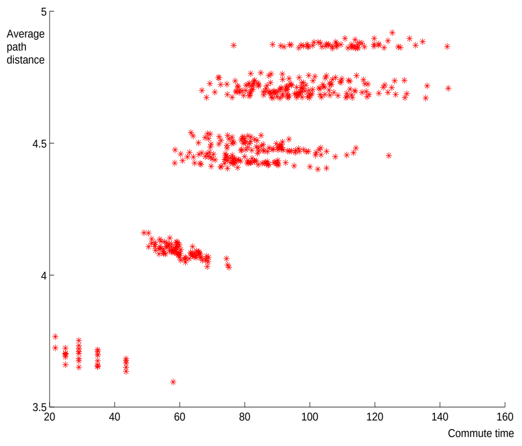
into different layers.

Figure 5.1(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 5.1(b), where the nodes connected by edges of the same colour belong to the same layer.

5.1.2 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 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.


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



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

Figure 5.2: 3D score visualisation and the scatter plot.

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 5.2(a) shows a visualisation of the score functions for the Delaunay graph in Figure 5.1(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 5.2(b) we show a scatter plot of commute times $CT(u, v)$ versus the average path length distance $\hat{d}(u, v)$. From this plot it is clear that the commute time varies more smoothly and has a longer range than the average path distance.

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_k denotes the set of nodes in the kth layer of the graph, then the permutation ρ minimises the cost function

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

5.1.3 Experiments

In this section, we carry out experiments based on our proposed multi-layer graph matching method. Firstly, we test on synthetic graphs with various sizes and then we compare our method with alternatives on the real-world data. Results show that our method is stable under structural corruption and outperform others with a considerable margin.

5.1.3.1 Synthetic Data

Our synthetic data is the same as the ones we have been using for partition matching in Chapter 3 Section 3.6.1. They are comprised of three randomly generated graphs with original nodes size 20, 40 and 60 respectively. In Figure 5.3, we show the original graphs together with their corresponding multi-layer representations. Here, different layers in each graph are coded with different colors for illustration.

To test the stability of our multi-layer representation, we corrupt the original graphs with structural error and match the corrupted graphs with the original ones. As in the previous experiments at Chapter 3, the effects of structural errors are simulated by randomly deleting nodes and re-triangulating the remaining nodes. Here we match each corrupted graph with its original one using their

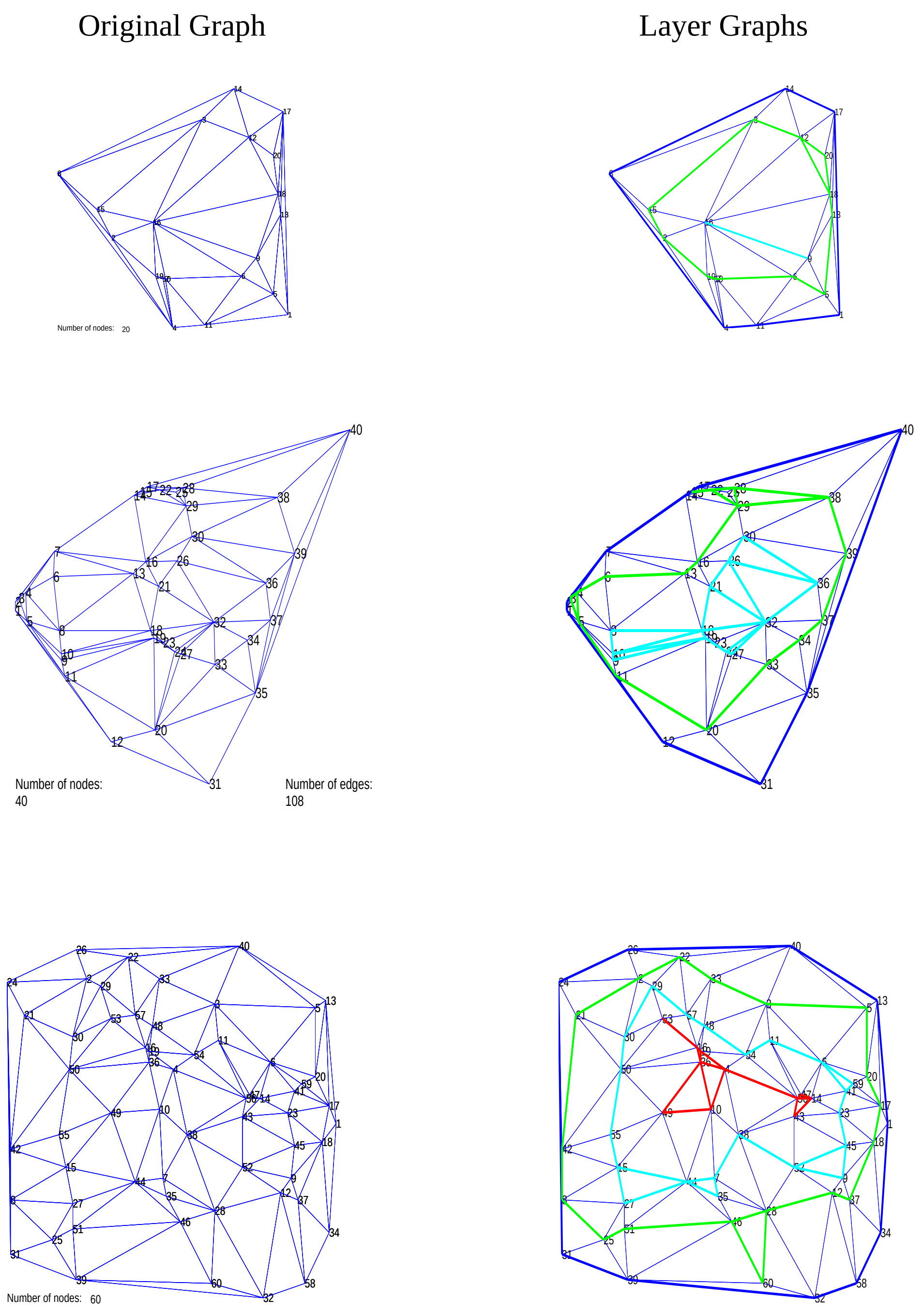


Figure 5.3: Synthetic graphs and their layered representations.

multi-layer representations. The matching result is compared with the ones from partition matching method and shown in Figure 5.4. Here the performance is based on an average of 50 trials for each graph set. From the figure, it is interesting to see that larger graphs (60 nodes in this case) give more stable performance than the middle size ones (40 nodes) and small ones (20 nodes). This is different from partition matching method. In their case, middle-size graphs give the best performance since large graphs do have the problem of finding the correct correspondencs of partitions. However, in multi-layer representation, this problem is eased by matching the corresponding layers. Moreover, each layer is a cycle graph and the matching of these cyle graphs are realitively simple and stable. As we can see from the figure, when there is no structural error (percentage of clutter equals zero), all three groups of synthetic graphs achieved 100% accurate. Finally, it is worthful to point out that multi-layer graph matching does worse than the partition method when there is severe structural corruption. This is because under that condition, graphs are corrupted so badly that graph layers are cut into pieces and unable to form stable structures.

5.1.3.2 Real-World Data

The data used in our study is furnished by a sequence of views of a model-house taken from different camera viewing directions. Similar to Section 3.6.2 in Chapter 3, we take two such sequences for our real-world data test. One of them is from CMU database (referred as CMU) and the other is from INRIA (referred as MOVI). Examples from each sequence together with their corresponding multi-layer representations are shown in Figure 5.5. In this figure, we have the original images on the top and their multi-layer graphs on the bottom. Different colors illustrate different layers. CMU house has 31 nodes and the corresponding multi-layer graph has four layers. MOVI house is larger. It has 140 nodes and as

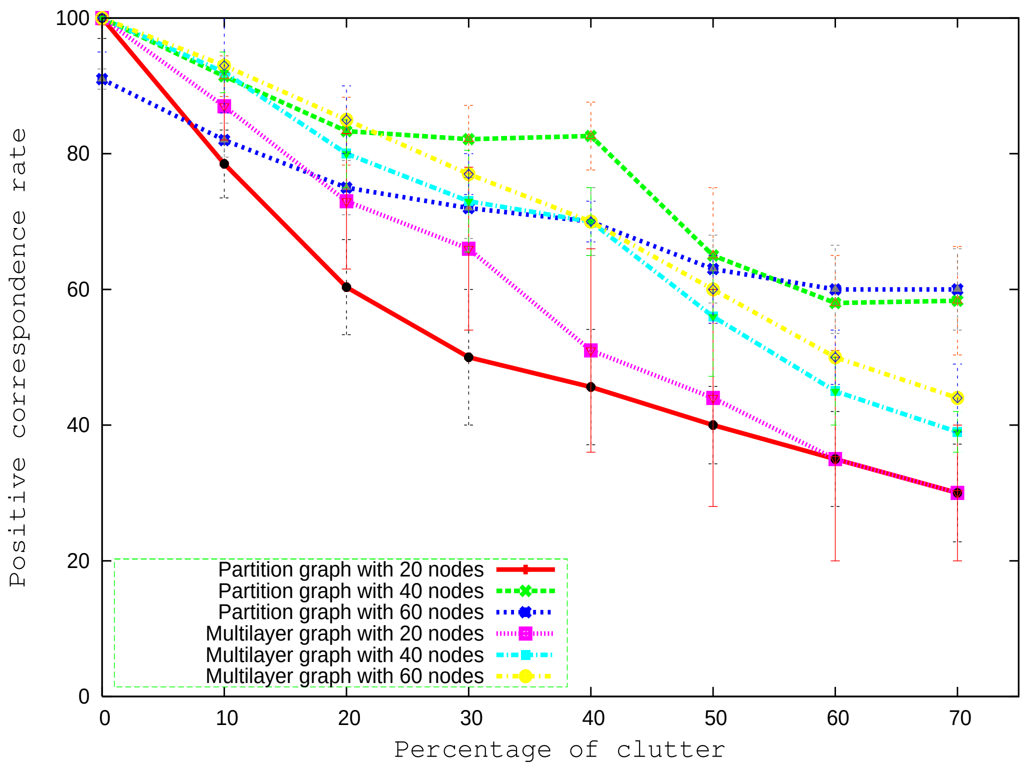


Figure 5.4: Comparison of multilayer graph matching method with partition matching method on synthetic data.

a result, it has five layers. In order to illustrate the variations of the house images as well as their multi-layer graph structure, we show the complete CMU house sequence overlayed by their multi-layer graph representations in Figure 5.7.

We have matched the first image to each of the subsequent images in the CMU sequence by using the multilayer matching method outlined earlier in this chapter. The results are compared with those obtained using the method of Luo and Hancock (Luo and Hancock, 2001) and the partition matching method of Qiu and Hancock (Chapter 3) in Table 5.1. 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. We have also compared the multilayer matching method with our partition matching method

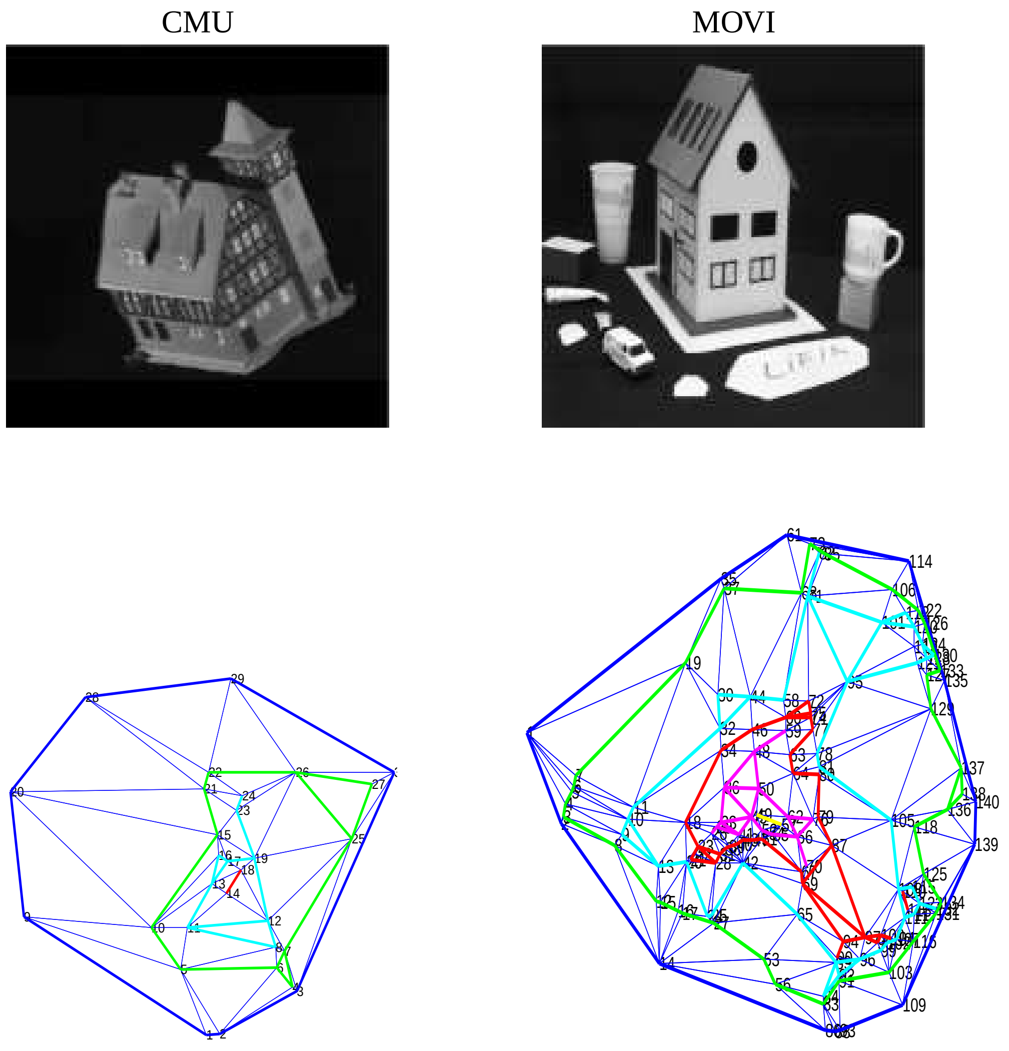


Figure 5.5: Real-world house images with their multi-layer graph representations.

on the MOVI sequence. To illustrate the results, Figure 5.6 shows 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 the partition matching method for large differences in viewing angles for the CMU house sequence. The performance of multilayer matching method on MOVI house sequence is also much better than the partition method. This is because multilayer graph matching method does not have the

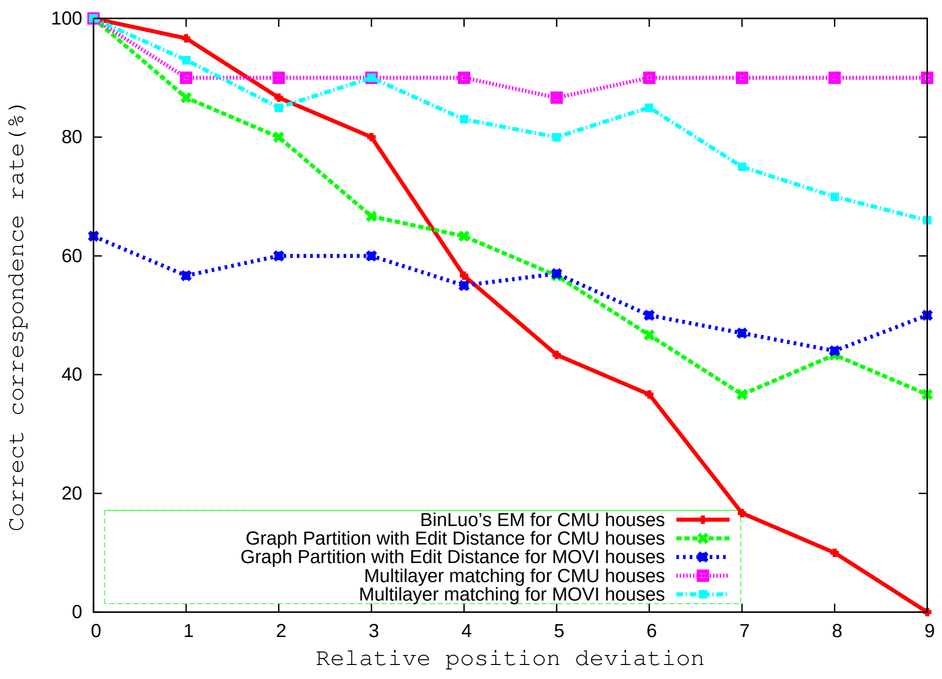


Figure 5.6: Comparison of three graph matching methods on two real-world image sets.

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	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	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 5.1: Correspondence allocation results and comparison with the methods.

problem of finding the correct corresponding layers. Nodes on the layers can be easily matched afterwards. While the partition matching method has problems of locating the correct correspondences between partitions when graphs become larger. This is important since MOVI houses are much larger than CMU ones and as a result, multilayer graph matching method could be considered one of the best to handle with large graph matching problems.

Figure 5.8 shows the results for some CMU 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%.

5.2 Minimum Spanning Tree Representation and Matching

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 this section, 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 section 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.

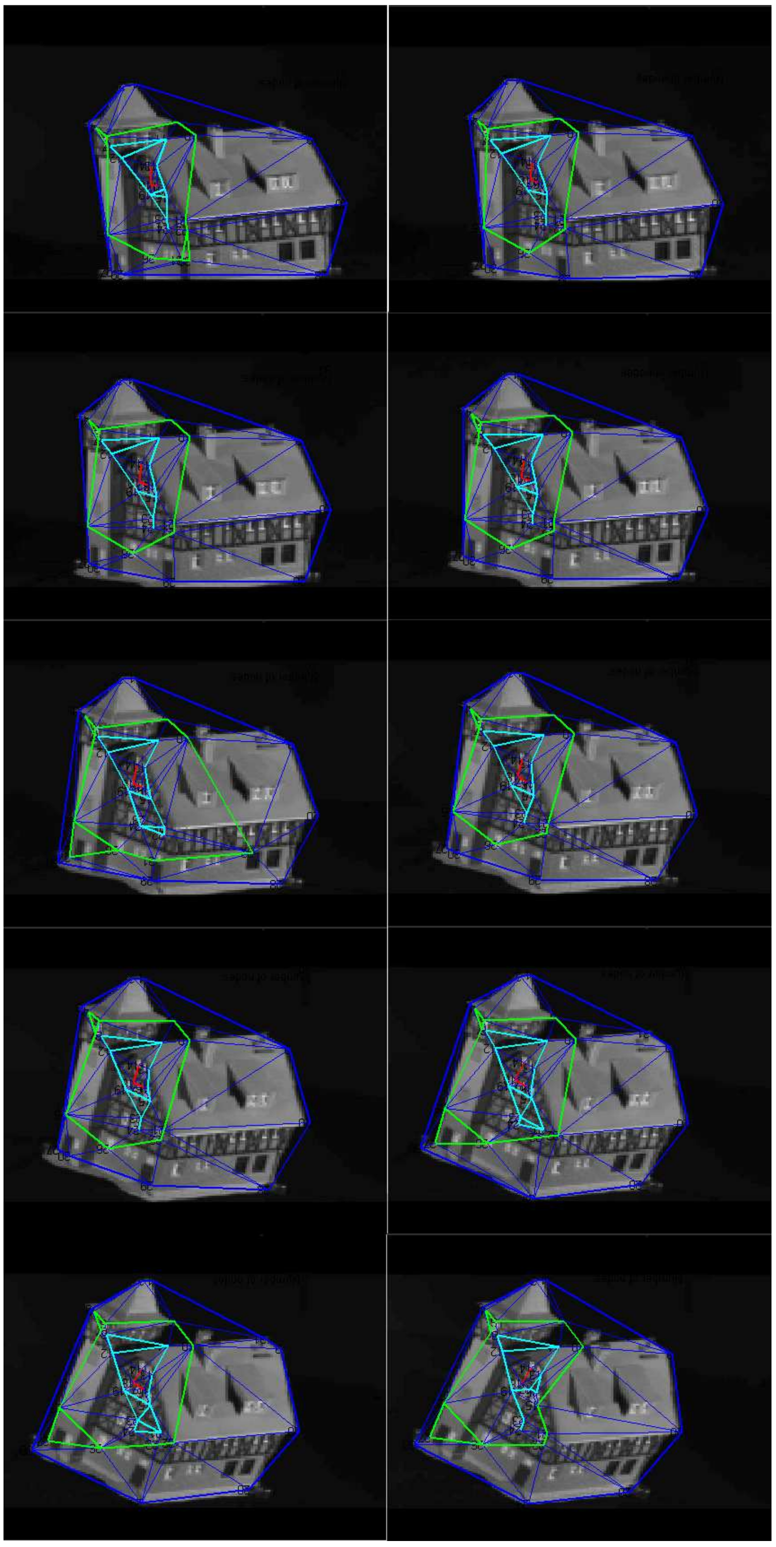
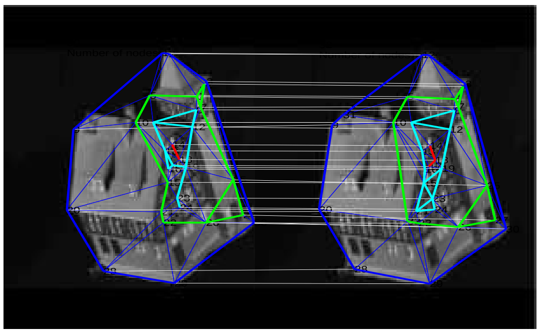
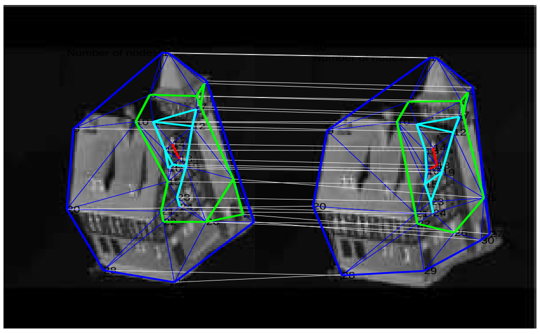


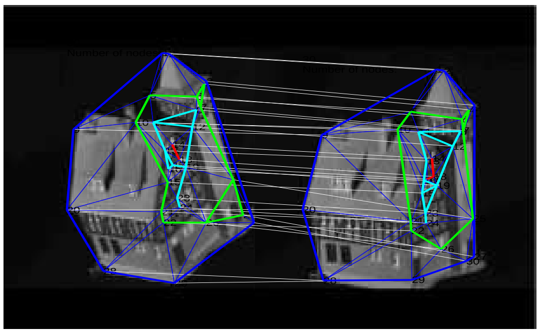
Figure 5.7: CMU house sequence.



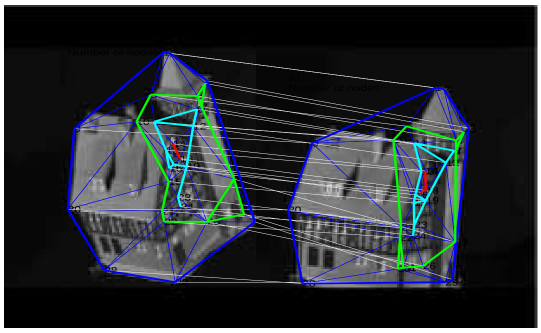
(a) 1st image to 2rd image.



(b) 1st image to 5th image.



(c) 1st image to 7th image.



(d) 1st image to 10th image.

Figure 5.8: Matched samples.

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.

5.2.1 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 Γ'. The weights of the edges in this graph are the commute-times. In another word, the weight matrix Ω of the new graph Γ' satisfies: $\Omega_{\Gamma'}(u, v) = CT(u, v)$. Our representation is based on the minimum spanning tree of the fully connected graph Γ' 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

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

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 (Prim, 1957) 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. 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 centre of a graph and has a large degree.

Two examples are shown in Figure 5.9 and Figure 5.10. In these two figures, we have shown two types of graphs. The first of these is the Delaunay 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.

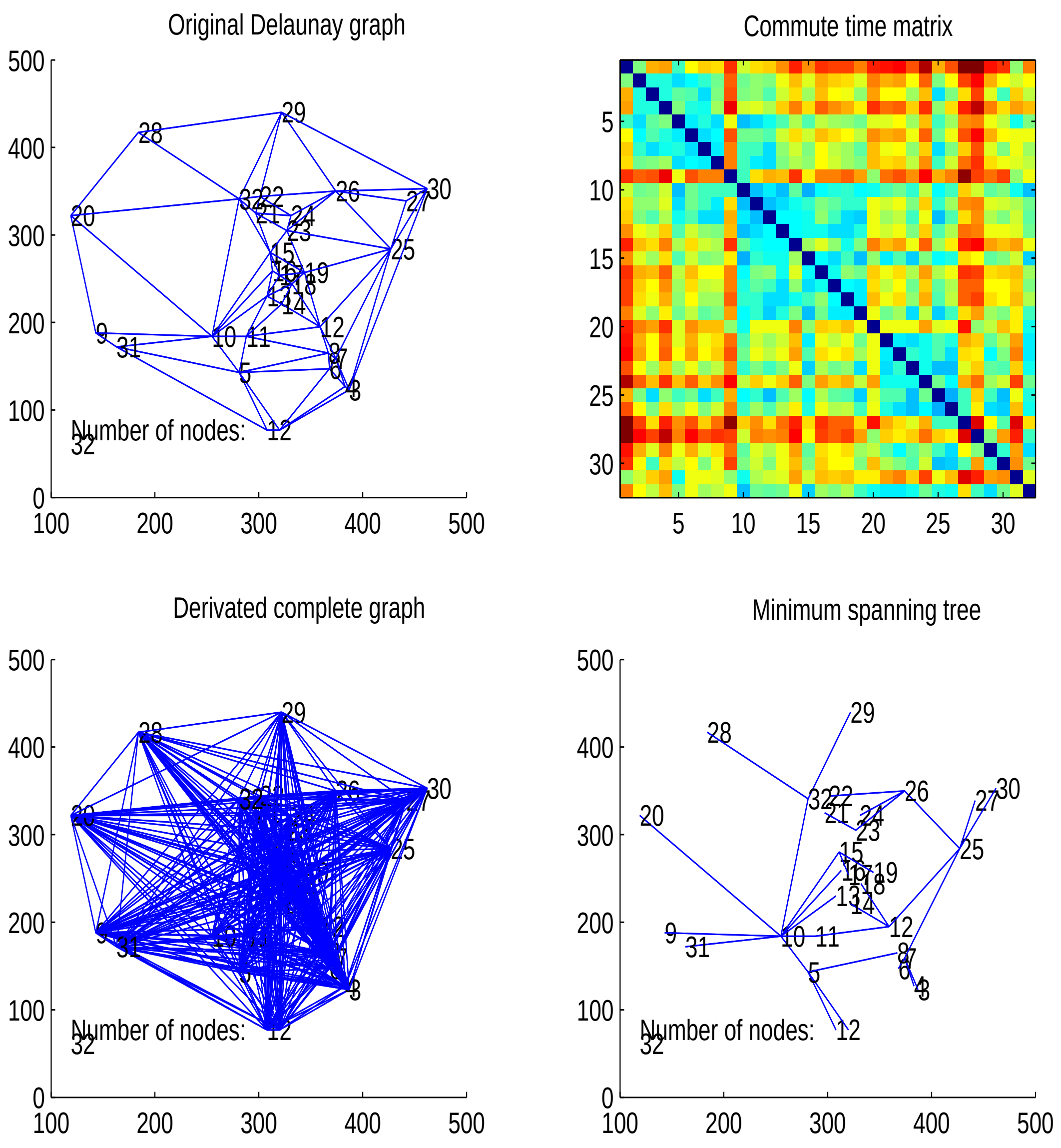


Figure 5.9: Delaunay graph example.

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 $p_t(u)$ is the probability of visiting node u of the graph under the random walk. The probability vector evolves under the equation

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

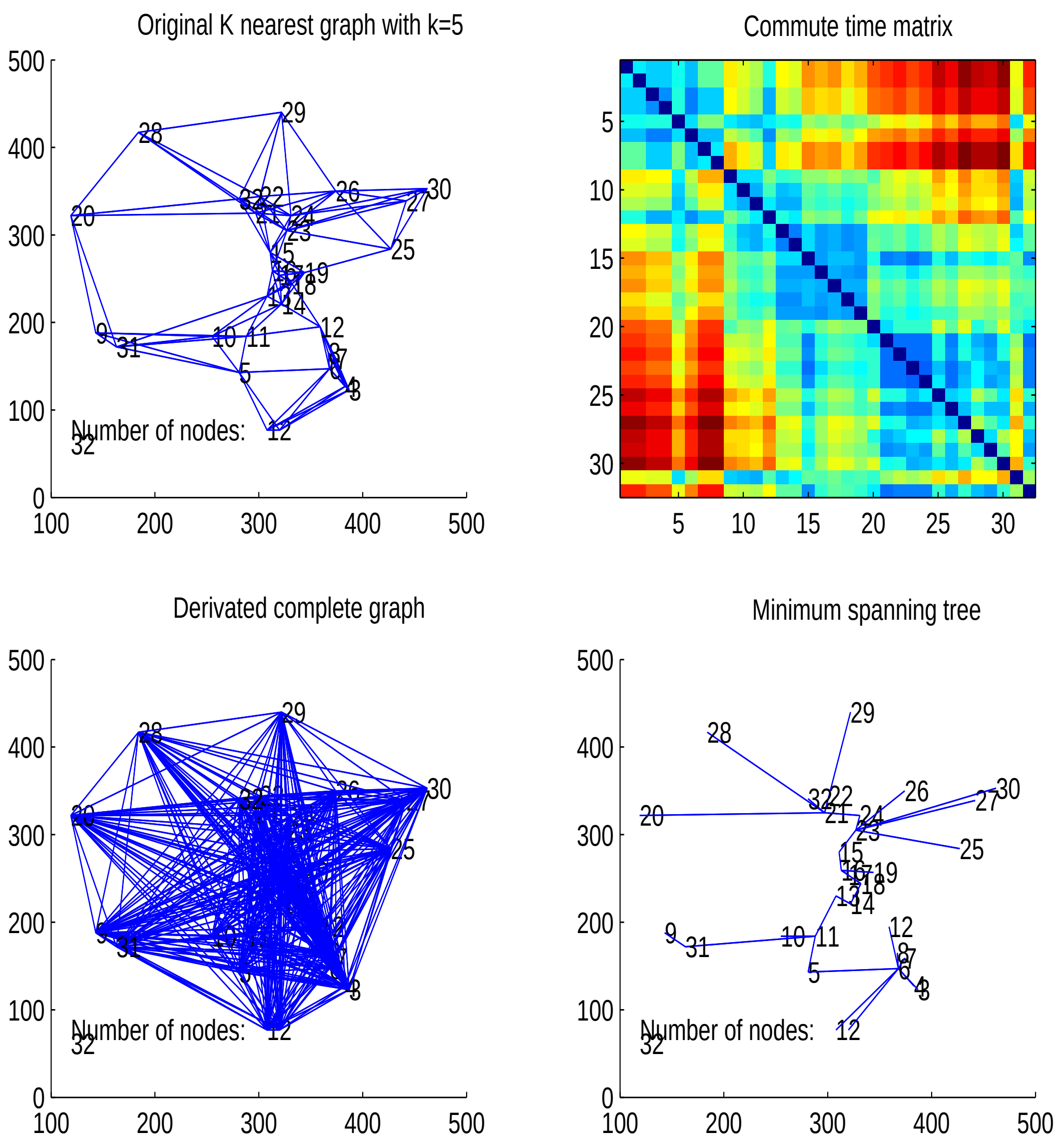


Figure 5.10: K nearest neighbour graph example.

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 trees associated with the heat kernel evolve with time.

In Figure 5.11 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 around the boundary, with branches extending it to the centre of the original graph. Hence, the structures are unstable and not suitable for matching.

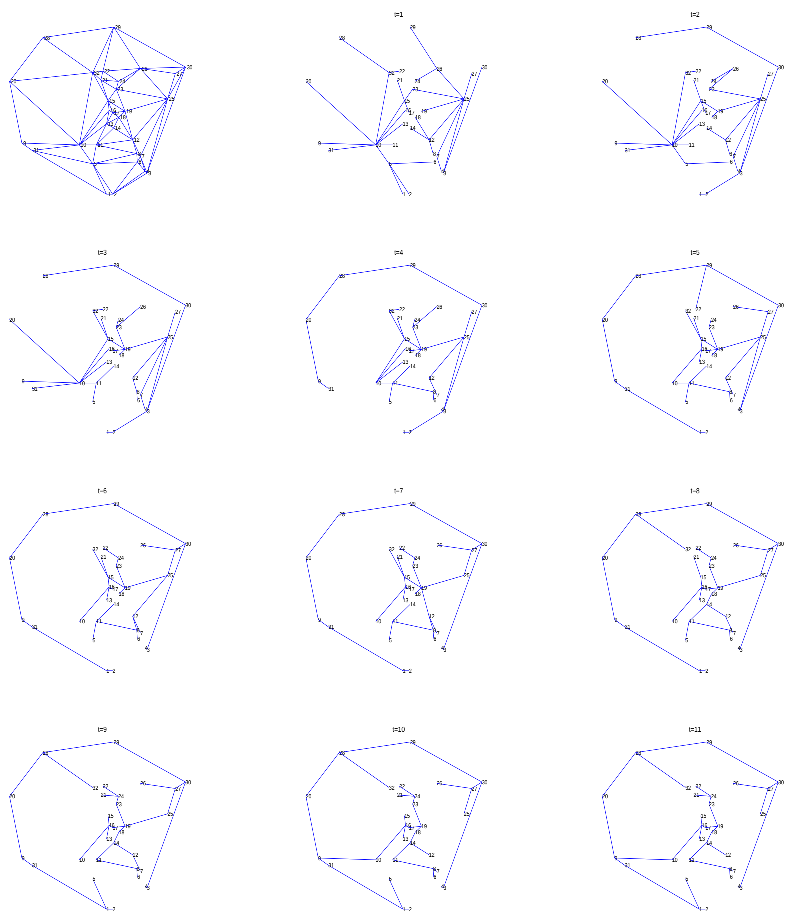


Figure 5.11: Minimum spanning tree with varying t.

5.2.2 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 (Torsello and Hancock, 2001) 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 method poses the matching problem as a max clique problem, and uses the relaxation labelling method of Pelillo (Pelillo et al., 1999; Pelillo, 1999) to obtain a solution.

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

1. Given two trees τ and τ', calculate their transitive closure TC_τ and $TC_{\tau'}$.

2. Construct the directed association graph (DAG) of TC_τ and $TC_{\tau'}$.

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 (Torsello and Hancock, 2001) or quadratic programming (Pelillo et al., 1999).

5.2.3 Experiments

The aim in this section is to illustrate 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.

5.2.3.1 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 (Prim, 1957).

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 5.12, 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.

A quantitative study on stability of spanning trees for these three sets of images is shown in Figure 5.12. Here we match the spanning tree of each image in the sequence to the first one using Torsello and Hancock's (Torsello and Han-

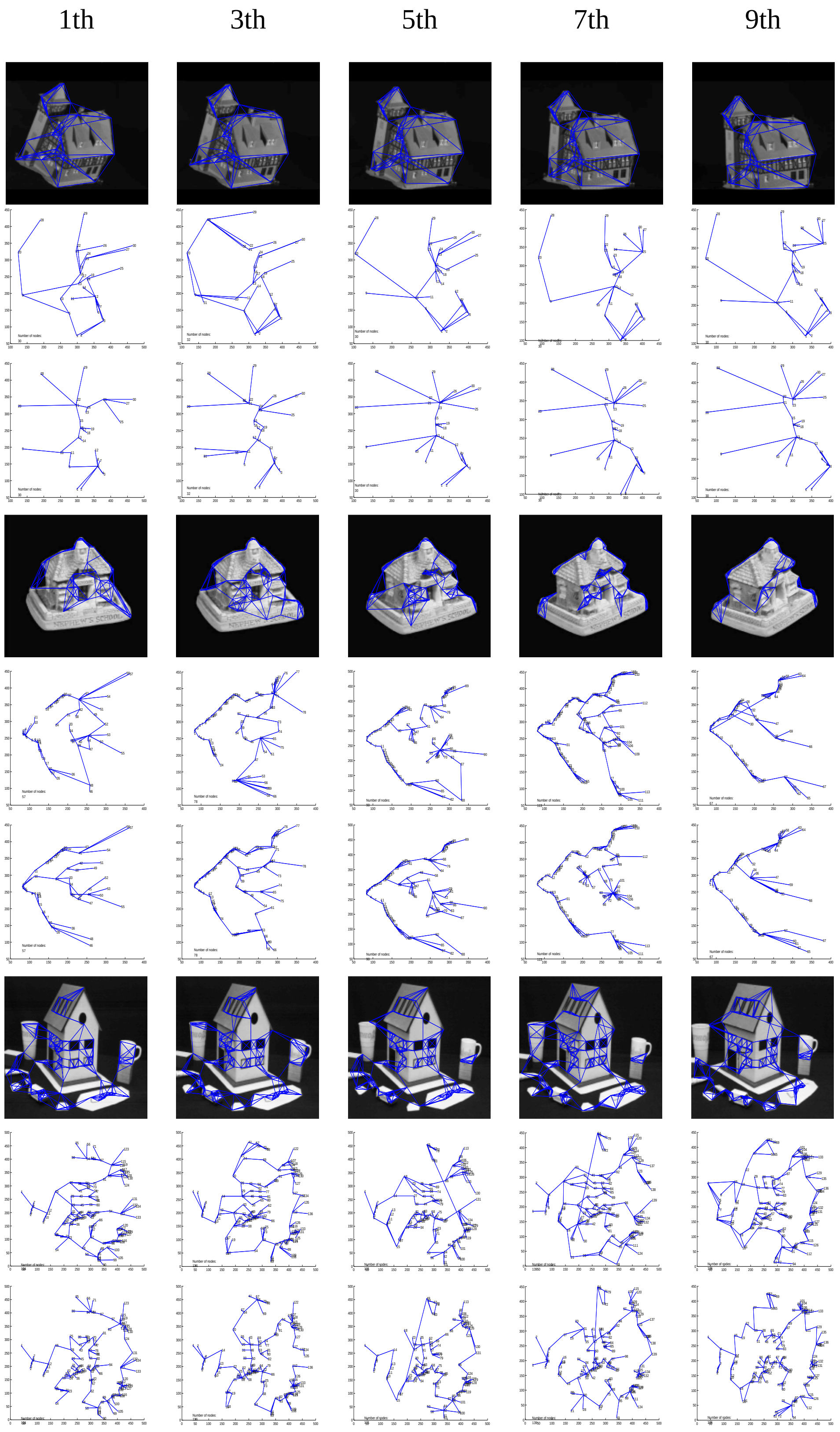


Figure 5.12: Three sequences of model houses with their spanning tree representation.

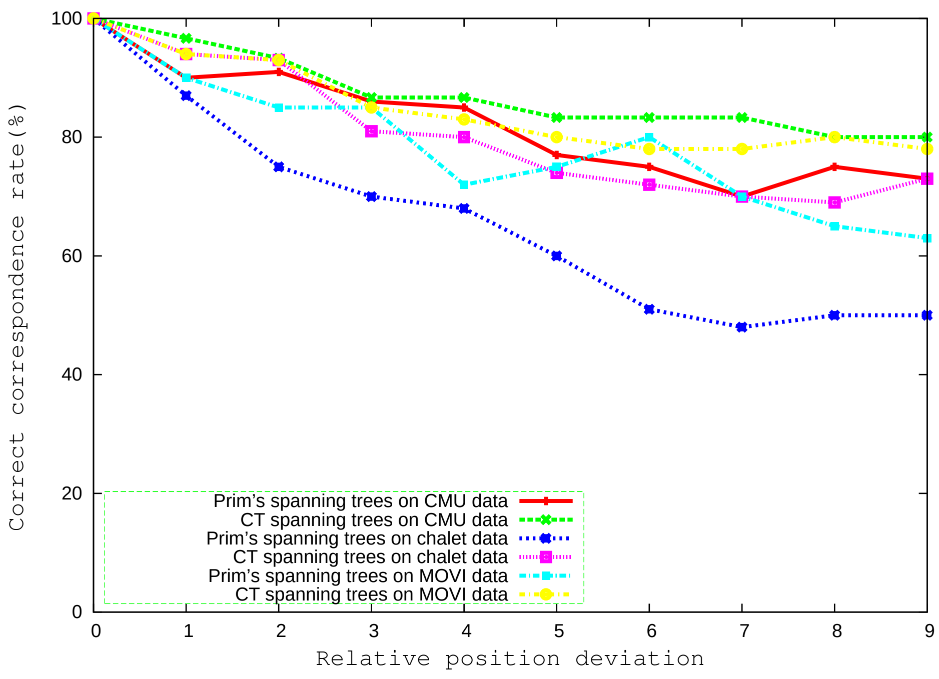


Figure 5.13: Stability comparison of spanning trees.

cock, 2001) tree matching method. It is clear that our method delivers better matching performance. It is also interesting to note that the spanning trees from MOVI data are more stable than chalet ones although the size of the former is much larger. This is due to the significant variations in nearest neighbour graph structure in chalet sequence. Some examples can been clearly seen in the forth row of Figure 5.12.

5.2.3.2 Inexact Graph 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 5.14, 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

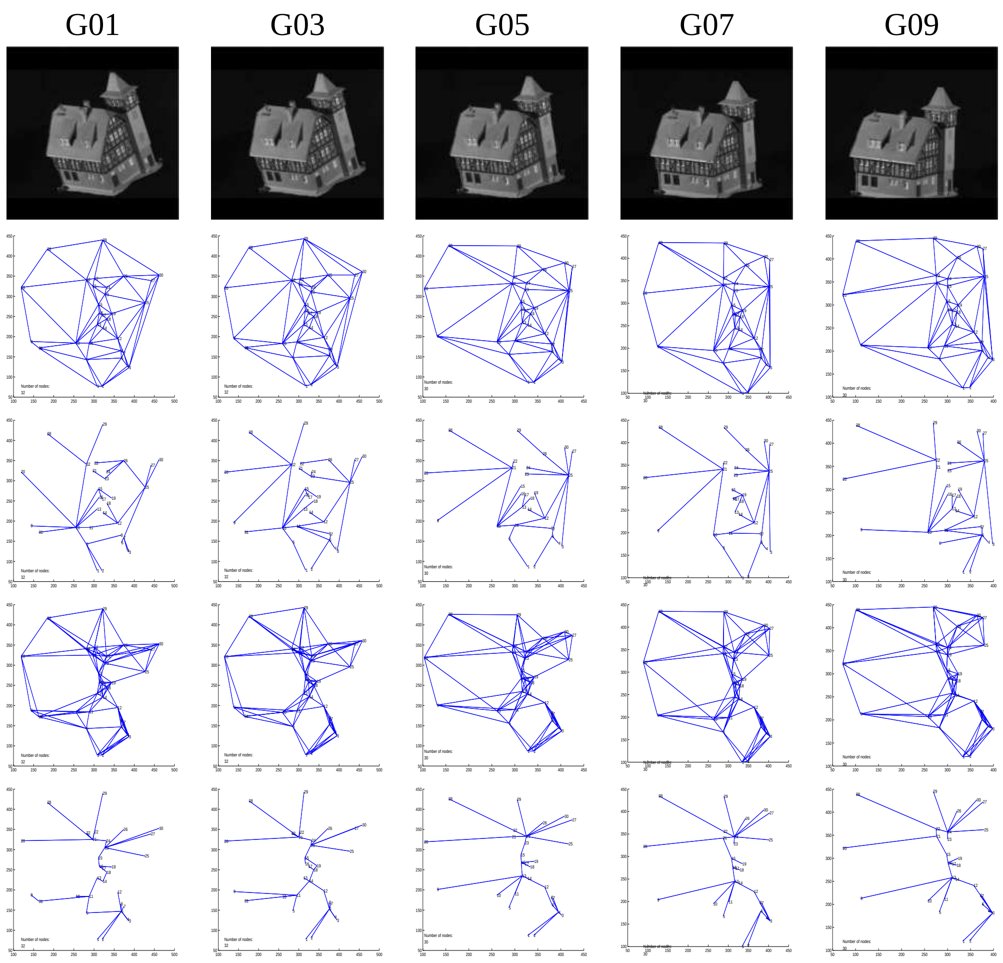


Figure 5.14: House images, their graphs and extracted trees.

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 (Torsello and Hancock, 2001). The results are compared with those obtained using the method of Luo and Hancock (Luo and Hancock, 2001) and the partition matching method of Qiu and Hancock (Chapter 3). Figure 5.15 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%.

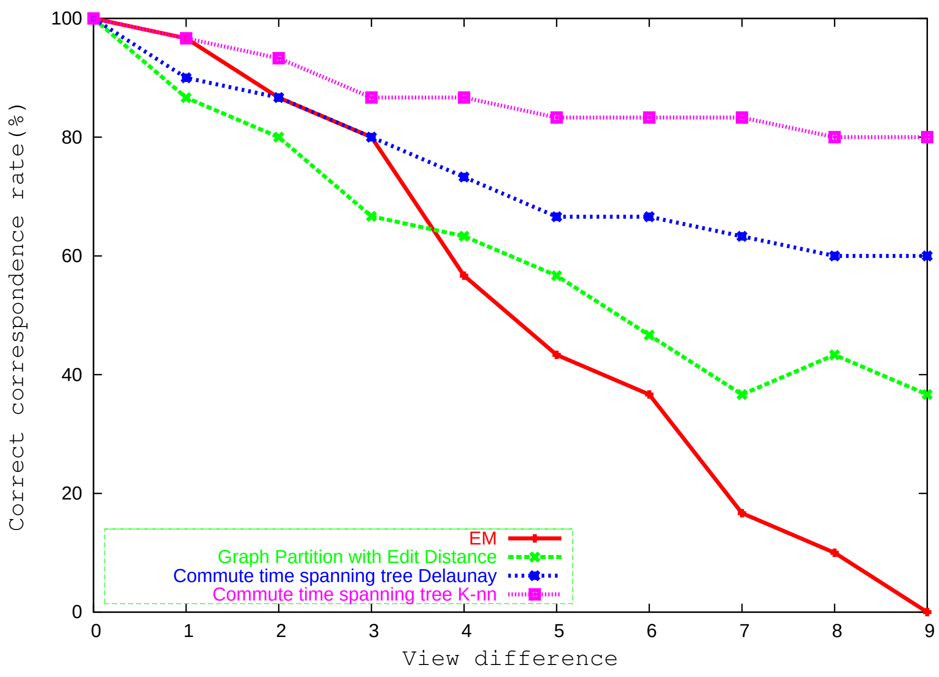


Figure 5.15: Comparison of results.

5.2.3.3 Inexact Graph Matching with Random Edge Corruption

We now focus on testing the stability of 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) randomly. In Figure 5.16 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 5.17. 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.

5.3 Comparison of the Two Simplified Graph Representations

We now explore the relative merits of the two graph simplification methods presented in this chapter. Figure 5.18 shows the fraction of correct matches for the images in the CMU house data-set. The curves in the plot are taken from Figures 5.6 and 5.15 . 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-

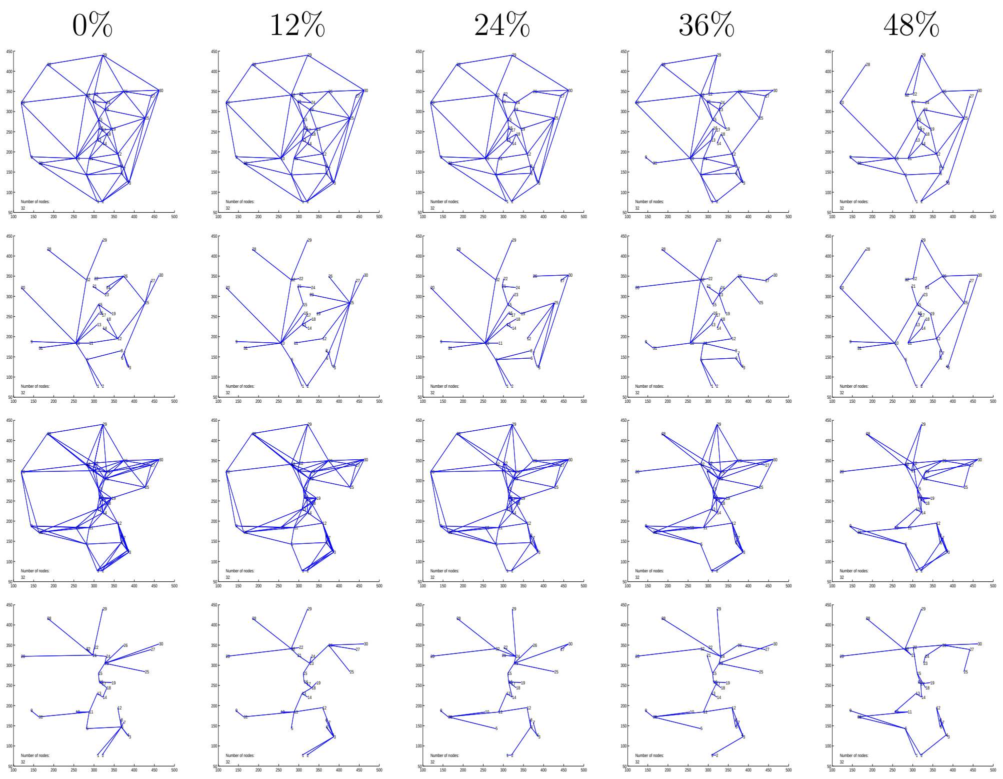


Figure 5.16: Random edge deletion.

layer representation is stable under graph variation and the composition of each layer is not modified significantly (see Figure 5.6 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. Figure5.19 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 5.14).

Overall the stability of the spanning tree representation is better than that

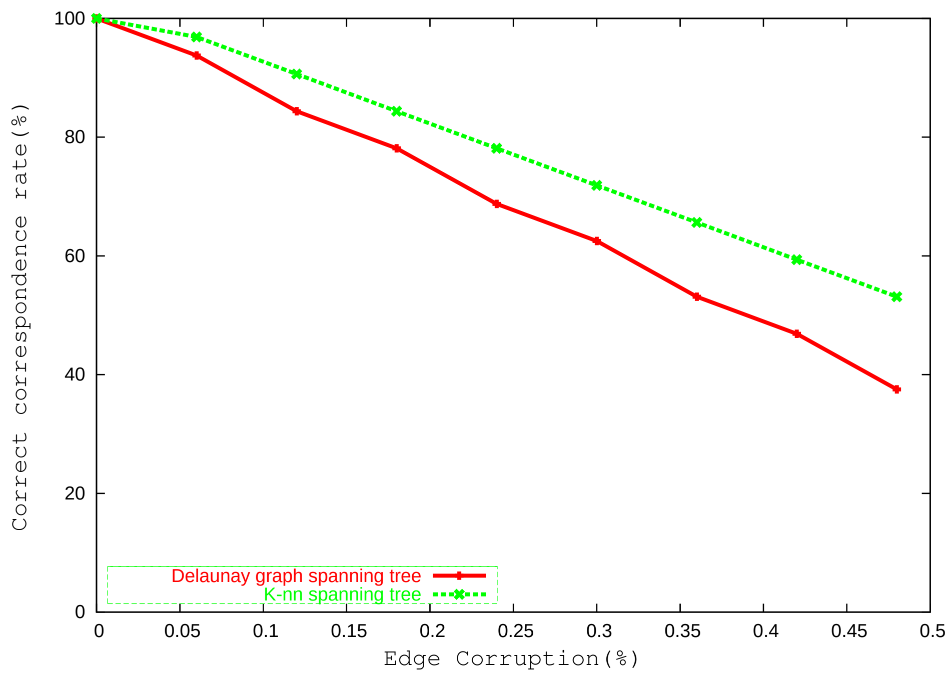


Figure 5.17: Graph corruption matching results.

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 5.20 with 12.6% of edges randomly pruned. In this example, the connectivity of the layer graph displayed in light blue is destroyed.

5.4 Commute Time for Grouping

In this section, first, we illustrate the grouping steps based on the commute times. Then we experiment it with synthetic and real-world data and compare the results with those from the normalised cut.

5.4.1 Grouping Steps

The idea of our segmentation algorithm is to use the spectrum of the commute time matrix for the purposes of grouping. In the normalised cut method, the

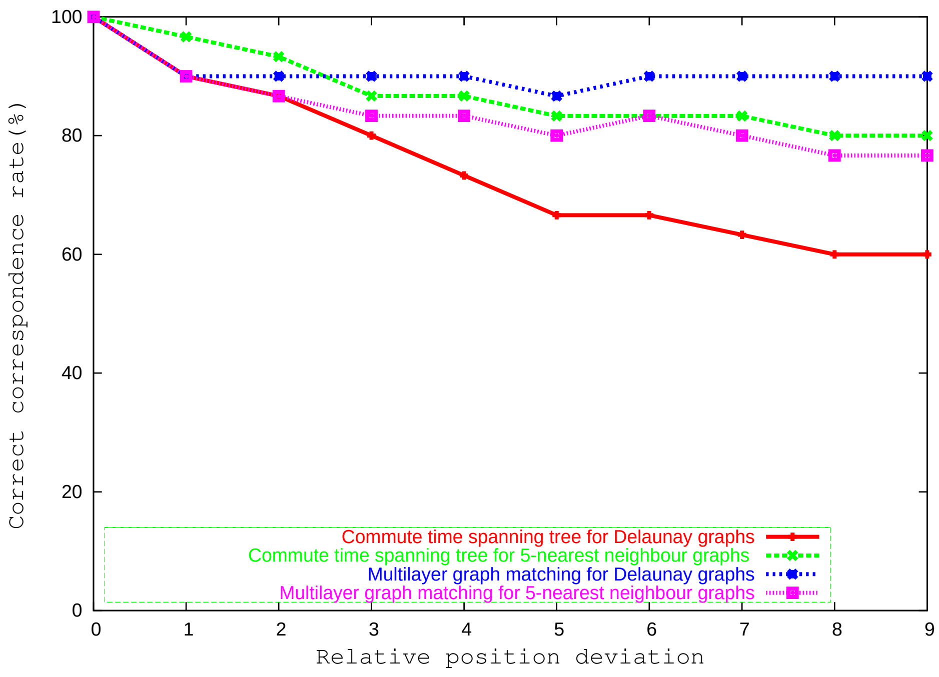


Figure 5.18: Comparison of the two methods on graph matching.

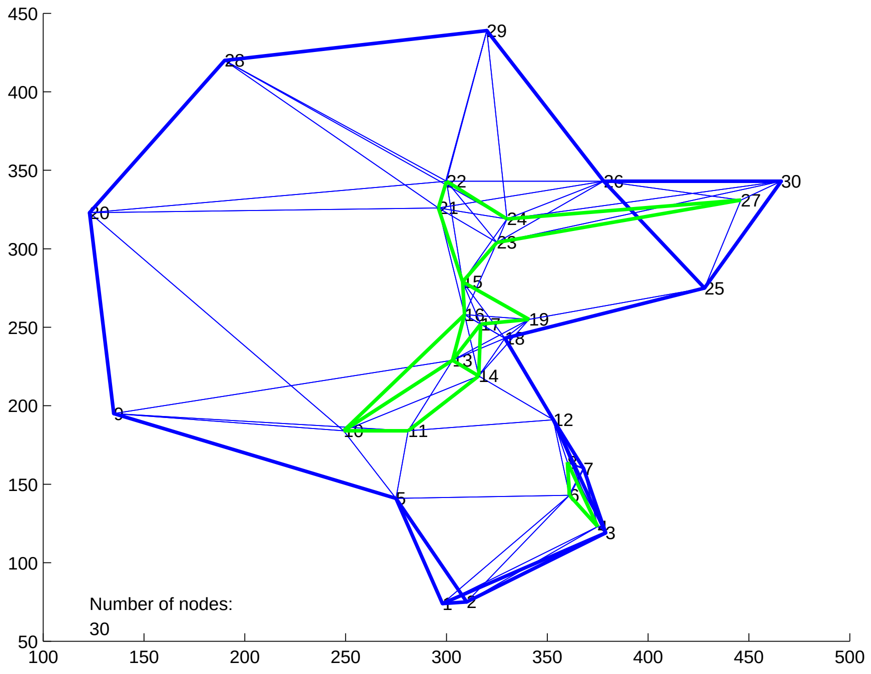


Figure 5.19: An example of multi-layer graph of a 5 nearest neighbour graph.

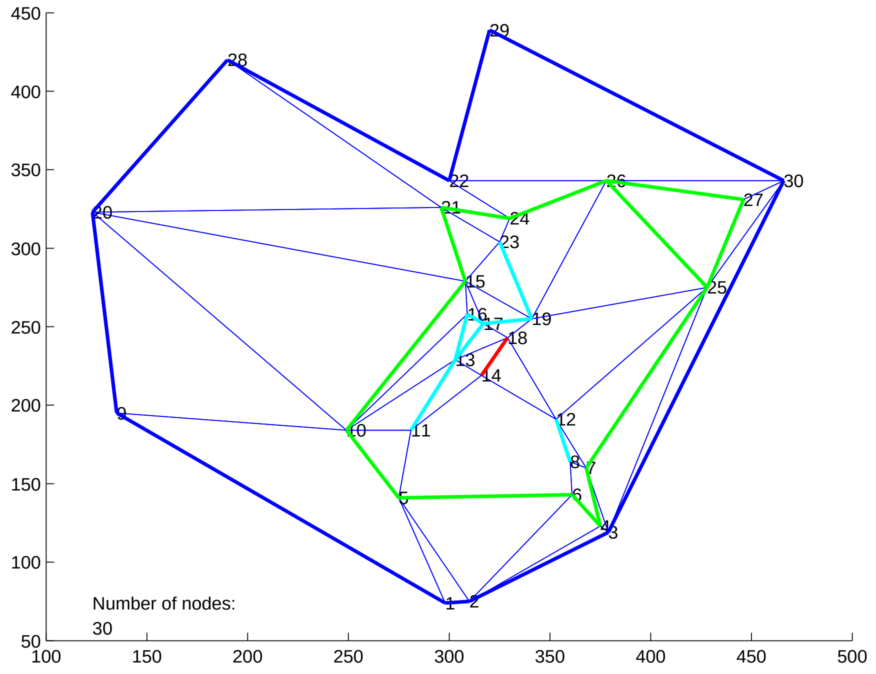


Figure 5.20: An example of a multi-layer graph with edge corruption.

eigenvector corresponding to the second smallest eigenvalue of the Laplacian matrix is utilised 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 $\Gamma = (V, E)$ where each pixel, or point, is taken as a node and each pair of nodes is connected by an edge. The weight on the edge is assigned according to the similarity between the two node as follows

   a) for a point-set, the weight between nodes u and v is set to be $\Omega(u, v) = \exp(-d(u, v)/\sigma_x)$, where $d(u, v)$ is the Euclidean distance between two points and σ_x controls the scale of the spatial proximity of the points.

b) for an image, the weight is:

$$\Omega(u,v) = \exp\left(\frac{-\|\mathbf{F}_u - \mathbf{F}_v\|_2}{\sigma_I}\right) * \begin{cases} \exp\left(\frac{-\|\mathbf{X}_u - \mathbf{X}_v\|_2}{\sigma_X}\right) & \text{if } \|\mathbf{X}_u - \mathbf{X}_v\|_2 < r \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

where $\mathbf{F}_u$ is either the intensity value at pixel u for a brightness image or the vector of RGB value for a colour image.

2. From the weight matrix Ω we compute the Laplacian $L = T - \Omega$.

3. Then we compute the *normalised* Green's function using Equation 4.2 and the eigen-spectrum of the *normalised* Laplacian $\mathcal{L}$.

4. From Equation 4.5, 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 to bipartition the weighted graph:
$CT(u,v) = vol \sum_{i=2}^{|V|} \frac{1}{\lambda'_i} \left(\frac{\phi'_i(u)}{\sqrt{d_u}} - \frac{\phi'_i(v)}{\sqrt{d_v}}\right)^2$.

6. Decide if the current partition should be sub-divided, and recursively repartition the component parts if necessary.

5.4.2 Experiments

In this section, we will illustrate some experimental results both on synthetic and real-world images on clustering and image segmentation.

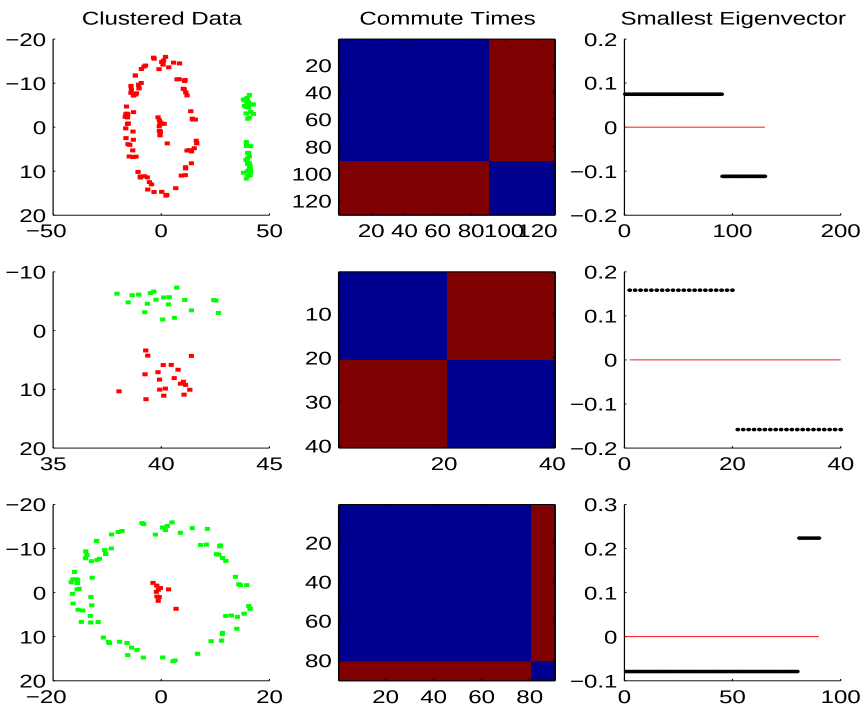


Figure 5.21: Data clustering by commute time cut

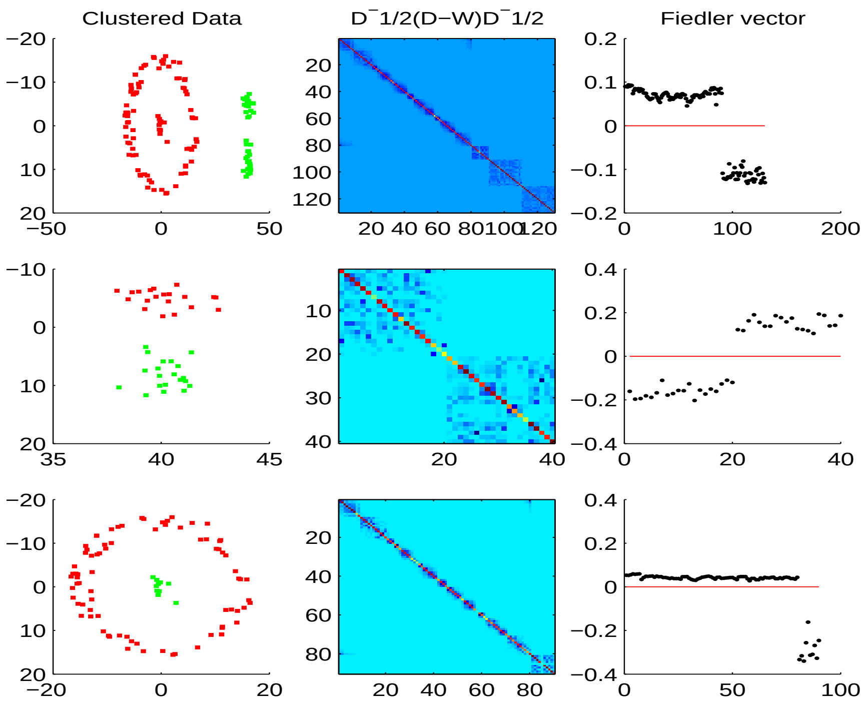


Figure 5.22: Data clustering by normalised cut

5.4.2.1 Point-set Clustering Examples

In Figure 5.21 and 5.22 we compare the results for point-set clustering using commute-times and the normalised cut. Here we set $\sigma = 1.5$. The sub-figures in both figures are organised 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 Figure 5.21 and 5.22. From the figures it is clear that both methods succeeded in grouping the data. However, the commute time method outperforms the normalised cut since its affinity matrix is more block like 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 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 Figure 5.21 compared to the one in Figure 5.22. From the figures, it is clear the distribution of the eigenvector delivered by our commute time matrix is nearly discrete. This is due to the strong block structure of the commute time matrix as illustrated in the middle of Figure 5.21 compared to the normalised affinity matrix in Figure 5.22.

5.4.2.2 Image Segmentation

We have compared our new method with that of Shi and Malik (Shi and Malik, 2000) on synthetic images subject to additive Gaussian noise. On the left-hand side of Figure 5.23, we show the results of using these two methods for segmenting a synthetic image composed of 3 rectangular regions with additive

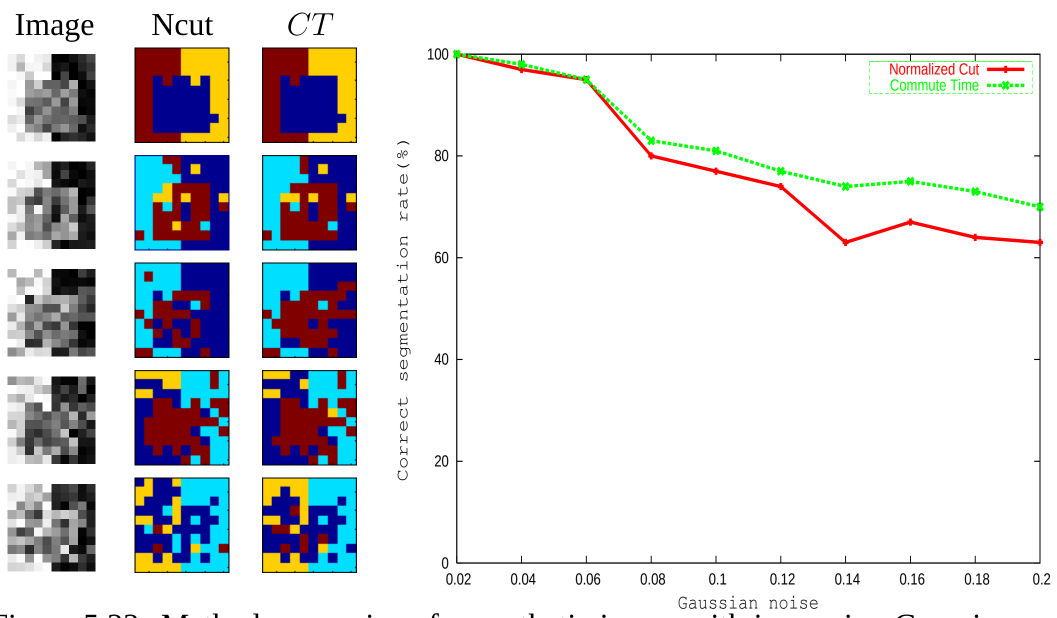


Figure 5.23: Method comparison for synthetic image with increasing Gaussian noise.

(zero mean and standard derivation increasing evenly from 0.04 to 0.20) random Gaussian noise. On the right hand side of Figure 5.23 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 Figure 5.24, we show some examples of our segmentation results and compare them with those obtained using the normalised cut. The aim here is to investigate the effect of adding and deleting link-weights at random. The first column shows the original image, the second column the original affinity matrix and the third colum 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 normalised cut and the commute time. For these images, Figure 5.25 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

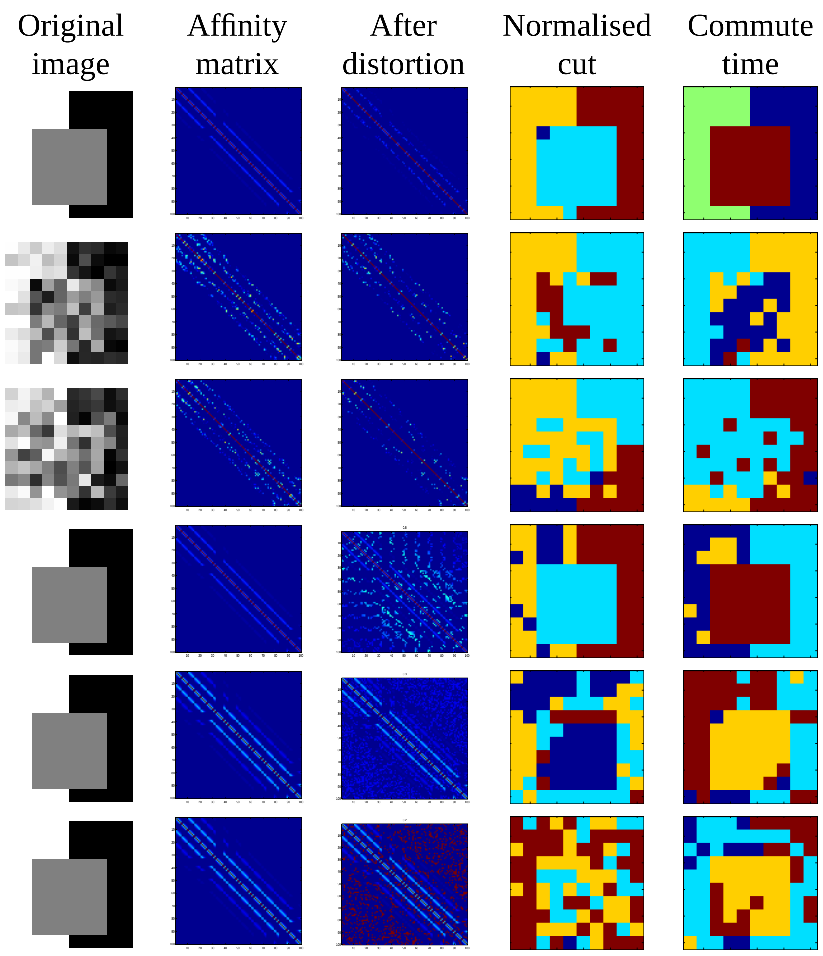


Figure 5.24: Examples of segmentation results with different link-weight distortion.

effect of link addition on the commute time method, the green curve the effect of link addition on the normalised 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 normalised cut. The main features to note from the plot are as follows. First, the commute time method is more robust to both link deletion and insertion than the normalised 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 normalised cut.

In Figure 5.26, we show eight real world images (from the Berkeley image

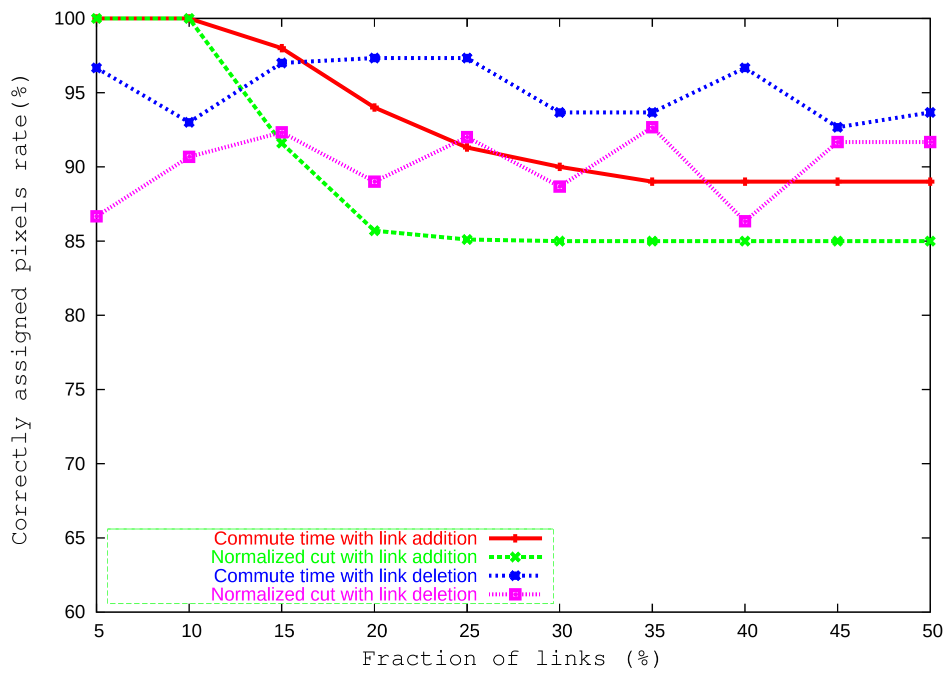


Figure 5.25: Method comparison for synthetic images with different link-weight distortion.

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 one shows the original image. The middle and right-hand panels show the results from two successive bipartitions.

For four of the real images, we compare our method with the normalised cut in Figures 5.27 and 5.28. The first column of each sub-figure shows the first, second and third bipartitions of the images. The second column shows the histogram of the components of the smallest eigenvector, and the 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 com-

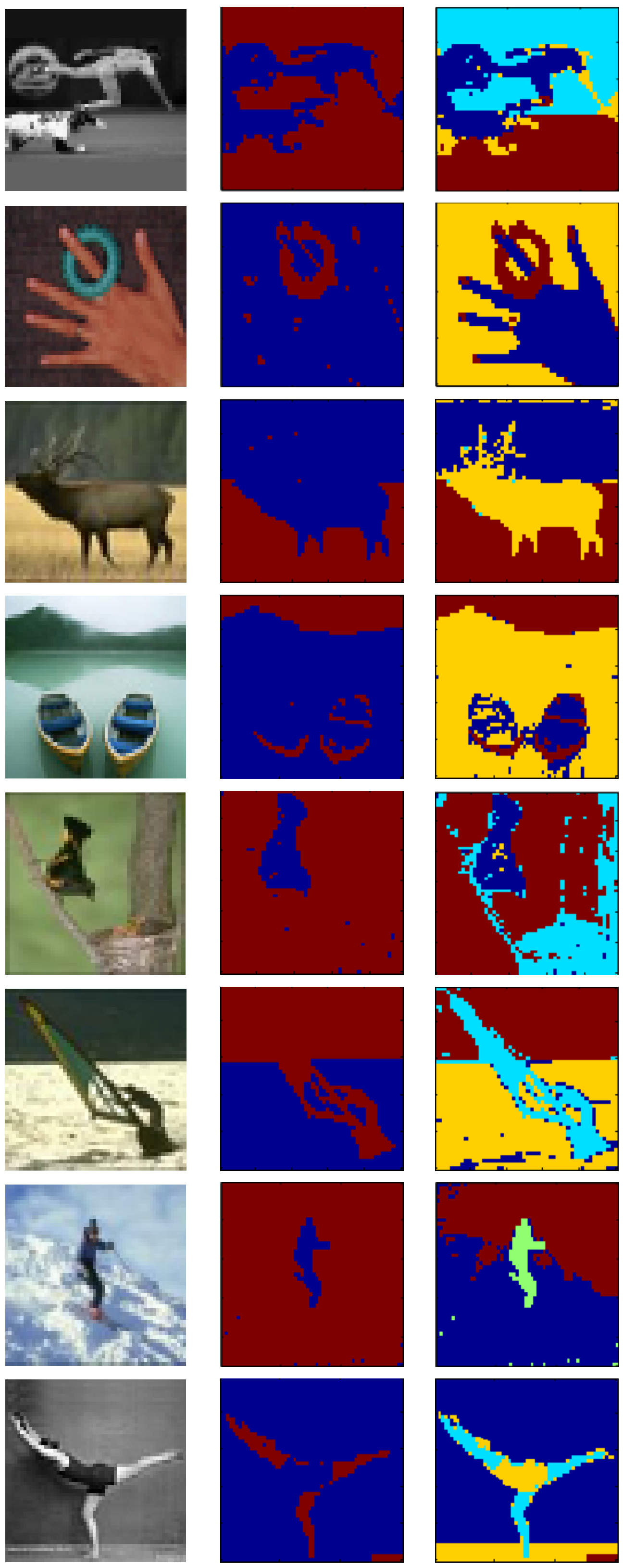
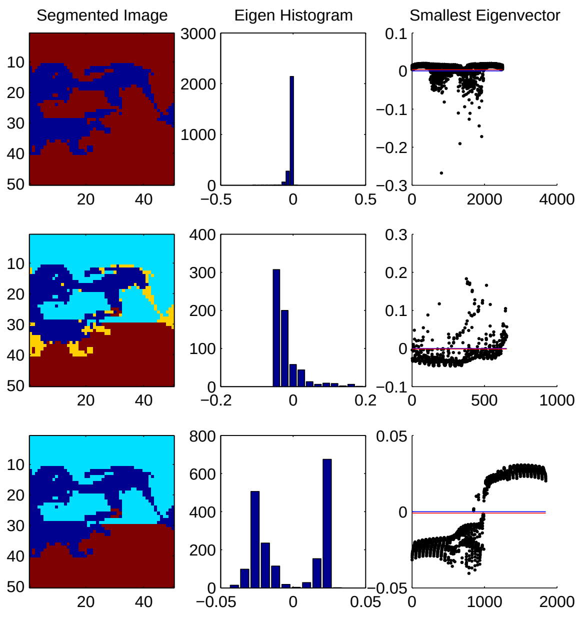
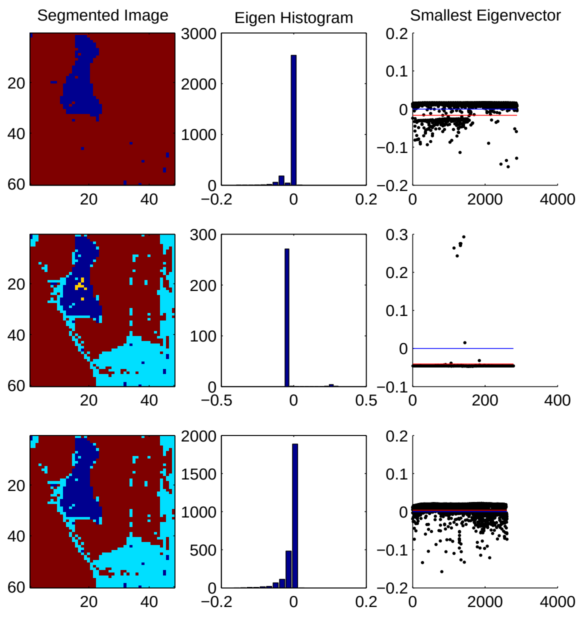


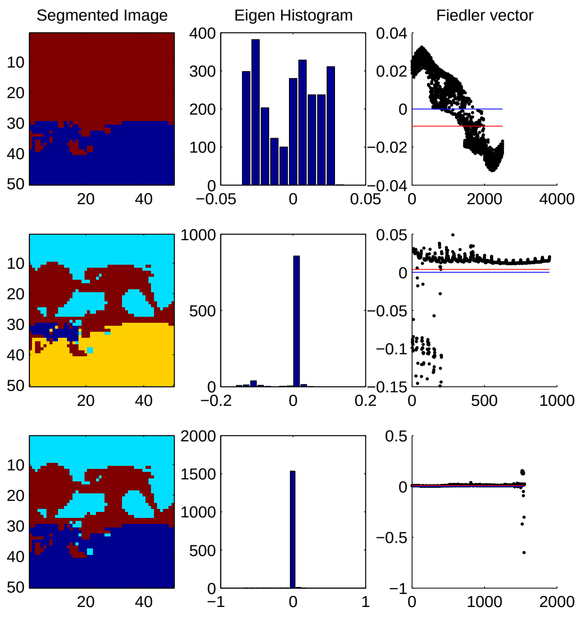
Figure 5.26: Real world segmentation examples.



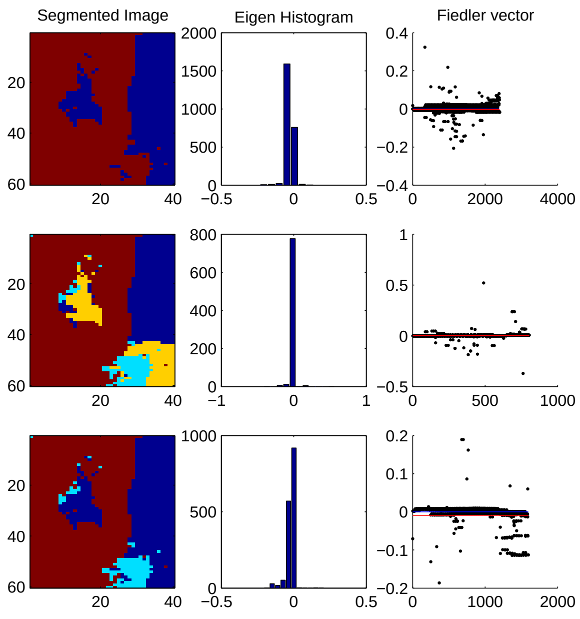
(a) Commute time for 50x50 image with $r = 8$ $\sigma_X = 0.5$ $\sigma_I = 0.1$



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

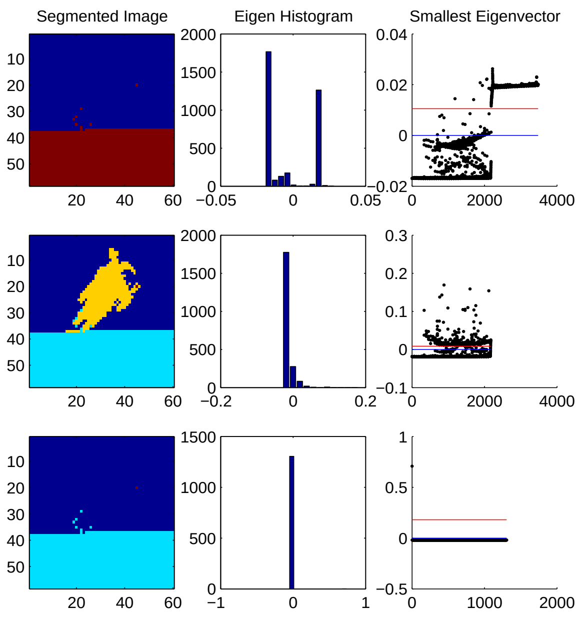


(c) Normalised cut for 50x50 image with $r = 5$ $\sigma_X = 2$ $\sigma_I = 0.05$

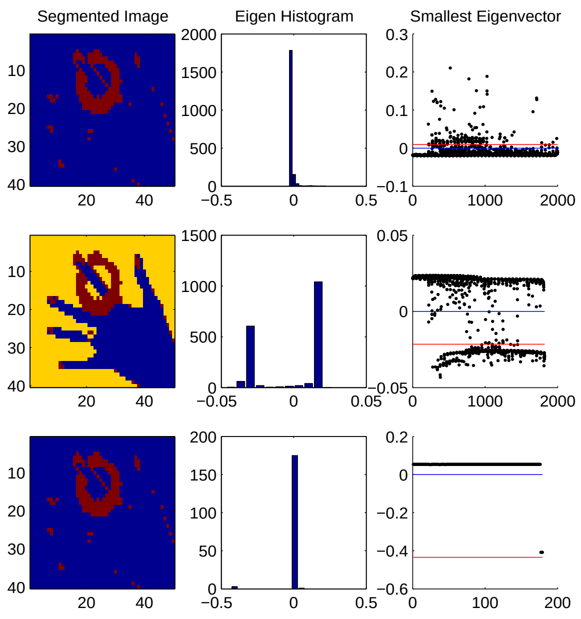


(d) Normalised cut for 60x40 image with $r = 5$ $\sigma_X = 0.05$ $\sigma_I = 0.01$

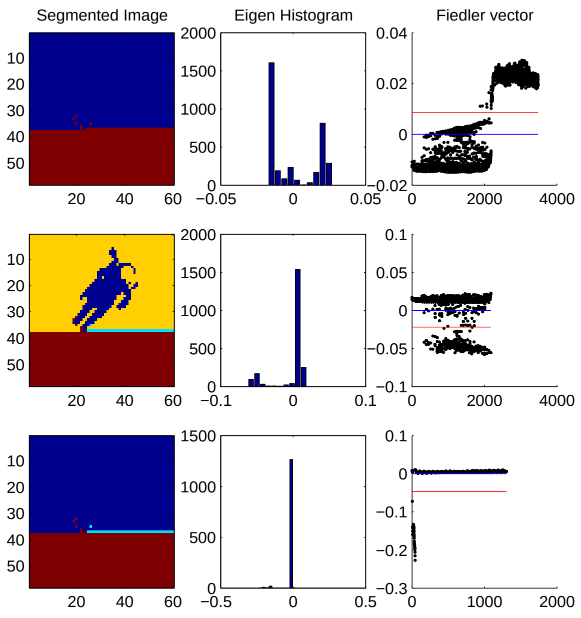
Figure 5.27: Detailed segmentation process in comparison.



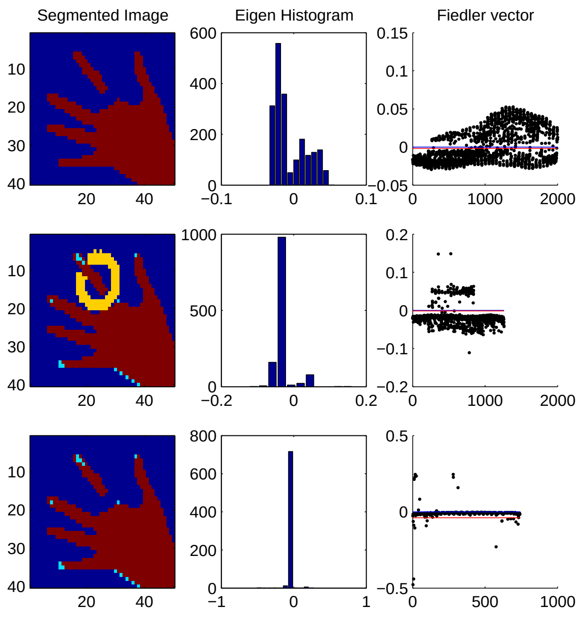
(a) Commute time for 60x58 image with $r = 5$ $\sigma_X = 0.1$ $\sigma_I = 0.03$



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



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



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

Figure 5.28: Detailed segmentation process in comparison.

mute time outperforms the normalised cut in both maintaining region integrity and continuity. For instance in the case of the baseball player, the background trademark and the limbs of the players are well segmented. In the case of the bird, the thin tree branch is detected. For the astronaut the boundary between space and the earth is detected. Finally, for the hand, the finger nails and ring are correctly segmented by the commute time method. Another important feature is that, once again, the eigenvector distribution is more stable and discriminates more strongly between clusters. This is illustrated in the second and third columns of Figure 5.27 and 5.28 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.

5.5 Multi-body Motion Tracking

The aim in this section 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. The idea 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. 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 aim is to embed feature points in a space that preserves commute time. The embedding co-ordinate matrix is found the premultiplying the transpose of the Laplacian eigenvector matrix by 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.

5.5.1 Factorisation 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

$$W = \begin{bmatrix} X \\ \hline Y \end{bmatrix}_{2F \times P}$$

The W matrix can be factorised 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 factorisation 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.

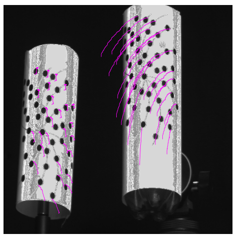
$$W = [U_1 \cdots U_N] \begin{bmatrix} \Sigma_1 & & \\ & \ddots & \\ & & \Sigma_N \end{bmatrix} \begin{bmatrix} R_1^T & & \\ & \ddots & \\ & & R_N^T \end{bmatrix}$$

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.

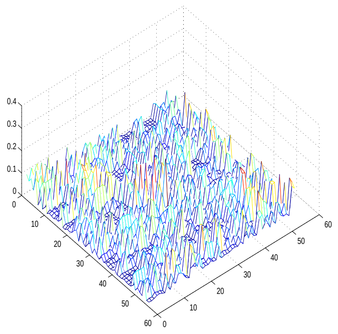
In a real multi-body tracking problem, the coordinates of the different objects are potentially permuted into a random order. As a result it is impossible to correctly recover the shape matrix S_k without knowledge of the correspondence order. Since the eigenvector matrix R is related to the shape matrix, the shape interaction matrix was introduced by Costeira and Kanade (Costeira and Kanade, 1997; Costeira and Kanade, 1995) to solve the multi-body separation problem. The shape interaction matrix is

$$Q = RR^T = \begin{bmatrix} S_1^T\Sigma_1^{-1}S_1 & 0 & \cdots & 0 \\ 0 & S_2^T\Sigma_2^{-1}S_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & \cdots & S_N^T\Sigma_N^{-1}S_N \end{bmatrix} \tag{5.2}$$

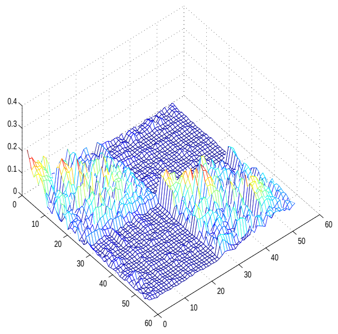
From Equation 5.2, the shape interaction matrix Q has the convenient properties that $Q_{uv} = 0$, if points u,v belong to different objects and $Q_{uv} \neq 0$, if points u,v belong to the same object. The matrix Q is also invariant to both the object motion and the selection of the object coordinate 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 (Costeira and Kanade, 1997; Costeira and Kanade, 1995) a greedy algorithm is used to permute the Q matrix into block diagonal form. An illustration is shown in Figure 5.29(a,b,c,d). This method works well only for the ideal case where there is no noise and outliers are not present. In Figures 5.29(e) and 5.29(f) we respectively show the effect of adding Gaussian noise to the Q matrix in 5.29(b) and the resulting permuted matrix. In the noisy case, the block structure is badly corrupted and object separation is almost impossible.



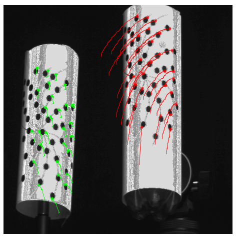
(a) Original picture with trails of the moving feature points.



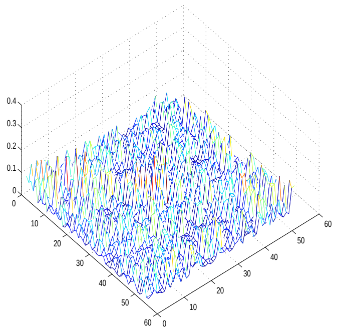
(b) Original Q matrix unsorted.



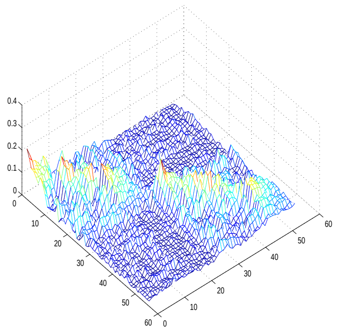
(c) Sorted Q by Costeira and Kanade's method.



(d) Object separation result.



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



(f) Sorted Q with noise.

Figure 5.29: A multi-body motion separation example using Costeira and Kanade's method.

5.5.2 Commute Time Applied to the Multi-body Motion Tracking Problem

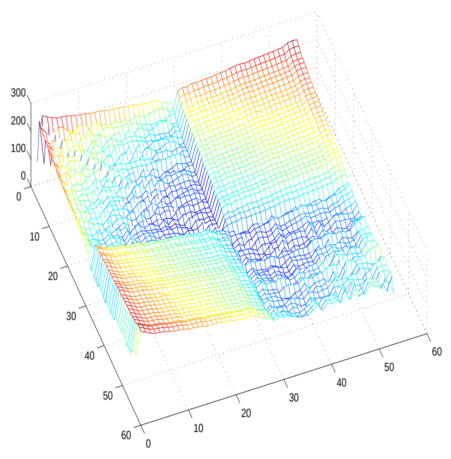
Having discussed some of the properties of the commute time embedding, in this section we will show how it may be used for multi-body motion analysis. As we have already seen, the shape interaction matrix Q introduced in the factorisation method is invariably contaminated by noise and this limits its 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:

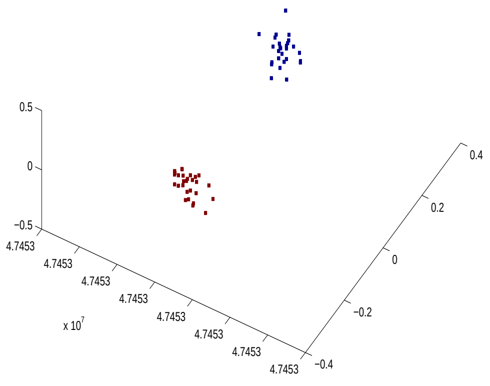
The algorithm we propose for this purpose has the following steps:

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 $L = \Phi\Lambda\Phi^T$.
4. Compute the commute time matrix CT using Λ and Φ from Equation 4.8.
5. Embed the commute time into a subspace of R^n using Equation 4.10 or 4.11.
6. Cluster the data points in the subspace using the K-means algorithm (MacQueen, 1967).

To illustrate the effectiveness of this method, we return to the example used in previous section. 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,



(a) Sorted commute time matrix.



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

Figure 5.30: Multi-body motion separation re-casted as a commute time clustering problem.

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. Moreover, when we add noise (Q with zero mean, standard derivation 0.8 Gaussian noise) and apply the clustering steps listed above we still recover a good set of objects (see Figure 5.29(d)). This is illustrated in Figure 5.30. 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.

5.5.3 Experiments

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

5.5.3.1 Synthetic Data

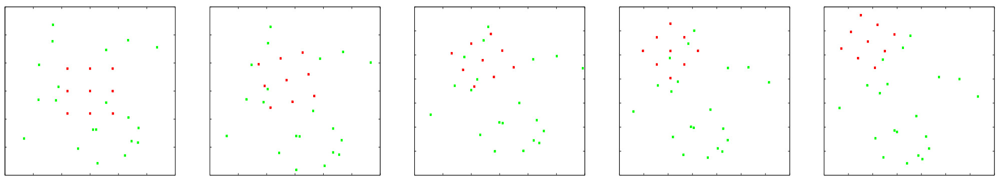


Figure 5.31: Synthetic image sequence.

Figure 5.31 shows a sequence of five consecutive synthetic images with 20 background points(green dots) and 20 foreground points(red dots) moving independently. We have added Gaussian noise of zero mean and standard deviation σ to the coordinates of these 29 points, and then cluster them into two groups.

We have compared our method with Costeira and Kanade's greedy algorithm (Costeira and Kanade, 1997; Costeira and Kanade, 1995), Ichimura's discrimination criterion method (Ichimura, 1999) and Kenichi's subspace separation method (Kanatani, 2001). In Figure 5.32 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 (Costeira and Kanade, 1997) and the discrimination criterion method (Ichimura, 1999). It also has a margin of advantage over the subspace separation method (Kanatani, 2001).

For an example with a Gaussian noise with $\sigma = 0.5$, the commute time matrix and the embedded subspace are shown in Figure 5.33(a) and 5.33(b) 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

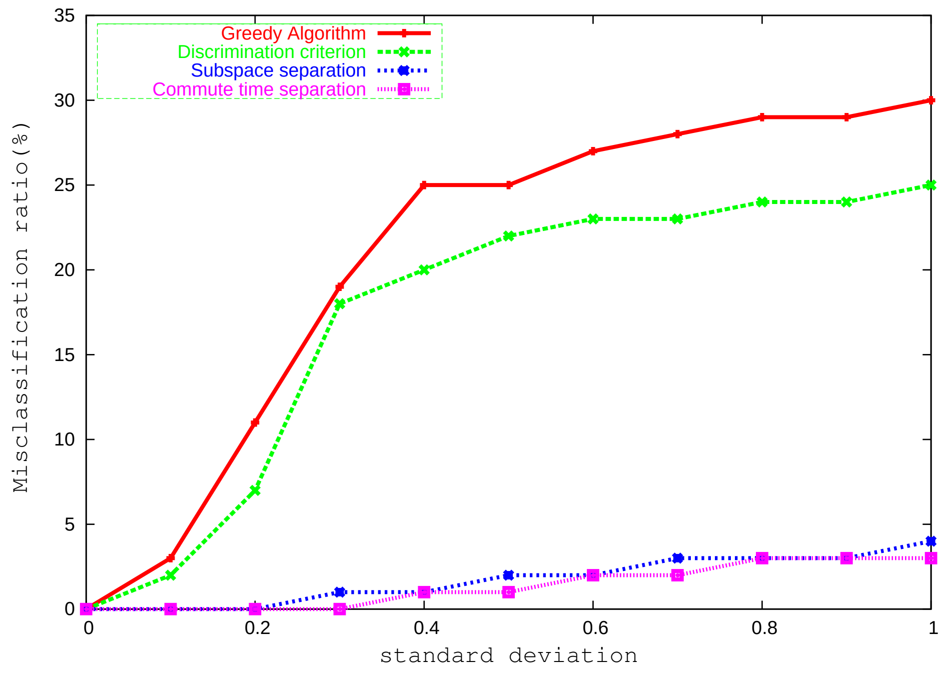


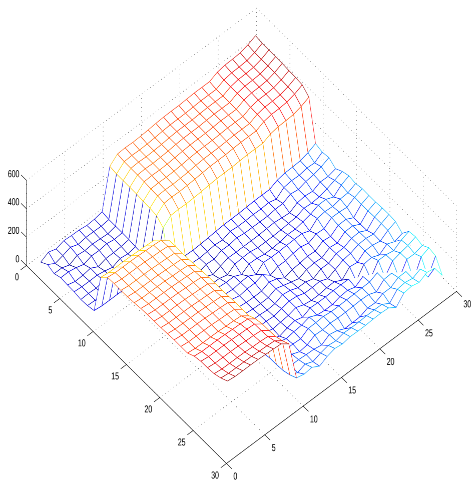
Figure 5.32: Method comparison.

embedding the points are easily separated.

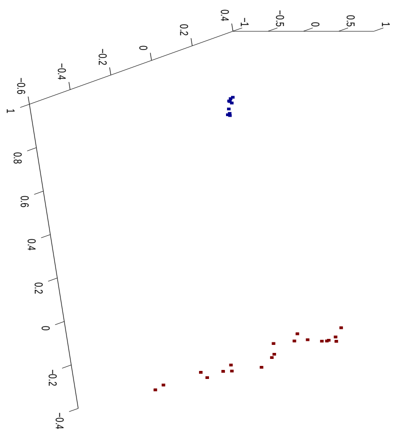
5.5.3.2 Real-world Motion Tracking

In this section we experiment with the commute time method on real-world multi-body motion tracking problems. Figure 5.34 shows five real-world video sequences with the successfully tracked feature points using the commute time method.

The first three rows are for the data used by Sugaya and Kanatani in (Sugaya and Kanatani, 2004; Sugaya and Kanatani, 2003). 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 rows in Figure 5.34 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 (Shi and Tomasi, 1994), and tracked the feature points using the commute time method.



(a) Sorted commute time matrix.



(b) Embedded subspace.

Figure 5.33: Synthetic data.

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 (Sugaya and Kanatani, 2004; Sugaya and Kanatani, 2003), we increase the number of detected moving objects from one to two, which makes the separation more difficult.

In the case of the forth row of Figure 5.34, our method not only separates the ducks correctly from the moving background, but it also separates the moving ducks from each other. The fifth row of Figure 5.34 is the most difficult one with two independently moving hands and a moving background. it also separates the wall from the floor correctly.

In Figure 5.35 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 labelled **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 labelled **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 and together with articulations. Even in the worst case, our method successfully separates the background as two different clusters as shown in Figure 5.36. The colours of the points in the embedded subspace is the same as the one shown in the fifth column of Figure 5.34.

For the same sequences, we compared our results with Costeira and Kanade's greedy algorithm (Costeira and Kanade, 1997), Ichimura's discrimination criterion method (Ichimura, 1999), Kanatani's subspace separation method (Kanatani, 2001) and Sugaya and Kanatani's multi-stage learning method (Sugaya and Kanatani, 2004). The comparison is shown in Table 5.2.

Table 5.2 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 (Costeira and Kanade, 1997) 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 (Ichimura, 1999) and the subspace separation method (Kanatani, 2001) 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 (Sugaya and Kanatani, 2004) delivers significantly better results due to its adaptive capabilities, but failed on our data.

	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**

Table 5.2: Separation accuracy for the sequences in Fig. 5.34.

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 recognised and suppressed by the embedding process. Outliers, on the other hand, are automatically rejected in the clustering step by the K-means algorithm.

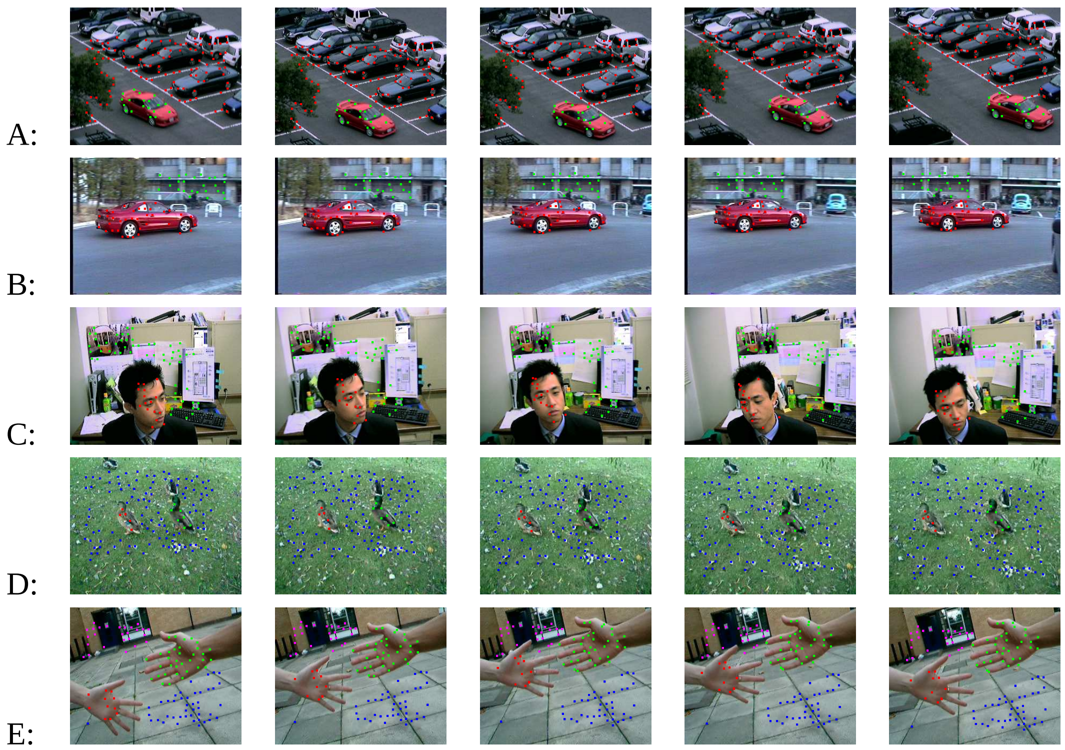


Figure 5.34: Real-world video sequences and successfully tracked feature points.

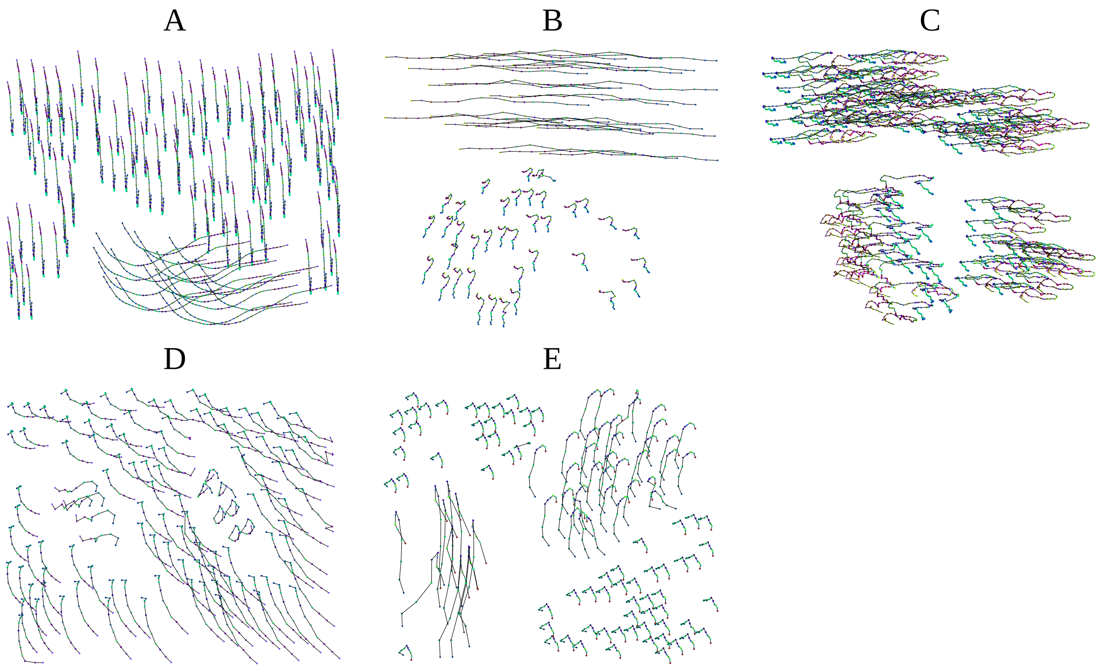


Figure 5.35: Feature point trajectories.

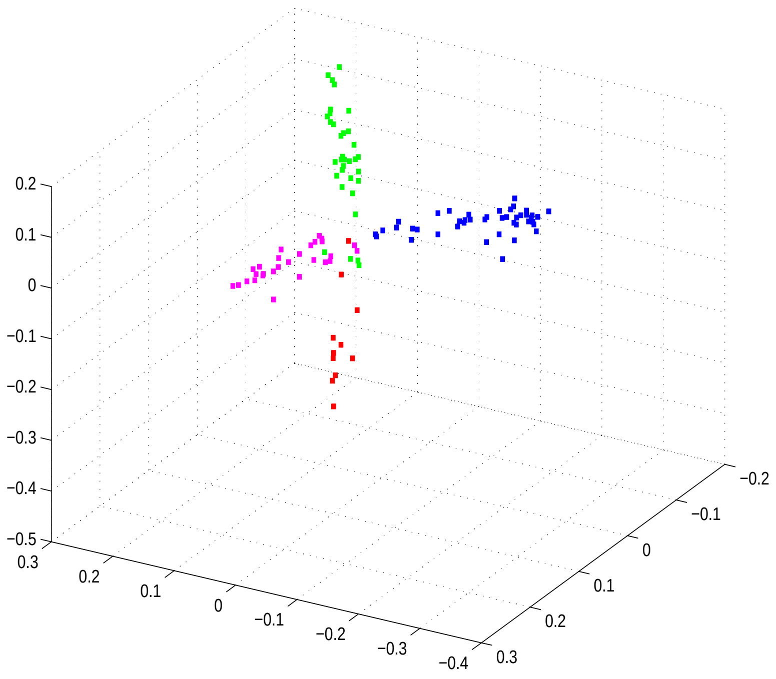


Figure 5.36: Sequence E embedded by commute time in a subspace.

5.6 Conclusions

The work presented in this chapter has focused on the application of commute times. To this end, we have shown how to use the commute time to develop two graph simplification algorithms. The first of these is a graph simplification method that uses the commute time to an auxiliary node. We show how the representation that results from this simplification can be used for the purposes of matching. 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 node information to be useful for the purposes of graph matching.

We have shown how commute time can be used for clustering and segmentation, and have compared to the Shi and Malik's method (Shi and Malik, 2000). Finally, we described how the multi-body motion tracking problem can be cast into a graph spectral setting using a commute time embedding method together with K-means clustering. To test its performance, We have compared our embedding method with a number of alternative tracking algorithms on both synthetic and real world data. Here it offers a convincing margin of improvement for noise-contaminated multi-body motion tracking.

Chapter 6

Conclusions and Future Work

The overall goal of this thesis was to exploit the properties of spectral graph theory for the purpose of solving a number of computer vision problems including object recognition, embedding, clustering and motion tracking. To this end, we have a) developed an inexact graph matching method based on the spectral decomposition of the graphs, b) described three distinct simplified graph representations and c) developed an effective embedding and clustering method for image segmentation and motion tracking.

6.1 Contributions

6.1.1 Inexact Graph Matching

Inexact graph matching has proved to be an intractable task in the computer vision literature. When spectral graph matching methods are used, then only graphs of the same size (Umeyama, 1988) can be matched. To overcome this problem, some approaches have been made but with high computational cost (Luo and Hancock, 2001). Here we solved the problem using a hierarchical matching method which is suitable for parallel computation.

Our starting point was to use the graph seriation method to decompose the graph into non-overlapping subgraphs (partitions). The pattern of partitions is similar for graphs with similar structure. The graph matching process can be realized by matching these partitions. We first found the correspondence of the partitions and then matched the elements in each pair of partitions separately. This gave us a two level matching framework and the potential to develop parallel graph matching technique. The advantage of this hierarchical matching scheme is its efficiency and ability to deal with graphs of different sizes. However, it has the disadvantages that there is a dependency of the node correspondences on the partition correspondences. Hence, incorrect partition correspondences can cause the node error to be amplified significantly.

The matching process was realized using a similarity based matching scheme and a probability based dictionary padding method. Similarity was measured using the string edit distance, where the strings were formed from the nodes of the partitions. Although this method is both effective and efficient, it relies on dismantling local graph structure, and in order to locate the node correspondences of each pair of nodes we need to employ back-tracking. A second and alternative matching method was to supplement two supercliques so that they are of same degree. To do this, we have padded the smaller one with a certain number of dummy nodes. This process did not destroy the superclique structure. However, its complexity increases exponentially with the number of dummy nodes that need to be inserted. To demonstrate their differences, we have compared these two methods together with four alternative methods. The alternatives are discrete relaxation (Wilson and Hancock, 1997), EM (Luo and Hancock, 2001), quadratic assignment (Gold and Rangarajan, 1996) and non-quadratic graduated assignment (Finch et al., 1998). Results suggested that the similarity based matching scheme outperformed the alternatives in terms of the correct matching of correspondences.

Furthermore, with the partitions in hand, we constructed a simplified graph representation based on the local partition neighbourhood. This new representation preserves the structural information of the original graph and can be used for graph clustering.

6.1.2 Simplified Graph Representations

Our contribution here was to draw on ideas from the heat diffusion process on a graph to develop a graph simplification method. This representation was based on extracting concentric layers from the graph. We realized this simplification by supplementing the original graph with an auxiliary node. This node was connected by edges to the nodes on the graph boundary. Then, we constructed the layer graphs using the distribution of heat diffusion. This process was governed by the heat equation and the solution was given by the heat kernel. The simplified graph representation is a series of concentric layers that can be matched by cyclic permutation. Although commute time was not involved in constructing the new graph representation, it played an important role in assigning scores to it. The score of each node was derived from the commute time to the auxiliary node. Nodes from different layers possess distinct commute times due to their difference in distance from the auxiliary node. Variance in graph structure does not disturb the pattern of commute time. As a result, the commute times on the nodes of the same layer can be used as an attribute for the purposes of matching. We have tested our method on real-world images and the results were promising. This method outperformed the alternatives by about 25% in the worst case. Our method delivers a more stable graph representation than alternative graph simplification methods such as the random walk (Robles-Kelly and Hancock, 2005a). This is because the pattern of concentric layers is less likely to be disturbed by structural noise. Our method is analogous to level set methods (Sethian, 1996) which evolve by front propagation rather than by heat diffusion.

Our second approach to graph simplification was based on the idea that the commute time matrix delivers a more stable representation than the adjacency matrix. The main obstacle of recasting the inexact graph matching problem as an inexact tree matching problem is the stability of the tree representation extracted from a graph. To overcome this problem, we constructed an auxiliary fully connected graph in which the weights were the commute times between pairs of nodes in the original graph. Tree representations were obtained by locating the minimum spanning trees on these complete graphs. To examine the performance of our method, we have tested the matching method on Delaunay graphs as well as on K-nearest neighbour(KNN) graphs. The superior performance of the method on the KNN graphs is probably due to the dense distribution of the edges around the central part of the graphs. These edges are preserved as the branches of the spanning tree. The success in producing a stable spanning tree from the auxiliary graph allowed us to further investigate the commute time matrix as a graph representation. The information supplied by the commute time matrix is richer than the normal adjacency matrix. If we take the adjacency matrix as a ”hard” representation, indicating only the connectivity of each pair of nodes, the commute time matrix is a ”soft-link” representation, giving a means of node.

6.1.3 Embedding and Clustering

The properties of the commute time preserving embedding were studied and a comparison with alternative embedding methods was presented. Two of the most important properties of the commute time embedding are that it preserves the maximal variance of data and that it maintains data proximity. This embedding scheme was successfully applied to the multi-body motion tracking problem. The aim here is to control the effect of noise in the factorisation method. We interpreted the shape-interaction matrix as an affinity matrix. From the associated Laplacian matrix we computed the corresponding commute time matrix. We used the commute time embedding to project the feature points into a subspace. The classification of different objects was achieved by applying a K-means clustering on the embedded feature points. Outliers and noise were suppressed by the clustering method. A set of experiments carried out on synthetic and real video sequences showed that our method performed quite well even under significant noise contamination.

The application of the commute time to image segmentation was presented in Chapter 5. Here, commute time provides a fine cluster cohesion measure that gives an enhanced block diagonal structure of the similarity matrix. Comparing our method with the normalised cut, we have developed a similar grouping algorithm based on recursive bipartition using the eigenvector of the commute time matrix. Experiments have been carried out on both synthetic images and real-world pictures. Our method outperformed the normalised cut in both maintaining region integrity and continuity. The importance of our method is that we have taken a different approach towards clustering. Rather than seeking a better cutting criteria, we here focused on exploiting and enhancing the cohesion relationships in the data. This greatly simplifies the clustering task.

In this thesis, spectral graph theory has been intensively studied and new methods have been developed facilitating more sophisticate use of the Laplacian eigenspectrum. Distinct from the existing methods, our approaches are concerned with using more eigenvectors of the affinity matrix. This enables us to use richer information from the original graph and develop more robust and efficient algorithms. At the same time, it inspires the directions to make more approaches towards solving various computer vision problems using spectral methods.

6.2 Future Work

The methods proposed in this thesis exhibit several shortcomings that need further research. Moreover, some of the topics addressed could be extended and investigated further.

For instance, in Chapter 3, we have restricted our method to deal only with Delaunay graphs. Since the partition is based on supercliques it is not applicable to non-planar graphs with crossed edges (such as the K nearest neighbour graph). This restriction needs to be overcome. One possible solution is to consider alternative feasible sub-structures embedded in the graph such as maximal cliques or dominant sets (Pavan and Pelillo, 2003a). Based on the cohesion relationship between nodes, the dominant set has been successfully applied to the image segmentation problem. Hence, further investigation of graph partition matching could yield some interesting results.

A second issue concerning our matching method is the size of the graphs. Since our method is based on hierarchical matching, the accuracy of matching in the early steps is critical. This is increasingly difficult as graphs become larger since the number of partitions increases.

Despite its effectiveness and efficiency, the multi-layer graph representation is vulnerable to structural corruption caused by edge and node deletion. This sometimes disturbs the connectivity of the layer graphs. One way of recovering from this problem could be to cast the matching process into a maximum likelihood estimation framework.

Another possible avenue of investigation is to further study potential applications of the proximity matrix. So far,we have used it only for stabilising partitions, hence it could be useful for segmentation and clustering.

Graph matching by comparing spanning trees suffers from problems due to the unstable structure of the tree representation and information loss. Although our tree representation is relatively stable for large graphs, there is clearly room for improvement in the accuracy of branch location. Another topic that merits further investigation is to study the relationship between our tree representation and the original graph.

Our factorisation method suppresses the effect of outliers and noise. However, it does not deal with degeneracy, dependency and missing data. Although these problems are difficult to handle, our framework could be extended to deal with them. For instance, the EM algorithm (Gruber and Weiss, 2004) has been employed in the factorisation method to deal with uncertainty and missing data. Furthermore, covariance-weighted factorisation (Anandan and Irani, 2002) and a refined shape interaction matrix has also been used (Zelnik-Manor and Irani, 2003). All these methods perform reasonably well and can easily be incorporated into our clustering framework.

Bibliography

Adolphson, D. (1977). Single machine job sequencing with precedence constraints. *J. SIAM Comput.*, 6:40–54.

Aizerman, M., Braverman, E., and Rozonoer, L. (1964). Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25:821–837.

Aldous, D. and Fill, J. (2003). *Reversible Markov Chain and Random Walks on Graphs*. Draft.

Alpert, C. and Yao, S. (1995). Spectral partitioning: The more eigenvectors, the better. In *ACM/IEEE Design Automation Conference*.

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

Atkins, J. E., Boman, E. G., and Hendrickson, B. (1998). A spectral algorithm for seriation and the consecutive ones problem. *SIAM Journal on Computing*.

Auscher, P., Coulhon, T., and Grigor'yan, A. (2003). Heat kernels, random walks, and analysis on manifolds and graphs. *AMS, Contemporary Mathematics*, 338.

Bai, X. and Hancock, E. R. (2004). Heat kernels, manifolds and graph embedding. In *SSPR*, pages 198–206.

Bai, X., Yu, H., and Hancock, E. R. (2004a). Graph matching using manifold embedding. In *ICIAR*, pages 352–359.

Bai, X., Yu, H., and Hancock, E. R. (2004b). Graph matching using spectral embedding and alignment. In *ICPR*, pages 398–401.

Barrow, H. G. and Popplestone, R. J. (1971). Relational descriptions in picture processing. *Machine Intelligence*, 6:377–396.

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

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

Biggs, N. L. (1974). *Algebraic Graph Theory*. Cambridge University Press, Cambridge, Books.

Brand, M. and Huang, K. (2003). A unifying theorem for spectral embedding and clustering. In *Ninth International Workshop on Artificial Intelligence and Statistics*.

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

Bunke, H. (1999). Error correcting graph matching: On the influence of the underlying cost function. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 21(9):917 – 922.

Caelli, T. and Kosinov, S. (2004). An eigenspace projection clustering method for inexact graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(4):515–519.

Carcassoni, M. and Hancock, E. R. (2000). Point pattern matching with robust spectral correspondence. In *CVPR*, pages 1649–1655.

Carcassoni, M. and Hancock, E. R. (2003). Spectral correspondence for point pattern matching. *Pattern Recognition*, 36(1):193–204.

Chavel, I. (1984). *Eigenvalues in Riemannian Geometry*. Academic Press, New York.

Christmas, W. J., Kittler, J., and Petrou, M. (1995). Structural matching in computer vision using probabilistic relaxation. *IEEE Trans. Pattern Anal. Machine Intell*, 17(8):749–764.

Chung, F. R. K. (1988). Labelings of graphs. *Academic Press, San Diego*, pages 151–168.

Chung, F. R. K. (1997). *Spectral Graph Theory*. CBMS series 92. American Mathmatical Society Ed.

Chung, F. R. K. and Yau, S. T. (1997). A combinatorial trace formula. *Tsing Hua lectures on geometry and analysis*, pages 107–116.

Chung, F. R. K. and Yau, S. T. (1999). Coverings, heat kernels and spanning trees. *The Electronic Journal of Combinatorics*, 6.

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

Coifman, R. R., Lafon, S., Lee, A. B., Maggioni, M., Nadler, B., Warner, F., and Zucker, S. W. (2005). Geometric diffusions as a tool for harmonic analysis and structure definition of data: Diffusion maps. *National Academy of Sciences*, 102(21):7426–7431.

Collatz, L. and Maas, C. (1987). On early papers on the eigenvalues of graphs. *Hamburger Beitrage zur angewandten Mathematik, Reihe A, Preprint 6*.

Collatz, L. and Sinogowitz, U. (1957). Spektren endlicher grafen. *Abh. Math. Sem. Univ. Hamburg*, 21:63–77.

Comaniciu, D. (2003). An algorithm for data-driven bandwidth selection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(2):281–2886.

Comaniciu, D. and Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(5):603–619.

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

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

Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Cross, A. D. J., Wilson, R. C., and Hancock, E. R. (1997). Inexact graph matching using genetic search. *Pattern Recognition*, 30(6):953–70.

Cvetkovic', D. and Rowlinson, P. (1990). The largest eigenvalue of a graph – a survey. *Linear and Multilinear Algebra*, 28:3–33.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society B*, 39(1):1–38.

Desai, M. and Rao, V. (1993). On the convergence of reversible markov-chains. *SIAM J. Matrix Analysis and Appl.*, 4:950–966.

Diaconis, P. and Stroock, D. (1991). Geometric bounds for eigenvalues of markov chains. *Ann. Appl. Probab.*, 1:36–61.

Diaz, J., Petit, J., and Serna, M. (2000). A survey on graph layout problems. Technical report LSI-00-61-R, Universitat Polit‘ecnica de Catalunya.

Donath, W. E. and Hoffman, A. J. (1972). Algorithms for partitioning of graphs and computer logic based on eigenvectors of connection matrices. *IBM Technical Disclosure Bulletin*, 15:938–944.

Doob, M., Gutman, I., Cvetkovic’, D., and Torgasev, A. (1988). *Recent results in the theory of graph spectra*. North Holland, Amsterdam, Journals.

Doob, M., Sachs, H., and Cvetkovic’, D. (1995). *Spectra of graphs - Theory and applications*. Johann Ambrosius Barth Verlag, Heidelberg-Leipzig (1995), the thirst revised and enlarged edition, Books.

Dulmage, A. L. and Mendensol, N. S. (1967). Graphs and matrices. *Graph Theory and Theoretical Physics*, pages 167–227.

Elgammal, A. M., Duraiswami, R., and Davis, L. S. (2003). Efficient kernel density estimation using the fast gauss transform with applications to color modeling and tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(11):1499–1504.

Ellis, R. B. (2002). Discrete green’s function for products of regular graphs. In *AMS national conference special session on graph theory*.

Eschera, M. A. and Fu, K.-S. (1986). An image understanding system using attributed symbolic representation and inexact graph matching. *Transactions on Pattern Analysis and Machine Intelligence*, 18(5).

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

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

Fiedler, M. (1973). Algebraic connectivity of graphs. *Czech. Math. Journal*, 23:298–305.

Fiedler, M. (1975). A property of eigenvectors of non-negative symmetric matrices and its application to graph theory. *Czechoslovak Mathematics Journal*, 25:619–633.

Fiedler, M. (1989). Laplacian of graphs and algebraic connectivity. *Combinatorics and Graph Theory, Banach Centre Publ. PWN Polish Scientific Publ. , Warshaw*, 25:57–70.

Fiedler, M. (1993). A geometric approach to the laplacian matrix of a graph. *Combinatorial and Graph-Theoretical Problems in Linear Algebra*, pages 73–98.

Finch, A. M., Wilson, R. C., and Hancock, E. R. (1998). An energy function and continuous edit process for graph matching. *Neural Computation*, 10(7):1873–1894.

Fischer, I. and Poland, J. (2005). Amplifying the block matrix structure for spectral clustering. In *IDSIA*.

Fischler, M. and Elschlager, R. (1973). The representation and matching of pictorical structures. *IEEE Transactions on Computers*, 26:67–92.

Fisher, R. A. (1936). The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188.

Garey, M. R., Johnson, D. S., and Stockmeyer, L. (1976). Some simplified np-complete graph problems. *Theoretical Computer Science*, 1:237–267.

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

Gold, S. and Rangarajan, A. (1996). A graduated assignment algorithm for graph matching. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(4):377–388.

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

Grigor'yan, A. (2000). Heat kernels on manifolds, graphs and fractals. In *European Congress of Mathematics*, pages 393–406.

Grone, R. (1991). On the geometry and laplacian of a graph. *Linear Algebra and Appl.*, 150:167–178.

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

Harper, L. H. (1964). Optimal assignmentsof numbers to vertices. *J. SIAM*, 12(1):131–135.

Harper, L. H. (1966). Optimal numberings and isoperimetric problems on graphs. *J. Combinatorial Theory*, 1(3):385–393.

He, X. and Niyogi, P. (2003). Locality preserving projections. In *NIPS*, pages 585–591.

Hotelling, H. (1933). Analysis of complex statistical variables in principal components. *J. Educational Psychology*, 24:417–441,498–520.

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

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323.

Juvan, M. and Mohar, B. (1992). Optimal linear labelings and eigenvalues of graphs. *Disc. Appl. Math*, 36:153–168.

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

Kannan, R., Vempala, S., and Vetta, A. (2000). On clusterings: Good, bad, and spectral. In *Proceedings of the 41st Annual Symposium on the Foundation of Computer Science*, pages 367–380.

Kirchhoff, G. (1847). ber die auflsung der gleichungen, auf welche man bei der untersuchung der linearen verteilung galvanischer strme gefhrt wird. *Ann. Phys. Chem.*, 72:497–508.

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

Kruskal, J. B. and Wish, M. (1978). *Multidimensional scaling*. Sage Publications,Beverly Hills.

Lafferty, J. and Lebanon, G. (2005). Diffusion kernels on statistical manifolds. *Journal of Machine Learning Research*, 6:129–163.

Lafon, S. and Lee, A. B. (2005). Diffusion maps: a unified framework for dimension reduction, data partitioning and graph subsampling. *submitted to PAMI*.

Levenshtein, V. I. (1966). Binary codes capable of correcting deletions insertions and reversals. *Soviet Physics-Doklandy*, 10(8):707–710.

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

Luo, B., Cross, A. D., and Hancock, E. R. (1998). Corner detection via topographic analysis of vector potential. *Proceedings of the 9 th British Machine Vision Conference*.

Luo, B., Cross, A. D., and Hancock, E. R. (1999). Corner detection via topographic analysis of vector potential. *Pattern Recognition Letters*, 20(6):635–650.

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

Luo, B., Wilson, R. C., and Hancock, E. R. (2004). Graph pattern spaces from laplacian spectral polynomials. In *ICIAR*, pages 327–334.

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

Magyar, G., Johnsson, M., and Nevalainen, O. (2000). An adaptive hybrid genetic algorithm for the three-matching problem. *IEEE Transactions on Evolutionary Computation*, 4(2):135–146.

Marcus, M. and Minc, H. (1964). *A Survey of Matrix Theory and Matrix Inequalities*. Allyn and Bacon, Inc.

Marzal, A. and Vidal, E. (1995). Computation of normalized edit distance and applications. *IEEE Trans. Systems, Man, and Cybernetics*, 25:202–206.

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

Merris, R. (1994). Laplacian matrices of graphs: a survey. *Linear Algebra Appl.*, 197-198:143–176.

Merris, R. (1995). A survey of graph laplacians. *Linear Algebra Appl.*, 39:19–31.

Merris, R. and Grone, R. (1987). Algebraic connectivity of trees. *Czechoslovak Math. J.*, 37(112):660–670.

Merris, R. and Grone, R. (1990). Ordering trees by algebraic connectivity. *Graphs Combin.*, 6(3):229–237.

Merris, R. and Grone, R. (1994). The laplacian spectrum of a graph ii. *SIAM J. Discrete Math.*, 7:221–229.

Messmer, B. T. and Bunke, H. (1998). A new algorithm for error-tolerant subgraph isomorphism detection. *IEEE PAMI*, 20:493–504.

Mohar, B. (1991). The laplacian spectrum of graphs, graph theory, combinatorics, and applications. *John Wiley, New York.*, pages 871–898.

Mohar, B. (1992). Laplace eigenvalues of graphs - a survey. *Discrete Math.*, 109:171–183.

Mohar, B. (1997). Some applications of laplace eigenvalues of graphs. *Graph Symmetry: Algebraic Methods and Applications*, 497 NATO ASI Series C:227–275.

Muradyan, D. O. and Piliposyan, T. E. (1980). Minimal numberings of vertices of a rectangular lattice. *Akad. Nauk. Armjan. SRR*, 1(70):21–27.

Myers, R. and Hancock, E. R. (1997). Genetic algorithm parameter sets for line labelling. *Pattern Recognition Letters*, 18(13):1363–1371.

Myers, R., Wilson, R. C., and Hancock, E. R. (2000). Bayesian graph edit distance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(6):628–635.

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

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

Pavan, M. and Pelillo, M. (2003a). Dominant sets and hierarchical clustering. In *ICCV*, pages 362–369.

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

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

Pelillo, M., Siddiqi, K., and Zucker, S. W. (1999). 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.

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

Petit, J. (2000). Combining spectral sequencing and parallel simulated annealing for the minla problem. Technical Report LSI-01-13-R, Departament de Llenguatges i Sistemes Informtics, Universitat Politcnica de Catalunya.

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

Rao, S. and Richa, A. W. (1998). New approximation techniques for some ordering problems. In *9th ACM-SIAM symposium on discrete algorithms*, pages 211–218.

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

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

Roweis, S. and Saul, L. (2000). Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):2323–2326.

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

Sanfeliu, A. and Fu, K. (1983). A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 13:353–362.

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

Scholkopf, B., Smola, A., and Muller, K. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10:1299–1319.

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

Sethian, J. A. (1996). *Level Set Methods*. Cambridge University Press, Cambridge, Books.

Shahrokhi, F., Sýkora, O., Székely, L. A., and Vrto, I. (2001). On bipartite drawings and linear arrangement problem. *J. SIAM Comput.*, 30(6):1773–1789.

Shapiro, L. and Brady, J. (1992). Feature-based correspondence: an eigenvector approach. *Image and Vision Computing*, 10(2):283–288.

Shapiro, L. G. and Haralick, R. M. (1985). A metric for comparing relational descriptions. *PAMI*, 7(1):90–94.

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

Shi, J. and Tomasi, C. (1994). Good features to track. In *CVPR*, pages 593–600.

Shokoufandeh, A., Dickinson, S. J., Siddiqi, K., and Zucker, S. W. (1999). Indexing using a spectral encoding of topological structure. *In Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, pages 491–497.

Silverman, B. W. (1986). *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, London.

Sinclair, A. (1991). Improved bounds for mixing rates of markov chains and multicommodity flow. *LFCS Report Series, ECS-LFCS-91-178, University of Edinburgh.*

Smola, A. and Kondor, R. (2003). Kernels and regularization on graphs. In *In Learning Theory and Kernel Machines, Berlin - Heidelberg, Germany, 2003. Springer Verlag.*

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

Spielman, D. A. and Teng, S. (1996). Spectral partitioning works: Planar graphs and finite element meshes. In *IEEE Symposium on Foundations of Computer Science*, pages 96–105.

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

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

Tenenbaum, J. B., de Silva, V., and Langford, J. C. (2000). A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323.

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

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

Wagner, R. A. and Fischer, M. (1974). The string-to-string correction problem. *J. ACM*, 21(1):168–173.

Weiss, Y. (1999). Segmentatoin using eigenvectors: a unifying view. In *ICCV.*, pages 975–982.

Williams, M., Wilson, R., and Hancock, E. R. (1999). Deterministic search for relational graph matching. *Pattern Recognition*, 32(7):1255–1271.

Wilson, R. C. and Hancock, E. R. (1997). Structural matching by discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):634–648.

Wilson, R. C., Luo, B., and Hancock, E. R. (2005). Pattern vectors from algebraic graph theory. *IEEE PAMI*, 27:1112–1124.

Wong, A. and You, M. (1985). Entropy and distance of random graphs with application to structrual pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:509–609.

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

Yau, S. T. and Schoen, R. M. (1988). *Differential geometry*. Science publication.

Yu, H. and Hancock, E. R. (2005a). Eigenspaces from seriated graphs. In *CAIP*, pages 179–187.

Yu, H. and Hancock, E. R. (2005b). Graph seriation using semi-definite programming. In *GbRPR*, pages 63–71.

Zahn, C. T. (1971). Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Transactions on Computers*, 20:68–86.

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

Zhang, Z. H., Chen, C. B., Sun, J., and Chan, K. L. (2003). Em algorithms for gaussian mixtures with split-and-merge operation. *Pattern Recognition*, 36(9):1973–1983.