Edge-based Operators for Graph Characterization

Furqan Aziz

Submitted for the degree of Doctor of Philosophy

Department of Computer Science
The University of York

February 2014

Abstract

This thesis addresses problems in computer vision and pattern recognition using graphs. The particular focus is on graph matching and characterization using edge-based operators. The thesis commences with a brief introduction in Chapter 1, followed by a review of the relevant literature in Chapter 2. The remainder of the thesis is organized as follows.

Chapter 3 discusses the structure of the Ihara coefficients and presents efficient methods to compute these coefficients. One of our contributions in this chapter is to propose a $O(k|V|^3)$ worst-case running time algorithm to compute the set of first k Ihara coefficients.Chapter 4 proposes efficient methods for characterizing labelled as well as unlabelled graphs. One of our contributions in this chapter is to propose a graph kernel based on backtrackless walks for labelled graphs, whose worst-case running time is the same as that of the kernel defined using random walks.

The next part of the thesis discusses the edge-based Laplacian and its applications. Chapter 5 introduces the concept of a metric graph and the eigensystem of the edge-based Laplacian. Our novel contribution in this chapter is to fully explore the eigenfunctions of the edge-based Laplacian and develop a method for explicitly calculating the edge-interior eigenfunctions. In Chapter 6, we define a wave equation on a graph and give a complete solution. The solution is used to define a signature to classify weighted as well as unweighted graphs. Chapter 7 presents another application of the edge-based Laplacian, where the edge-based heat diffusion process is used to define a signature for points on the surface of a three-dimensional shape. It is called the edge-based heat kernel signature (EHKS) and it can be used for shape segmentation, correspondence matching and shape classification. Finally, in Chapter 8 we provide concluding remarks and discuss directions for future research.

Contents

List of Figures

List of Tables

Acknowledgements

My deepest and sincere gratitude goes first and foremost to my supervisors Professor Edwin Hancock and Professor Richard Wilson for their immense support and encouragement over the past three years. I have been very fortunate to work with them and learn from them. I will always remain indebted to them for their guidance, regular meetings and prompt email replies that helped me complete this dissertation in a timely manner.

My gratitude also goes to my assessor Dr. William Smith for his rigorous assessment of my work. My sincere thanks to all my colleagues and friends at the CVPR group for the help they provided when needed. I would also like to thank Samuel Rota Buló for his collaboration work.

Many thanks are due to the Institute of Management Sciences, Peshawar, Pakistan for granting me the study leave, and to the Higher Education Commission, Pakistan for providing me the financial support.

I am grateful to all my friends in UK for their help and support during my stay at York and for making my life easier and more enjoyable while I was far away from my home. I would also like to thank all my friends back in Pakistan for their regular discussions, motivation and encouragement.

Lastly and most importantly, I wish to thank all my family members for their constant support. My very special thanks go to my mother, my father Abdul Aziz and my brother Imran Aziz to whom I owe everything I am today. I am grateful to them for educating me and preparing me for my future. This work would have been impossible without their unconditional love and support throughout my life.

Declaration

I declare that the research described in this thesis is original work, which I undertook at the University of York during 2010 - 2014. Except where stated, all of the work contained within this thesis represents the original contribution of the author.

Some parts of this thesis have been published in conference proceedings and journals; where items were published jointly with collaborators, the author of this thesis is responsible for the material presented here.

Journal Papers

- Furqan Aziz, Richard C. Wilson, Edwin R. Hancock. Backtrackless Walks on a Graph. In *IEEE Transactions on Neural Networks and Learning Systems*, Volume 24, Issue 6, Pages 977-989.

- Richard C. Wilson, Furqan Aziz, Edwin R. Hancock. Eigenfunctions of the Edge-Based Laplacian on a Graph. In *Journal of Linear Algebra and its Applications*, Volume 438, Issue 11, Pages 4183-4189.

- Samuel Rota Buló, Edwin R. Hancock, Furqan Aziz and Marcello Pelillo. Efficient Computation of Ihara coefficients using the Bell Polynomial Recursion. In *Journal of Linear Algebra and its Applications*,Volume 436, Issue 5, Pages 1436-1441.

Conference Papers

- Furqan Aziz, Richard C. Wilson, Edwin R. Hancock. Analysis of Gaussian Wave Packet Signature of a Graph. In Proceedings of 15th International Conference on *Computer Analysis of Images and Patterns*, 2013, York, UK.

- Furqan Aziz, Richard C. Wilson, Edwin R. Hancock. Graph Characterization using

Gaussian Wave Packet Signature. In Proceedings of 2nd Workshop on *Similarity Based Pattern Recognition*, 2013, York, UK.

- Furqan Aziz, Richard C. Wilson, Edwin R. Hancock. Gaussian Wave Packet on a Graph. In Proceedings of 9th Workshop on *Graph Based Representation*, 2013, Vienna, Austria. LNCS Vol. 7877, Pages 224-233.

- Furqan Aziz, Richard C. Wilson, Edwin R. Hancock. Shape Signature using the Edge-Based Laplacian. In Proceedings of 15th *International Conference on Pattern Recognition* 2012, Tsukuba, Japan. IEEE ICPR, Pages 1594-1597.

- Furqan Aziz, Richard C. Wilson, Edwin R. Hancock. Shape Analysis using the Edge-Based Laplacian. In Proceedings of 9th Workshop on *Structural and Syntactic Pattern Recognition and Statistical Techniques in Pattern Recognition*, 2012, Hiroshima, Japan. LNCS Vol. 7626, Pages 382-390.

- Furqan Aziz, Richard C. Wilson, Edwin R. Hancock. Kernelising the Ihara Zeta Function. In Proceedings of 14th *International Conference on Computer Analysis of Images and Patterns*, 2011, Seville, Spain. LNCS Vol. 6854, Pages 219-227.

- Furqan Aziz, Richard C. Wilson, Edwin R. Hancock. Graph Characterization via Backtrackless Paths. In Proceedings of 1st Workshop on *Similarity Based Pattern Recognition*, 2011, Venice, Italy. LNCS Vol. 7005, Pages 149-162.

Chapter 1

Introduction

This chapter provides an introduction and motivation for the research presented in this thesis. We will explain why we are interested in the graph-based methods for characterization and matching. The chapter commences by introducing the problems encountered in machine learning and pattern recognition and what are the solutions available for these problems. This is followed by our research goals. The chapter concludes by giving an outline of the rest of the thesis.

1.1 The problems

Graphs-based methods have attracted the interest of many researchers and scholars from different areas of research in recent years. This is due to the fact that graphs are considered as one of the most generic data structures. Therefore pattern analysis and learning tasks involving graphs arise in a number of domains such as computer vision, natural language processing, data mining and complex networks. In computer vision, graphs are widely used to abstract images. These graphs are acquired by selecting the feature points from the images which become vertices of the graph. These vertices are then connected based on some criteria, e.g., nearest neighbour. A similar approach is used in three-dimensional computer vision, where the feature points are selected on the surface of the three-dimensional model. A chemical compound can be represented by a weighted labelled graph, where nodes carry the label of atoms and edges represent bonds between nodes. A social network can be represented by a graph where nodes represent persons and edges represent relationships.

Once graphs are extracted, we can perform different tasks such as matching and clus-

tering. However, learning with graphs is difficult. The reason is that graphs are not vectors. Moreover, there is no natural ordering of vertices of the graph and so there is no straightforward way to embed them in a vector space. Due to these reasons, traditional vectorial methods from the statistical machine learning cannot be directly applied to graphs.

Several methods have been proposed to cope with the difficulties that arise in learning with graphs. These methods can be categorized as either inexact methods or decomposition methods. Methods falling into the former category include the use of approximate methods to compute graph-edit distance [13,64], and those falling into the latter category include those that decompose a graph using permutation invariant characteristics. These characteristics are related to the topological structure of the graph. Examples include vertex number, edge number, diameter etc. A more sophisticated approach is to decompose the graph into substructures such as paths, random walks or cycles and use the frequencies of these substructures to embed graphs into a higher-dimensional space.

One of the most successful recent developments in the machine learning community is the use of kernel methods to characterize labelled graphs. For a set of graphs Γ, a graph kernel implicitly embeds graphs into a higher-dimensional feature space. For such a kernel $\kappa : \Gamma \times \Gamma \to \mathbb{R}$, it is known that a map $\psi : \Gamma \to \mathcal{H}$ into a Hilbert space $\mathcal{H}$ exists such that $\kappa(G, G') =< \psi(G), \psi(G') >$, for all $G, G' \in \Gamma$. Kernel methods are popular in the literature and have been extensively used to characterize graphs. However, the problem with the existing kernel methods is that either they are not expressive, or they are computationally very expensive. One of the goals in this thesis is to define efficient graph kernels that overcome the problems with existing kernels and have the same running time.

Over the recent years, there has been an increasing interest in developing new methods for problems arising in the graph-based learning which are based on solutions of partial differential equations defined using the vertex-based Laplacian of the graph. The vertex-based Laplacian is a discrete analogue of the continuous Laplacian in analysis and can be used to translate physical equations from the continuous domain to the discrete graph-theoretic domain. To this end, Xiao et al. [87] have used the heat kernel to embed the nodes of a graph in a Euclidean space. The heat kernel is a solution of the heat equation and is a compact representation of the path-length distribution on a graph. They have used the resulting distribution of the embedded nodes to construct a pattern vector for

characterizing a graph. In related works, Xiao et al. [86] have used the trace of the heat kernel to characterize graphs. Zhang et al. [90] have used the diffusion process for image smoothing. Coifman and Lafon [16] have proposed a probabilistic framework which is based upon diffusion processes on graphs for dimensionality reduction. Escolano et al. [21] have defined the thermodynamic depth complexity for characterizing the graph structure. Recently, Suau et al. [76] have analyzed the Schrödinger operator in the context of graph characterization.

The use of solutions of partial differential equations defined using the vertex-based Laplacian is also becoming increasingly popular in three-dimensional computer vision. Here the goal is to define signatures for points on the surface of a three-dimensional shape that can be used for the purpose of shape matching and clustering. These local signatures can be combined in different ways to define a global signature for shape classification and retrieval. For example, Sun et al. [77] have used the diagonal of the heat kernel to define the so called heat kernel signature (HKS). Castellani et al. [14] have used the HKS to define a global heat kernel signature (GHKS) and have used this for brain classification. Aubry et al. [2] have proposed the wave kernel signature (WKS), which is based on the solution of the Schrödinger equation.

One of the limitations of the discrete Laplacian is that it cannot be used to link most of the results from the analysis of continuous Laplacian to a direct graph-theoretic analogue. As a result, although the heat equation can be mapped from the continuous domain onto a graph, the wave equation cannot be mapped in this way. To translate more complex results from the analysis to the graph-theoretic domain, one way is to treat the edges of the graph as interval of real length. Such graphs are referred to as "Quantum Graphs" in Physics literature. The graph now lives in two spaces, i.e., vertex-space (which is a point-like measure) and edge-space (Lebeguese measure). Functions therefore can exist on both edges and vertices of the graph. This results in two part Laplacian, i.e., the discrete vertex-based Laplacian and the edge-based Laplacian. Although the discrete Laplacian has been extensively used in the literature, little work has been done on graph characterization using solutions of the partial differential equations defined using the edge-based Laplacian. Initial work by ElGhawalby and Hancock [19] has revealed some of the potential uses of the edge-based Laplacian.

1.2 Our goals

Our goal in this thesis is to develop efficient graph-based methods, addressing the problems in computer vision and pattern recognition. Specifically our goals in this thesis are the following:

1. To define efficient vectorial representations for unlabelled graphs to which the statistical methods can be directly applied. In particular, we explore the use of backtrackless walks on a graph and the coefficients of the reciprocal of the Ihara zeta function (also referred to as the Ihara coefficients), which are related to the frequencies of the prime cycles in the graph.

2. To define a graph kernel based on backtrackless walks for characterizing labelled graphs with higher accuracy. We also develop efficient methods that can be used to compute such a kernel, and whose worst-case running is the same as that of the random walk kernel.

3. To study the edge-based Laplacian of a graph. We give methods for explicitly computing the eigenfunctions of the edge-based Laplacian of a graph. This also reveals a connection between the edge-based Laplacian and both random walks and backtrackless walks on a graph.

4. To solve more complex partial differential equations on a graph using the edge-based Laplacian and use its solution for characterizing graphs. In particular, we give a complete solution of a wave equation on a graph, where the initial condition is Gaussian wave packets on edges of the graph, and use it to define vectorial representation for both weighted and unweighted graphs. Such representations can be used to classify graphs with higher accuracy.

5. To define a signature for a point on the surface of a three-dimensional shape using a solution of the heat equation defined using the edge-based Laplacian. Such signatures can be used for correspondence matching and shape segmentation, and can be combined in a number of ways for shape classification and retrieval.

1.3 Contributions

To achieve these goals, we make the following contributions in this thesis.

1.3.1 Efficient computation of Ihara coefficients

The Ihara zeta function has recently attracted attention in the pattern analysis and machine learning literature. For instance, Zhao and Tang [91] have used Savchenko's formulation of the zeta function [68], expressed in terms of cycles, to generate merge weights for clustering over a graph-based representation of pairwise similarity data. Their formulation is based on a representation of oriented line graph, which is an intermediate step in the development of the Ihara zeta function. Watanabe and Fukumizu [80] have presented an approach to the analysis of loopy belief propagation (LBP) by establishing a formula that connects the Hessian of the Bethe free energy with the edge Ihara zeta function. Ren et al. [62] have investigated the use of the Ihara zeta function for clustering graphs. They have extended this work to weighted graphs [61] and hypergraphs [57]. In more recent work, Ren et al. [56] demonstrate a relationship between the Ihara zeta function and the discrete-time quantum walks on graphs.

Unfortunately, despite its attractions as a compact representation of graph structure, applications of the Ihara zeta function have been limited due to the computational overheads required to calculate it. For the graph $G = (V, E)$, with node-set V and edge-set E, the adjacency matrix T of the oriented line graph of G is of size $2|E|$. For graphs of large edge density, this makes computing the spectrum of T and hence the Ihara zeta function or its polynomial coefficients burdensome.

One of our contributions in this thesis is to establish a relationship between the Ihara coefficients and the Bell polynomials. This relationship can then be used for two purposes. Firstly, it can be used to find out interesting properties of each of the Ihara coefficients in terms of frequencies of prime cycles of different lengths in the graph. Secondly, it allows us to efficiently compute the low order Ihara coefficients, which are related to simple cycles of smaller length in the graph. We provide a $O(|V|^3)$ worst-case running time algorithm to compute such coefficients, which is better than the $O(|E|^3)$ worst-case running time of previously known algorithms. Here $|V|$ represents the number of vertices and $|E|$ represents the number of edges in the graph.

1.3.2 Defining efficient graph kernels based on backtrackless walks

Kernel methods are becoming popular for characterizing labelled graphs. One of our contributions in this thesis is to define a kernel based on backtrackless walks on a graph. The idea is based on the random walk kernel, which is one of the most popular graph

kernels. There are two advantages of using backtrackless walks over random walks. Firstly, backtrackless walks avoid tottering, and so it increases the characterization power of the kernel. Secondly, since a backtrackless walk is determined by the adjacency matrix of the oriented line graph, which is closely akin to the discrete time quantum walk on a graph [20], it is less prone to the problem of failing to distinguish graphs due to the cospectrality of the adjacency matrix of the graph. The problem with existing kernels based on backtrackless walks or prime cycles is their computational complexity. In this thesis, we also propose efficient methods to compute such kernels, that can characterize graphs with higher accuracy and whose worst-case running time remains the same as that of the kernel defined using random walks.

1.3.3 Pattern vectors from backtrackless walks and Ihara coefficients

One of the problems with graph structure is that there is no order relation. For this reason graphs cannot be directly embedded in a feature space and traditional statistical methods cannot be directly applied to graphs. Therefore we need methods that can be used to embed graphs in a feature space suitable for machine learning tasks. One way for graph embedding is to use the frequencies of a particular substructure in the graph. In this thesis, our goal is to use frequencies of backtrackless walks and prime cycles of smaller length to embed a graph into a higher-dimensional feature space. A prime cycle is a closed backtrackless and tail-less walk. Since Ihara coefficients are related to the frequencies of prime cycle of smaller length, we use them to embed md2 graphs (i.e., the graphs where the degree of each vertex is at least 2) in feature space. One of our contributions in this thesis is to define pattern vectors based on backtrackless walks and Ihara coefficients and propose efficient methods to compute such pattern vectors.

1.3.4 PDEs using the edge-based Laplacian

In this thesis, we develop methods for explicitly computing the set of all eigenfunctions of the edge-based Laplacian of a graph. Such eigenfunctions can be classified into two types, vertex-supported eigenfunctions and edge-interior eigenfunctions. We prove that the vertex supported eigenfunctions are determined by the structure of random walks on the graph while the edge-interior eigenfunctions are determined by the structure of backtrackless walks on the graph. We give explicit methods to compute the set of all eigenfunctions. Once the eigensystem of the edge-based Laplacian is known, we can use

it to define other complex partial differential equations on graph which are more closely related to equations in analysis. In particular, we give a complete solution of the wave equation and the heat equation using the edge-based Laplacian, where the initial conditions are a Gaussian wave packet and a Gaussian heat packet respectively. We use the solutions of these equations for graph characterization.

1.3.5 Three-dimensional shape signatures

Our last contribution in this thesis is to apply the proposed method for the three-dimensional shape analysis. For this purpose we use the solution of the heat equation, called the heat kernel, which is defined using the edge-based Laplacian of the graph. We define a signature for points on the shape that can be used to embed the shape in a vector space. The proposed signature can be used for correspondence matching and shape segmentation. Moreover local signatures can be combined to define a global signature that can be used for shape retrieval and classification purposes.

1.4 Thesis structure

The remainder of the thesis is organized as follows: Chapter 2 reviews the research literature related to the work presented in the thesis. Chapter 3 presents efficient methods to compute the Ihara coefficients. Chapter 4 presents efficient methods to characterize both labelled and unlabelled graphs using backtrackless walks and Ihara coefficients. In Chapter 5, we give method to explicitly compute the set of all eigenfunctions of the edge-based Laplacian and show its relationship with random walks and backtrackless random walks on a graph. Chapter 6 provides a solution of the edge-based wave equation on a graph where the initial condition is a Gaussian wave packet on the edges of the graph. We also propose methods for characterizing both weighted and unweighted graphs using the solution of the wave equation. In Chapter 7, we use the solution of heat equation defined using the edge-based Laplacian of a graph for three-dimensional shape analysis. Finally, Chapter 8 concludes the work in this thesis and points out possible directions for future research.

Chapter 2

Literature Survey

Graph-based methods are popular tools for high order structure learning [62]. In this thesis, our aim is to develop graph-based methods for high order structures that can be used for characterization, matching and segmentation of such objects. In the light of this aim, in this chapter we review relevant research material on graph theory for characterization and matching.

2.1 Graph models in applications

Graphs are one of the most general data structures. Most of the other data structures can be considered as simple instances of graphs. It is for this reason that graph-based methods are widely used in many applications including network analysis [46], World Wide Web [10], and problems in machine learning [43]. Some of the fields of applications for graphs are given below.

Image analysis: In computer vision, graph-based methods have been successfully used for image recognition, correspondence matching and image segmentation. To extract a graph from images, first the feature points are extracted which become the nodes of the graph. These nodes are then connected according to some defined criteria. Figure 2.1 shows an example.

Three-dimensional shape analysis: A three-dimensional shape can be represented by a mesh that approximates the bounding surface of the shape. Graph-based methods can be used for partial matching and shape retrieval. Figure 2.2 shows an example where the bounding surface of a three-dimensional elephant is approximated by a

Figure 2.1: Graph representation of an image

mesh.

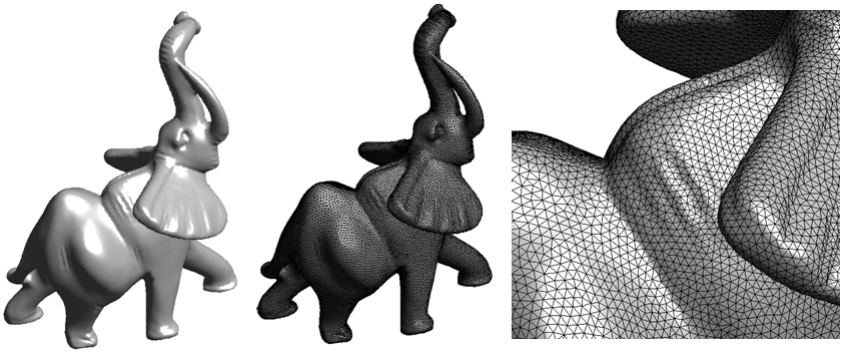


Figure 2.2: Graph representation of a three-dimensional model

Chemoinformatics: Graphs are the natural representation for chemical compounds, where nodes represent atoms and bonds represent edges. The goal here in chemoinformatics is to predict the characteristics of the molecule from their graph structure [7].

Bioinformatics: Graph-based methods are becoming increasingly popular in molecular biology. The most challenging benchmark dataset that originate from bioinformatics is the protein-protein interaction network (PPI) [85]. The objective here is to find the similarity between proteins and enzymes represented in this fashion. Figure 2.3 shows an example, where a protein fragment is modeled as a graph.

Network analysis: Recently, the field of network science has emerged as a new paradigm for the analysis of patterns represented as large and often very complex graphs. Examples of such systems are social networks where nodes represent individuals and edges represent relationships and World Wide Web where nodes represent web

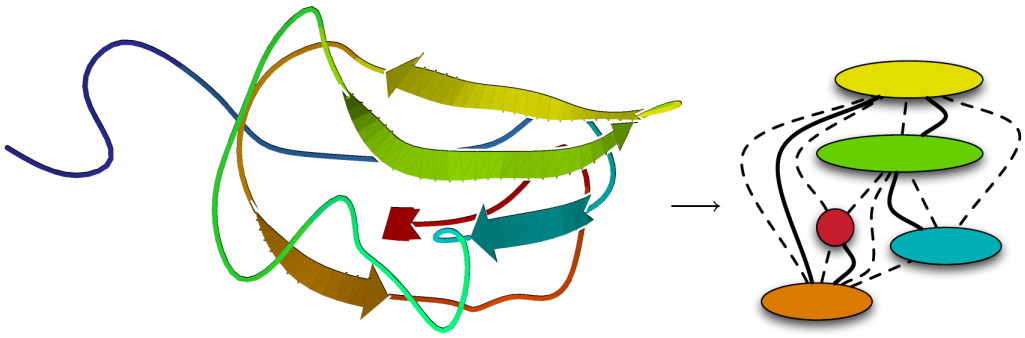


Figure 2.3: Left: Structure of E. coli protein fragment APO-BCCP87 [89], ID 1a6x in the Protein Data Bank [6]. Right: Borgwardt et al.'s [9] graph representation for this protein fragment. Nodes represent secondary structure elements, and edges encode neighbourhood along the amino acid chain (solid) respectively, in Euclidean three-dimensional space (dashed) graph representation of a protein structure [79]

pages and edges represent hyperlinks. Graph-based methods can be used to identify interesting properties of these networks.

2.2 Graph matching and graph kernels

One of the most widely studied problems in the graph theory is to measure similarity between graphs, when the nodes and edges of the graphs are assigned labels. Given two graphs G and G' from the space of graphs Γ, the similarity problem is to find a function

$$sim : \Gamma \times \Gamma \to \mathbb{R},$$

such that $sim(G, G')$ measures the similarity of G and G'. The simplest solution is to find a one-to-one correspondence between the nodes of the two graphs. However, exact graph matching is not practical due to two reasons. Firstly, the graph isomorphism problem is not known to be in P and this makes the exact solution computationally intractable and, secondly, the process of acquiring graphs may introduce some noise that will result in additional/missing edges/vertices. To overcome this problem, inexact and decomposition methods have been used instead. Inexact methods include the use of approximate methods to compute graph edit-distance [13]. Decomposition methods include the idea of decomposing graphs into substructures such as walks [26], paths [8], cycles [31], and trees [53]. Similarity between graphs can be measured using the frequencies of matching substructures.

One of the advantages of the decomposition methods is that they lead to kernels that can be computed in polynomial time. For this reason kernel methods are becoming popular in recent years. However, instead of decomposing a graph into substructures, kernel methods compute the similarities between graphs by assuming an implicit embedding of graphs in a vector space. This usually involves a first step of applying a kernel function $\kappa(G, G')$ to each pair of graphs (G, G') to form a Gram matrix of observed data. In this way, the original data are implicitly mapped into a higher-dimensional feature space. In other words, the kernel implicitly assumes the mapping $\mathcal{T} : \Gamma \rightarrow \psi(G)$ of the graph to a new space. A kernel acts as a dot product in the new feature space, i.e., $\kappa(G, G') =< \psi(G), \psi(G') >$. Although originally developed for vector-data, there has recently been a concerted effort to extend these methods to the structural domain, i.e., to measuring the similarity of strings, trees and graphs. While there is an order relation in strings and trees, graphs represent more difficult structures to kernelise since there is no order relation. Due to this reason the construction of graph kernels has proved to be particularly challenging.

One of the most popular graph kernels is the random walk kernel, which is based on the idea of counting the frequencies of matching walks in the two input graphs [26]. The first step in the random walk kernel is to construct a product graph of the two input graphs in such a way that a walk on the product graph corresponds to a simultaneous walk on each of the two graphs. The kernel can then be computed by counting the number of random walks of different lengths of the product graph. The advantage of random walk kernel is the expressivity of its feature space. Besides, it can be computed in polynomial amount of time and its execution can be further accelerated by using geometric series expansion [26]. Recent literature contains a number of well documented problems with the random walk kernel. These include the problems that different graphs are mapped to the same point in the feature-space of the random walk (this can be attributed to the cospectrality of graphs), and the fact that random walks totter and may visit the same edges and nodes multiple times. Both of these problems mean that the ability of the graph kernel, to discriminate graphs of different structures, is reduced.

To overcome the problems with the random walk kernel, a number of extensions and alternative approaches for defining kernels have been proposed. Mahé et al. [42] have improved the expressivity of the random walk kernel by using label enrichment and using random walks that avoid backtracking. The process of label enrichment adds additional

labels to the nodes of the graph-based on the degree of the node and the degrees of the neighbouring nodes. This increases the accuracy of the resulting kernel. By avoiding backtracking, tottering can be prevented which increases the expressivity of the resulting kernel. Mahé et al. [42] have experimentally shown that these two extensions can increase the performance of the resulting kernel. However, to compute the backtrackless walks, the graph is transformed into a representation that captures the backtrackless structure of the graph. This process introduces additional nodes in the graph, which increases the running time of the resulting kernel and the resulting kernel may not be practical in many real-world situations.

An alternate approach to random walk kernel is the "all-path" kernel which is defined on the set of all edge walks without repetitions of the same edge [8]. In [8], Borgwardt et al. have shown that the "all-path" kernel is a valid graph kernel. However, since finding the set of all paths in a graph is NP-hard, such kernels are not practical. One way to overcome this problem is to use a subset of paths rather than the set of all paths. Borgwardt et al. [8] have defined a kernel which is based on the k-shortest paths in a graph. They have introduced shortest-path kernel and k-shortest-paths kernel, and have experimentally shown that shortest-path kernels can characterize a graph with higher accuracy than the random walk kernel.

As an alternative to the random walk kernel, kernels based on subtrees and cyclic patterns have also been defined. In [31], Horvath et al. have defined a kernel which is based on simple cycles and tree patterns in a graph. To compute the kernel, they first extract the set of all simple cycles from the graph. To add more information to the kernel, they also consider the graph obtained by removing the edges associated with all simple cycles. The resulting graph is a forest consisting of the set of bridges in a graph. These cyclic patterns are then used to construct the kernel. One of the problems with the cyclic pattern kernel is that its computation is NP-hard [31]. Therefore such kernels can only be applied to the families of graphs where the number of cyclic patterns are polynomially bounded, and are hence limited in practical use.

The disadvantages of these methods originate due to competing requirements in the graph kernel design. Borgwardt et al. [8] have identified the following four requirements that a graph kernel should satisfy:

1. The kernel should be a good measure of similarity for graphs.

2. Its computation should be possible in polynomial time.

3. The kernel must be positive definite.

4. It should be applicable to all graphs, not just a small subset of graphs.

Additionally, a kernel should be able to differentiate between non-isomorphic graphs which are cospectral with respect to their matrix representations. Existing graph kernels have difficulties with achieving at least one of these goals. Some of the positive definite graph kernels can be computed in polynomial time, but are not expressive. Others are expressive, but are computationally expensive, sometimes even NP-hard. The problem of defining efficient positive definite graph kernels for measuring the similarity between graphs remains a challenging problem in the machine learning community.

2.3 Pattern vectors

Pattern analysis using graph structures has proved to be a challenging problem. The reason is that graphs have no ordering relations and so they cannot be easily converted to pattern vectors. Hence, the classical statistical methods from pattern recognition or machine learning cannot be directly applied to graphs without first converting them to pattern vectors. To overcome this problem, one way is to extract a pattern vector from a graph that captures the structure of the graph in a way which is permutation invariant. These features can be the number of nodes and edges, the node degrees etc.

Another approach for graph characterization is to extract permutation invariant characteristic from the matrix representation of the graphs. The matrix representation can be the adjacency matrix or the closely related Laplacian matrix of the graph. In [83], Wilson et al. have used the spectral decomposition of the Laplacian matrix and basis sets of symmetric polynomials to convert the graphs into pattern vectors. These pattern vectors are complete, unique and continuous and more importantly they are permutation invariant. They have also explored different methods to embed the vectors in a pattern space suitable for clustering including principle component analysis (PCA), multidimensional scaling (MDS) and locality preserving projection (LLP).

Ren et al. [59] have used the coefficients of the characteristic polynomials of the matrix representation of the graph to embed the graph into a higher-dimensional feature space. The characteristic polynomial is the determinant $det\,(\lambda I - M)$, where I is the identity matrix, M is the matrix representation of the graph, and λ is the variable of the polynomial. With the appropriate choice of the matrix, these coefficients capture the cyclic

structure of the graph [62,70]. They have explored number of different matrix representations, including the adjacency matrix, the Laplacian matrix which is the adjacency minus the diagonal degree matrix, the Perron-Frobenius operator which is the adjacency matrix of a transformed version of the graph, and the adjacency matrix of the positive support of third power of the discrete time quantum walk matrix. They have experimentally shown that polynomial coefficients perform better than the graph spectra.

Another approach for graph embedding, related to the embedding methodology we propose in this thesis, is to use the frequencies of substructures as feature for the vectorial representation of the graph. These substructures can be random walks, shortest paths, or cycles in the graph. For example Kashima et al. [35] use frequent path finding algorithm that finds m frequent paths for constructing feature space. Recently, Ren et al. [62] have explored the use of the Ihara zeta function as a mean of gauging cycle structure in graphs. The Ihara zeta function is computed by first converting a graph into the equivalent oriented line graph, and then computing the characteristic polynomial of the resulting structure. The coefficients of the characteristic polynomials are related to the frequencies of prime cycles of different sizes, and can be computed in polynomial time from the eigenvalues of the adjacency matrix of the oriented line graph. The method can be easily extended from simple graphs to both weighted graphs [61] and hypergraphs [60].

Riesen et al. [65] have proposed a general approach for transforming graphs into n-dimensional real vector spaces by means of prototype selection and graph edit distance computation. The key idea is to use the distances of an input graph to a number of training graphs as vectorial description of the graph. Their method is based on the idea of mapping the pattern vectors into dissimilarity spaces [51] [50]. This idea was also applied to strings by Spillmann et al. in [73]. The main challenge in this method is the selection of prototype. Riesen et al. [65] have proposed the following five prototype selectors

Centres: selects the m prototypes situated in the centre of graph set Γ

Random: randomly selects m prototypes from graph set Γ

Spanning: iteratively selects m prototypes as follows. The first prototype selected is the set median graph. Each additional prototype selected by the spanning prototype selector is the graph which is furthest away from the already selected prototype graphs.

k-centers: choose m graphs from Γ so that they are evenly distributed with respect to

the dissimilarity information.

Targetsphere: This method first selects the centre graph and the graph which is furthest away from the centre. The remaining $m-2$ are selected by locating the graphs that are nearest to the interval borders in terms of edit distance.

Many other approaches have been proposed to embed the graph in a feature space. Jouili et al. [34] have proposed a graph embedding technique based on the constant shift embedding which transforms a graph to a real vector. The constant shift embedding increases all dissimilarities by an equal amount to produce a set of Euclidean distances. This set of distances can be realized as the pairwise distances among a set of points in a Euclidean space. Xiao et al. [87] have used the solution of the heat equation, called the heat kernel, to embed the nodes of a graph into a higher-dimensional Euclidean space. They have used the geometric properties of the resulting embedding to characterize the graph.

2.4 Diffusion processes on graphs

The traditional discrete graph Laplacian Δ_v acts as an operator which is defined only on the vertices of a graph [15]. The discrete Laplacian has proved to be a useful tool in the analysis of graphs and has found applications in a number of areas including computer vision, image processing, and machine learning. For example, Fiedler [22] has used the eigenvector corresponding to smallest positive eigenvalue of the Laplacian for the purpose of partitioning graph.

One of the most popular and successful applications of discrete Laplacian in machine learning and pattern recognition is the dimensionality reduction. The generic problem of dimensionality reduction is the following. Given a set of k points $x_1, x_2, ..., x_k$ in n dimensional space $\mathbb{R}^n$, find a set of k points $y_1, y_2, ..., y_k$ in $\mathbb{R}^m$ $(m << n)$ such that y_i represents x_i. Belkin et al. [3] have used Laplacian eigenmaps that considers the problem of constructing the representation for data lying on a lower-dimensional manifold embedded into a higher-dimensional space. They have shown that the embedding provided by the Laplacian eigenmap is optimal and it preserves the local information. Given k points $x_1, x_2, ..., x_k$ in $\mathbb{R}^n$, the algorithms performs the following three steps:

Constructing the adjacency graph: The adjacency graph is constructed by choosing the points as nodes. Two nodes are connected according to either ϵ-neighbourhoods

(i.e., Nodes i and j are connected by an edge if the Euclidean distance between the two points is less than ϵ) or n-nearest-neighbour (i.e., nodes i and j are connected by an edge if i is among n nearest neighbours of j or vice versa).

Choosing the weights: Two variations for weighting the edges were proposed. The first one assigns a weight of 1 to every edge of the graph. The second one uses the function $w_{ij} = \exp\left(-\|x_i - x_j\|^2/t\right)$. The reason for choosing the later is that it is related to the heat equation on the graph and captures the geometric properties of the data.

Eigenmaps: Once the graph is constructed and weights are assigned, the third step is to compute eigenvalues and eigenvectors for the generalized eigenvector problem $Lf = \lambda Df$, where D is the diagonal weight matrix, and $L = D - W$ is the Laplacian matrix. The eigenmap is constructed by choosing the m eigenvector corresponding to first m smallest positive eigenvalues, i.e., $x_i \rightarrow (f_1(i), f_2(i), ..., f_m(i))$.

The idea of the Laplacian eigenmap was extended by Czaja et al. [84]. They have introduced Schrödinger eigenmap which is a generalization of the Laplacian eigenmap. Schrödinger eigenmap allows experts to input data in the form of additional, labelled information (called potential) on a data-dependent graph to improve the detection and classification processes. This additional information becomes potential for the Schrödinger operator. Hence, Schrödinger eigenmaps not only capture the geometry of the underlying graph, but they also capture the dynamics of the labelled data. The Schrödinger eigenmap can be applied to a wide range of high-dimensional biomedical data analysis problems that require the flexibility to add expert data in a fully automated classification problem [84]. The authors in [84] have applied the method to gene expression analysis and showed higher accuracy results.

Coifman and Lafon [16] have proposed a probabilistic framework which is based upon diffusion processes on graphs for dimensionality reduction. They have used the eigenvectors of Markov matrix (the normalized Laplacian) to construct coordinates called diffusion maps. They have shown that diffusion maps can generate an efficient representation of complex geometric structure and can be used for finding meaningful geometric descriptions of data. Nadler et al. [45] have observed that the eigenvector of the normalized Laplacian are discrete approximation of the eigenfunctions of a Fokker-Plank operator with reflecting boundary conditions.

The partial differential equations on graphs defined using the discrete Laplacian, have

been also used as a tool for anisotropic image smoothing [81]. The anisotropic image smoothing is an implementation of smoothing filters that smoothes the image while preserving the edge details. Tschumperle et al. [78] has proposed a generic framework for anisotropically smoothing multivalued images that uses a solution of a partial differential equation while preserving natural curvature constraints. Zhang et al. [90] have used diffusion process on graphs using the discrete Laplacian for the purpose of anisotropic image smoothing. The first step is to construct a weighted attributed graph from the image, and compute the associated Laplacian matrix. Diffusion across this weighted graph structure with time is captured by the heat equation, and the solution, i.e., the heat kernel, is found by exponentiating the Laplacian eigensystem with time. Image smoothing is affected by convolving the heat kernel with the image. The method has the effect of smoothing within regions, but does not blur region boundaries. They experimented their method with both gray-scale and colour images.

The discrete Laplacian defined over the vertices of a graph, however, cannot link most results in analysis to a graph-theoretic analogue. For example the wave equation $u_{tt} = -\Delta u$, defined with discrete Laplacian, does not have finite speed of propagation. The reason is that the discrete Laplacian is an approximation of Laplacian on discrete points only and hence results from calculus seem unnatural using the discrete Laplacian. Therefore we need improved calculus on graph that can link most of the results from analysis to graph-theoretic domain. In [23, 24], Friedman and Tillich have developed a calculus on graphs which provides strong connections between graph theory and analysis. This approach has a number of advantages. It allows the application of many results from analysis directly to the graph domain, and opens up the use of many new partial differential equations on graphs. As an example, they define a wave equation which has a finite speed of propagation, in contrast to the usual wave equation on a graph [24].

In the graph calculus of Friedman and Tillich, the graph is given a geometric realization by associating an interval with each edge of the graph. Functions may therefore exist both at the vertices and on the interior of edges. From this starting point they develop a divergence and, most importantly, graph Laplacian. This type of Laplacian has found application in the physics literature where the interpretation is as the limiting case of a "quantum wire" [32,66] . The graph Laplacian consists of two parts; namely a vertex-based Laplacian and an edge-based Laplacian. Friedman and Tillich also demonstrate that for edgewise-linear functions the edge-based Laplacian is zero and the graph Laplacian reduces

to the traditional discrete graph Laplacian. On the other hand, for functions where the vertex-based Laplacian is zero, they obtain the edge-based Laplacian only. This results in a setting which is substantially different from the traditional approach. While the method leads to the definition of both a divergence operator and a Laplacian, it is not exhaustive in the sense that the edge-based eigenfunctions are not fully specified.

2.5 Three-dimensional shape analysis

A three dimensional shape can be conveniently represented by a mesh that approximates the bounding surface of the shape. Although such representations are convenient for rendering and visualization purposes, they are not suitable for analysis tasks such as automatic shape matching and automatic shape classification. For the analysis tasks the key idea is to define an informative and discriminative feature descriptor that characterizes each point on the surface of the three dimensional shape. Generally these techniques use a feature vector in $\mathbb{R}^n$ [2,77], which contains both local and global information for that point. These feature descriptors can be used in many ways for analyzing three-dimensional shapes. For correspondence matching, the descriptors are used to find potential correspondence among pairs of points on two different shapes [2,77]. For clustering the parts of a shape, the signatures can be used to identify semantically coherent parts of an object [1, 67]. Local descriptors can be combined in different ways to define a global shape signature and this can be used for shape classification or recognition [14,48]. Ovsjanikov et al. [49] have generalized it to the notion of map that puts in correspondence real-valued functions rather than points on the shapes.

To be effective, Bronstein [11] has enumerated a list of the following desired properties, which a descriptor should have:

Localization: a small displacement of a point on the manifold should greatly affect the descriptor computed at it.

Sensitivity: when a point on a shape is queried against another similar shape, a small set of best matches of the descriptor should contain a correct match with high probability.

Discriminativity: the descriptor should be able to distinguish between shapes belonging to different classes.

Invariance: the descriptor should be invariant or at least insensitive to a certain class of transformations that the shape may undergo.

Efficiency: the descriptor should capture as much information as possible within as little number of dimensions as possible.

Earlier work on three-dimensional shapes was based on the fact that isometric surfaces share the same geometric structure, also known as the "first fundamental form". For example, all possible bendings of a given surface that includes all length preserving deformations without tearing or stretching the surface are considered to be isometric. One of the early works on three-dimensional shape analysis for recognition and classification was the construction of probability distributions and was reported by Osada et al. [48]. They construct a signature for a three-dimensional model as a probability distribution sampled from a shape function measuring the geometric properties of the three-dimensional model. They call this generalization of geometric histogram as a shape distribution [48]. To define their signature they define a number of shape functions based on different geometric properties of the shape. They have experimented with the following functions and have shown their ability to classify three-dimensional models.

A3 Measures the angle between three random points on the surface of a three-dimensional model.

D1 Measures the distance between a fixed point and one random point on the surface. The fixed point is usually chosen as the centroid of the boundary of the model.

D2 Measures the distance between two random points on the surface.

D3 Measures the square root of the area of the triangle between three random points on the surface.

D4 Measures the cube root of the volume of the tetrahedron between four random points on the surface.

Shape distribution has the advantage of being invariant under controlled transformations. However, the retrieval performance of the shape distributions is not sufficient, failing to distinguish shapes that are quite different. To improve its performance, Ohbuchi et al. [47] proposed a modification of D2 shape signature referred to as mutual angle-distance histogram (AD) and mutual absolute-angle distance histogram (AAD) shape feature. Unlike the D2, which is a 1D histogram, they used 2D histograms.

The statistical signatures proposed by Osada et al. [48] and Ohbuchi et al. [47] are simple and somewhat robust to small perturbations. However, its classification rate is considered to be low. Elad et al. [18] have proposed an efficient method for computing bending invariant shape signature for isometric surfaces which is based on geodesic distances. Their method is based on two numerical procedures. The first is the fast marching on triangulated domains (FMTD) [71] that efficiently calculates geodesic distances on triangulated curved surfaces. This is followed by a multidimensional scaling (MDS) technique. They have shown that their approach is useful for nonrigid isometric surface classification and can also be helpful in identifying nonrigid objects that are partially occluded.

Unfortunately, shape signatures defined using geodesic distances are sensitive to the local topology of the shape. As a result such signatures have limited use. Recently, there is an increasing interest in descriptors obtained from the spectral decomposition of the Laplace-Beltrami operator associated with a shape. Reuter et al. [63] have used the eigenvalues of the Laplace-Beltrami operator to define shape signature. They have shown that their signature contains enough information to classify shapes and they have the following properties:

Isometry Congruent solids (or isometric surfaces) should have the same fingerprint being independent of the solid's given representation. For some applications it is necessary that the fingerprint is independent of the object's size.

Similarity Similar shaped solids should have similar fingerprints. The fingerprint should depend continuously on the shape deformation.

Efficiency The effort needed to compute those fingerprints should be reasonable.

Compression In addition, it would also be desirable that the fingerprint data should not be redundant, i.e., a part of it could not be computed from the rest of the data.

Physicality Furthermore, it would be nice if an intuitive geometric or physical interpretation of the meaning of the fingerprints would be available.

Shape signature defined by Reuter et al. [63] fails when two non-isometric shapes share the same spectra. Such shapes are called isospectral shapes. To overcome this problem, Rustamov [67] has defined a shape signature, referred to as global point signature (GPS), which is based on both the eigenvalues and eigenfunctions of the Laplace-Beltrami

operator. This work was also inspired by Lévy's work [40]. In his paper, Lévy [40] has shown that the eigenfunction of the Laplace-Beltrami operator can give interesting information about the geometry of the shape. Possible applications of this representation include signal processing on surfaces, geometry processing, registration and pose transfer, and segmentation and parameterization. Given a point p on the surface, the global point signature is defined as

$$\mathrm{GPS}(p) = \left(\frac{1}{\sqrt{\lambda_1}} \Phi_1(p), \frac{1}{\sqrt{\lambda_2}} \Phi_2(p), \frac{3}{\sqrt{\lambda_3}} \Phi_3(p), ... \right).$$

GPS has many applications including shape segmentation. Local signatures are combined using G2 distribution (a modification of D2 distribution [48]) to give a global signature for shape classification [67]. Although GPS overcomes the problem of failing to distinguish isospectral shapes, it introduces the problem of eigenfunctions sign correction. This is due to the fact that the signs and the ordering of the Laplace-Beltrami eigenfunctions can flip from one pose to another. This makes it difficult to identify segments over different poses. Another limitation of GPS (and the other methods discussed so far) is that it is a global signature and cannot be used to detect partial symmetries or to perform partial matching of articulated shapes.

To overcome these problems with GPS, Sun et al. [77] have used a heat diffusion process to define signatures and this is referred to as the heat kernel signature (HKS). HKS is defined by sampling the retained heat at a number of time points. Given a point p on the surface of the shape, its heat kernel signature ($HKS(x) : \mathbb{R}^+ \to \mathbb{R}$) is defined as

$$HKS(x,t) = k_t(x,x),$$

where $k_t(x,x)$ is the fundamental solution of the Heat equation, called heat kernel. They showed that the HKS contains sufficient information to characterize points uniquely. By sampling the retained heat at different times, they have shown that HKS captures both the local and the global properties of the shape. Hence, it can be used for both partial shape matching and shape classification. HKS has the following desirable properties [77]:

- It organizes information about the intrinsic geometry of a shape in an efficient, multi-scale way.
- It is stable under perturbations of the shape.
- It is concise and commensurable, but remains informative.

- It can be estimated faithfully and efficiently.

Many extensions and modifications of HKS have been proposed. Raviv et al. [55] have extended the idea of HKS to robust isometry-invariant volumetric descriptors and show their utility in shape retrieval. Their work is based on the fact that in many cases modelling shape deformations as approximate isometries of the volume of an object rather than its boundary can capture the properties of non-rigid deformations with higher accuracy. Castellani et al. [14] have used HKS to define a signature for shape classification, called the global heat kernel signature (GHKS). The GHKS is defined by accumulating the local heat kernel values observed at each point into a histogram for a fixed number of scales. They have applied GHKS to magnetic resonance imaging (MRI) data and used it to detect brain morphological abnormalities.

Another physically motivated feature descriptor, the wave kernel signature (WKS), was proposed by Aubry et al. [2], which uses the wave like solutions. WKS was proposed as a solution to the excessive sensitivity of the HKS to low frequency information. WKS is the solution of Schrödinger equation and it represents the average probability of measuring a quantum mechanical particle at a specific location. They have experimentally shown that WKS allows for more accurate feature matching than the HKS. A comparison of properties of WKS and HKS can be found in [11]. The author has analyzed both descriptors and proposed a generic family of descriptors that generalize both the HKS and the WKS.

2.5.1 Summary

In this chapter, we have reviewed the relevant research literature on the spectral graph theory for characterization and matching graphs. We have also analysed deficiencies and problems with the existing methods. Based on the review of the relevant literature, we may draw several conclusions.

Firstly, the Ihara zeta function provides potential approaches to characterize graphs. The original form of the Ihara zeta function is defined on cycle frequencies and thus is closely related to graph topologies. Although the Ihara zeta function is generally an infinite product, its reciprocal can be written as a determinant of a matrix whose coefficients are related to the frequencies of prime cycles in the graph. This identifies a possible way to use these coefficients for graph characterization. However, the computation of the Ihara coefficients requires us to transform the graph into oriented line graph, whose size can be $O(|V|^2)$ in the worst-case. One of our contributions in this thesis is to develop

efficient methods to compute the Ihara coefficients. In Chapter 3, we develop a relationship between Ihara coefficients and Bell polynomial that can be used to efficiently compute these coefficients. This relationship also helps us to understand the structure of each Ihara coefficient in terms of frequencies of prime cycles of different lengths in the graph.

Secondly, to characterize labelled graphs, several graph kernels have been defined on substructures like random walks, shortest paths, and cycles. However, they all suffer from one or the other of the following two problems. Either they are not expressive enough or their high computational costs make them not suitable in practical situations. In Chapter 4, we propose a graph kernel which is based on backtrackless walks on a graph. Such kernel offers a more powerful representation of graph compared to the alternative random walk kernel. Moreover we also propose efficient method to compute such kernels, whose worst-case running time is the same as that of the kernel defined using random walks. To characterize unlabelled graphs, we use pattern vectors that are composed of Ihara coefficients and backtrackless walks on a graph.

Thirdly, the traditional discrete Laplacian defined over the vertices of a graph does not link most of the results from the analysis of continuous Laplacian to a direct graph-theoretic analogue. To overcome this problem, Friedman and Tillich [23] have developed a calculus on graph. This formalism can be used to translate more complex equation from analysis to graph-theoretic domain. While Friedman and Tillich find the eigenvalues of the edge-based Laplacian, and give some of its eigenfunctions explicitly, they do not give a method for computing the entire eigensystem of the graph. In Chapter 5, we develop explicit methods to compute the eigenfunctions of the edge-based Laplacian. This also reveals a connection between the edge-based Laplacian and the adjacency matrix of both the line graph and the oriented line graph.

Fourthly, although the discrete Laplacian has been extensively used in the literature, little work has been done on graph characterization using the solutions of partial differential equations defined using the edge-based Laplacian. Initial work by ElGhawalby and Hancock [19] has revealed some of the potential uses of the edge-based Laplacian. In Chapter 6, we explore the use of the wave equation defined using the edge-based Laplacian for graph characterization. We solve the wave equation on a graph and use its solution to characterize graphs.

Finally, to explore the use of the edge-based Laplacian to solve problems in computer vision, in Chapter 7, we present a novel method for defining pose-invariant signatures for

non-rigid three-dimensional shapes based on the edge-based heat kernel. The signature proves useful for both shape segmentation and feature point correspondence. To illustrate the utility of our method, we apply it to segmenting and classifying non-rigid three-dimensional shapes represented in terms of meshes. One of our novel contributions in this chapter is to define adjacency matrix for the mesh in a way that captures both the topological and geometric properties of the shape itself. We then use the resulting adjacency matrix to find the eigenfunctions of the edge-based Laplacian.

Above all, the work presented in this thesis addresses the shortcomings in the existing methods for characterization and matching graphs and aims to improve their performance and efficiency. We will compare our proposed methods with the state-of-the-art-methods and discuss in detail our contributions to the research literature in the subsequent chapters.

Chapter 3

Ihara Coefficients and Bell Polynomials

The Ihara zeta function has proved to be a powerful tool in the analysis of graph structures [62]. It is determined by the frequencies of prime cycles of a finite graph $G = (V, E)$. The reciprocal of the Ihara zeta function can be characterized in terms of a characteristic polynomial of the adjacency matrix T of the oriented line graph associated to G. The coefficients of this polynomial, referred to as Ihara coefficients, have been used to characterize graphs in a permutation-invariant manner, and allow for an efficient evaluation of the Ihara zeta function.

Despite its attractions as a compact representation of graph structure, the use of Ihara coefficients for graph representation has been limited in the literature due to the computational overheads required to calculate such coefficients. This is due to the fact that to compute the Ihara coefficients, we need to transform the graph into an oriented line graph [62]. For the graph $G = (V, E)$, with node-set V and edge-set E, the adjacency matrix T of the oriented line graph, which is also called the *Perron-Frobenius operator*, is of size $2 \times |E|$ (i.e., twice the number of edges in the graph). Therefore the worst-case running time of computing Ihara coefficients is $O(|E|^3)$. For graphs of large edge density, this makes computing the spectrum of T and hence the Ihara zeta function or its polynomial coefficients burdensome.

In this chapter, we establish a relationship between the Ihara coefficients and the Bell polynomials. This has two advantages. Firstly, it allows us to determine the structure of each of these coefficients in terms of number of simple cycles and prime cycles in the graph. Secondly, this relation can be used to efficiently compute these Ihara coefficients.

We develop an efficient recursive scheme that can be used to compute the set of the Ihara coefficients in $O(|E|^2)$ worst-case time, provided that the eigenvalues of T are known. We also provide an efficient recursive method to compute the set of low order Ihara coefficients in $O(|V|^3)$ worst-case time, which is better than any known method for computing such coefficients.

3.1 Graphs

Before introducing the Ihara zeta function, in this section we provide some basic definitions and notations that will be used throughout the thesis.

3.1.1 Definitions

A *graph* $G = (V, E)$ consists of a finite nonempty set V of *vertices* and a finite set $E \subseteq V \times V$ of unordered pairs of vertices, called *edges*. If $\{u, v\} \in E$, we say that u is *adjacent* to v. A graph is *simple* if it has no edge of the from $\{u, u\}$ and if there are no repeated edges. In the remaining of this thesis we will use the term graph to refer to a simple graph. For a *weighted graph*, there exists a mapping $w : E \to \mathbb{R}$, which assigns a unique weight to every edge of the graph.

A *directed graph* or *digraph* $D = (V_D, E_D)$ consists of a finite nonempty set V_D of vertices and a finite set $E_D \subset V_D \times V_D$ of ordered pairs of vertices, called *arcs*. So a digraph is a graph with an orientation on each edge. For an arc (u, v), v is called the *head* and u is called the *tail* of the arc. For the arc (u, v), the *inverse arc* is (v, u). A digraph D is called *symmetric* if for every arc of D, its inverse arc is also an arc of D. There is a one-to-one correspondence between the set of symmetric digraphs and the set of graphs, given by identifying an edge of the graph with an arc and its inverse arc on the digraph on the same vertices. We denote by $SDG(G)$ the symmetric digraph associated with the graph G. Figure 3.1(b) shows the symmetric digraph of the graph of Figure 3.1(a). The original graph G has 4 vertices and 5 edges, while the symmetric digraph $SDG(G)$ has 4 vertices and 10 arcs.

The *complement* or *inverse* of a graph G is a graph with the same vertex set but whose edge set consists of the edges not present in G. The complement is denoted by $\overline{G} = (\overline{V}, \overline{E})$, where

$$\overline{V} = V,$$

and

$$\overline{E} = \{(u, v) : (u, v) \notin E\}.$$

The *line graph* $LG(G) = (V_L, E_L)$ is constructed by replacing each arc of $D(G)$ by a vertex. These vertices are connected if the head of one arc meets the tail of another. Therefore

$$V_L = \{(u, v) \in D(G)\},$$

$$E_L = \{((u, v), (v, w)) : (u, v) \in D(G), (v, w) \in D(G)\}.$$

The *oriented line graph* $OL(G) = (V_O; E_O)$ is constructed in the same way as the $L(G)$ except that reverse pairs of arcs are not connected, i.e., $((u, v), (v, u))$ is not an arc. The vertex and edge sets of $OL(G)$ are therefore

$$V_O = \{(u, v) \in D(G)\},$$

$$E_O = \{((u, v), (v, w)) : (u, v) \in D(G), (v, w) \in D(G), u \neq w\}.$$

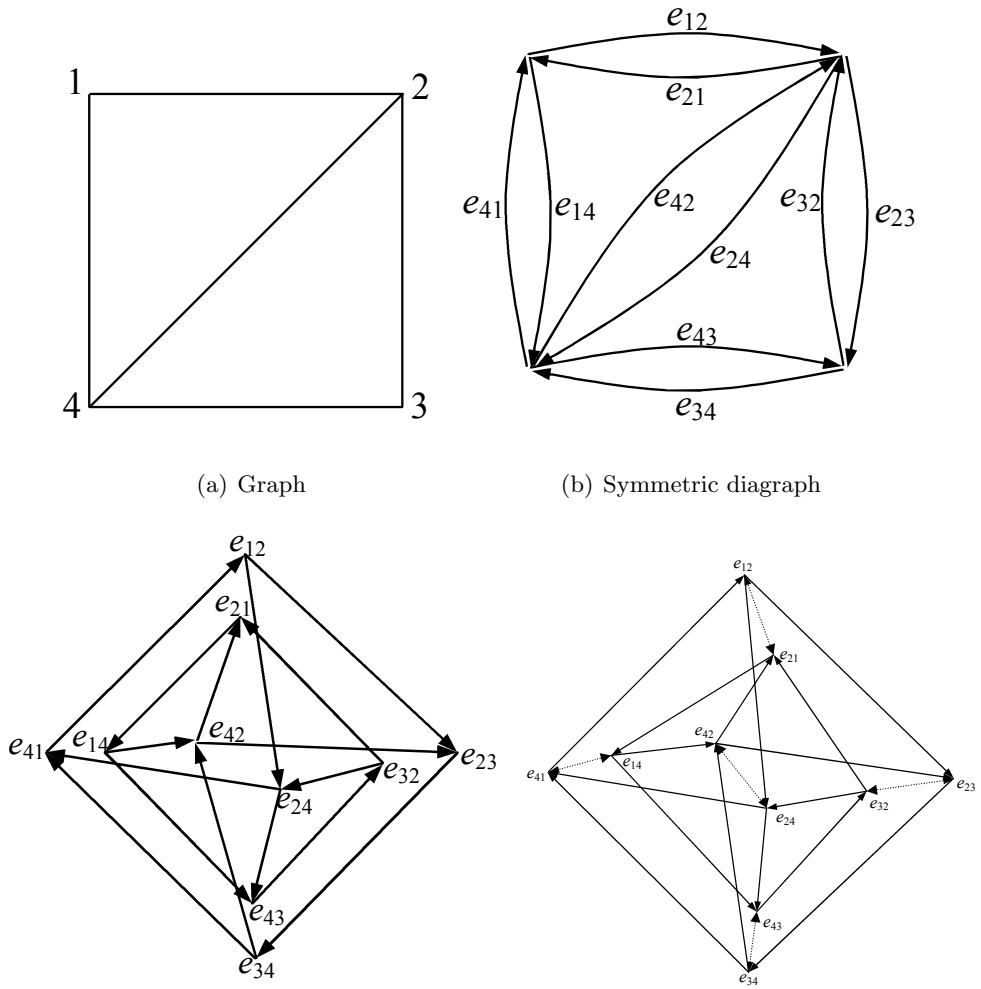
Figure 3.1(c) and Figure 3.1(d) show the oriented line and line graph of the graph of Figure 3.1(a) respectively. Note the dotted arcs in line graph do not exist in the oriented line graph. The number of vertices in both the line graph and oriented line graph are twice the number of edges in the original graph (10 in this case).

3.1.2 Walks on a graph

In this thesis, we need several definitions for walks and cycles.

Definition 3.1.1 A *walk* w of *length* k in a graph is a sequence of vertices $v_1, v_2, ...v_{k+1}$ where $v_i \in V$ such that v_i and v_{i+1} are adjacent. The length of the walk is the number of edges traversed by the walk. A walk has *backtracking* if $v_{i-1} = v_{i+1}$, for some i, $2 \leq i \leq k$, where k is the length of the walk. A walk is *backtrackless* if it has no backtracking. A *path* is a walk that does not have any repeated vertices. A graph can have an infinite number of walks but can only have a finite number of paths.

Definition 3.1.2 A *cycle* of length k is a walk $v_1, v_2, ...v_k$ such that $v_1 = v_k$. A cycle is *simple* if it has no repeated vertices other than v_1 and v_k. The *r-multiple* of a cycle ξ is the cycle ξ^r formed by going r times around ξ. A cycle ξ has a *tail* if ξ has no backtracking but ξ^2 has backtracking. A cycle is *primitive* if it is not the r-multiple of some other cycle for



(a) Graph

(b) Symmetric diagraph

(c) Oriented line graph

(d) Line graph

Figure 3.1: Graph and its transformed versions

$r \geq 2$ (but it can have repeated vertices or edges). We impose an equivalence relation on cycles via cyclic permutation, i.e., two cycles $\xi_1 = (v_1, v_2, ..., v_n)$ and $\xi_2 = (u_1, u_2, ..., u_n)$ are said to be *equivalent* if for some $\alpha \in \mathbb{Z}/n\mathbb{Z}$, $v_i = u_{i+\alpha}$ for all $i \in \mathbb{Z}/n\mathbb{Z}$. Note that the direction of travel does matter so traversing a cycle in the opposite direction does not give a cycle equivalent to the original one. A *prime cycle* is the equivalence class of primitive cycles which have no backtracking or tails, written as $[C]$.

In the graph of Figure 3.2, the sequence $\text{RW} = (v_1, v_2, v_3, v_2, v_5)$ is a random walk, while $\text{BW} = (v_1, v_2, v_3, v_4, v_5)$ is a backtrackless walk. The cycle $\xi_1 = (v_1, v_2, v_3, v_4, v_5, v_2, v_1)$ is a backtrackless cycle but it has a tail (since ξ_1^2 has backtracking). The cycle $\xi_2 = (v_2, v_3, v_4, v_5, v_2)$ is a prime cycle since both ξ_2 and ξ_2^2 are backtrackless cycles. It is also

a simple cycle, because it has no repeated vertices, other than v_2.

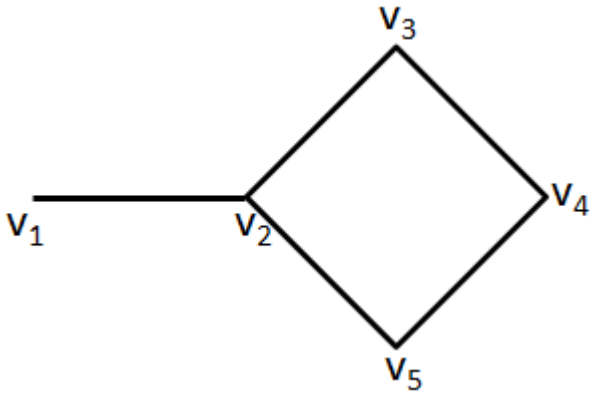


Figure 3.2: A simple graph with 5 vertices

A random walk on the vertices of $L(G)$ represents a sequence of edges traversed by a walker on the original graph G. Similarly, a random walk on the vertices of $OL(G)$ defines a sequence of edges in a random walk on G where backtracking steps are not allowed. This is because the edge $((u,v),(v,u))$ is not included in the $OL(G)$.

3.1.3 Matrix representation for graphs

There are different ways to represent a graph using matrix. The most widely used are the adjacency matrix and the Laplacian matrix.

Definition 3.1.3 The *adjacency matrix* A of a graph $G = (V, E)$ is a $|V| \times |V|$ matrix, whose $(u,v)^{th}$ entry is given as

$$A(u,v) = \begin{cases} 1, & \text{if } (u,v) \in E; \\ 0, & \text{otherwise.} \end{cases}$$

The adjacency matrix of a weighted graph is given as

$$A(u,v) = \begin{cases} w_{uv}, & \text{if } (u,v) \in E; \\ 0, & \text{otherwise.} \end{cases}$$

where w_{uv}, represents the weight of the edge (u,v).

The $(u,v)^{th}$ entry of the n^{th} power of the adjacency matrix of an unweighted graph represents the number of random walks of length n from node u to node v. Since $OL(G)$ prevents backtracking, the $(u,v)^{th}$ entry of the n^{th} power of the adjacency matrix of $OL(G)$

represents the number of backtrackless walks of length $n+1$ from the tail of the edge u to the head of the edge v in the original graph.

Definition 3.1.4 The *Laplacian matrix* L of a graph $G = (V, E)$ is a $|V| \times |V|$ matrix, whose $(u, v)^{th}$ entry is given as

$$L(u,v) = \begin{cases} -1, & \text{if } (u,v) \in E; \\ d(u), & \text{if } u = v; \\ 0, & \text{otherwise.} \end{cases}$$

where $d(u)$ represents the *degree* of the vertex u, i.e., the number of vertices adjacent to the vertex u.

The Laplacian matrix can also be computed as $L = D - A$, where D is the diagonal degree matrix, i.e., the matrix with the vertex degree on diagonal and zeros elsewhere. The Laplacian matrix of a graph is the discrete analogue of the continuous Laplacian operator in the analysis, and therefore it can be used to translate equations from analysis to graph-theoretic domain.

Definition 3.1.5 The *spectrum* of a graph is the set of the eigenvalues associated with the matrix representation of the graph.

Definition 3.1.6 Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are said to be *isomorphic* if there exists a bijection $f : V_1 \to V_2$, such that any two vertices u and v are adjacent in G_1 if and only if $f(u)$ and $f(v)$ are adjacent in G_2. In other word, two graphs are isomorphic, if one can be transformed into the other by relabeling the vertices.

Figure 3.3 shows an example of isomorphic graphs, where the mapping is $\{1, 2, 3, 4, 5\} \to \{2, 1, 3, 4, 5\}$.

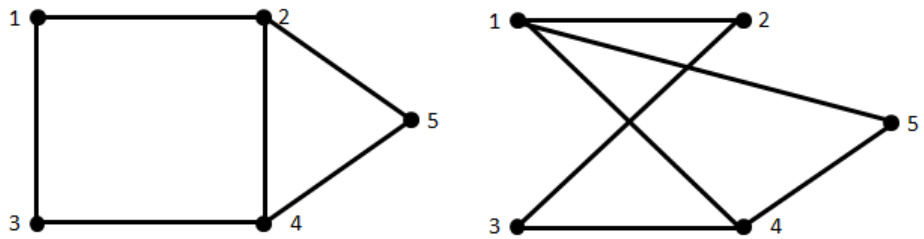
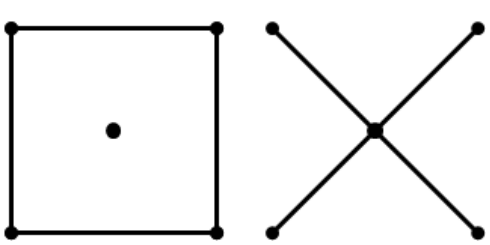


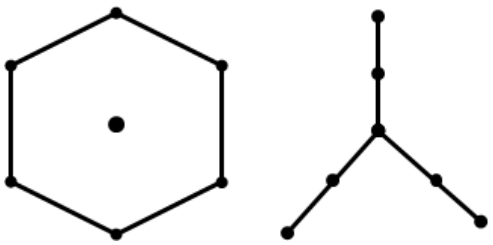
Figure 3.3: Isomorphic graphs

Definition 3.1.7 Let A_1 and A_2 be the adjacency matrices of two non-isomorphic graphs G_1 and G_2 respectively and $\overline{A_1}$ and $\overline{A_2}$ be the adjacency matrices of their complement graphs, i.e., $\overline{G_1}$ and $\overline{G_2}$ respectively. G_1 and G_2 are called cospectral with respect to their adjacency matrices, if A_1 and A_2 have the same spectrum. Similarly G_1 and G_2 are cospectral with respect to the adjacency matrices of their complements, if $\overline{A_1}$ and $\overline{A_2}$ have the same spectrum.

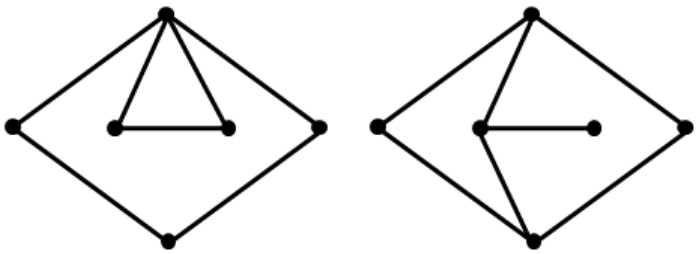
Non-isomorphic graphs can be cospectral with respect to other matrix representations. Figure 3.4 shows some examples of cospectral graphs [72]. The graphs of Figure 3.4(a) are cospectral with respect to their adjacency matrices, sharing the same spectrum, i.e., $\{-2, 0, 0, 0, 2\}$. The graphs of Figure 3.4(b), on the other hand, are cospectral with respect to both their adjacency matrices and the adjacency matrices of their complements. The spectrum of these graphs is $\{-2, -1, -1, 0, 1, 1, 2\}$, and the spectrum of the complements of these graphs is $\{-2, -2, -1.3723, 0, 0, 1, 4.3723\}$. Similarly the graphs of Figure 3.4(c) are cospectral with respect to their Laplacian matrices, sharing the spectrum $\{0, 0.7639, 2, 3, 3, 5.2361\}$.



(a) Cospectral graphs with respect to their adjacency matrices



(b) Cospectral graphs with respect to adjacency matrices of graphs as well as their complement



(c) Cospectral graphs with respect to their Laplacian matrices

Figure 3.4: Examples of cospectral graphs

3.2 The Ihara zeta function

We now introduce the Ihara zeta function.

3.2.1 Definition

The *Ihara zeta function* associated with a graph G is a function of the complex variable $u \in \mathbb{C}$ defined as

$$\zeta_G(u) = \prod_{c \in [C]} \left(1 - u^{l(c)}\right)^{-1}, \tag{3.1}$$

where $[C]$ runs over the set of all equivalence classes of prime cycles of G, and $l(c)$ denotes the length of the prime cycle.

The Ihara zeta function can also be expressed in terms of a power series of the variable u [37]:

$$\zeta_G(u) = \exp\left(\sum_{m=1}^{\infty} \frac{N_m}{m} u^m\right), \tag{3.2}$$

where N_m represents the number of prime cycles of length m.

One of the advantages of the Ihara zeta function is that its inverse, $\zeta_G(u)^{-1}$, can be written in the form of a determinant of a matrix of size $2|E| \times 2|E|$, whose coefficients are related to the frequencies of prime cycles in the graph. This makes the use of the Ihara zeta function of practical utility.

3.2.2 The reciprocal of the Ihara zeta function

The Ihara zeta function can also be written in the form of a determinant expression [37]:

$$\zeta_G(u) = \frac{1}{\det(I - uT)}, \tag{3.3}$$

where T, the Perron-Frobenius operator, is the adjacency matrix of the oriented line graph of the original graph. The size of T is $2|E| \times 2|E|$, where $|E|$ denotes the cardinality of E, i.e., the number of edges of G, and I is the $2|E| \times 2|E|$ identity matrix. Since the reciprocal of the Ihara zeta function can be written in terms of a determinant of the matrix T, therefore it can be expressed in the form of a polynomial of degree at most $2|E|$ (and exactly $2|E|$ for $md2$ graphs, i.e., the graphs where the degree of each vertex is at least two.), i.e.,

$$\zeta_G(u)^{-1} = \det(I - uT) = c_0 + c_1 u + c_2 u^2 + c_3 u^3 + ... + c_{2|E|} u^{2|E|}, \tag{3.4}$$

where $c_0, c_1, ..., c_{2|E|}$ are the coefficient of the reciprocal of the Ihara zeta function, referred to as the Ihara coefficients.

The above coefficients can be computed as a summation of a series of determinants [12, 58]:

$$c_n = \sum_{\binom{2|E|}{2|E|-k}} \begin{vmatrix} b_{1,1} & b_{1,2} & ... & b_{1,2|E|} \\ b_{2,1} & b_{2,2} & ... & b_{2,2|E|} \\ \vdots & \vdots & \ddots & \vdots \\ b_{2|E|,1} & b_{2|E|,2} & ... & b_{2|E|,2|E|} \end{vmatrix} . \tag{3.5}$$

This requires the computation of $\binom{2|E|}{2|E|-k}$ determinants of size $2|E| \times 2|E|$ to find one coefficient c_k.

3.3 Ihara coefficients and Bell polynomials

In this section, we give our first major contribution. We provide an explicit representation of the Ihara coefficients in terms of the complete Bell polynomials. Later we will show how this relationship can be used to relate the low order Ihara coefficient to the frequencies of simple cycles in the graph. We will also use this relationship to develop algorithms for efficiently computing Ihara coefficients, whose worst-case running time is better than the previously known algorithms.

3.3.1 Bell polynomials

Bell polynomial is a combinatorial identity, named in honor of Eric Temple Bell. We distinguish between partial and complete Bell polynomials. The *partial Bell polynomials* are a triangular array of polynomials given by

$$B_{n,k}(x_1, \ldots, x_{n-k+1}) = \sum \frac{n!}{\prod_{\ell=1}^{n-k+1} j_\ell!} \prod_{\ell=1}^{n-k+1} \left(\frac{x_\ell}{\ell!}\right)^{j_\ell} , \text{ for } 1 \le k \le n ,$$

where the sum extends over all sequences $j_1, j_2, \ldots, j_{n-k+1}$ of non-negative integers such that

$$\sum_{\ell=1}^{n-k+1} j_\ell = k \quad \text{and} \quad \sum_{\ell=1}^{n-k+1} j_\ell \, \ell = n .$$

The sum of the partial Bell polynomials $B_{n,k}(x_1, \ldots, x_{n-k+1})$ over all values of k gives the *complete Bell polynomials* $B_n(x_1, \ldots, x_n)$, i.e.,

$$B_n(x_1, \ldots, x_n) = \sum_{k=1}^{n} B_{n,k}(x_1, \ldots, x_{n-k+1}) .$$

The complete Bell polynomials $B_n(x_1, \ldots, x_n)$ have the following generating function:

$$\exp\left(\sum_{n=1}^{\infty} \frac{x_n}{n!} u^n\right) = \sum_{n=0}^{\infty} \frac{1}{n!} B_n(x_1, \ldots, x_n) u^n, \tag{3.6}$$

and, additionally, it can be shown that they satisfy the following identity:

$$B_n(x_1, \ldots, x_n) = \begin{vmatrix} x_1 & \binom{n-1}{1} x_2 & \binom{n-1}{2} x_3 & \binom{n-1}{3} x_4 & \ldots & x_n \\ -1 & x_1 & \binom{n-2}{1} x_2 & \binom{n-2}{2} x_3 & \ldots & x_{n-1} \\ 0 & -1 & x_1 & \binom{n-3}{1} x_2 & \ldots & x_{n-2} \\ 0 & 0 & -1 & x_1 & \ldots & x_{n-3} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & -1 & x_1 \end{vmatrix}. \tag{3.7}$$

The Bell polynomial has an interesting combinatorial meanings. If the integer n is partitioned into a sum in which "1" appears j_1 times, "2" appears j_2 times, and so on, then the number of partitions of a set of size n that collapse to that partition of the integer n when the members of the set become indistinguishable is the corresponding coefficient in the polynomial.

3.3.2 Relationship between Ihara coefficient and Bell polynomial

An interesting relation between the Ihara coefficients and the complete Bell polynomials can be established by combining Equation 3.2 and Equation 3.4, i.e.,

$$\sum_{m \geq 0} c_m u^m = \exp\left(-\sum_{m \geq 1} \frac{N_m}{m} u^m\right). \tag{3.8}$$

The coefficient c_k, can be computed by evaluating the k^{th} derivative of Equation 3.6 at $u = 0$. The first five Ihara coefficients are

$$\begin{aligned} c_0 &= 1, \\ c_1 &= -N_1, \\ c_2 &= \frac{1}{2!}\left(-N_2 + N_1^2\right), \\ c_3 &= \frac{1}{3!}\left(-2N_3 + 3N_2 N_1 - N_1^3\right), \\ c_4 &= \frac{1}{4!}\left(-6N_4 + 8N_3 N_1 + 3N_2^2 - 6N_2 N_1^2 + N_1^4\right). \end{aligned}$$

In general, the n^{th} Ihara coefficient is given by

$$c_n = \sum_{k_1, k_2, \ldots, k_n} \left(-\frac{x_1}{1}\right)^{k_1} \left(-\frac{x_2}{2}\right)^{k_2} \ldots \left(-\frac{x_n}{n}\right)^{k_n}, \tag{3.9}$$

where $k_1 + 2k_2 + 3k_1 + ... + nk_n = n$ and $x_k = -(k-1)!N_k$. Hence, we can write c_n in terms of Bell polynomials:

$$c_n = \frac{1}{n!}\left(\sum_{k=1}^{n} B_{n,k}\left(x_1, x_2, ..., x_{n-k+1}\right)\right), \tag{3.10}$$

$$= \frac{1}{n!}B_n\left(x_1, x_2, ..., x_n\right), \tag{3.11}$$

where $B_{n,k}\left(x_1, x_2, ..., x_{n-k+1}\right)$ are partial Bell polynomials and $B_n\left(x_1, x_2, ..., x_n\right)$ is the complete Bell polynomial.

The following theorem proves this relationship of the Ihara coefficients in terms of the complete Bell polynomials.

Theorem 3.3.1 Let G be a graph, T be the adjacency matrix of $OL(G)$ and let c_n denote the n^{th} Ihara coefficient related to G. The following identity holds:

$$c_n = \frac{B_n(\alpha_1, \ldots, \alpha_n)}{n!},$$

where $\alpha_\ell = -(\ell-1)!N_l$.

Proof 3.3.2 By Equation 3.2 and Equation 3.6 we have

$$\begin{aligned}
\zeta_G^{-1}(u) &= \exp\left(-\sum_{n=1}^{\infty}\frac{N_n}{n}u^n\right), \\
&= \exp\left(\sum_{n=1}^{\infty}\frac{\alpha_n}{n!}u^n\right), \\
&= \sum_{n=0}\frac{1}{n!}B_n(\alpha_1, \ldots, \alpha_n)u^n.
\end{aligned}$$

Differentiating the above equation n times, and evaluating the result at $u = 0$, we get

$$\frac{d^n}{du^n}\zeta_G^{-1}(0) = B_n(\alpha_1, \ldots, \alpha_n).$$

Now the coefficients c_n of the reciprocal of the Ihara zeta function can be derived from Equation 3.4 in terms of derivatives of $\zeta_G^{-1}(u)$ evaluated at $u = 0$ as follows:

$$c_n = \frac{1}{n!}\frac{d^n}{du^n}\zeta_G^{-1}(0). \tag{3.12}$$

From the above results we conclude

$$c_n = \frac{B_n(\alpha_1, \ldots, \alpha_n)}{n!}.$$

□

This result justifies a possible method for computing the Ihara coefficients, which makes use of Equation 3.7. Indeed, one could compute the Ihara coefficient by calculating the determinant of a matrix, which has in general complexity $O(|E|^3)$. Since each α_i depends on N_i, the number of prime cycles of length i, the relations also provides a possible method for determining the structure of each Ihara coefficient in terms of frequencies of prime cycles of different lengths.

3.4 The structure of Ihara coefficients

As mentioned earlier, the relationship that we have established in Section 3.3 between Ihara coefficients and the Bell polynomials can be used to characterize the structure of each Ihara coefficient. In particular, we give an alternate proof of the following theorem where we prove that the low order Ihara coefficients are related to the frequencies of simple cycles in the graph. This theorem was first proved by Scott and Storm [70]. However, the relationship established in the previous section between Ihara coefficients and the Bell polynomials helps simplify the proof.

Theorem 3.4.1 *(Scott and Storm [70])* Let G be a simple graph with $\zeta_G(u)$ be its Ihara zeta function. Let $c_1, c_2, ..., c_n$, be its Ihara coefficients. Then $c_1 = c_2 = 0$. Furthermore, c_3, c_4 and c_5 are the negatives of twice the number of triangles, squares, and pentagons in G respectively.

Proof 3.4.2 This can be proved by using the relationship between the Ihara coefficients and the Bell polynomials.

- Since $c_1 = -N_1$ and since there are no loops in a simple graph so c_1 is always zero.
- Similarly c_2 is always zero since $c_2 = \frac{1}{2!}\left(-N_2 + N_1^2\right)$ and there are no multiple edges or loops in a simple graph.
- $c_3 = \frac{1}{3!}\left(-2N_3 + 3N_2N_1 - N_1^3\right) = -\frac{1}{3}N_3$. Since G is a simple graph, N_3 depends on the number of triangles in a graph. Each triangle will contribute 6 to N_3. This is due to the fact that each node in a triangle gives us two backtrackless and tailless paths of length 3 traversed in opposite directions. Therefore $N_3 = 6 \times$ number of triangles in G and hence the coefficient c_3 is equal to the negative of twice the number of triangles in the graph G.

- Similarly the coefficients $c_4 = -\frac{1}{4}N_4$ and $c_5 = -\frac{1}{5}N_5$, and so c_4 and c_5 are equal to the negative of twice the number of rectangles and pentagons in the graph G respectively. □

Note that the high order Ihara coefficients are related to number of prime cycles of different lengths. For example $c_6 = -\frac{1}{6}N_6 + \frac{1}{18}N_3^2$, so c_6 depends on the number of prime cycles of length 6 and the number of triangles in the graph. Indeed, in [70] Scott and Storm have shown that the coefficient c_6 is the negative of twice the number of hexagons in G plus four times the number of pairs of edge disjoint triangles plus twice the number of pairs of triangles with a common edge, while c_7 is the negative of twice the number of heptagons in G plus four times the number of edge disjoint pairs of one triangle and one square plus twice the number of pairs of one triangle and one square that share a common edge. Hence, the high order Ihara coefficients may give redundant information about the structure of the graph.

Since the structure of the Ihara coefficients is determined by the frequencies of prime cycles in the graph, they can be used to determine the girth of the graph.

Definition 3.4.3 For a graph $G = (V, E)$, the girth of G is the length of the shortest cycle in G.

We now show that the relationship between the Ihara coefficients and Bell polynomials can be used to show that the girth of the graph is determined by the Ihara coefficients. This result was first shown by Horton in his Ph.D dissertation [30] and later by Scott and Storm [70].

Theorem 3.4.4 *(Horton [30])* Let r be the girth of a simple graph $G = (V, E)$. Then, $c_k = 0$ for $1 \leq k < r$. Moreover, c_r is the negative of twice the number of cycles of length r.

Proof 3.4.5 From the relationship between the Ihara coefficients and Bell polynomials, it is straightforward to show that $c_k = 0$ for $1 \leq k < r$. This is because each c_k, for $1 \leq k < r$ depends on the number of cycles of length smaller than r, but the length of smallest cycle in the graph is r.

Using the relationship between the Ihara coefficients and Bell polynomials again, it can be shown that $c_r = -\frac{1}{r}N_r$, where r is the girth of the graph and N_r depends on the number of simple cycles of length r (since there are no cycles of smaller length in G). Each

cycle will contribute $2r$ to N_r, because each node in the cycle will give two closed paths of length r traversed in opposite directions. Therefore N_r = Negative of twice the number of cycles of length r in G. This completes the proof. □

This shows that the smallest non-zero Ihara coefficient c_i is determined by the number of cycles of length i in the graph. Indeed, using Theorem 3.3.1, it can be shown that if r is the girth of the graph $G = (V, E)$, then $c_i = 0$, whenever $0 < i < r$, and c_i = negative of twice the number of cycles of length i, whenever $r \leq i < 2r$.

3.5 A recursive formula for the complete Bell polynomials

We introduce here a recursive formula that can be used for the computation of the Bell polynomials. This result will then play a fundamental role in next section, where we propose our efficient method for computing the Ihara coefficients.

Although the recursive relation for the Bell polynomials is already known (see, *e.g.* [4, 5]), we provide here a self-contained proof for completeness. To this end, we prove the following lemma first, which can be considered as a special instance of the general Leibniz rule for the n^{th} derivative of the product of functions.

Lemma 3.5.1 Let $h(u) = \exp(f(u))$. Then for all $n \geq 1$,

$$h^{(n)}(u) = \sum_{\ell=0}^{n-1} \binom{n-1}{\ell} f^{(n-\ell)}(u) h^{(\ell)}(u)\,,$$

where $f^{(n)}(u)$ and $h^{(n)}(u)$ denote the nth-order derivatives of f and h, respectively.

Proof 3.5.2 We proceed by induction. For $n = 1$ we can easily see that

$$h^{(1)}(u) = \frac{d}{du}\left[\exp(f(u))\right] = f^{(1)} \exp(f(u)) = f^{(1)}(u) h(u)\,.$$

For the general case n we have

$$\begin{aligned} h^{(n)}(u) &= \frac{d}{du} h^{(n-1)}(u) \\ &= \left[\sum_{\ell=0}^{n-2} \binom{n-2}{\ell} f^{(n-\ell)}(u) h^{(\ell)}(u)\right] + \left[\sum_{\ell=0}^{n-2} \binom{n-2}{\ell} f^{(n-1-\ell)}(u) h^{(\ell+1)}(u)\right], \end{aligned}$$

where we used the inductive hypothesis for $h^{(n-1)}(u)$. By taking the term $\ell = 0$ out of the first summation and the term $\ell = n-2$ out of the second one, we obtain after simple

algebra:

$$\begin{aligned} h^{(n)}(u) &= f^{(n)}(u)h(u) + f^{(1)}(u)h^{(n-1)}(u) \\ &\quad + \sum_{\ell=0}^{n-3} \binom{n-2}{\ell+1} f^{(n-1-\ell)}(u)h^{(\ell+1)}(u) + \binom{n-2}{\ell} f^{(n-1-\ell)}(u)h^{(\ell+1)}(u), \\ &= f^{(n)}(u)h(u) + f^{(1)}(u)h^{(n-1)}(u) + \sum_{\ell=0}^{n-3} \binom{n-1}{\ell+1} f^{(n-1-\ell)}(u)h^{(\ell+1)}(u), \\ &= \sum_{\ell=0}^{n-1} \binom{n-1}{\ell} f^{(n-\ell)}(u)h^{(\ell)}(u)\,. \end{aligned}$$

□

Theorem 3.5.3 The complete Bell polynomials can be expressed using the following recursive formula:

$$B_n(x_1, \ldots, x_n) = \begin{cases} \sum_{\ell=0}^{n-1} \binom{n-1}{\ell} x_{n-\ell} B_\ell(x_1, \ldots, x_\ell), & \text{if } n > 0 \\ 1, & \text{otherwise}\,. \end{cases}$$

Proof 3.5.4 Let $f(u) = \sum_{n=1}^{\infty} \frac{x_n}{n!} u^n$. Then it easy to see that $f^{(n)}(0) = x_n$. Similarly, let $h(u) = \sum_{n=0}^{\infty} \frac{1}{n!} B_n(x_1, \ldots, x_n) u^n$. Then $h^{(n)}(0) = B_n(x_1, \ldots, x_n)$. By Equation 3.6 and by Lemma 3.5.1 the result derives. □

3.6 Efficient computation of Ihara coefficients

The recursive formula for the computation of the Bell polynomials introduced in Theorem 3.5.3, combined with the fact that the Ihara coefficients can be expressed in terms of the Bell polynomials, leads to a novel, easy and efficient way of computing the Ihara coefficients. Our final contribution in this chapter is to provide efficient recursive methods for computing the Ihara coefficients, which are based on the following theorem.

Theorem 3.6.1 Let G be a graph and T be the adjacency matrix of $OL(G)$. The Ihara coefficients related to G can be computed through the following recursive formula:

$$c_n = \begin{cases} -\frac{1}{n} \sum_{\ell=0}^{n-1} (N_{n-\ell})\, c_\ell, & \text{if } n > 1; \\ 1, & \text{otherwise}\,. \end{cases}$$

Proof 3.6.2 By Theorem 3.5.3 and Lemma 3.3.1 we have that

$$c_n = \frac{1}{n!}\sum_{\ell=0}^{n-1}\binom{n-1}{\ell}\alpha_{n-\ell}\ell! c_\ell = \sum_{\ell=0}^{n-1}\frac{\alpha_{n-\ell}}{n(n-1-\ell)!}c_\ell = -\frac{1}{n}\sum_{\ell=0}^{n-1}(N_{n-\ell})\,c_\ell\,.$$

□

3.6.1 Computing the set of Ihara coefficients

Using the fact that the number of prime cycles can be computed from the trace of the matrix T, the coefficient of the reciprocal of the Ihara zeta function can be recursively computed as

$$c_n = \begin{cases} -\frac{1}{n}\sum_{\ell=0}^{n-1}\mathrm{Tr}\left(T^{n-\ell}\right)c_\ell, & \text{if } n > 1; \\ 1, & \text{otherwise}\,. \end{cases}$$

Note that $\mathrm{Tr}\,(T^n)$ can be computed efficiently in terms of the eigenvalues $\lambda_1, \ldots, \lambda_{|E|}$ of matrix T, since $\mathrm{Tr}\,(T^n) = \sum_{i=1}^{|E|}\lambda_i^n$. Assuming the eigenvalues of T are known, we can compute, by means of Theorem 3.6.1, the Ihara coefficients incrementally starting from $n = 1$ by exploiting for each new coefficient the previously computed ones. By doing so, the complexity of the computation of all the Ihara coefficients is $O(|E|^2)$. It is worth noting that this complexity becomes $O(|E|^3)$ if the spectrum of T is not available, since a preliminary eigenvalue decomposition of T is required in order to compute the terms $\mathrm{Tr}\,(T^n)$.

3.6.2 Computing low order Ihara coefficients

As mentioned in Section 3.4, the low order Ihara coefficients are related to frequencies of simple cycles of small lengths in a graph. The high order Ihara coefficients, on the other hand, can give us redundant information. One of our goals in this thesis is to use the low order Ihara coefficients for characterizing graphs. In order to compute the low order Ihara coefficients, we give an algorithm that computes a constant number of low order Ihara coefficients and runs in $O(|V|^3)$ time in worst-case.

We commence by introducing a $|V| \times |V|$ matrix A_k for a graph $G = (V, E)$, whose $(u, v)^{th}$ entry is the total number of backtrackless walks of length k, for $k \geq 1$, i.e.,

$$[A_k]_{u,v} = \text{number of paths in } G \text{ of length } k$$
$$\text{with no backtracking starting at } u \text{ and ending at } v.$$

Here (u, v) runs over the vertices of G. Since there is no backtracking in paths of unit length, we define $A_1 = A$. To locate the paths of higher lengths, we use the following theorem [75] that recursively computes the matrix A_k for $k \geq 2$.

Theorem 3.6.3 *(Stark and Terras [75])* Let A be the adjacency matrix of a graph G and Q be a $|V| \times |V|$ diagonal matrix whose u^{th} diagonal entry is the degree of the u^{th} node minus 1. Then

$$A_k = \begin{cases} A, & \text{if } k = 1; \\ A^2 - (Q + I), & \text{if } k = 2; \\ A_{k-1}A - A_{k-2}Q, & \text{if } k \geq 3. \end{cases} \tag{3.13}$$

The proof of the above theorem can be found in [75]. □

Note that A_2 can be computed using one matrix multiplication and two matrix additions, and therefore it requires $O(|V|^3 + |V|^2)$ operations. Similarly A_k, for $k \geq 3$, is recursively computed and requires one matrix addition and two matrix multiplications, and therefore a total of $O(|V|^3 + |V|^2)$ operations. The worst-case running time of computing A_k is therefore $O(k \times |V|^3)$. When k is bounded by $O(1)$, then the running time of computing A_k becomes $O(|V|)^3$.

To efficiently compute the low order Ihara coefficients, we use the fact that the number of prime cycles of length n can be computed from the matrices A_k as [75]:

$$N_m = \mathrm{Tr}\left(A_m - (Q - I) \sum_{j=1}^{[(m-1)/2]} A_{m-2j} \right). \tag{3.14}$$

Using the above equation we can compute N_m in $O(m \times |V|^3)$ worst-case time. This is because the computation of each A_m requires $O(m \times |V|^3)$ time. Hence, the worst-case running time of computing N_m is $O(|V|^3)$, when m is bounded by $O(1)$, i.e., when m is constant. The execution can be accelerated by using dynamic programming and storing the temporary sums in memory. Once $N_1, N_2, ..., N_m$ are known, we can use the following recursive formulation to compute the Ihara coefficients:

$$c_n = \begin{cases} \frac{-1}{n} \sum_{k=0}^{n-1} N_{n-k} c_k, & \text{if } n > 1; \\ 1, & \text{otherwise.} \end{cases} \tag{3.15}$$

The worst-case running time of computing the first n Ihara coefficients is bounded by $O(n|V|^3)$. To compute the fixed number of Ihara coefficients the worst-case running time

becomes $O(|V|^3)$. It is worth noting that the complexity becomes $O(|E| \times |V|^3)$, if we wish to compute the set of all Ihara coefficients.

3.7 Summary

In this chapter, we have developed a relationship between the Ihara coefficients and the Bell polynomial. This relationship allows us to study the structure of each of these coefficients as well as to compute these coefficients in an efficient way. The Ihara coefficients are related to prime cycles in the graph, and therefore can be used to characterize the graphs in a permutation invariant manner. We have provided a method that computes the set of all Ihara coefficients in $O(|E|^2)$ time, provided the spectrum of the oriented line graph is known. We have also provided a method to compute the set of low order Ihara coefficients in $O(|V|^3)$ time.

Chapter 4

Backtrackless Walks on a Graph

In this chapter, we explore the use of backtrackless walks and prime cycles for characterizing graphs. The reason for using backtrackless walks and prime cycles is that they avoid tottering, and hence can increase the discriminative power of the resulting graph representation. We present methods for characterizing both labelled and unlabelled graphs. For labelled graphs, we define graph kernels using backtrackless walks on the graph. For unlabelled graphs, we construct pattern vectors that are composed of backtrackless walks and Ihara coefficients. We also provide efficient methods to compute these kernels and pattern vectors. For graph kernel, we present an $O(|V_\times|^6)$ time algorithm, which is better than the $O(|V_\times|^{12})$ worst-case running time of the previously known algorithms. Similarly for unlabelled graphs, we present an $O(|V|^3)$ algorithm which is better than the $O(|V|^6)$ worst-case running time of the previously known algorithms. In the experimental evaluation, we apply the proposed methods to cluster labelled as well as unlabelled graphs. The results show that the use of backtrackless walks and prime cycles instead of random walks can increase the accuracy of recognition.

Another advantage of using backtrackless walks and Ihara coefficients is that these methods are based on the adjacency matrix of the oriented line graph, which is closely related to the discrete time quantum walk on a graph [56]. Emms et al. [20] have demonstrated the ability of Quantum walks to distinguish strongly regular graphs. This justifies the use of backtrackless walks for the purpose of distinguishing non-isomorphic cospectral graphs. Indeed, in the experimental section we will demonstrate that the random walk cannot distinguish graphs which are cospectral with respect to their adjacency matrices as well as the adjacency matrices of their complements. On the other hand, pattern vectors constructed from the Ihara coefficients or backtrackless walks can distinguish such graphs.

4.1 Graph kernels

In this section, we provide methods for characterizing labelled graph using graph kernels. We commence by introducing a graph kernel and define the random walk kernel. Next we discuss backtrackless walk kernels and provide efficient methods to compute such kernels.

Definition 4.1.1 A *labelled graph* is a graph along with an additional set of labels $\mathcal{L}$ and a function $l : V \cup E \to \mathcal{L}$ which assigns a label to each edge and/or vertex of the graph.

A labelled graph can be *vertex-labelled* which assigns labels to the vertices only, *edge-labelled* which assigns labels to the edges only, or *fully-labelled* which assigns labels to both edges and vertices. Any edge-labelled (or vertex-labelled) graph can be considered fully labelled if we consider that all of the vertices (or edges) of the graph are assigned the same label. Similarly an unlabelled graph can be considered as a labelled graph, that assigns the same label to each vertex and edge of the graph. If a labelled graph G is transformed to a line graph $LG(G)$ or an oriented line graph $OL(G)$, then the vertices and edges of the $LG(G)$ or $OL(G)$ are assigned the same labels of the corresponding edges and vertices of G respectively.

Definition 4.1.2 A *graph kernel* is a positive definite kernel on the set of graphs Γ. For such a kernel $\kappa : \Gamma \times \Gamma \to \mathbb{R}$ it is known that a map $\psi : \Gamma \to \mathcal{H}$ into a Hilbert space $\mathcal{H}$ exists, such that $\kappa(G, G') = \langle \psi(G), \psi(G') \rangle$ for all $G, G' \in \Gamma$ [69].

Graph kernels can be defined on different substructures such as random walks [26], shortest paths [8], cyclic patterns [31], and trees [53] in the graph. One of the most popular polynomial time algorithms for graph characterization is the random walk kernel [26]. The idea behind the random walk kernel is to measure the similarity between graphs based on the frequencies of matching random walks of different lengths.

As mentioned earlier, one of the problems with the random walk graph kernel is that of tottering. A tottering walk can move to one direction and then immediately returns to the starting position. This results in many redundant paths in the graphs. For example in the chemical data structure of citric acid in Figure (4.1), the sequence O=C-C-C-C-H may correspond to a walk on 6 different atoms (red dashed in the figure) or a tottering walk on 4 different atoms (green solid in the figure). These redundant paths may decrease the discriminative power of the resulting kernel.

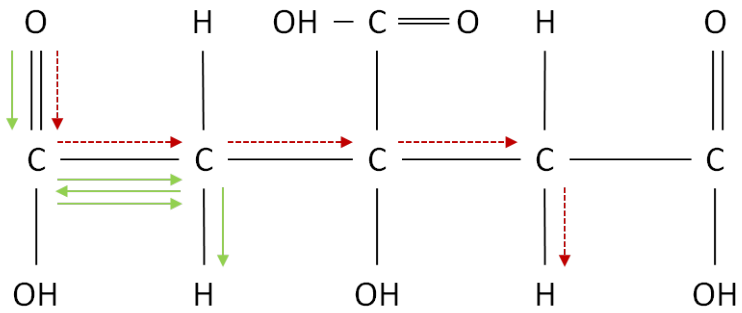


Figure 4.1: Structure of citric acid

One way to avoid this problem is to use the set of all paths or simple cycles in the graph because a path or a cycle does not contain repeated edges and therefore avoids tottering. However, finding the set of all paths or cycles is an NP-hard problem and is not practical in most cases [8] [31]. Therefore such kernels can only be applied to a limited families of graphs, where the numbers of simple cycles or paths can be computed in polynomial time. Another way to avoid tottering is to use backtrackless walks or prime cycles instead of random walks. In this section, we present methods for characterizing graphs using backtrackless walks on the graph. We give methods for characterizing both labelled and unlabelled graphs and propose efficient methods to compute such kernels. The worst-case running time of the proposed method is the same as that of the method which uses random walks on the graph.

4.1.1 Random walk kernel

The random Walk Kernel is based on the idea of counting the number of matching random walks of different lengths in the two input graphs. Instead of decomposing the graphs into random walks of different lengths, Gärtner et al. [26] have proposed an efficient method for locating all the pairs of matching random walks in two input labelled graphs, which is based on the idea of direct product graph (labelled). A direct product graph is a construction that allows an implicit embedding of the input graphs into a higher-dimensional feature space. It is defined as follows:

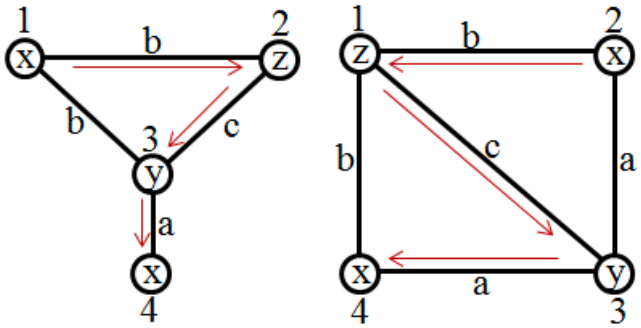
Definition 4.1.3 Given two input graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, their direct product graph (labelled), $G_\times = (V_\times, E_\times)$, is defined as

$$V_\times(G_\times) = \{(v_1, v_2) \in V_1 \times V_2 : label(v_1) = label(v_2)\},$$

and

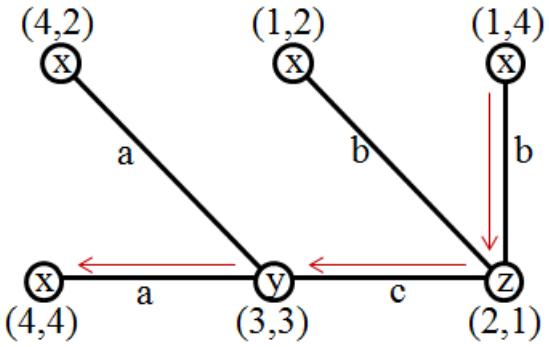
$$E_\times(G_\times) = \{((u_1, u_2), (v_1, v_2)) \in V_\times^2(G_\times) : \\ (u_1, v_1) \in E_1 \wedge (u_2, v_2) \in E_2 \wedge label(u_1, v_1) = label(u_2, v_2)\}.$$

Figure 4.2(c) shows an example of a direct product graph of the two input graphs of Figure 4.2(a) and Figure 4.2(b). A labelled random walk on the direct product graph corresponds to a labelled random walk on the two input graphs. For example the labelled sequence $(x, b, z, c, y, a, x, a, y)$ in the product graph corresponds to a labelled random walk in each of the input graphs. Similarly a labelled backtrackless walk on the direct product graph corresponds to a labelled random walk on the two input graphs. For example, in the case of Figure 4.2, the labelled sequence (x, b, z, c, y, a, x) in the product graph corresponds to a labelled backtrackless walk in each of the input graphs.



(a) Labelled graph

(b) Labelled graph



(c) Direct product graph

Figure 4.2: Two labelled graphs and their direct product graph

Definition 4.1.4 Given two input labelled graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, the random walk kernel can be computed using the direct product graph $G_\times = (V_\times, E_\times)$ as:

$$\kappa(G_1, G_2) = \sum_{u,v=1}^{|V_\times|} \sum_{k=0}^{\infty} \epsilon_k \left[X^k\right]_{u,v}, \tag{4.1}$$

where X is the adjacency matrix of $G_\times$ and $\epsilon_1, \epsilon_2, ...$ is a sequence of constants chosen in such a way that the kernel converges [26]. Note that, we define $X^0 = I_{|V|\times|V|}$ which counts the total number of random walks of length 0, i.e., the number of matching pairs of vertices in the two input graphs.

4.1.2 Backtrackless walk kernel

In this section, we introduce the backtrackless walk kernel and propose efficient method for computing such a kernel. Backtrackless walk kernel is a modification to random walk kernel, and is based on the idea of counting the number of matching backtrackless walks instead of random walks in the two input graphs. To define a kernel on backtrackless walks, one way is to convert the graph into a form that captures the backtrackless structure of the graph. This can be done by transforming the graph G into oriented line graph, $OL(G)$, since the $OL(G)$ captures the backtrackless structure of the graph. The kernels can be defined on the oriented line graph of the direct product graph. Another transformation was proposed by Mahé in [42], which is defined as follows:

Definition 4.1.5 Given a graph $G = (V, E)$, the transformed graph, $G' = (V', E')$, is defined as

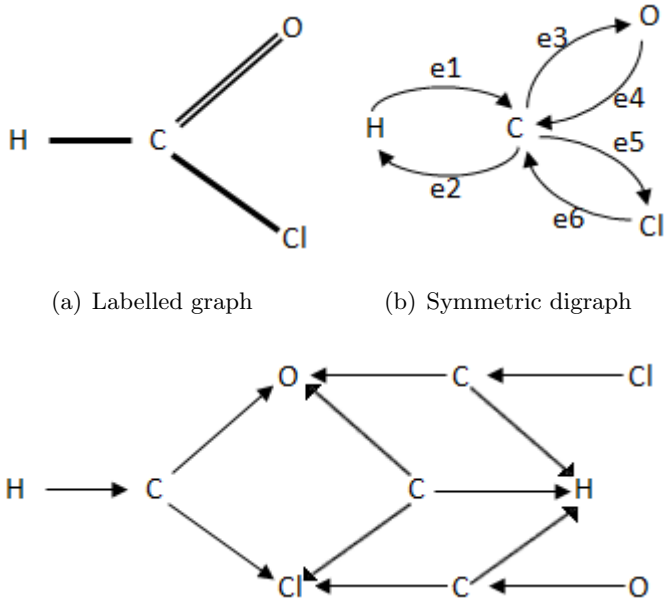
$$V' = V \cup E,$$

and

$$\begin{aligned} E' &= \{(v, (v, w)) : v \in V, (v, w) \in E\} \\ &\cup \{((u, v), (v, w)) : (u, v), (v, w) \in E, u \neq w\}. \end{aligned}$$

The transformed graph is labelled as follows. For a node $v' \in V$, the label is $l'(v') = l(v')$, if $v' \in V$, or $l'(v') = l(v)$ if $v' = (u, v) \in E$. For an edge $e' = (u', v')$ between two vertices u' and $v' \in V \cup E$ and $v' \in E$, the label is simply given by $l'(e') = l(v')$. The construction of G' and its labeling are illustrated in Figure 4.3.

A random walk on the transformed graph corresponds to a backtrackless walk on the original graph and a random walk on the product graph of the transformed graph corresponds to a random walk on each of the transformed graphs (and hence a backtrackless



(a) Labelled graph (b) Symmetric digraph

(c) Transformed graph

Figure 4.3: A labelled graph and its transformed graph

walk on each of the original graphs). Therefore the product graph of the transformed graphs of the two input graphs can be used to count the number of matching backtrackless walks in the two input graphs. The backtrackless walk kernel for the two input graphs G_1 and G_2 is defined in a similar way as

$$\kappa\left(G_1, G_2\right) = \sum_{u,v=1}^{|\mathcal{V}_\times|} \sum_{k=0}^{\infty} \epsilon_k \left[Y^k\right]_{u,v}, \tag{4.2}$$

where Y is the adjacency matrix of the direct product graph of the two transformed graphs (instead of the adjacency matrix of the direct product graph of the original graphs).

However, the use of backtrackless kernels in practice is limited because of the computational cost of such kernels. For a graph $G = (V, E)$, the size of the transformed graph is $|E| + |V|$. Therefore, the size of the product graph of the transformed graphs is $(|E_1| + |V_1|) \times (|E_2| + |V_2|)$. In the worst-case, when $|E| = O\left(|V|^2\right)$, the size of the product graph can be $O\left(|V_1|^2 \times |V_2|^2\right)$. In such cases, the computational cost of the kernel can be $O\left(|V|^{12}\right)$, when both G_1 and G_2 have $|V|$ vertices. This is due to the fact that the computation of the kernel requires the inversion of the adjacency matrix of the product graph [79], and this generally requires cubic time. For this reason, kernels based on transformed graphs may not be practical in most cases.

To overcome the above mentioned problem, in this chapter we present a method that efficiently computes a kernel based on backtrackless walks using the adjacency matrix of the original graph instead of the transformed graph. For this purpose we use the $|V| \times |V|$ matrix A_k defined in Section 3.6.2, whose $(u, v)th$ entry is given by

$$[A_k]_{u,v} = \text{number of walks in } G \text{ of length } k \text{ with no backtracking starting at } u \text{ and ending at } v.$$

Here u, v run over the vertices of G. Since there is no backtracking in paths of unit length, we define $A_1 = A$. Also we define $A_0 = I_{|V|\times|V|}$, which counts the total number of backtrackless walk of length 0. To locate the walks of higher lengths, we use the theorem 3.6.3 [75] that recursively computes the matrix A_k for $k \geq 2$, i.e., we compute each A_k, for $k \geq 0$ as

$$A_k = \begin{cases} I_{|V|\times|V|}, & \text{if } k = 0; \\ A, & \text{if } k = 1; \\ A^2 - (Q + I), & \text{if } k = 2; \\ A_{k-1}A - A_{k-2}Q, & \text{if } k \geq 3. \end{cases}$$

We now define backtrackless walk kernel using the adjacency matrix of the direct product graph.

Definition 4.1.6 Given two input graphs G_1 and G_2, the backtrackless walk kernel is defined as

$$\kappa(G_1, G_2) = \sum_{u,v=1}^{|V_\times|} \sum_{k=0}^{\infty} \epsilon_k [X_k]_{u,v}. \tag{4.3}$$

where X is the adjacency matrix of $G_\times$. The (u, v)th entry of X_k is the number of backtrackless walks of length k in G, starting from vertex u and ending at vertex v. Note that X_0 counts the total number of backtrackless walk of length 0, i.e., the number of matching pairs of vertices in the two input graphs.

The kernel defined here is a valid positive definite kernel if we choose a sequence of positive coefficients ϵ_i such that (4.3) converges [79]. Here we propose to choose $\epsilon_i = \epsilon^i$ for $i \geq 1$ and $0 < \epsilon < 1$. The value of ϵ depends on the particular dataset to which we are applying the kernel. In practice, we approximate the infinite sum of Equation 4.3 by computing it for the first k_{max} terms, where k_{max} is some small and fixed number.

The reason for ignoring the backtrackless walks of higher length is that they contain some redundant information and, therefore, can reduce the performance of the resulting kernel.

4.1.3 Timing analysis

In this section, we have proposed a method to compute a kernel, which is based on backtrackless walks in a graph. However, it remains an important question that how the proposed method compares to known methods in terms of computational complexity. Here we compare the time complexity and the execution time of the proposed method with alternative methods.

Suppose we are dealing with two graphs $G_1 = (V_1, E_2)$ and $G_2 = (V_2, E_2)$. Suppose that the number of vertices in both graphs are bounded by n, i.e., $|V_1| = O(n)$ and $|V_2| = O(n)$. The random walk kernel is computed from the adjacency matrix of the direct product graph, whose size in the worst-case can be $O(n^2)$. Since the random walk kernel is computed by matrix multiplication or matrix inversion [79], which generally requires cubic time in terms of the size of the matrix, therefore the worst-case running time of the random walk kernel is $O(n^6)$.

To compute the backtrackless walk kernel using transformed graph, we need to find the direct product graph of the transformed graphs of the original graphs. The size of the transformed graph of the graph G is $O(|V| + |E|)$ and the size of the product graph of the transformed graphs of G_1 and G_2 is $O\left((|V_1| + |E_1|) \times (|V_2| + |E_2|)\right)$. In the worst-case, when the graph is dense, i.e., when $E = O(V^2)$, the size of each transformed graph can be $O(n^2)$ and the size of the direct product graph can be $O(n^4)$. Therefore the worst-case running time of the backtrackless walk kernel using transformed graph can be $O(n^{12})$.

Finally, the proposed kernel, defined in Equation 4.3, is computed using the adjacency matrix of the direct product graph of the original graphs, whose size in worst-case is $O(n^2)$. The running time of the propose kernel is therefore $O(n^6)$, if we successively compute A_k using recursion of Theorem 3.6.3. Note that, while the worst running time of the proposed kernel and the random walk kernel are the same, in practice the random walk kernel runs a little faster than the one defined here. This is due to the fact that to compute A_k (for $k > 2$), we require two matrix multiplications, while to compute A^k, we require only one matrix multiplication.

We now compare the execution time of the proposed method with alternative methods. For this purpose, we randomly generate 1000 graphs, each with 100 nodes and around 500

edges. For each graph, we compute random walks of length 10, backtrackless walks of length 10 using graph transformation method [42], and backtrackless walks of length 10 using proposed method. The average execution time for each method on 1000 graphs is shown in Table 4.1. The table also shows the values of the standard error. The standard error (SE) estimates the standard deviation of the sample mean and It is defined as the standard deviation divided by the square root of the number of samples, i.e.,

$$SE = \frac{\sigma}{\sqrt{n}},$$

where σ is the standard deviation and n is the number of samples. From the results of Table 4.1, It is clear that even on sparse graphs our method performs very well compared to that based on the transformed graph. Note that, as mentioned earlier, the execution time for the random walk is slightly less than that of the backtrackless walk.

Table 4.1: Execution time comparison

Method	Execution time (ms)	Standard error (ms)
Random walk kernel	2.7035	0.0237
Backtrackless walk kernel (proposed)	3.1584	0.0274
Backtrackless walk kernel (transformed)	129.7705	0.9141

4.2 Pattern vectors

Although the kernel defined in (4.3) can be used for both labelled and unlabelled graphs, it can be very inefficient in case of unlabelled graphs. This is due to the fact that the size of the product graph is $|V_1| \times |V_2|$, when the input graphs have no labels. The running time can be improved by a method called label enrichment [46], which assigns labels to each vertex of the graph based on its degree and the degree of its neighbours. However, in the case where the degrees distribution for a graph is nearly uniform (e.g., in Delaunay triangulation) then such methods cannot be very useful.

Unlabelled graphs can be efficiently characterized by defining a vector representation for the graph based on the frequency with which a particular substructure appears. This will allow us to embed the graph into a higher-dimensional feature space. In this section, we use backtrackless walks and Ihara coefficients for embedding the graphs in a vector space. We also provide efficient methods for computing these pattern vectors.

4.2.1 Pattern vectors using backtrackless walks

To characterize unlabelled graphs, we use the pattern vector $\overrightarrow{bw}(G) = [l_1, l_2, ..., l_k]$, where l_n is the number of backtrackless walks of length n, and k is the length of the pattern vector. As with the backtrackless walk kernel, the high order terms in the pattern vector are ignored since they contain redundant information. The value of l_1 can be computed from the adjacency matrix of the graph. To compute l_n for $n \geq 2$, one way is to use T, the adjacency matrix of the oriented line graph. As a result the coefficients l_n for $n \geq 2$ can be computed as

$$l_n = \sum_{i,j=1}^{|\mathcal{V}|} \left[T^{n-1}\right]_{i,j} . \tag{4.4}$$

The running time of computing each l_n using (4.4) is $O(|E|^3)$ which in worst-case is $O(|V|^6)$. To reduce the computational cost, we use A_n, that can be computed using the recursion in Theorem (3.6.3), instead of T^{n-1}. So we compute each l_n, for $n \geq 1$, as

$$l_n = \sum_{i,j=1}^{|\mathcal{V}|} [A_n]_{i,j} . \tag{4.5}$$

Using (4.5), the cost of computing each l_n in worst-case is now $O(|V|^3)$.

One of the disadvantages of using the pattern vector defined above is that walks with larger lengths may give rise to redundant information. One way to overcome this problem, is to define the pattern vector as $\overrightarrow{bw}(G) = [\epsilon_1 l_1, \epsilon_2 l_2, ..., \epsilon_k l_k]$, where $(\epsilon_1, \epsilon_2, ..., \epsilon_k)$ is a sequence of weights. These weights are assigned in such a way that the walks of shorter lengths get higher weights. We select these weights in the same way as we did for labelled graphs, i.e., we choose $\epsilon_i = \epsilon^i$ for $i \geq 1$ and for some $0 < \epsilon < 1$. Another approach to choose these constants is to use the exponential decay function, i.e., $\epsilon_i = \exp(-\lambda i)$, where λ is the decay constant. One of the disadvantages of using proper weights is that the value of ϵ (or λ) is not fixed and need to be learned from the training dataset.

To compare the performance of the pattern vector $\overrightarrow{bw}(G)$, we also define pattern vector composed of random walks. Such a pattern vector can be constructed by using the higher powers of the adjacency matrix, i.e., the pattern vector of length k from random walks for the graph G is defined as $\overrightarrow{rw}(G) = [w_1, w_2, ..., w_k]$, where

$$w_n = \sum_{i,j=1}^{|\mathcal{V}|} [A^n]_{i,j} . \tag{4.6}$$

Here A is the adjacency matrix of the graph G. As with the pattern vector $\overrightarrow{bw}(G)$, another approach is to define the pattern vector from random walks as $\overrightarrow{rw}(G) = [\epsilon_1 w_1, \epsilon_2 w_2, ..., \epsilon_k w_k]$.

Here $(\epsilon_1, \epsilon_2, ..., \epsilon_k)$ is a sequence of weights, chosen in the same way as for the pattern vector from backtrackless walks.

4.2.2 Pattern vectors from Ihara coefficients

In Chapter 3, we have shown that the Ihara coefficients are related to the frequencies of prime cycles in the graph and can be computed in polynomial time. Since a prime cycle is defined as a closed backtrackless and tail-less walk, Ihara coefficients are directly related to closed backtrackless (and tail-less) walks on a graph. Here we propose to use the low order Ihara coefficients of the Ihara zeta function for clustering graphs. The reason is that the low order Ihara coefficients are related to simple cycles of small lengths. The high order Ihara coefficients, on the other hand, can give us redundant information. In Chapter 3, we have proposed an algorithm that computes the constant number of low order Ihara coefficients and runs in $O(|V|^3)$ time in worst-case.

To characterize graphs using the Ihara coefficients we propose to use the pattern vector $\overrightarrow{ic}(G) = [c_3, c_4, c_5, ...c_k]$ for the graph G, where c_i is the i^{th} Ihara coefficient. Since $c_0 = 1$, $c_1 = 0$, and $c_2 = 0$, we have ignored these coefficients in the pattern vector. Also since, as mentioned earlier, the high order Ihara coefficients contain some redundant information, we only keep the low order Ihara coefficients. As with the pattern vector for backtrackless walks, a more sophisticated approach is to define the pattern vector using a sequence of weights chosen in such a way that the high order Ihara coefficients are assigned lower weights. The reason for assigning smaller weights to higher Ihara coefficients is that the coefficients besides c_3, c_4 and c_5 provide redundant information. So an alternative approach is to use the pattern vector $\overrightarrow{ic}(G) = [\epsilon_1 c_3; \epsilon_2 c_4; \epsilon_3 c_5; ...\epsilon_k c_{k+2}]$, where $(\epsilon_1, \epsilon_2, ..., \epsilon_k)$ is a sequence of weights and can be chosen in the same way as for $\overrightarrow{bw}(G)$.

The difference between the pattern vector $\overrightarrow{bw}(G)$ and the pattern vector $\overrightarrow{ic}(G)$ is worth noting. The latter depends on the number of closed backtrackless and tail-less walks in the graph and can be computed from the diagonal of the matrix T. The former depends on the number of all the backtrackless walks in the graph and hence uses all the information contained in the matrix T. So the pattern vector $\overrightarrow{ic}(G)$ can be useful when the graph exhibits a cyclic structure (e.g. Delaunay triangulation). However, when the graph has too many branches (e.g. chemical structures of molecules or Gabriel graphs), then the pattern vector $\overrightarrow{bw}(G)$ can be more informative.

4.3 Experiments

In this section, we apply our proposed method to both real-world and synthetic datasets. The purpose of the experiments on synthetic datasets is to evaluate whether the backtrackless walks can distinguish between different graphs under controlled levels of similarity (i.e., structural modifications) measured using graph edit distance. For the real-world data we have selected two different datasets, Mutag [74] and COIL [44].

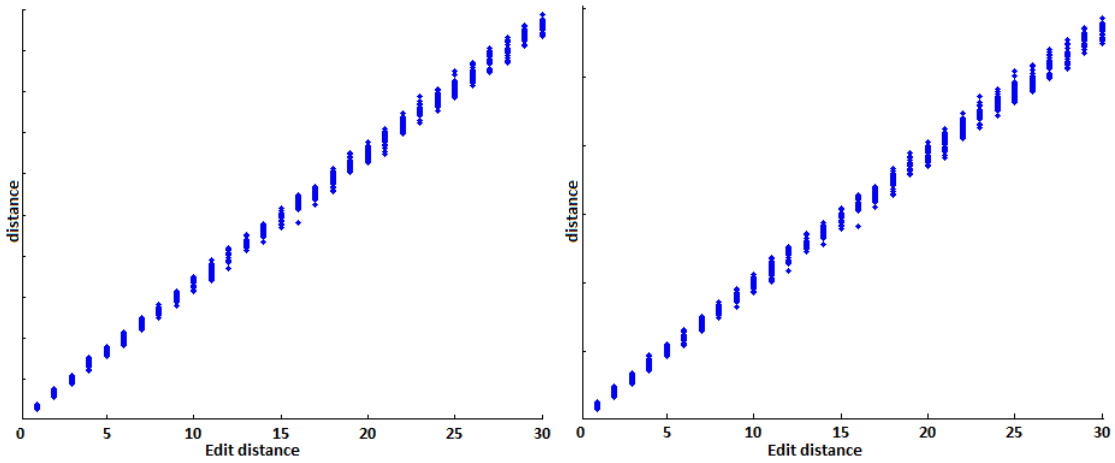
4.3.1 Synthetic data

We commence by investigating the relationship between graph edit distance and the Euclidean distance for the pattern vector $\overrightarrow{bw}(G)$ as well as $\overrightarrow{rw}(G)$. The edit distance between two graphs G_1 and G_2 is the minimum edit cost taken over all sequences of edit operations that transform G_1 to G_2 [13]. For this purpose, we generate 100 random points in Euclidean space and construct a Delaunay triangulation over the point positions. A Delaunay triangulation for a set P of points in a Euclidean space is a triangulation, $DT(P)$, such that no point in P is inside the circumcircle of any triangle in $DT(P)$ [17]. We use the resulting graph with 100 nodes and 288 edges as our seed graph.

We next generate 1000 graphs by randomly deleting up to 30 edges of the seed graph. The edit cost between seed graph and the newly generated graph is then equal to the number of edges deleted. For each graph we compute backtrackless walks of length up to 10 and construct a pattern vector in the form $\overrightarrow{bw}(G) = [l_1, l_2, ..., l_{10}]$. We compute the distance between the vector v_i and $\overrightarrow{v_j}$ as $d_{ij} = \sqrt{(\overrightarrow{v_i} - \overrightarrow{v_j})^T (\overrightarrow{v_i} - \overrightarrow{v_j})}$. The experimental results are shown in Figure 4.4, which shows the distance between pattern vectors composed of backtrackless walks of seed graph and edited graph as a function of edit distance, i.e., number of edges deleted. Similarly Figure 4.4(b) shows the distance between pattern vectors composed of random walks of the seed graph and the edited graph as a function of the edit distance. The small variance in Figure 4.4(a) compared to Figure 4.4(b) shows that backtrackless walks offer more stability to noise.

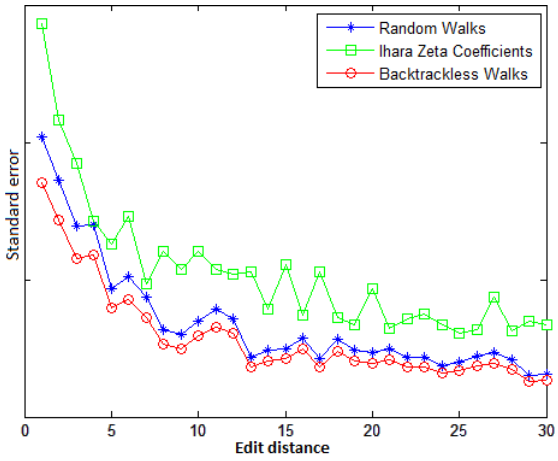
To compare the stability of the pattern vector $\overrightarrow{bw}(G)$ with the pattern vector $\overrightarrow{rw}(G)$, we have shown the SE as a function of edit distance in Figure 4.4(c). The smaller SE for backtrackless walks shows that it provides a more stable representation of the graph when compared to the random walks. The figure also compares the SE for the pattern vector $\overrightarrow{ic}(G)$. The higher SE for the Ihara coefficients is due the fact that randomly deleting the edges from the graph can produce a graph with too many branches. It illustrates that

the Ihara coefficients cannot provide a good measure of similarity for the graphs when branches are present.



(a) Backtrackless walk

(b) Random walk



(c) Standard error

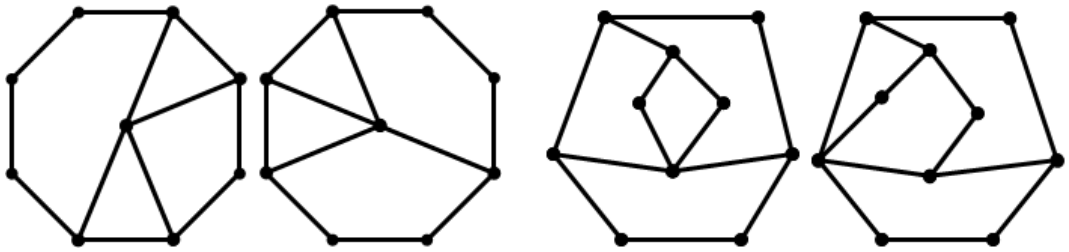
Figure 4.4: Effect of edit distance

4.3.2 Cospectral graphs

One of the advantages of using pattern vectors constructed either from backtrackless walks or Ihara coefficients is that they are less prone to the problem of degeneracies to the cospectrality of non-isomorphic graphs. This is due to the fact that backtrackless walks and Ihara coefficients are determined by the spectrum of the adjacency matrix of the oriented line graph, which is related to the discrete time quantum walk on a graph [56]. Recently, Setyadi and Storm [72] have shown the ability of Ihara zeta function to distinguish some

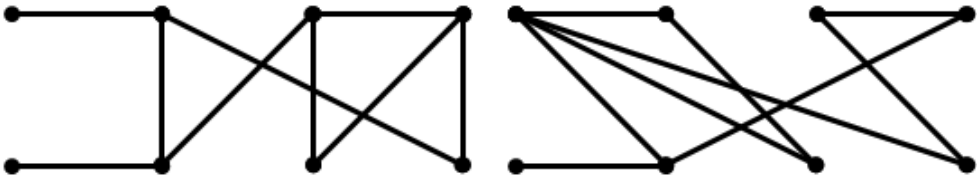
cospectral graphs up to 11 vertices. In this section, we experimentally demonstrate that the pattern vector extracted from a random walk $\overrightarrow{rw}(G)$ cannot distinguish between non-isomorphic graphs which are cospectral with respect to both their adjacency matrices and the adjacency matrices of their complements. On the other hand pattern vectors composed of backtrackless walks $\overrightarrow{bw}(G)$ and the Ihara coefficients $\overrightarrow{ic}(G)$ can distinguish such graphs.

Figure 4.5 shows three pairs of non-isomorphic graphs with 9 [27], 10 [29] and 8 vertices respectively. The pairs of graphs G_1 and G_2 (Figure 4.5(a)), G_3 and G_4 (Figure 4.5(b)), and G_5 and G_6 (Figure 4.5(c)) are cospectral with respect to both their adjacency matrices and the adjacency matrices of their complement respectively. Table 4.2 shows the values of the pattern vectors $\overrightarrow{bw}(G)$, $\overrightarrow{rw}(G)$ and $\overrightarrow{ic}(G)$. These results show that the pattern vector $\overrightarrow{rw}(G)$ cannot distinguish graphs which are cospectral with respect to both their adjacency matrices and the adjacency matrices of their complements. However, the pattern vectors $\overrightarrow{bw}(G)$ and $\overrightarrow{ic}(G)$ can distinguish such graphs.



(a) Cospectral graphs G_1 and G_2

(b) Cospectral graphs G_3 and G_4



(c) Cospectral graphs G_5 and G_6

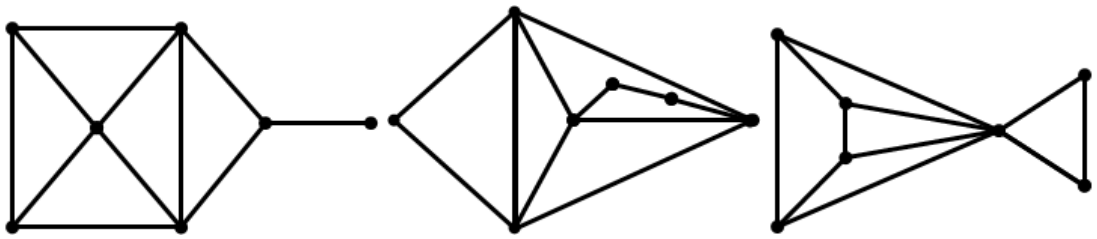
Figure 4.5: Pairs of graphs which are cospectral with respect to their adjacency matrices as well as the adjacency matrices of their complements

Note that, while the pattern vector $\overrightarrow{rw}(G)$ cannot distinguish graphs which are cospectral with respect to both their adjacency matrices and the adjacency matrices of their complements, we observed that such pattern vectors can distinguish the graphs which are cospectral with respect to their adjacency matrices only. Figure 4.6 shows three non-isomorphic graphs [29] G_1 (Figure 4.6(a)), G_2 (Figure 4.6(b)), and G_3 (Figure 4.6(c)), each with 7 vertices. These graphs are cospectral with respect to their adjacency ma-

Table 4.2: Pattern vectors composed of backtrackless walk, random walks, and Ihara coefficients for the cospectral graphs of Figure 4.5.

Method	Pattern vector
$\overrightarrow{bw}(G_1)$	[24 44 84 154 276 498 900 1620 2916 5246]
$\overrightarrow{bw}(G_2)$	[24 44 84 154 276 496 888 1582 2832 5088]
$\overrightarrow{bw}(G_3)$	[26 46 78 128 210 348 586 976 1608 2646]
$\overrightarrow{bw}(G_4)$	[26 46 78 128 210 346 584 972 1598 2636]
$\overrightarrow{bw}(G_5)$	[18 28 40 54 76 106 148 196 260 358]
$\overrightarrow{bw}(G_6)$	[18 28 40 48 64 92 120 152 200 270]
$\overrightarrow{rw}(G_1)$	[24 68 196 570 1664 4868 14256 41772 122432 358896]
$\overrightarrow{rw}(G_2)$	[24 68 196 570 1664 4868 14256 41772 122432 358896]
$\overrightarrow{rw}(G_3)$	[26 72 196 542 1484 4098 11242 31018 85176 234868]
$\overrightarrow{rw}(G_4)$	[26 72 196 542 1484 4098 11242 31018 85176 234868]
$\overrightarrow{rw}(G_5)$	[18 46 114 288 722 1824 4590 11594 29222 73796]
$\overrightarrow{rw}(G_6)$	[18 46 114 288 722 1824 4590 11594 29222 73796]
$\overrightarrow{ic}(G_1)$	[-4 -2 -4 -2 0 5 12 8 16 1]
$\overrightarrow{ic}(G_2)$	[-4 -2 -4 0 0 1 8 4 20 20]
$\overrightarrow{ic}(G_3)$	[-2 0 -2 -1 0 2 2 1 2 1]
$\overrightarrow{ic}(G_4)$	[-2 0 -2 1 0 0 0 1 2 0]
$\overrightarrow{ic}(G_3)$	[0 -2 -8 -6 0 -3 0 24 20 1]
$\overrightarrow{ic}(G_4)$	[0 -2 -8 -4 -2 -7 4 20 16 16]

trices. Graphs G_1 and G_2 are also cospectral with respect to the adjacency matrices of their complements, $\overline{G_1}$ and $\overline{G_2}$. However, $\overline{G_3}$ is neither cospectral to $\overline{G_1}$ nor to $\overline{G_2}$ with respect to the adjacency matrix. Table 4.3 shows the values of the pattern vectors $\overrightarrow{bw}(G)$, $\overrightarrow{ic}(G)$ and $\overrightarrow{rw}(G)$ for the three graphs of Figure 4.6. It is clear from the table that the pattern vectors $\overrightarrow{bw}(G)$ and $\overrightarrow{ic}(G)$ can distinguish all the three graphs. On the other hand the pattern vector $\overrightarrow{rw}(G)$ can only distinguish G_3 from the other two graphs, but cannot distinguish between G_1 and G_2. Note that, if the graphs are regular then cospectrality of the adjacency matrix and that of the oriented line graph's adjacency matrix are equivalent. Therefore the pattern vectors $\overrightarrow{bw}(G)$ and $\overrightarrow{ic}(G)$ can only distinguish non-isomorphic cospectral graphs which are non-regular.



(a) G_1 (b) G_2 (c) G_3

Figure 4.6: Cospectral graphs

Table 4.3: Pattern vectors composed of backtrackless walk, random walks, and Ihara coefficients for the cospectral graphs of Figure 4.6

Method	Pattern vector
$\overrightarrow{bw}(G_1)$	[22 54 134 324 782 1900 4606 11124 26902 65132]
$\overrightarrow{bw}(G_2)$	[22 54 134 318 750 1782 4238 10054 23854 56654]
$\overrightarrow{bw}(G_3)$	[22 58 134 294 734 1726 3870 9294 21982 50286]
$\overrightarrow{rw}(G_1)$	[22 76 264 920 3208 11192 39048 136248 475400 1658808]
$\overrightarrow{rw}(G_2)$	[22 76 264 920 3208 11192 39048 136248 475400 1658808]
$\overrightarrow{rw}(G_3)$	[22 80 272 960 3328 11648 40576 141696 494208 1724800]
$\overrightarrow{ic}(G_1)$	[-10 -12 -14 13 54 60 50 -69 -110 -140]
$\overrightarrow{ic}(G_2)$	[-10 -12 -8 17 48 34 8 -56 -44 -12]
$\overrightarrow{ic}(G_3)$	[-10 -10 -8 13 44 41 8 -62 -26 -44]

4.3.3 Real-world datasets

In this section, we compare our method with alternatives on real-world data. We choose two real-world datasets namely Mutag [74] and COIL [44]. We then compare our method to a number of existing methods. We use KNN classifier to measure the classification accuracy on this data.

Mutag:

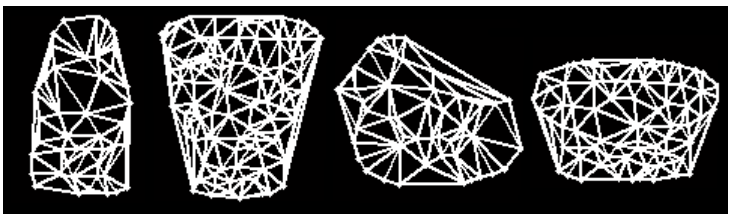
The Mutagenesis dataset consists of a set of chemical compounds, first introduced to the machine learning community in [74]. The data consists of two classes, one which produces mutagenic activity and one which does not. The goal, from the point of view of classification, is to identify the mutation-causing molecules from their structure. There are 125 chemicals in the active class and 63 in the inactive class.

The molecular structure of each chemical is represented by a labelled graph. The atoms form the vertices of the graph and are encoded with a label representing their periodic table group. The edges are weighted 0.5 for a single bond, 1 for a double bond and 0.75 for an aromatic carbon-carbon bond. In the second part of our experiments, we use it as an unlabelled dataset by ignoring the attributes of edges and vertices.

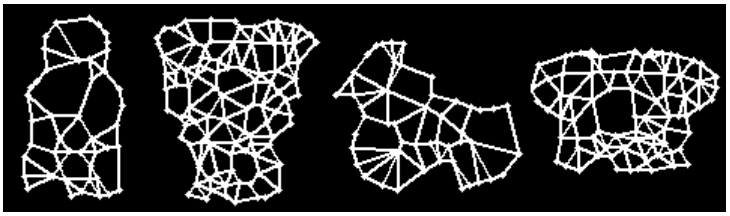
COIL:

The second dataset consists of graphs extracted from the images in the Columbia object image library (COIL) dataset. This dataset contains views of three-dimensional objects under controlled viewer and lighting condition. For each object in the database there are 72 equally spaced views. The objective here is to cluster different views of the same object onto the same class. To establish a graph on the images of objects, we first extract feature points from the image. For this purpose, we use the Harris corner detector [28]. We then construct a Delaunay graph using the selected feature points as vertices. Figure 4.7(a) shows some of the object views (top row) used for our experiments and Figure 4.7(b) shows the corresponding Delaunay triangulations.

(a) COIL



(b) Delaunay triangulation



(c) Gabriel graphs

Figure 4.7: COIL objects, Delaunay triangulations, and Gabriel graphs

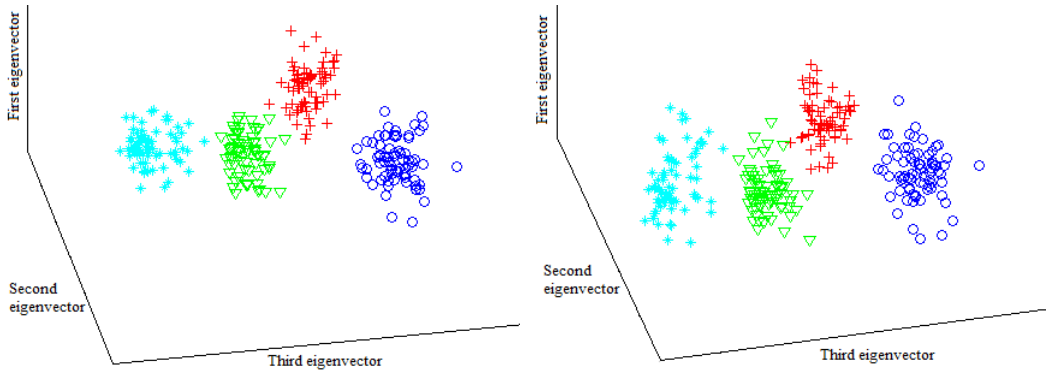
4.3.4 Experiments and results

To evaluate the performance of our kernel on labelled graphs we use the Mutag dataset. We have compared our method with the random walk kernel [26]. The classification accuracies are estimated using 10-fold cross-validation. In 10-fold cross validation, the original data is partitioned into 10 subsamples of equal size. Of these 10 subsamples a single subsample is retained as the validation data for testing the model, and the remaining 9 subsamples are used as training data. This process of cross-validation is then repeated 10 times, with each time a different subsample is used as the test data. Table 4.4 shows the classification accuracies and SE. The classification accuracy of the kernel based on backtrackless walk is 91.1%, while the classification accuracy of the kernel based on random walks is 90.0%. Note that although the accuracy of recognition is increased, the improvement is not very significant. As demonstrated in Section 4.3.2, backtrackless walks are more powerful in distinguishing non-isomorphic cospectral graphs. This may suggest that the small increase in the performance of backtrackless walk kernel over the random walk kernel is due to the set of graphs which are cospectral with respect to their adjacency matrix representations.

Table 4.4: Experimental results of Mutag dataset

Method	Accuracy
Random walk kernel	90.0%
Backtrackless walk kernel	91.1%

To evaluate the performance of the pattern vector extracted from backtrackless walks on unlabelled graphs, we apply our method to the Delaunay triangulations extracted from COIL dataset which are unlabelled graphs. Here we choose four different objects each with 72 different views. To visualize the results we perform principal component analysis (PCA) on the pattern vectors $\overrightarrow{bw}(G)$. PCA is mathematically defined [33] as an orthogonal linear transformation that transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on. Figure 4.8(a) shows the results of the PCA embedding of pattern vectors $\overrightarrow{bw}(G)$ on the first three principal components. For comparison, a similar result of the PCA embedding of pattern vectors $\overrightarrow{rw}(G)$ on the first three principal components is shown in Figure 4.8(b).



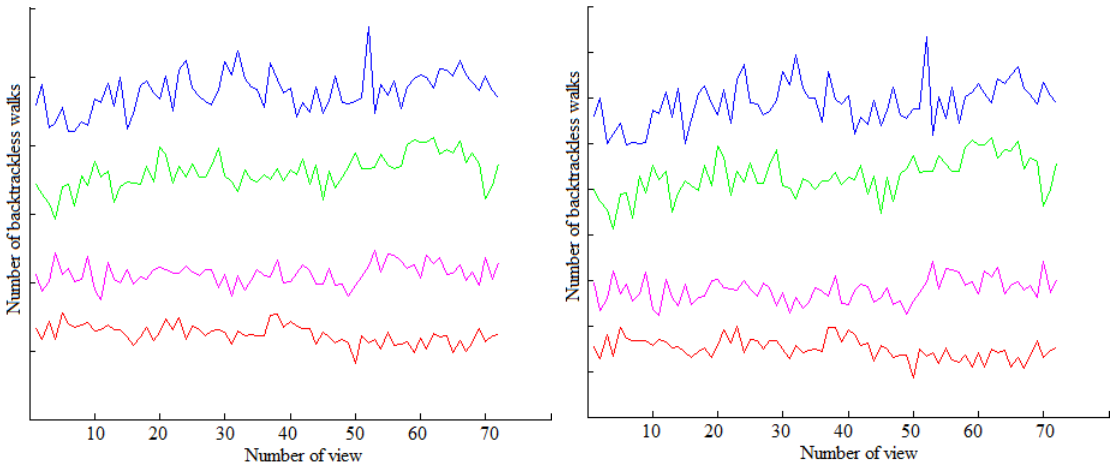
(a) Backtrackless walk (b) Random walk

Figure 4.8: Comparison of performance of clustering

To investigate the performance of backtrackless walks in more detail, we have plotted the frequencies of backtrackless walks of lengths 3, 6, and 10 against the view numbers in Figure 4.9. The results show that the backtrackless walks of shorter lengths can distinguish the objects very well. However, as the length of the walks is increased, the variation in the views of same objects increases and the discriminative power decreases. For example, the walks of length 10 in Figure 4.9(c) shows high variation compare to the walks of length 6 in Figure 4.9(b). This is due to the fact that backtrackless walks of length greater than 5 can add noise to the structural representation of the graph. For example, a backtrackless walk of length 6 might correspond to a random walk on six different edges of a graph or a random walk on three different edges that traverses a triangle twice. Hence, although the use of backtrackless walk can reduce noise in the structural representation of the graph due to redundancy, the problem is not completely avoided.

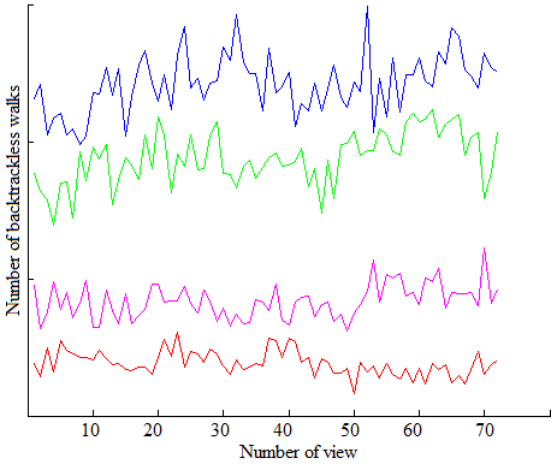
To compare the performance of proposed methods, we construct pattern vectors of length 10 from the random walks, backtrackless walks and Ihara coefficients for the four objects used in the previous experiment. We measure the accuracy of each method using 10-fold cross validation. The results in Table 4.5 shows that the Ihara coefficients can be useful to classify Delaunay graphs. This is due to the fact that the Delaunay triangulation are $md2$ graphs (i.e., the graphs in which the degree of each vertex is at least two) and are cyclic in nature. The results also suggest that the accuracy can be increased by using a backtrackless walk instead of a random walk.

We now compare the performance of the proposed methods on non-cyclic graphs, i.e., graphs having branches. For this purpose we use the unlabelled Mutag data set and the



(a) BW of length 3

(b) BW of length 6



(c) BW of length 10

Figure 4.9: Number of backtrackless walks of different lengths for graphs extracted from COIL dataset

Table 4.5: Experimental results of COIL (Delaunay triangulations)

Method	Accuracy
$\overrightarrow{rw}(G)$	94.8%
$\overrightarrow{bw}(G)$	95.5%
$\overrightarrow{ic}(G)$	98.6%

COIL objects represented using a Gabriel graph [25] rather than a Delaunay triangulation. The Gabriel graph for a set of n points is a subset of Delaunay triangulation, which connects two data points v_i and v_j for which there is no other point v_k inside the open

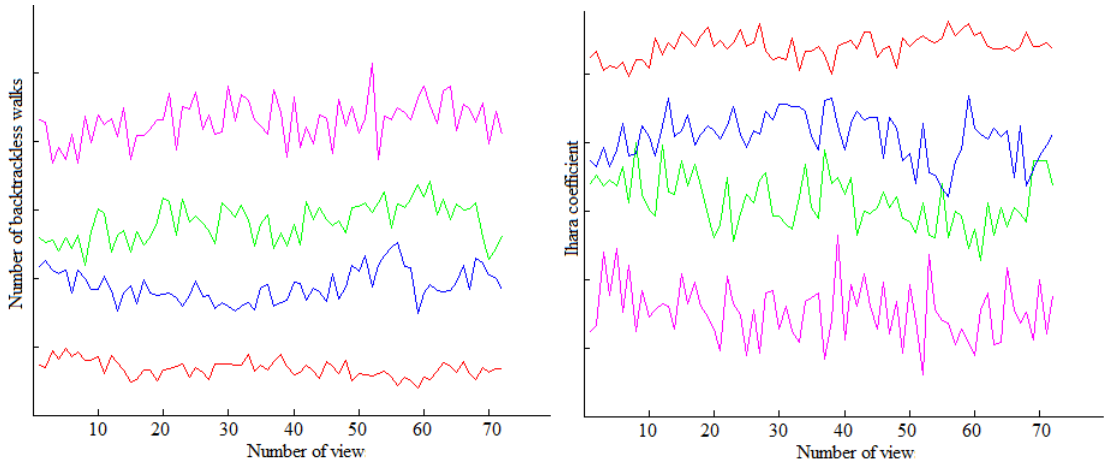
ball whose diameter is the edge (v_i, v_j). The reason for this modification is to reduce the frequency of cycles of smaller lengths and to introduce some branches in the graph. The Gabriel graph of some of the COIL objects are shown in Figure 4.7(c).

Figure 4.10 compares the backtrackless walk of smaller length and low order Ihara coefficients. It is clear from the figure that not only the backtrackless walks can distinguish the different objects with higher accuracy but it also offers small variations in different views of the same object as compared to the Ihara coefficients. The reason for high variation is that Gabriel graphs may introduce branches. Table 4.6 shows the classification accuracies of different methods, which suggest that the pattern vector $\overrightarrow{bw}(G)$ give better performance when the graphs have branches.

Table 4.6: Experimental results (pattern vectors)

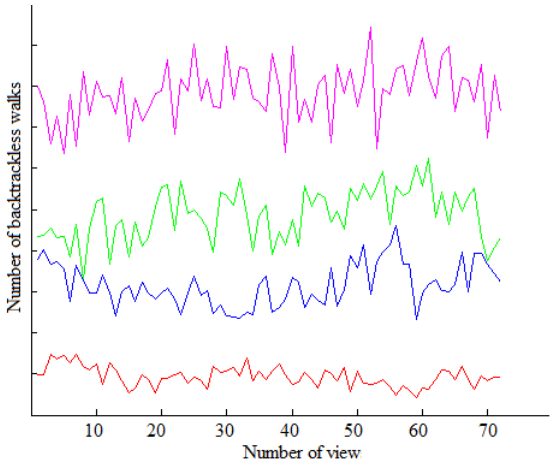
Method	Dataset	Accuracy
$\overrightarrow{rw}(G)$	Mutag (unlabelled)	89.4%
$\overrightarrow{bw}(G)$	Mutag (unlabelled)	90.5%
$\overrightarrow{ic}(G)$	Mutag (unlabelled)	80.5%
$\overrightarrow{rw}(G)$	COIL (Gabriel Graph)	97.2%
$\overrightarrow{bw}(G)$	COIL (Gabriel Graph)	97.6%
$\overrightarrow{ic}(G)$	COIL (Gabriel Graph)	90.6%

To illustrate the advantage of using weights for the pattern vector, we select three different objects from COIL data set each with 30 different views. We select feature points on the objects and construct Delaunay graphs over the point positions. For each graph, we compute the pattern vectors using weighted Ihara coefficients and backtrackless walks and unweighted Ihara coefficients and backtrackless walks. To compare the performance, we cluster the graphs using *k-means clustering* [41]. k-means clustering is a method which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean. We compute *Rand index* [54] of these clusters which is a measure of the similarity between two data clusters. Table 4.7 compares the Rand indices of both methods. It is clear that choosing an appropriate constant can increase the performance of the pattern vectors.

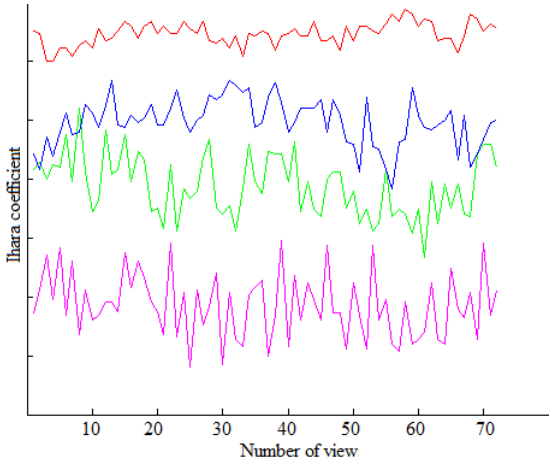


(a) BW of length 2

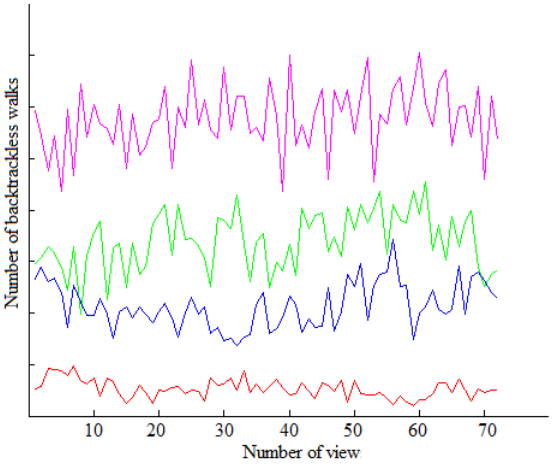
(b) 3rd IC



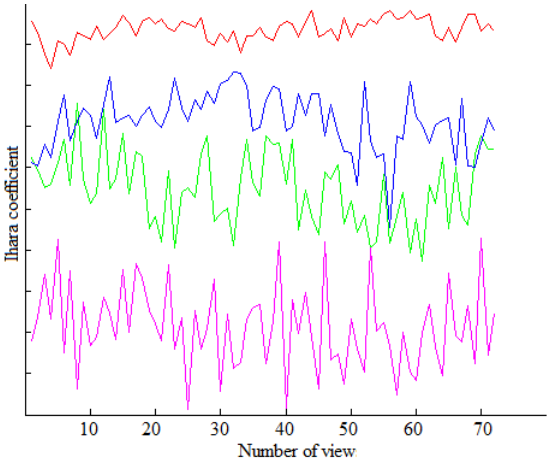
(c) BW of length 3



(d) 4th IC



(e) BW of length 4



(f) 5th IC

Figure 4.10: Backtrackless walks and Ihara coefficients of Gabriel graphs

Table 4.7: Rand indices (comparison of pattern vector with and without weights)

Method	without weights	with weights
$\overrightarrow{bw}(G)$	0.8692	0.8809
$\overrightarrow{ic}(G)$	0.8699	0.9191

4.4 Summary

In this chapter, we have explored the use of backtrackless walks and Ihara coefficients for characterizing labelled as well as unlabelled graphs. To efficiently characterize the labelled graphs we have proposed the use of a graph kernel which is based on backtrackless walks in a graph. We have also presented efficient methods to compute such a kernel. The worst-case running time of the proposed method is the same as that of a kernel defined using random walks. To characterize unlabelled graphs, we have used feature vectors that are composed of backtrackless walks and Ihara coefficients. These feature vectors can be used to embed the graph in a vector space. The worst-case running time of computing these feature vectors is $O(|V|^3)$. Experiments have been conducted on both labelled and unlabelled graphs. Results show not only that backtrackless walks and Ihara coefficients are effective for graph clustering but they are also more powerful in distinguishing cospectral graphs.

Chapter 5

Eigenfunctions of the Edge-Based Laplacian

The traditional discrete graph Laplacian operator [15] has proved to be a useful tool in the analysis of graphs and has found applications in a number of areas. For example, the heat kernel, which is derived from the graph Laplacian, has been used to define graph kernels [36, 39] in the machine learning literature. Sun et al. [77] used the heat kernel on the mesh representing a three-dimensional shape to create a heat kernel signature for describing shape. In [2], Aubry et al. used wave-like solutions of Schrödinger's equation to form an alternative shape descriptor, referred to as the wave kernel signature. The graph Laplacian was used by Coifman and Lafon [16] for dimensionality reduction of data. There are many other applications of graph Laplacian in the literature.

Despite its numerous applications, one of the limitations of the discrete Laplacian is that it cannot link results from the analysis of continuous Laplacian to a direct graph-theoretic analogue. For example, the wave equation defined using the discrete Laplacian does not have a finite speed of propagation [24]. One way to overcome this problem is to bring edges into the picture and treat them as one-dimensional real length intervals. Such graphs are referred to as *Quantum Graphs* [38] or *Geometric Graphs* in the literature. With this approach functions can now exist on both edges and vertices, and equations from analysis can be translated to graphs in a more natural way. Quantum graphs can be considered as simplified models, where one is interested in propagation of waves of various natures (electromagnetic, acoustic, etc) through a one-dimensional system [38].

In [24], Friedman and Tillich have studied a new type of wave equation on graphs that involves a reasonable analogue of wave equation in analysis and has a finite speed

of propagation. The graph Laplacian, that they have studied, consists of two parts; a vertex-based Laplacian and an edge-based Laplacian. They also demonstrate that for edgewise-linear functions the edge-based Laplacian is zero and the graph Laplacian reduces to the traditional discrete graph Laplacian. On the other hand, for functions where the vertex-based Laplacian is zero, they obtain the edge-based Laplacian only. This results in a setting which is substantially different from the traditional approach. In the remainder of this chapter, we will concern ourselves with the edge-based Laplacian.

While Friedman and Tillich find the eigenvalues of the edge-based Laplacian, and give some of its eigenfunctions explicitly, they do not give a method for computing the remaining eigenfunctions. In this chapter, we explore the eigenfunctions of the edge-based Laplacian on a graph and develop methods to explicitly compute these eigenfunctions. The eigenfunctions of the edge-based Laplacian are of two types, i.e., vertex-supported eigenfunctions and edge-interior eigenfunctions. One of our contributions in this chapter is to show a relationship between the edge-interior eigenfunctions and the eigenvectors of the adjacency matrix of the oriented line graph. This relationship gives us an explicit method to compute edge-interior eigenfunctions. We also show a relationship between the vertex-supported eigenfunctions and the eigenvectors of the line graph. The analysis shows that the eigenfunctions of the edge-based Laplacian are related to both the random walk and the backtrackless random walk on a graph.

We commence this chapter by introducing the edge-based Laplacian of the graph and giving a brief details of the formalism of Friedman and Tillich [24]. Next we discuss the set of eigenfunctions supported at the vertices, and demonstrate the relationship of these eigenfunctions to the classical random walk on the graph. Then, from an analysis of functions supported only on the interior of edges, we develop a method for explicitly calculating eigenfunctions which take this form. This reveals a connection between the edge-based Laplacian and the backtrackless random walk on the graph, which in turn is linked to some properties of the quantum walk on a graph and its Ihara zeta function.

5.1 The edge-based Laplacian

In this section, we give brief details of the formalism of Friedman and Tillich [23]. Let $G = (V, E)$ be a graph with a geometric realization $\mathcal{G}$. The geometric realization is the metric space consisting of the vertices V and a closed interval of length l_e associated with each edge $e \in E$. The graph has a (possibly empty) boundary set $\partial G \subset G$. We assume

that the boundary is separated; i.e., each boundary vertex is incident on only one edge. The graph $\mathring{G}=\{\mathring{V},\mathring{E}\}$ excludes boundary vertices and any incident edges $\mathring{G}= G\backslash\partial G$. We associate an edge variable with each edge. Let $e = (u, v)$ be an edge with interval variable x_e. The edge variable x_e equals zero where the edge meets vertex u and equals one at vertex v.

Definition 5.1.1 (Friedman and Tillich [23]) A *vertex measure*, $\mathcal{V}$ is a measure supported on the vertices with $\mathcal{V}(v) > 0$ for all $v \in V$. For our purposes it suffices to take $\mathcal{V}(v) = 1$ for all $v \in V$. An *edge measure*, $\mathcal{E}$ is a measure supported on the interior of edges. $\mathcal{E}(v) = 0$ for all $v \in V$ and the restriction to the interior of edges is the Lebesgue measure.

Let f be a function defined on the graph (on both edges and vertices). We take $f(u)$ to mean the value of f at vertex u and $f(e, x_e)$ to mean the value of f at position x_e along edge e. Since we have different volume measures on the edges and vertices, we must take care in dealing with integrating factors since they are different on edges and vertices. We use $d\mathcal{V}$ for the vertex integration factor and $d\mathcal{E}$ for the edge integration factor. As a result, we have a two-part Laplacian:

$$\Delta f = \Delta_V f d\mathcal{V} + \Delta_E f d\mathcal{E}, \tag{5.1}$$

where Δ_V is the *vertex-based Laplacian* and Δ_E is the *edge-based Laplacian*. When restricting to an edgewise linear functions, the edge-based Laplacian becomes zero, i.e., $\Delta_E = 0$. In this case Δ reduces to the tradional discrete Laplacian which is defined as $L = D - A$, where D is the digonal degree matrix, while A is the adjacency matrix.

Since graph Laplacians are usually given as positive definite operators, the edge-based Laplacian is minus the usual calculus Laplacian (i.e., second derivative)

$$\Delta_E f = -\nabla_{\text{calc}} \cdot \nabla f. \tag{5.2}$$

The vertex-based Laplacian turns out to be

$$\Delta_V f = \frac{1}{\mathcal{V}(v)} \sum_{e \ni v} \mathbf{n}_{e,v} \cdot \nabla f|_e(v). \tag{5.3}$$

Here $\mathbf{n}_{e,v}$ is the *outward-pointing* unit normal. In other words, for an edge (u, b) it points from u to v at the vertex v, and from v to u at the vertex u.

Definition 5.1.2 A function is said to be *edge-based* if $\Delta_V f = 0$. For edge-based functions, the Laplacian consists of only the edge-based part, $\Delta f = \Delta_E f d\mathcal{E}$.

For a function f to be edge-based, the following condition applies.

$$\sum_{e \ni v} (-1)^{1-x_{e,v}} \nabla f(e, x_{e,v}) = 0 \, \forall v, \tag{5.4}$$

or in other words, the sum of the outward-pointing gradients must be zero.

5.2 Vertex-supported edge-based eigenfunctions

In this section, we discuss the vertex supported eigenfunctions of the edge-based Laplacian. We also develop a relationship between vertex-supported eigenpairs and eigenpairs of the line graph. Friedman and Tillich [24] demonstrate the connection between the eigenpairs of the edge-based Laplacian and the eigenpairs of the row-normalised adjacency matrix, provided that all edge lengths are equal. However, the analysis is not valid when the edge lengths vary. For this reason, in the remainder of this thesis, we assume that the edge lengths on the graph are equal. The relationship between the eigenfunctions of the edge-based Laplacian and the eigenvectors of the row-normalized adjacency matrix follows directly from the observation that Δ_E is essentially the familiar Laplacian of calculus and therefore admits eigenfunctions of the form $\phi(e, x_e) = C(e)\cos(\omega x_e + B(e))$ where ω^2 is the eigenvalue. The eigenfunction is edge-based and so applying condition (5.4) to the eigenfunction gives

$$\sum_{e \ni v} \frac{\phi(u) - \phi(v)\cos\omega}{\sin\omega} = 0, \tag{5.5}$$

for any (ω^2, ϕ), which form an eigenpair for Δ_E, when ω is not a multiple of π.

Definition 5.2.1 A *principal eigenpair* is an eigenpair of Δ_E with $0 \leq \omega \leq 2\pi$ where ω^2 is the smallest magnitude eigenvalue of a sequence of eigenpairs of the form $((\omega + 2n\pi)^2, C(e)\cos[(\omega + 2n\pi)x_e + B(e)])$, $n \in \mathbb{N}$ with the same coefficients $B(e)$ and $C(e)$.

Let the vector $g \neq 0$ be the restriction of an eigenfunction ϕ to the vertices, taken in some particular order, i.e., $g_u = \phi(u)$. All eigenfunctions in the same sequence as ϕ have a vertex restriction equal to g.

The row-normalised adjacency matrix $\tilde{A}$ for interior vertices $v \in \mathring{V}$ is given by

$$\tilde{A}_{ij} = \frac{A_{ij}}{\sum_j A_{ij}}, \tag{5.6}$$

where A is the usual graph adjacency matrix.

Theorem 5.2.2 (Friedman and Tillich [24]) Let $\mathcal{G}$ be the geometric realization of a graph and let $\tilde{A}$ be its row-normalized adjacency matrix. Each eigenvalue λ of $\tilde{A}$, with $\lambda \notin \{-1, 1\}$ corresponds to two principal eigenvalues of $\mathcal{G}$

$$\cos^{-1}\lambda \quad \text{and} \quad 2\pi - \cos^{-1}\lambda, \tag{5.7}$$

each with the same multiplicity as λ. The corresponding eigenvector g is the vertex restriction of the sequence of eigenfunctions based on the principal eigenvalue ω.

Equation (5.5) allows us to determine the eigenfunctions. For a principal eigenvalue ω the principal eigenfunction is $\phi(e, x_e) = C(e)\cos(B(e) + \omega x_e)$ with

$$C(e)^2 = \frac{g(v)^2 + g(u)^2 - 2g(u)g(v)\cos\omega}{\sin^2\omega}, \tag{5.8}$$

$$\tan B(e) = \frac{g(v)\cos\omega - g(u)}{g(v)\sin\omega}. \tag{5.9}$$

There are two solutions to Equations (5.8) and (5.9) which are $\{-C(e), B_0(e) + \pi\}$ and $\{C(e), B_0(e)\}$ but both give the same eigenfunction. The sign of $C(e)$ must be chosen correctly to match the phase, so that $C(e)\cos(B(e)) = g(u)$. Since B and C are uniquely determined, this comprises all the eigenfunctions of this form.

Friedman and Tillich give a pair of eigenvalues as above for each eigenvalue of the row-normalized adjacency matrix, but we note that the eigenpair given by $\omega = 2\pi - \cos^{-1}\lambda$ actually corresponds to the same sequence of eigenpairs as $\omega = \cos^{-1}\lambda$, but with negative values of n. There is, therefore, a single sequence for each eigenvalue with

$$\omega = \cos^{-1}\lambda, \tag{5.10}$$

$$\phi(e, x_e) = C(e)\cos\left[B(e) + (\omega + 2\pi n)x_e\right], n \in \mathbb{Z}. \tag{5.11}$$

The value $\lambda = 1$ is always an eigenvalue of $\tilde{A}$. This corresponds to a principal eigenvalue of $\omega = 0$ in Δ_E and therefore the corresponding eigenfunction is $\phi(e, x_e) = C(e)\cos(B(e))$ which is constant on the vertices. If $\partial G \neq \emptyset$ then this eigenfunction does not exist. If $\partial G = \emptyset$ then we obtain a single principal eigenpair of $\omega = 0$ and $\phi(e, x_e) = C$.

The value $\lambda = -1$ will be an eigenvalue of $\tilde{A}$ if $\mathring{G}$ is bipartite. If $\partial G \neq \emptyset$ then the eigenfunction must be zero on the vertices. We defer the evaluation of eigenfunctions not supported on V for the next section. If $\partial G = \emptyset$ then a single principal eigenpair exists with $\omega = \pi$ and $\phi(e, x_e) = C\cos(\pi x_e)$. The eigenfunction alternates in sign between the two partitions of the graph.

This comprises all principal eigenpairs which are supported on the vertices ($g \neq 0$) [24]. The eigenfunctions supported on the vertices are therefore directly determined by the eigensystem of $\tilde{A}$.

5.2.1 Random walks and line graphs

We now show a relationship between the eigenpairs of graph and line graph.

Proposition 5.2.1 Let $\tilde{A}$ be the row-normalised adjacency matrix of G and $\tilde{U}$ be the row-normalised adjacency matrix of the line graph of G. Each eigenpair $\{\lambda, g\}$ of $\tilde{A}$ corresponds to an eigenpair $\{\mu, h\}$ of $\tilde{U}$ with

$$\mu = \lambda, \tag{5.12}$$

$$h_{uv} = A_{uv} g_v. \tag{5.13}$$

Proof 5.2.3 We may write the row-normalized adjacency matrix of the LG as

$$\begin{aligned} \tilde{U}_{uv,wx} &= \frac{A_{uv} A_{wx} \delta_{vw}}{d_x}, \\ &= A_{uv} \tilde{A}_{wx} \delta_{vw}. \end{aligned} \tag{5.14}$$

We have

$$\begin{aligned} \sum_{w,x} \tilde{U}_{uv,wx} h_{wx} &= \sum_{w,x} A_{uv} \tilde{A}_{wx} \delta_{vw} A_{wx} g_x, \\ &= A_{uv} \sum_x \tilde{A}_{vx} g_x, \\ &= \lambda A_{uv} g_v = \mu h_{uv}. \end{aligned} \tag{5.15}$$

□

The vertex-supported eigenfunctions of the edge-based Laplacian are therefore determined by the structure of the random walk on the graph. As we shall show later, the remaining eigenfunctions are determined by the structure of the backtrackless random walk.

5.3 Edge-interior eigenfunctions

We now proceed to our main result. We show that edge-supported eigenfucntion can be determined from the eigenvectors of the oriented line graph. These eigenfunctions of the edge-based Laplacian are zero on the vertices of $\mathcal{G}$ and therefore must have a principal frequency of $\omega \in \{\pi, 2\pi\}$. We start with the case $\omega = \pi$.

Proposition 5.3.1 Principal eigenfunctions of the edge-based Laplacian with principal eigenvalue $\omega = \pi$ and which are zero on the vertices of $\mathcal{G}$ are of the form

$$\phi(e, x_e) = C(e)\cos(\frac{\pi}{2} + \pi x_e), e \in \mathring{E},$$

with

$$\sum_{e \ni v} C(e) = 0 \, \forall v \in \mathring{V}. \tag{5.16}$$

Proof 5.3.1 Since the boundary is separated, the eigenfunctions must be zero on any boundary edge and any edge incident on a boundary vertex. As a result, we may concern ourselves only with $\mathring{G}$. The eigenfunction $\phi(e, x_e)$ is zero at both vertices incident on edge e, giving values of $B(e) = \pi/2$ and $B(e) = 3\pi/2$. However, both values give the same eigenfunction (with a different sign for $C(e)$). We may therefore take $B(e) = \pi/2$. The gradients at either end of the edge are $\nabla\phi(e, x_e = 0) = -\pi C(e)\sin(\frac{\pi}{2})$ and $\nabla\phi(e, x_e = 1) = -\pi C(e)\sin(\frac{3\pi}{2})$. Applying condition (5.4) to this eigenfunction, we obtain

$$\sum_{e \ni v} C(e) = 0.$$

□

Hence, in order to find eigenfunctions of this type, we must find a set of coefficients attached to the edges which sum to zero at every vertex.

Proposition 5.3.2 The principal eigenfunctions of the edge-based Laplacian with principal eigenvalue $\omega = 2\pi$ and which are zero on the vertices of $\mathcal{G}$ are of the form

$$\phi(e, x_e) = C(e)\cos(\frac{\pi}{2} + 2\pi x_e), e \in \mathring{E},$$

with

$$\sum_{e \ni v} (-1)^{1-x_{e,v}} C(e) = 0 \, \forall v \in \mathring{V}. \tag{5.17}$$

Proof 5.3.2 The proof is essentially the same as for the previous proposition. However, the gradients at either end of the edge are $\nabla\phi(e, x_e = 0) = -2\pi C(e)\sin(\frac{\pi}{2})$ and $\nabla\phi(e, x_e = 1) = -2\pi C(e)\sin(\frac{5\pi}{2})$. Applying Condition (5.4), we obtain

$$\sum_{e \ni v} (-1)^{1-x_{e,v}} C(e) = 0.$$

□

We may interpret this condition as follows: Consider each undirected edge of $\mathring{G}$ as a pair of directed edges. Associate a value $C(e)$ with each directed edge in the direction of increasing x_e (i.e., from $x_e = 0$ to $x_e = 1$) and a value $-C(e)$ with the reverse edge. Then the condition above is that the sum of incoming directed edges at a vertex must be zero.

We may write these conditions for both principal eigenvalues in the following form. Let $w_{uv} = \pm C(e)$ be the values associated with each edge, as described above. The sign is always positive for eigenvalues $\omega = \pi$ and alternates in sign depending on the direction of edge traversal for $\omega = 2\pi$. Then we may form a matrix W with elements $W_{uv} = w_{uv}A_{uv}$. The conditions for $\omega = \pi$ are then

$$W1 = 0, \tag{5.18}$$

$$W = W^T, \tag{5.19}$$

and for $\omega = 2\pi$ they are

$$W1 = 0, \tag{5.20}$$

$$W = -W^T, \tag{5.21}$$

where 1 is the vector of all-ones.

The oriented line graph of G ($OL(G)$) represents the structure of a backtrackless random walk on G in the sense that a random walk on the vertices of $OL(G)$ generates a sequence of edges which can be traversed in a backtrackless random walk on G. In [56], Ren et al. have demonstrated that the adjacency matrix of the oriented line graph, T, is equal to the positive support of a quantum walk on the graph G. This matrix is also related to the Ihara zeta function of the graph since it is equal to the Perron-Frobenius operator on $OL(G)$ and therefore can be used to calculate the Ihara zeta function $\zeta_G(u)$ using $\zeta_G^{-1}(u) = \det(I - uT)$. We now show that the eigenvectors of T corresponding to eigenvalues of $\lambda = \pm 1$ determine the structure of the edge-interior eigenfunctions.

Theorem 5.3.3 Let T be the adjacency matrix of $OL(G)$ and s be an eigenvector of T with eigenvalue $\lambda = 1$. Then $s_{uv} = -s_{uv}$ and $\sum_u s_{uv} = 0$, and s provides a solution for W in the case of $\omega = 2\pi$. Similarly, if $\lambda = -1$ then $s_{uv} = s_{uv}$ and $\sum_u s_{uv} = 0$, and s provides a solution for W in the case of $\omega = \pi$.

Proof 5.3.4 In [56] Ren et al. demonstrate that $s_{uv} = A_{uv}w_{uv}$ is an eigenvector of T if $\sum_v A_{uv}w_{uv} = 0$ and either $w_{uv} = -w_{vu}$ or $w_{uv} = w_{vu}$. If $w_{uv} = -w_{vu}$ the eigenvalue is

$\lambda = 1$, and if $w_{uv} = w_{vu}$ then the eigenvalue is $\lambda = -1$. These are precisely the solutions for W (and for $C(e)$). Since the eigenvectors with $\lambda = \pm 1$ span the space of possible solutions, we obtain $|E| - |V| + 1$ linearly independent solutions for $\lambda = 1$ and $|E| - |V|$ linearly independent solutions for $\lambda = -1$ which are all the available solutions according to [24]. □

The structure of the eigenfunctions which are not supported on the vertices is therefore determined by the eigenvectors of the backtrackless random walk on the graph G.

5.4 Summary

We have explored the eigenfunctions of the edge-based Laplacian and given explicit forms for all of these eigenfunctions. Our analysis provides a method for computing the eigenfunctions from the eigenvectors of the oriented line graph, which are zero on the vertices. We demonstrate the connection between the eigenfunctions and both the classical random walk and backtrackless random walk. The edge-based Laplacian therefore shares elements of both of these. As noted by Friedman and Tillich [24], this approach is closer to traditional analysis than the usual discrete graph Laplacian. In particular, it allows us to formulate wave equations and relativistic heat equations which have more usual properties and may be of great use in the study of networks where distance and propagation speed are important.

Chapter 6

Gaussian Wave Packet on a Graph

The solutions of partial differential equations defined using the discrete Laplacian have been extensively used to solve problems that arise in computer vision and machine learning over the recent years. For example Xiao et al. [87] have used the solution of heat equation for graph embedding and clustering. Similarly Zhang et al. [90] have used the diffusion process for anisotropic image smoothing. While the use of partial differential equations defined using the discrete Laplacian has been extensively investigated in literature, little work has been done on the use of partial differential equations defined using the edge-based Laplacian to solve such problems. Initial work by ElGhawalby and Hancock [19] has revealed some of the potential uses of the edge-based Laplacian.

In this chapter, we investigate approaches for characterizing the weighted as well as the unweighted graphs, which are based on the solutions of the wave equation on a graph. The wave equation is defined using the edge-based Laplacian of the graph and it provides a richer and potentially more expressive means of characterizing graphs than the more widely studied heat equation. Unfortunately the wave equation whose solution gives the kernel is less easily solved than the corresponding heat equation. This is due to the reason that the eigenfunctions of the edge-based Laplacian are more complex than that of the node-based Laplacian. In this chapter we present a solution to the wave equation, where the initial condition is a Gaussian wave packet(s) on the edge(s) of the graph. One of our novel contributions in this chapter is to propose a global signature of a graph which is based on the amplitudes of the waves at different edges of the graph over time and that can be used to characterize both weighted and unweighted graphs.

We commence this chapter by defining the eigensystem of the edge-based Laplacian of the graph and how it can be computed. We also discuss how to normalize these eigenfunc-

tions, and to find a complete set of orthonormal bases of the edge-based Laplacian. Next we give a complete solution of a wave equation for both bipartite and non-bipartite graphs, where the initial condition is a Gaussian wave packet(s) on the edge(s) of the graph. We use the solution of the wave equation to define signatures, called the wave packet signature (WPS), for both weighted graphs and unweighted graphs. The proposed signatures are based on the amplitudes of waves on the edges over time. In the experiment section, we perform the proposed method on real-world data and compare its performance with other state-of-the-art methods.

6.1 Edge-based eigensystem

We commence this chapter with a brief review of the eigenvalues and eigenfunctions of the edge-based Laplacian [24, 82] and discuss how to find a complete set of orthonormal bases for the edge-based Laplacian of a graph. Let $G = (V, E)$ be a graph with a geometric realization which is a metric space consisting of vertices V with a closed interval of constant length associated with each edge $e = (u, v) \in E$. We associate an edge variable x_e with each edge that represents the standard coordinate on the edge with $x_e(u) = 0$ and $x_e(v) = 1$. For the work presented in this chapter, it will suffice to assume that the graph is finite with empty boundary (i.e., $\partial G = 0$) and edge-length is 1. As explained in the previous chapter, there are two types of eigenfunctions of the edge-based Laplacian.

6.1.1 Vertex supported edge-based eigenfunctions

The vertex-supported eigenpairs of the edge-based Laplacian can be expressed in terms of the eigenpairs of the normalized adjacency matrix of the graph. Let A be the adjacency matrix of the graph G, and $\tilde{A}$ be the row-normalized adjacency matrix, i.e., the $(i, j)th$ entry of $\tilde{A}$ is given as $\tilde{A}(i, j) = A(i, j)/\sum_{(k,j)\in E} A(k, j)$. Let $(g(v), \lambda)$ be an eigenvector-eigenvalue pair for this matrix. Note that g is defined on vertices and may be extended along each edge to an edge-based eigenfunction. Let ω^2 and $\phi(e, x_e)$ denote the edge-based eigenvalue and eigenfunction respectively. Then the vertex-supported eigenpairs of the edge-based Laplacian are given as follows:

1. For each $(g(v), \lambda)$ with $\lambda \neq \pm 1$, we have an eigenvalue ω^2 with $\omega = \cos^{-1}\lambda$. Since there are multiple solutions to $\omega = \cos^{-1}\lambda$, we obtain an infinite sequence of eigenfunctions; if $\omega_0 \in [0, \pi]$ is the principal solution, the eigenvalues are $\omega = \omega_0 + 2\pi n, n \in$

$\mathbb{Z}$. The eigenfunctions are $\phi(e, x_e) = C(e)\cos(B(e) + \omega x_e)$ where

$$C(e)^2 = \frac{g(v)^2 + g(u)^2 - 2g(v)g(u)\cos(\omega)}{\sin^2(\omega)},$$

and

$$\tan(B(e)) = \frac{g(v)\cos(\omega) - g(u)}{g(v)\sin(\omega)}.$$

There are two solutions here, $\{C, B_0\}$ or $\{-C, B_0 + \pi\}$ but both give the same eigenfunction. The sign of $C(e)$ must be chosen correctly to match the phase.

2. $\lambda = 1$ is always an eigenvalue of $\tilde{A}$. We obtain a principle frequency $\omega = 0$, and since $\phi(e, x_e) = C\cos(2n\pi x_e)$, therefore $\phi(e, v) = \phi(e, u) = C$, which means the eigenfunction is constant on the vertices.

3. $\lambda = -1$ is an eigenvalue of $\tilde{A}$, if the graph is bipartite. We obtain a principle frequency $\omega = \pi$, and since $\phi(e, x_e) = C\cos(\pi x_e + 2n\pi x_e)$, therefore $\phi(e, v) = C$ and $\phi(e, u) = -C$, which means the eigenfunction is constant on vertices, with an alternating sign on both sides of bipartition.

6.1.2 Edge-interior eigenfunctions

The edge-interior eigenfunctions are those eigenfunctions which are zero on vertices and therefore must have a principle frequency of $\omega \in \{\pi, 2\pi\}$. These eigenfunctions can be determined from the eigenvectors of the adjacency matrix of the oriented line graph.

1. The eigenvector corresponding to the eigenvalue $\lambda = 1$ of the oriented line graph provides a solution in the case $\omega = 2\pi$, and we obtain $|E| - |V| + 1$ linearly independent solutions.

2. Similarly the eigenvector corresponding to the eigenvalue $\lambda = -1$ of the oriented line graph provides a solution in the case $\omega = \pi$. If the graph is bipartite, then we obtain $|E| - |V| + 1$ linearly independent solutions. If the graph is non-bipartite, then we obtain $|E| - |V|$ linearly independent solutions.

This comprises all the principal eigenpairs which are only supported on the edges.

6.1.3 Normalization of eigenfunctions

Note that although these eigenfunctions are orthogonal, they are not normalized. To normalize these eigenfunctions we need to find the normalization factor corresponding to

each eigenvalue. Let $\rho(\omega)$ denotes the normalization factor corresponding to eigenvalue ω. Then

$$\rho^2(\omega) = \sum_{e\in\mathcal{E}} \int_0^1 \phi^2\left(e, x_e\right) dx_e.$$

Evaluating the integral, we get

$$\rho(\omega) = \sqrt{\sum_{e\in\mathcal{E}} C(e)^2 \left[\frac{1}{2} + \frac{\sin\left(2\omega + 2B(e)\right)}{4\omega} - \frac{\sin(2B(e))}{4\omega}\right]}.$$

Once we have the normalization factor in hand, we can compute a complete set of orthonormal bases by dividing each eigenfunction with the corresponding normalization factor. Therefore the orthonormalized eigenfunction corresponding to eigenvalues ω^2 is $\phi(e, x_e) = \frac{C(e)}{\rho(\omega)} \cos(B(e) + \omega x_e)$.

Note that the constant eigenfunctions $\phi(e, x_e) = C\cos(2n\pi x_e)$ corresponding to principal frequency $\omega = 0$ are different for the case when $n = 0$ and when $n > 0$. When $n = 0$, these eigenfunctions are constant on vertices as well as edges, and therefore in this case $C = \sqrt{\frac{1}{|E|}}$, since, $\sum_E \int_0^1 \left(\frac{1}{\sqrt{E}}\right)^2 dx_e = 1$. When $n > 0$, the eigenfunctions corresponding to principal frequency $\omega = 0$ are constant on vertices but not on edges. They take the form $\phi(e, x_e) = C\cos(2n\pi x_e)$ on the edges. These eigenfunctions must be normalized, so $C = \sqrt{\frac{2}{E}}$, since $\sum_E \int_0^1 \left(\sqrt{\frac{2}{E}}\cos(2\pi n x_e)\right)^2 dx_e = 1$. Once normalized, these eigenfunctions form a complete set of orthonormal bases for $L^2(\mathcal{G}, \mathcal{E})$.

6.2 General solution of the wave equation

In this section we give a general solution of a wave equation on a graph. Let a graph coordinate $\mathcal{X}$ defines an edge e and a value of the standard coordinate on that edge x. The eigenfunctions of the edge-based Laplacian are

$$\phi_{\omega,n}(\mathcal{X}) = C(e,\omega)\cos\left(B(e,\omega) + \omega x + 2\pi n x\right).$$

The edge-based wave equation is

$$\frac{\partial^2 u}{\partial t^2}(\mathcal{X}, t) = \Delta_E u(\mathcal{X}, t). \tag{6.1}$$

To solve the above equation, we look for separable solutions of the form $u(\mathcal{X}, t) = \phi_{\omega,n}(\mathcal{X})g(t)$. This gives us

$$\phi_{\omega,n}(\mathcal{X})g''(t) = g(t)\left(\omega + 2\pi n\right)^2 \phi(\omega, n),$$

which gives a solution for the time-based part as

$$g(t) = \alpha_{\omega,n} \cos\left[(\omega + 2\pi n)t\right] + \beta_{\omega,n} \sin\left[(\omega + 2\pi n)t\right].$$

By superposition, we obtain the general solution of the wave equation as:

$$\begin{aligned} u\left(\mathcal{X}, t\right) \quad = \quad & \sum_{\omega} \sum_{n} C(e,\omega) \cos\left[B(e,\omega) + \omega x + 2\pi n x\right] \\ & \{\alpha_{\omega,n} \cos\left[(\omega + 2\pi n)t\right] + \beta_{\omega,n} \sin\left[(\omega + 2\pi n)t\right]\}. \end{aligned} \tag{6.2}$$

Here the coefficients $\alpha_{\omega,n}$ and $\beta_{\omega,n}$ depend on the initial conditions of the wave equation.

6.2.1 Initial conditions

Since the wave equation is second order partial differential equation, we can impose initial conditions on both position and speed

$$u(\mathcal{X}, 0) = p(\mathcal{X}),$$

$$\frac{\partial u}{\partial t}(\mathcal{X}, 0) = q(\mathcal{X}),$$

and we obtain

$$p(\mathcal{X}) = \sum_{\omega} \sum_{n} \alpha_{\omega,n} C(e,\omega) \cos\left[B(e,\omega) + \omega x + 2\pi n x\right], \tag{6.3}$$

$$q(\mathcal{X}) = \sum_{\omega} \sum_{n} \beta_{\omega,n} (\omega + 2\pi n) C(e,\omega) \cos\left[B(e,\omega) + \omega x + 2\pi n x\right]. \tag{6.4}$$

We can obtain the coefficients $\alpha_{\omega,n}$ and $\beta_{\omega,n}$ using the orthogonality of the eigenfunctions. So from Equation 6.3, we get

$$\alpha_{\omega,n} = \sum_{e} C(e,\omega) \frac{1}{2} \left[F_{\omega,n} + F^*_{\omega,n}\right], \tag{6.5}$$

where

$$F_{\omega,n} = e^{iB} \int_0^1 dx p(e,x) e^{i\omega x} e^{i2\pi n},$$

We can find $\beta_{\omega,n}$, similarly to the above, using Equation 6.4. So we get

$$\beta_{\omega,n}(\omega + 2\pi n) = \sum_{e} C(e,\omega) \frac{1}{2} \left[G_{\omega,n} + G^*_{\omega,n}\right], \tag{6.6}$$

where

$$G_{\omega,n} = e^{iB} \int_0^1 dx q(x,e) e^{i(\omega+2\pi n)x} = e^{iB} \int_0^1 dx p'(x,e) e^{i(\omega+2\pi n)x}.$$

6.3 Gaussian wave packet

In this section, we give a complete solution of the wave equation, where the initial condition is a Gaussian wave packet on the edge of the graph. For this purpose, we assume that the initial position of the wave equation is a Gaussian wave packet $p(e,x) = e^{-a(x-\mu)^2}$ on one particular edge and zero everywhere else. Then we have

$$\begin{aligned} F_{\omega,n} &= e^{iB}\int_0^1 dx e^{-a(x-\mu)^2} e^{i\omega x} e^{i2\pi n x}, \\ &= e^{iB} e^{i\mu\omega} e^{-\frac{\omega^2}{4a}} \int_0^1 dx e^{-a\left(x-\mu-\frac{i\omega}{2a}\right)^2} e^{i2\pi n x}. \end{aligned}$$

Let the Gaussian wave packet is fully contained on one edge, i.e., $p(x,e)$ is only supported on this edge, then

$$F_{\omega,n} = e^{iB} e^{i\mu\omega} e^{-\frac{\omega^2}{4a}} \int_{-\infty}^{\infty} dx e^{-a\left(x-\mu-\frac{i\omega}{2a}\right)^2} e^{i2\pi n x}.$$

Solving the above equation, we get

$$F_{\omega,n} = \sqrt{\frac{\pi}{a}} e^{i[B+\mu(\omega+2\pi n)]} e^{-\frac{1}{4a}(\omega+2n\pi)^2}.$$

Similarly to the above, solving for $F^*_{\omega,n}$, we obtain

$$F^*_{\omega,n} = \sqrt{\frac{\pi}{a}} e^{-i[B+\mu(\omega+2\pi n)]} e^{-\frac{1}{4a}(\omega+2n\pi)^2},$$

and therefore, the coefficient $\alpha_{\omega,n}$ can be found as

$$\alpha_{\omega,n} = \sqrt{\frac{\pi}{a}} e^{-\frac{1}{4a}(\omega+2n\pi)^2} C(e,\omega)\cos[B+\mu\left(\omega+2\pi n\right)]. \tag{6.7}$$

Since $p(x,e)$ is zero at both ends the coefficients β can be found straightforwardly, as

$$\beta_{\omega,n} = \sqrt{\frac{\pi}{a}} e^{-\frac{1}{4a}(\omega+2n\pi)^2} C(e,\omega)\sin[B+\mu\left(\omega+2\pi n\right)]. \tag{6.8}$$

6.3.1 Complete reconstruction

Let f be the edge on which the initial function is non-zero. Let the Gaussian is fully contained on one edge. Then the solution is given as:

$$\begin{aligned} u(\mathcal{X},t) &= \sum_{\omega} \sqrt{\frac{\pi}{a}} C(\omega,e) C(\omega,f) \sum_n e^{-\frac{1}{4a}(\omega+2\pi n)^2} \\ &\quad \cos\left[B(\omega,e)+\omega x+2\pi n x\right] \cos\left[B(\omega,f)+(\omega+2\pi n)(t+\mu)\right]. \end{aligned} \tag{6.9}$$

For a particular sequence with principal eigenvalue ω, we need to calculate

$$u_\omega = \sum_n \sqrt{\frac{\pi}{a}} e^{-\frac{1}{4a}(\omega+2\pi n)^2} \cos\left[B(\omega,e)+\omega x+2\pi n x\right] \cos\left[B(\omega,f)+(\omega+2\pi n)(t+\mu)\right].$$

Writing the cosine in exponential form, we obtain

$$\begin{aligned} u_w &= \sum_n \sqrt{\frac{\pi}{a}} e^{-\frac{1}{4a}(\omega+2\pi n)^2} \\ &\quad \times \frac{1}{4}\Big[e^{i[B(e,\omega)+B(f,\omega)]} e^{i(\omega+2\pi n)(x+t+\mu)} + e^{-i[B(e,\omega)+B(e,\omega)]} e^{-i(\omega+2\pi n)(x+t+\mu)} \\ &\quad + e^{i[B(e,\omega)-B(f,\omega)]} e^{i(\omega+2\pi n)(x-t-\mu)} + e^{-i[B(e,\omega)-B(e,\omega)]} e^{-i(\omega+2\pi n)(x-t-\mu)}\Big]. \end{aligned}$$

We need to evaluate terms like $\sum_n \frac{\pi}{a} e^{-\frac{1}{4a}} e^{i[B(e,\omega)+B(f,\omega)]} e^{i(\omega+2\pi n)(x+t+\mu)}$, where the values of ω and n depend on the particular eigenfunction sequence under evaluation.

Let $\mathcal{W}(z)$ be z wrapped to the range $[-\frac{1}{2}, \frac{1}{2})$, i.e.,

$$\mathcal{W}(z) = z - \left\lfloor z + \frac{1}{2} \right\rfloor.$$

Here $\lfloor x \rfloor$ is the floor function, i.e., the largest integer not greater than x. Solving for all cases (i.e., for all values of ω and n), the complete solution becomes:

$$\begin{aligned} u(\mathcal{X}, t) &= \sum_{\omega \in \Omega_a} \frac{C(\omega, e)C(\omega, f)}{2} \Bigg(e^{-a\mathcal{W}(x+t+\mu)^2} \cos\left[B(e,\omega) + B(f,\omega) + \omega \left\lfloor x + t + \mu + \frac{1}{2} \right\rfloor\right] \\ &\quad + e^{-a\mathcal{W}(x-t-\mu)^2} \cos\left[B(e,\omega) - B(f,\omega) + \omega \left\lfloor x - t - \mu + \frac{1}{2} \right\rfloor\right]\Bigg) \\ &\quad + \frac{1}{2|E|}\left(e^{-a\mathcal{W}(x+t+\mu)^2} + e^{-a\mathcal{W}(x-t-\mu)^2}\right) \\ &\quad + \sum_{\omega \in \Omega_b} \frac{C(\omega, e)C(\omega, f)}{4}\left(e^{-a\mathcal{W}(x-t-\mu)^2} - e^{-a\mathcal{W}(x+t+\mu)^2}\right) \\ &\quad + \sum_{\omega \in \Omega_c} \frac{C(\omega, e)C(\omega, f)}{4}\Big((-1)^{\lfloor x-t-\mu+\frac{1}{2}\rfloor} e^{-a\mathcal{W}(x-t-\mu)^2} \\ &\quad - (-1)^{\lfloor x+t+\mu+\frac{1}{2}\rfloor} e^{-a\mathcal{W}(x+t+\mu)^2}\Big). \end{aligned} \tag{6.10}$$

where Ω_a represents the set of vertex-supported eigenvalues and Ω_b and Ω_c represent the set of edge-interior eigenvalues respectively, i.e., π and 2π. The second term in the summation comes from the constant eigenvalue, i.e., $\omega = 0$.

Similarly for a bipartite graph, the solution becomes

$$\begin{aligned} u(\mathcal{X},t) &= \sum_{\omega\in\Omega_a}\frac{C(\omega,e)C(\omega,f)}{2}\left(e^{-a\mathcal{W}(x+t+\mu)^2}\cos\left[B(e,\omega)+B(f,\omega)+\omega\left\lfloor x+t+\mu+\frac{1}{2}\right\rfloor\right]\right.\\ &\left.+\,e^{-a\mathcal{W}(x-t-\mu)^2}\cos\left[B(e,\omega)-B(f,\omega)+\omega\left\lfloor x-t-\mu+\frac{1}{2}\right\rfloor\right]\right)\\ &+\frac{1}{2|E|}\left(e^{-a\mathcal{W}(x+t+\mu)^2}+e^{-a\mathcal{W}(x-t-\mu)^2}\right)\\ &+\frac{C(\omega,e)C(\omega,f)}{4}\left((-1)^{\lfloor x-t-\mu+\frac{1}{2}\rfloor}e^{-a\mathcal{W}(x-t-\mu)^2}+(-1)^{\lfloor x+t+\mu+\frac{1}{2}\rfloor}e^{-a\mathcal{W}(x+t+\mu)^2}\right)\\ &+\sum_{\omega\in\Omega_b}\frac{C(\omega,e)C(\omega,f)}{4}\left(e^{-a\mathcal{W}(x-t-\mu)^2}-e^{-a\mathcal{W}(x+t+\mu)^2}\right)\\ &+\sum_{\omega\in\Omega_c}\frac{C(\omega,e)C(\omega,f)}{4}\left((-1)^{\lfloor x-t-\mu+\frac{1}{2}\rfloor}e^{-a\mathcal{W}(x-t-\mu)^2}\right.\\ &\left.-(-1)^{\lfloor x+t+\mu+\frac{1}{2}\rfloor}e^{-a\mathcal{W}(x+t+\mu)^2}\right). \end{aligned} \tag{6.11}$$

Here the third sum is due to the constant eigenfunction with alternating sign corresponding to the principal frequency $\omega=\pi$. Note that the last two terms in Equation 6.10 as well as Equation 6.11 depend on the number of edges and vertices in the graph. For example if the graph is a tree (which is always bipartite), then Equation 6.11 gives the solution with last two terms disappeared since $|E|-|V|=-1$.

For a weighted graph, we assume a Gaussian wave packet on every edge of the graph, whose amplitude is multiplied by the weight of that particular edge, and solve the wave equation for this case. Let w_{ij} be the weight of the edge (i,j). The solution for bipartite graph in this case becomes

$$\begin{aligned} u(\mathcal{X},t) &= \sum_{(u,v)\in E}w_{i,j}\sum_{\omega\in\Omega_a}\frac{C(\omega,e)C(\omega,f)}{2}\\ &\left(e^{-a\mathcal{W}(x+t+\mu)^2}\cos\left[B(e,\omega)+B(f,\omega)+\omega\left\lfloor x+t+\mu+\frac{1}{2}\right\rfloor\right]\right.\\ &\left.+\,e^{-a\mathcal{W}(x-t-\mu)^2}\cos\left[B(e,\omega)-B(f,\omega)+\omega\left\lfloor x-t-\mu+\frac{1}{2}\right\rfloor\right]\right)\\ &+\frac{1}{2|E|}\left(e^{-a\mathcal{W}(x+t+\mu)^2}+e^{-a\mathcal{W}(x-t-\mu)^2}\right)\\ &+\frac{C(\omega,e)C(\omega,f)}{4}\left((-1)^{\lfloor x-t-\mu+\frac{1}{2}\rfloor}e^{-a\mathcal{W}(x-t-\mu)^2}+(-1)^{\lfloor x+t+\mu+\frac{1}{2}\rfloor}e^{-a\mathcal{W}(x+t+\mu)^2}\right)\\ &+\sum_{\omega\in\Omega_b}\frac{C(\omega,e)C(\omega,f)}{4}\left(e^{-a\mathcal{W}(x-t-\mu)^2}-e^{-a\mathcal{W}(x+t+\mu)^2}\right)\\ &+\sum_{\omega\in\Omega_c}\frac{C(\omega,e)C(\omega,f)}{4}\left((-1)^{\lfloor x-t-\mu+\frac{1}{2}\rfloor}e^{-a\mathcal{W}(x-t-\mu)^2}\right.\\ &\left.-(-1)^{\lfloor x+t+\mu+\frac{1}{2}\rfloor}e^{-a\mathcal{W}(x+t+\mu)^2}\right). \end{aligned} \tag{6.12}$$

As with the case of unweighted graphs, the last three terms depend on the number of vertices and edges in the graph and whether the graph is bipartite or not. For example for non-bipartite graphs, the third term in the above equation disappears.

To show the evolution of Gaussian wave packet on a graph, we take a graph with 5 nodes and 6 edges. We assume the initial condition is a Gaussian wave packet on a single edge and zero elsewhere. Figure 6.1(a) shows the results for the times $t = 0$, $t = 1$, $t = 2$ and $t = 3$ in a three-dimensional space. Note that when the wave packet hits a node with degree greater than 2, some part of the packet is reflected back while the other part is equally distributed to the connecting edges. Figure 6.1(b) shows a similar analysis but with a different initial condition. Here we assume that initially a Gaussian wave packet exists on every edge of the graph and show its evolution for the times $t = 0$, $t = 1$, $t = 2$ and $t = 3$.

6.4 Gaussian wave packet signature

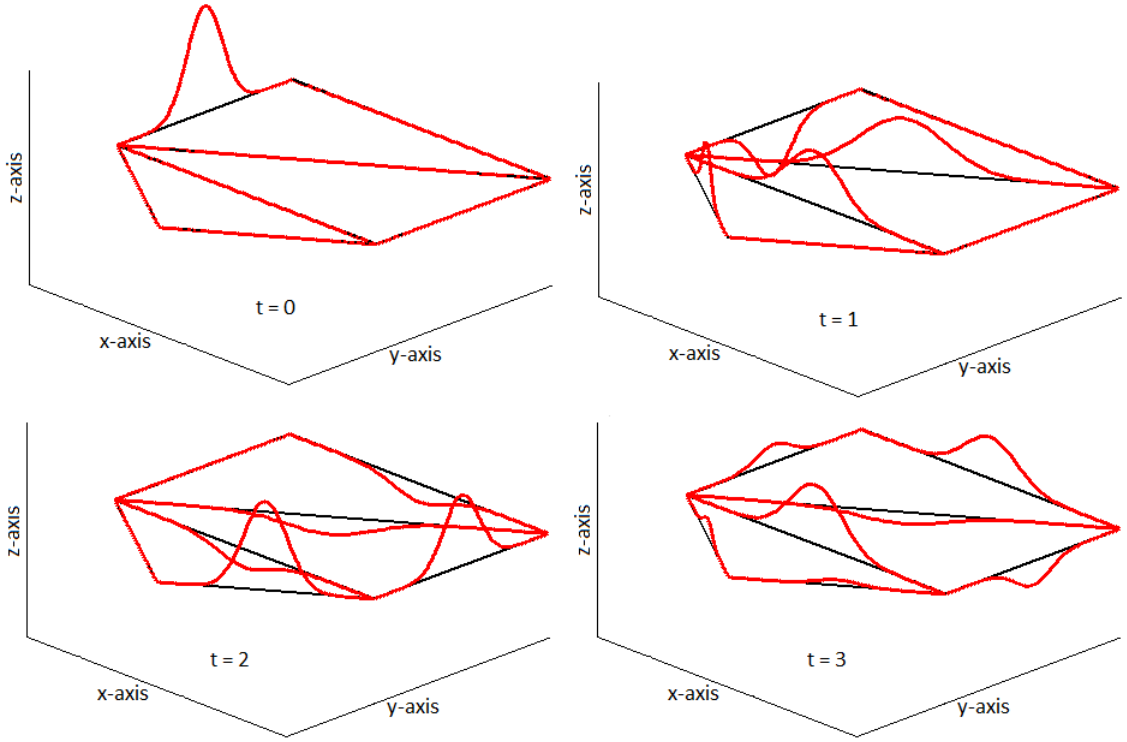
To define a signature for the weighted as well as the unweighted graphs, we use the amplitudes of the waves on the edges of the graph over time. For an unweighted graph, we assume that the initial condition is a Gaussian wave packet on a single edge of the graph. For this purpose we select the edge $(u, v) \in E$, such that u is the highest degree vertex in the graph and v is the highest degree vertex in the neighbours of u. In case, if this edge is not uniquely defined, we arbitrarily select amongst all possible candidate edges. For this case Equation 6.10 provides a solution if the graph is non-bipartite, while Equation 6.11 provides the solution if the graph is bipartite. For weighted graph, we assume a wave packet on every edge whose amplitude is multiplied by the weight of the edge. Equation 6.12 provides a solution in this case. We define the local signature, called the *wave packet signature* (WPS), of an edge as

$$\text{WPS}(\mathcal{X}) = [u(\mathcal{X}, t_0), u(\mathcal{X}, t_1), u(\mathcal{X}, t_2), ...u(\mathcal{X}, t_n)].$$

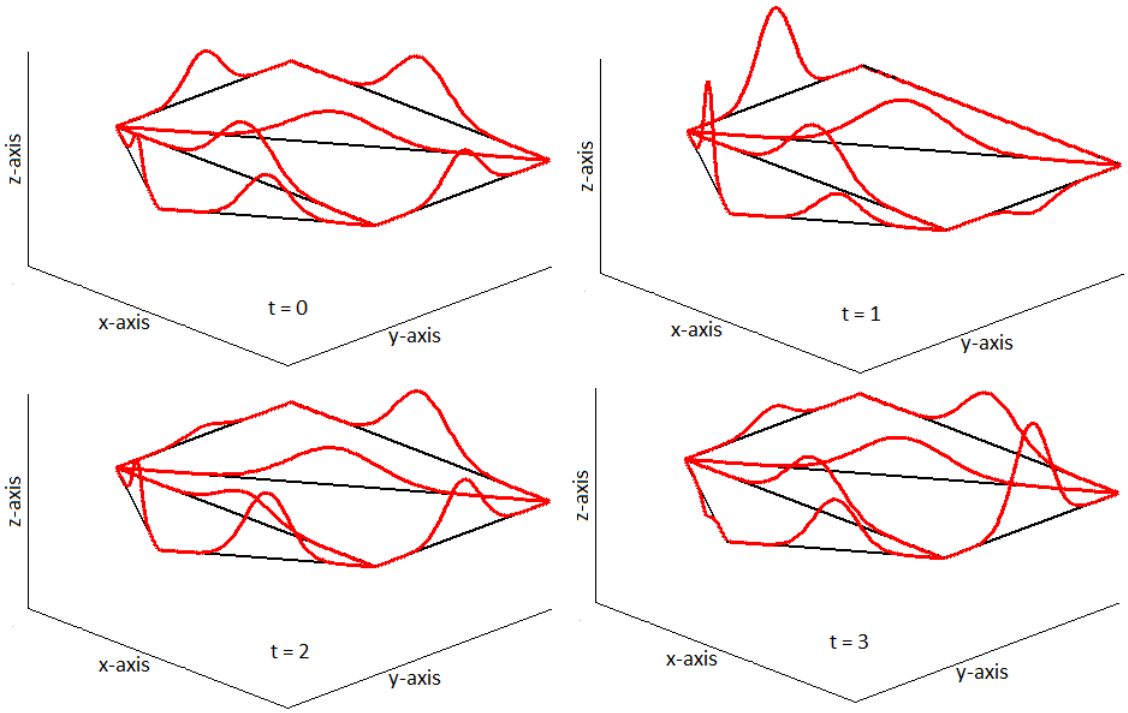
Given a graph G, we define its *global wave packet signature* (GWPS) as

$$\text{GWPS}(G) = \text{hist}\left(\text{WPS}(\mathcal{X}_1), \text{WPS}(\mathcal{X}_2), , ..., \text{WPS}(\mathcal{X}_{|E|})\right), \tag{6.13}$$

where hist(.) is the histogram operator which bins the list of arguments $WPS(\mathcal{X}_1)$, $WPS(\mathcal{X}_2), , ..., WPS(\mathcal{X}_{|E|})$. Here $\mathcal{X}_i$ represents the the value of standard coordinate x_e on the edge e_i. In the experiment section we choose $x_e = 0.5$.



(a) Evolution of a single wave packet on a graph



(b) Evolution of multiple wave packets on a graph

Figure 6.1: Solution of wave equation on a graph with 6 vertices and 8 edges

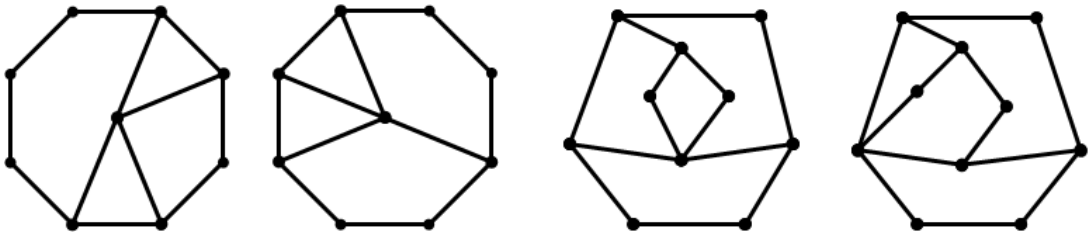
6.5 Experiments

In this section, we test the proposed method on cospectral graphs and real-world data. We also compare the performance of the proposed method with other alternative methods and report the results. We commence by demonstrating the ability of the GWPS to distinguish between cospectral graphs.

6.5.1 Cospectral graphs:

One of the advantages of using the solution of equations defined using the edge-based Laplacian is that it is less prone to the problem of failing to distinguish graphs due to cospectrality of the Laplacian or adjacency matrices. This is due to the fact that the structure of edge-interior eigenfunctions of the edge-based Laplacian are determined by the structure of backtrackless walks on the graph which is more powerful tool in distinguishing non-isomorphic cospectral graph as compared to the random walks on the graphs.

To demonstrate this we find GWPS of the pairs of cospectral graphs of Figure 6.2(a) and Figure 6.2(b). These pairs of graphs are cospectral with respect to both their adjacency matrices and the adjacency matrices of their compliments. Figure 6.3(a) and Figure 6.3(b) show the GWPS for these graphs. Results show the ability of the GWPS to distinguish graphs which are cospectral with respect to their adjacency matrices and adjacency matrices of their compliments. Note that in Chapter 4, we have shown that such graphs cannot be distinguished by random walks on graphs.

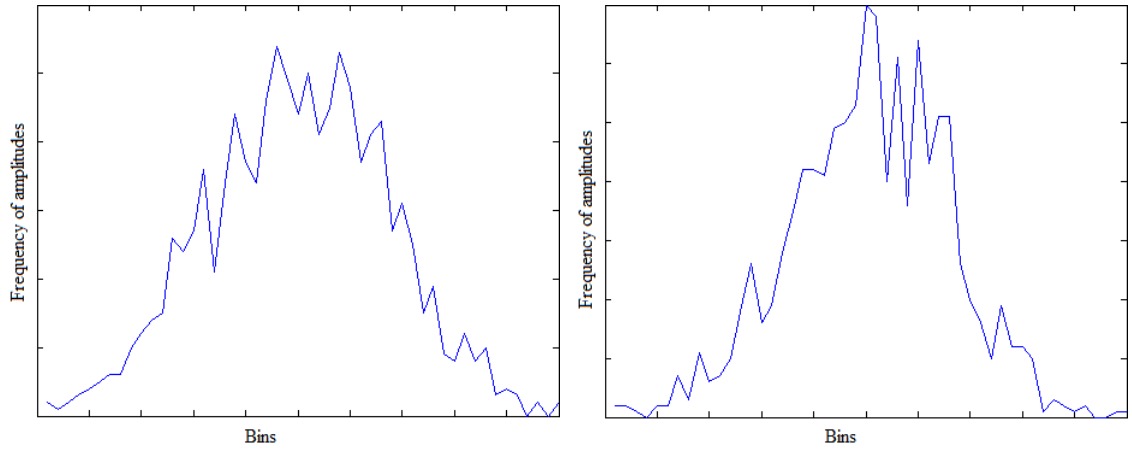


(a) Cospectral graphs, each with 9 vertices and 12 edges

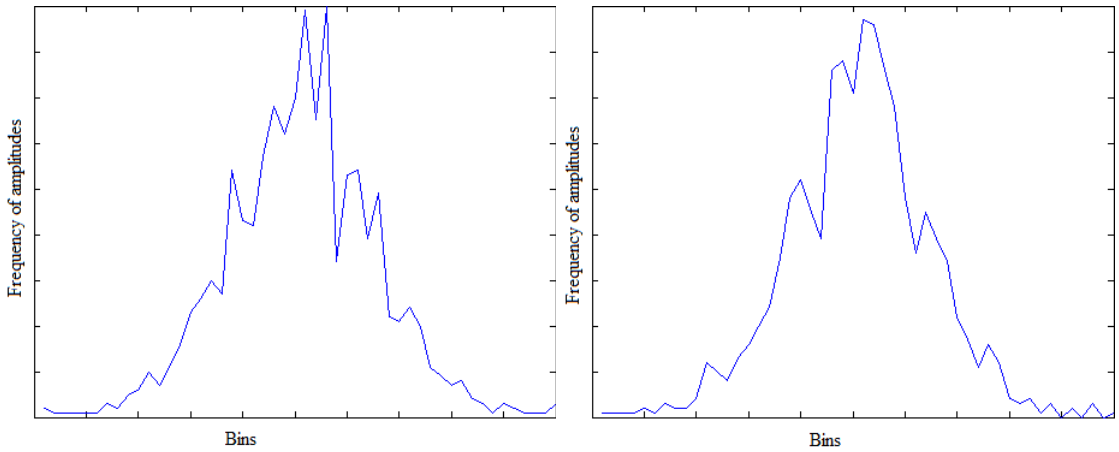
(b) Cospectral graphs, each with 10 vertices and 13 edges

Figure 6.2: Examples of cospectral graphs with respect to adjacency matrices of graphs and adjacency matrices of their compliment graphs

Similarly Figure 6.5 shows the GWPS for the pair of graphs of Figure 6.4, which are cospectral with respect to their Laplacian matrices. These graphs cannot be distinguished



(a) Histogram for the graphs of Figure 6.2(a)



(b) Histogram for the graphs of Figure 6.2(b)

Figure 6.3: Histograms for cospectral graphs

by the heat kernel trace [86] of graphs. However, as shown in Figure 6.5, GWPS can distinguish such graphs.

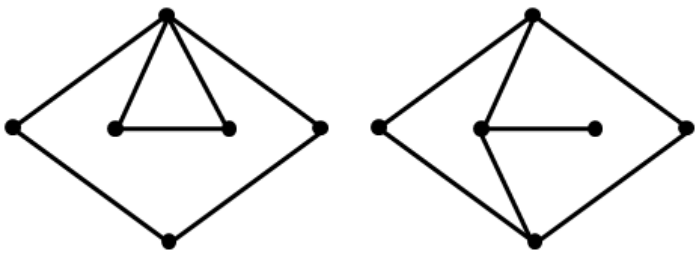


Figure 6.4: Cospectral graphs with respect to their Laplacian matrices

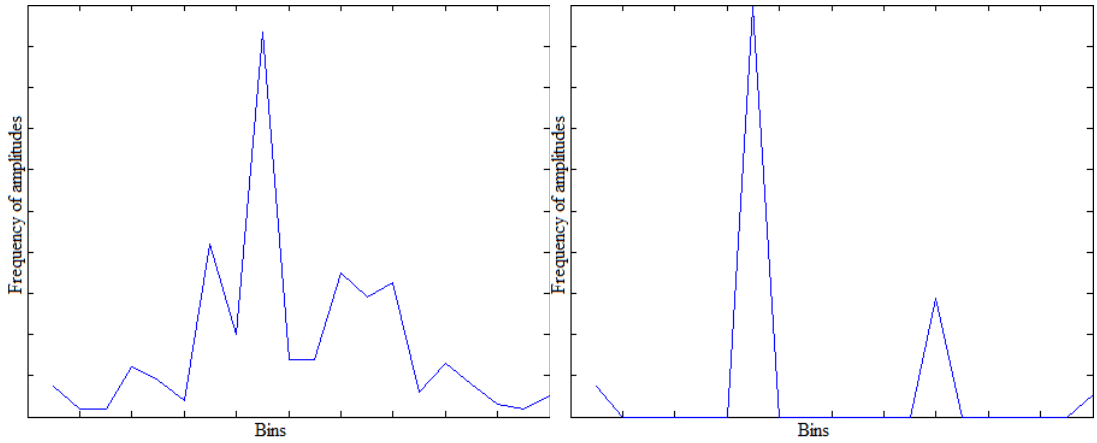


Figure 6.5: Histogram for the graphs of Figure 6.4, which are cospectral with respect to their Laplacian matrices

6.5.2 Real-world data

In this section, we perform an experimental evaluation of the GWPS on different graphs. These graphs are extracted from the images in the Columbia object image library (COIL) dataset [44]. These are Delaunay triangulation (DT), Gabriel graphs (GG), and relative neighbourhood graphs (RNG).

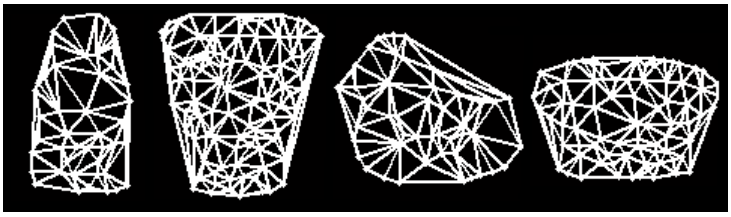
Delaunay triangulation: A Delaunay triangulation for a set P of points in a Euclidean space is a triangulation, $DT(P)$, such that no point in P is inside the circumcircle of any triangle in $DT(P)$ [17].

Gabriel graphs: The Gabriel graph for a set of n points is a subset of Delaunay triangulation, which connects two data points v_i and v_j for which there is no other point v_k inside the open ball whose diameter is the edge (v_i, v_j).

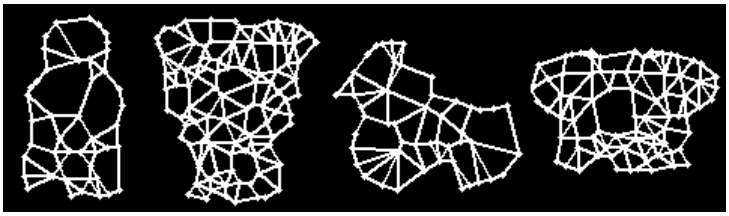
Relative neighbourhood graphs: Like the Gabriel graph, the relative neighbourhood graph is also a subset of Delaunay triangulation. In this case, a lune is constructed on each Delaunay edge. The circles enclosing the lune have their centres at the end-points of the Delaunay edge; each circle has a radius equal to the length of the edge. If the lune contains another node then its defining edge is pruned from the relative neighbourhood graph.

Figure 6.6(b), Figure 6.6(c), and Figure 6.6(d) show the DT, GG and RNG of the corresponding COIL objects of Figure 6.6(a) respectively.

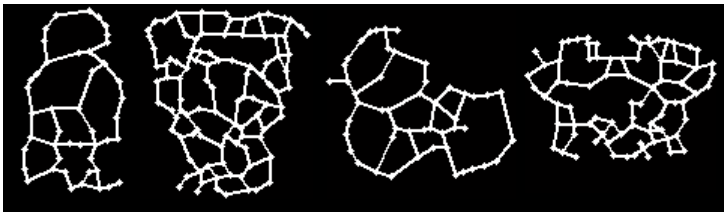
(a) COIL



(b) DT



(c) GG

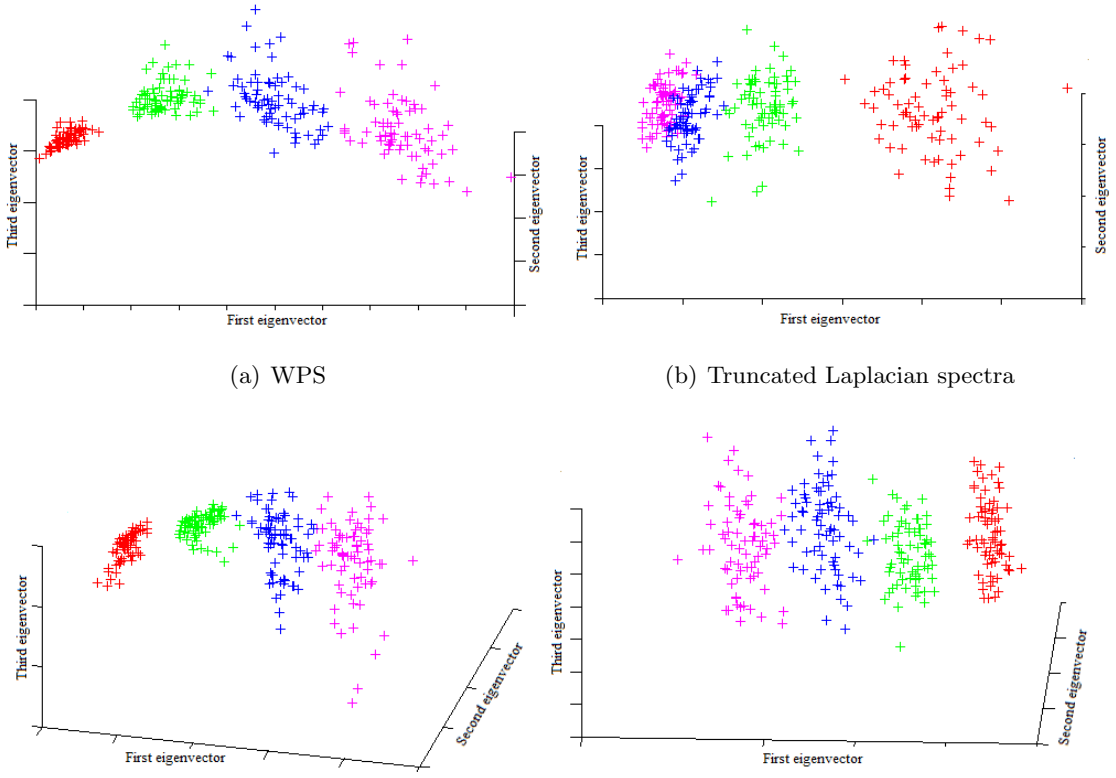


(d) RNG

Figure 6.6: COIL objects and their extracted graphs

Unweighted graphs: We commence by experimenting the proposed method with the Delaunay triangulations. To compute the GWPS, we first compute WPS for each edge by setting $x_e = 0.5$. We set $t_{max} = 100$ and $t_{min} = 20$ to allow the wave packet to be distributed over the whole graph. We then compute the GWPS for the graph by fixing 100 bins for histogram.

To visualize the results, we have performed principal component analysis (PCA) on the GWPS. Figure 6.7(a) shows the results of the embedding of the pattern vectors on the first three principal components. Results demonstrate the ability of GWPS to provide good separation between objects of different classes.



(a) WPS

(b) Truncated Laplacian spectra

(c) Random walk

(d) Ihara coefficients

Figure 6.7: Comparison of clustering results

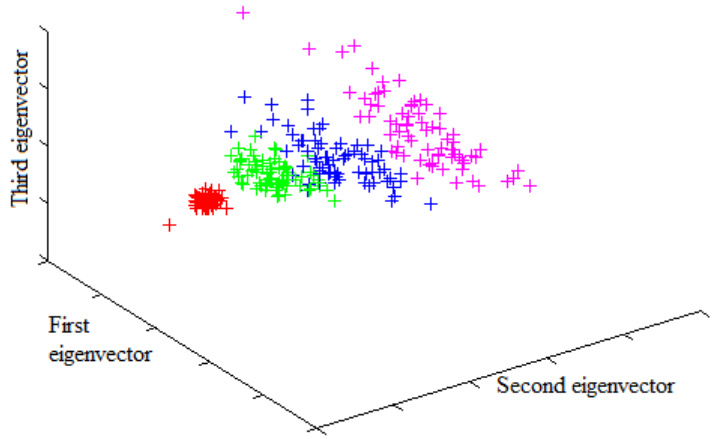
To measure the performance of the proposed method we compare it with truncated Laplacian, pattern vector from random walks and pattern vector from Ihara coefficients. To compare the performance, we cluster the pattern vectors using *k-means clustering* [41]. The rand indices for these methods are shown in Table 6.1. It is clear from the table that the proposed method can classify the graphs with higher accuracy as compared to alternative methods. To visualize the results, we have applied PCA on the resulting pattern vectors for each method. Figure 6.7(b), Figure 6.7(c), and Figure 6.7(d) shows results for truncated Laplacian, pattern vector from random walks on graphs [26], and pattern vector from Ihara coefficients [62] respectively.

We now compare the performance of the proposed method on GG and RNG. The purpose of comparing the performance on GG and RNG is twofold. Firstly, since both the GG and RNG are subset of DT, it allows us to analyze the performance of the proposed method under controlled structural modifications. Secondly, since both GG and RNG reduce the frequencies of cycles of smaller length and introduce branches in the graph, it allows us to analyze the performance of the proposed method on non-cyclic graphs.

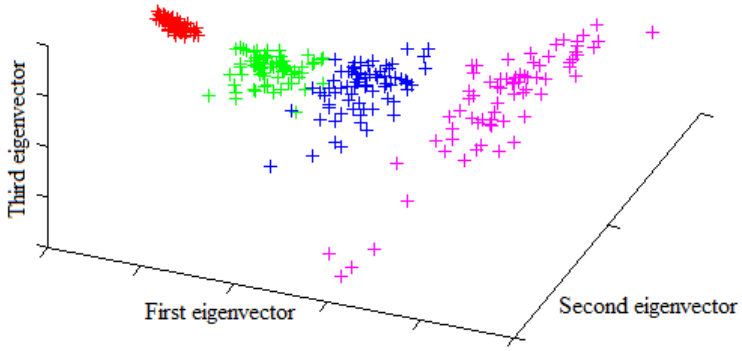
Table 6.1: Experimental results on Delaunay triangulation

Method	Performance
Wave Kernel Signature	0.9965
Random Walk Kernel	0.9526
Ihara Coefficients	0.9864
Truncated Laplacian	0.9737

To visualize the results, we have applied PCA on GWPS computed from GG and RNG. Figure 6.8(a) and Figure 6.8(b) show the visual results of the proposed method on GG and RNG respectively. Results suggest that the proposed method can separate the objects of different classes under controlled structural error.



(a) Clustering GG



(b) Clustering RNG

Figure 6.8: Clustering results of GWPS

Table 6.2 compares the performance of the three methods, which shows that the proposed method performs well under controlled structural modifications compared to other methods. Note that a drop in the performance of the Ihara coefficients is due to the fact that the Ihara coefficients cannot provide a good measure of similarity for the graphs when branches are present. This is because the Ihara coefficients are related to the frequencies of prime cycles in the graph. GG as well as RNG reduces the frequencies of cycles of smaller length and may also introduce branches.

Table 6.2: Experimental results on GG and RNG

Method	GG	RNG
Wave Kernel Signature	0.9511	0.8235
Random Walk Kernel	0.9115	0.8197
Ihara Coefficients	0.8574	0.7541
Truncated Laplacian	0.8188	0.7790

Properties of WPS: We now look at the characteristic of the proposed GWPS. For large graphs, the histogram distribution of the wave amplitudes over time (i.e., GWPS) closely follows a Gaussian distribution. Figure 6.9 shows the distribution of the GWPS of a single view of 4 different objects in COIL dataset and a Gaussian fit for each signature. Given GWPS of length N, the parameters of the Gaussian curve can be found as

$$\mu = \frac{1}{N}\sum_{i=1}^{n} GWPS(i),$$

and

$$\sigma = \sqrt{\frac{1}{N-1}\sum_{i=1}^{n}(GWPS(i)-\mu)}.$$

Results suggest that the Gaussian curve fits nicely to the GWPS.

In our next experiment, we have selected 4 different objects from COIL data set with all the 72 view and extracted their DT and GG. Next we compute their GWPS and find the parameters of the Gaussian curve for each signature. In Figure 6.10(a) and Figure 6.10(b), we have plotted the values of the standard deviations of Gaussian fit for all the DT and GG respectively. Results show that the standard deviation of Gaussian curve provides a good separation between the objects of different classes. Table 6.3 shows the mean value of the standard deviation and the standard error for each of the 4 objects. These results

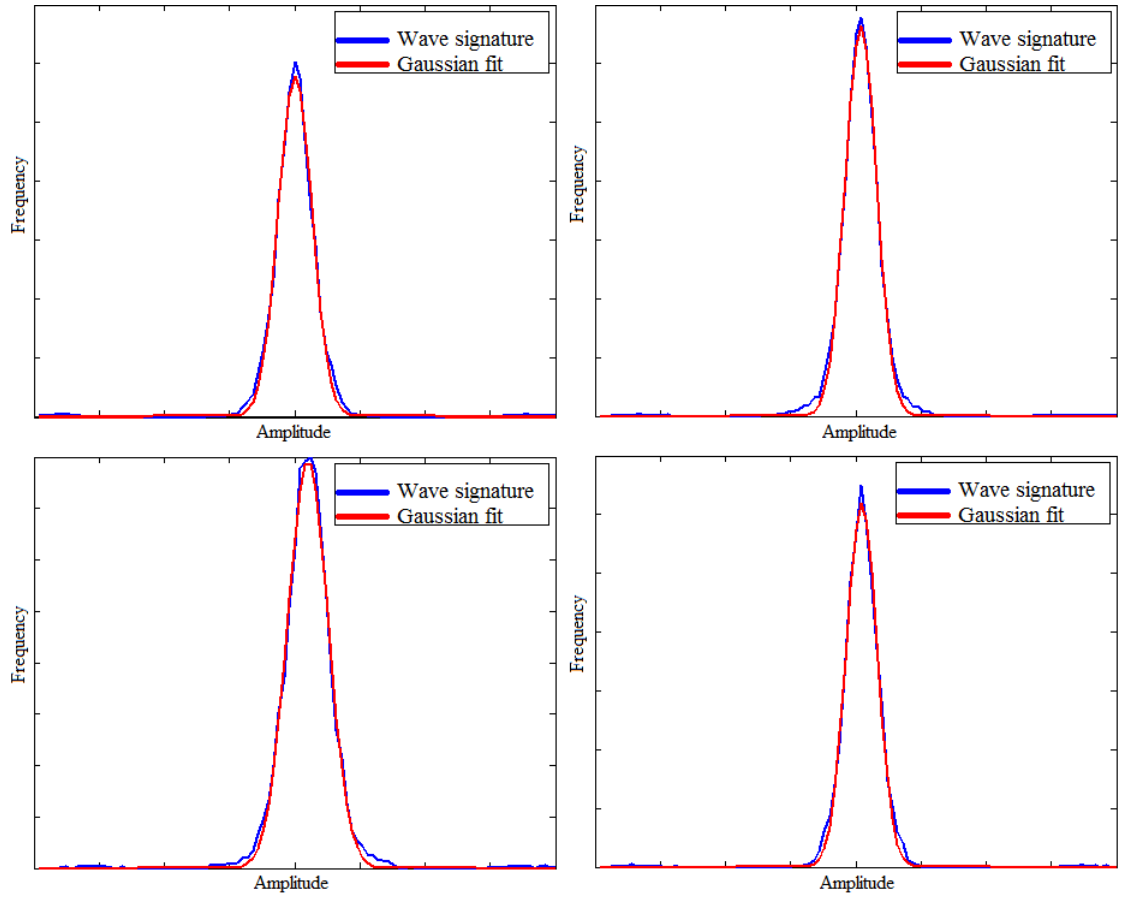


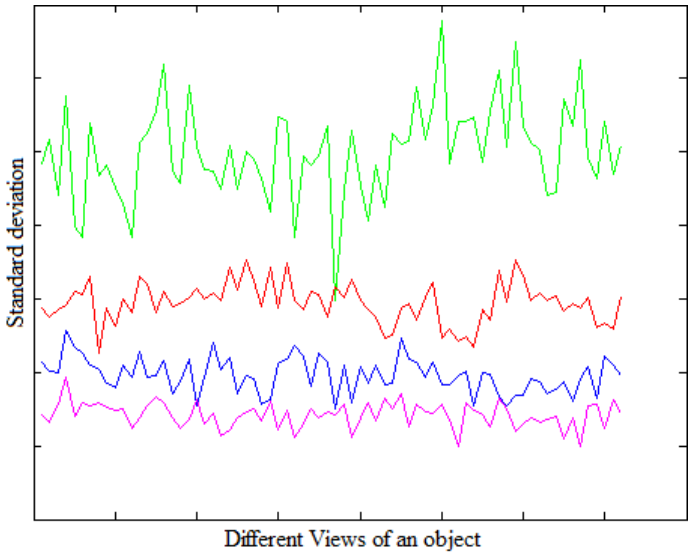
Figure 6.9: Gaussian fit of GWPS

suggest that the properties of the Gaussian distribution can be used to classify objects of different classes.

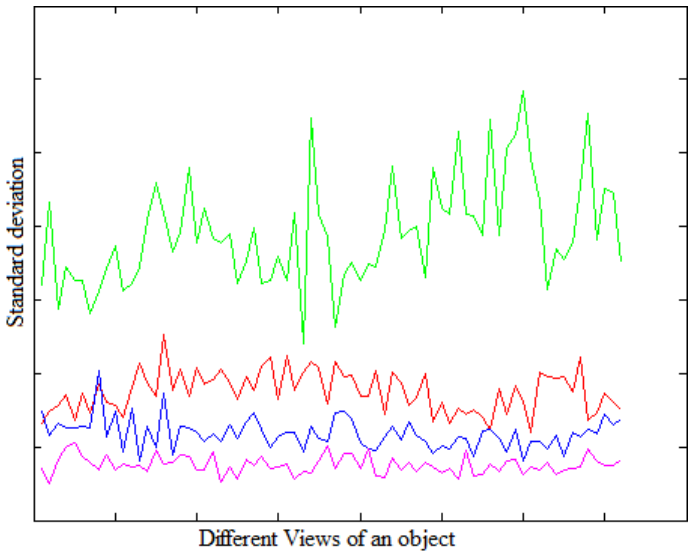
Table 6.3: Average values of standard deviation

	Standard Deviation	Standard Error
Object 1	0.1400	1.54×10^{-3}
Object 2	0.0989	6.57×10^{-4}
Object 3	0.0793	5.64×10^{-4}
Object 4	0.0685	4.07×10^{-4}

Weighted graphs: In our final experiment, we test the performance of the proposed GWPS on weighted graphs. For this purpose, we have selected the same four objects from the COIL dataset with all the 72 views. We have extracted the Gabriel graphs for each of



(a) Delaunay triangulations



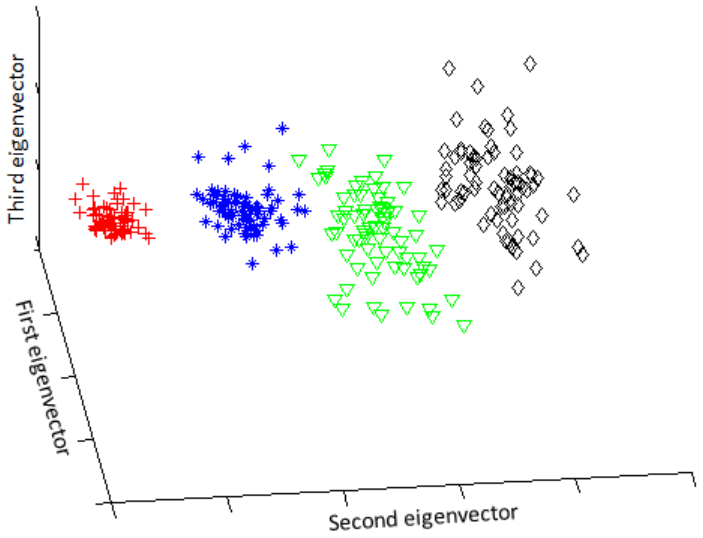
(b) Gabriel graphs

Figure 6.10: Standard deviations of all views of four different objects

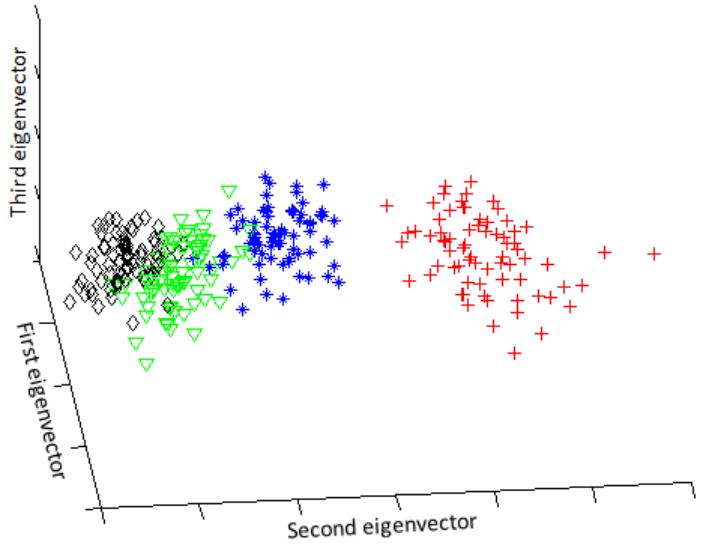
these views. The edges are weighted with the exponential of the negative distance between two connected vertices, i.e., $w_{ij} = \exp[-k||x_i - x_j||]$ where x_i and x_j are coordinates of corner points i and j in an image and k is a scalar scaling factor. For each weighted graph, we have computed GWPS. To compare the performance, we have also computed the truncated Laplacian for each graph.

To visualise the results, we have applied PCA to both GWPS and truncated Laplacian. Figure 6.11(a) shows the clustering results of GWPS, while Figure 6.11(b) shows the

clustering result of the truncated Laplacian. Results show that the WPS provides a better separation between weighted graphs of different classes as compared to the truncated Laplacian. To compare the performance we have computed the rand indices for both methods. Table 6.4 shows that GWPS gives higher accuracy as compared to alternate methods. These results of GWPS on both weighted and unweighted graphs suggest that edge-based methods perform well compared to the vertex-based methods.



(a) GWPS



(b) Truncated Laplacian

Figure 6.11: Clustering results on weighted graphs

Table 6.4: Experimental results on weighted graphs

Method	Accuracy
Wave Packet Signature	0.9931
Truncated Laplacian	0.8855

6.6 Timing analysis

The solution of wave packet signature can be computed in $O(|E|^2)$ time, once all the eigenfunctions are computed. This is due to the fact that computing the amplitude for each edge requires $O(|E|)$ time and there are $|E|$ edges. The vertex supported eigenfunctions can be computed in $O(|V|^3)$ time. However, to compute the edge-interior eigenfunctions, we require to convert the graph into an oriented line graph and find the eigenvectors corresponding to the eigenvalues -1 and 1. So the running time for computing the edge-interior eigenfunctions is $O(|E|^3)$. Therefore the running time of computing GWPS is $O(|E|^3)$. Note that this time can be accelerated by using **eigs** routine in Matlab.

6.7 Summary

In this chapter, we have used the solution of the wave equation on a graph to characterize both weighted and unweighted graphs. The wave equation is solved using the edge-based Laplacian of a graph. The advantage of using the edge-based Laplacian over vertex-based Laplacian is that it allows the direct application of many results from analysis to graph-theoretic domain. Our novel contribution in this chapter was to solve a wave equation on a graph, where the initial condition is a Gaussian wave packet(s) on the edge(s) of the graph. We use the amplitude of the waves on the edge over time to define a signature, called the global wave packet signature (GWPS) to characterize graph. Experimental evaluations show that the proposed signature can perform better as compared to the feature vectors that are based on the discrete Laplacian of the graph.

Chapter 7

Shape Analysis using the Edge-based Laplacian

In this chpater our purpose is twofold. Firstly, we shall solve a heat equation on a graph using the edge-based Laplacian, where the initial condition is a Gaussian heat packet on a single edge of a graph. Secondly, we use the fundamental solution of the heat equation defined using the edge-based Laplacian to define a signature for the points on the surface of a three-dimensional shape.

The key idea in the analysis of three-dimensional deformable shapes is to define an informative and a discriminative feature descriptor that characterizes each feature point on the surface of the shape. Generally these techniques use a feature vector in $\mathbb{R}^n$ [1, 77], which contains both local and global information for that point. These feature descriptors can be used in many ways for analyzing three-dimensional shapes. For correspondence matching, the descriptors are used to find potential correspondence among pairs of points on two different shapes [1, 77]. For clustering the parts of a shape, the signatures can be used to identify semantically coherent parts of an object [2, 67]. Local descriptors can be combined in different ways to define a global shape signature and this can be used for shape classification or recognition [14, 48].

One of our novel contributions in this chapter is to construct the adjacency matrix of a mesh, that represents a three-dimensional shape, in a way that capture the geometric as well as the topological properties of the shape. Next we find the eigenfunctions and eigenvalues of the edge-based Laplacian of the proposed adjacency matrix. Based on the eigenpairs of the edge-based Laplacian, we propose a local signature, called the edge-based heat kernel signature (EHKS) that can be used for segmentation, matching and clustering

three-dimensional shapes. Using local EHKS, we define a global edge-based heat kernel signature (GEHKS), which can be used for shape classification and shape retrieval. In the experiment section, we show the applications of the proposed signature for shape segmentation, shape matching, and shape classification.

We commence this chapter by solving a heat equation on a graph using the edge-based Laplacian, where the initial condition is a Gaussian heat packet on a particular edge of the graph. We demonstrate the solution on a graph with 5 vertices and 7 edges. Next we define signatures that can be used for segmentation, matching and retrieval of three-dimensional shapes. In the experiment section, we perform numerous experiments to demonstrate the effectiveness of the proposed method and compare it with the alternative state-of-the-art methods.

7.1 General solution of the heat equation

In this section we give a similar solution to the heat equation as we did for the wave equation in the previous chapter. We assume that the initial condition is a Gaussian heat packet on a single edge of the graph and see its evolution with time. We will demonstrate the result by simulating the solution on a graph with 5 vertices and 7 edges. To commence, let $G = (V, E)$ be a graph with empty boundary. Suppose a graph coordinate $\mathcal{X}$ defines an edge e and a value of the standard coordinate on that edge x. The eigenfunctions of the edge-based Laplacian are

$$\phi_{\omega,n}(\mathcal{X}) = C(e,\omega)\cos\left(B(e,\omega) + \omega x + 2\pi n x\right).$$

The edge-based heat equation is

$$\frac{\partial h}{\partial t}(\mathcal{X},t) = \Delta_E h(\mathcal{X},t). \tag{7.1}$$

We look for separable solutions of the form $h(\mathcal{X},t) = \phi_{\omega,n}(\mathrm{X})g(t)$. This gives

$$\phi_{\omega,n}(\mathcal{X})g'(t) = g(t)\left(\omega + 2\pi n\right)^2 \phi(\omega,n),$$

which gives a solution for the time-based part as

$$g(t) = \alpha_{\omega,n} e^{-((\omega+2n\pi)^2 t}.$$

By superposition, we obtain the general solution

$$\begin{aligned} h\left(\mathcal{X},t\right) &= \sum_{\omega}\sum_{n} C(e,\omega)\cos\left[B(e,\omega) + \omega x + 2\pi n x\right] \\ &\quad \{\alpha_{\omega,n} e^{-((\omega+2n\pi)^2 t}\}. \end{aligned} \tag{7.2}$$

7.1.1 Initial conditions

Let $p(\mathcal{X})$ be the initial condition, i.e., at $t = 0$, $h(\mathcal{X}, t) = p(\mathcal{X})$. Then

$$p(\mathcal{X}) = \sum_{\omega} \sum_{n} \alpha_{\omega,n} C(e, \omega) \cos\left[B(e, \omega) + \omega x + 2\pi n x\right].$$

We can find $\alpha_{\omega,n}$ using the orthogonality of the eigenfunctions. So we get

$$\alpha_{\omega,n} = \sum_{e} C(e, \omega) \frac{1}{2} \left[F_{\omega,n} + F^*_{\omega,n}\right],$$

where

$$F_{\omega,n} = e^{iB} \int_0^1 dx p(e, x) e^{i\omega x} e^{i2\pi n}.$$

7.1.2 Solution of the heat equation for a Gaussian heat packet

Let the initial position be a normally distributed heat packet $p(e, x) = e^{-a(x-\mu)^2}$ on one particular edge and zero everywhere else. Then we have

$$\begin{aligned} F_{\omega,n} &= e^{iB} \int_0^1 dx e^{-a(x-\mu)^2} e^{i\omega x} e^{i2\pi n x}, \\ &= e^{iB} e^{i\mu\omega} e^{-\frac{\omega^2}{4a}} \int_0^1 dx e^{-a\left(x-\mu-\frac{i\omega}{2a}\right)^2} e^{i2\pi n x}. \end{aligned}$$

Let the Gaussian heat packet is fully contained on one edge, i.e., $p(x, e)$ is only supported on one edge and zero elsewhere. Then

$$F_{\omega,n} = e^{iB} e^{i\mu\omega} e^{-\frac{\omega^2}{4a}} \int_{-\infty}^{\infty} dx e^{-a\left(x-\mu-\frac{i\omega}{2a}\right)^2} e^{i2\pi n x}.$$

Solving the above equation, we get

$$F_{\omega,n} = \sqrt{\frac{\pi}{a}} e^{i[B+\mu(\omega+2\pi n)]} e^{-\frac{1}{4a}(\omega+2n\pi)^2}.$$

Similarly to the above, solving for $F^*_{\omega,n}$, we obtain

$$F^*_{\omega,n} = \sqrt{\frac{\pi}{a}} e^{-i[B+\mu(\omega+2\pi n)]} e^{-\frac{1}{4a}(\omega+2n\pi)^2},$$

and therefore, we get

$$\alpha_{\omega,n} = \sqrt{\frac{\pi}{a}} e^{-\frac{1}{4a}(\omega+2n\pi)^2} C(e, \omega) \cos[B + \mu(\omega + 2\pi n)]. \tag{7.3}$$

Let f be the edge on which the initial function is non-zero. Then the solution becomes

$$\begin{aligned} h(\mathcal{X}, t) &= \sum_{\omega} \sqrt{\frac{\pi}{a}} C(\omega, e) C(\omega, f) \sum_{n} e^{-\frac{1}{4a}(\omega+2\pi n)^2} e^{-(\omega+2n\pi)^2 t} \\ &\quad \cos\left[B(\omega, e) + \omega x + 2\pi n x\right] \cos\left[B(\omega, f) + (\omega + 2\pi n)\mu\right]. \end{aligned} \tag{7.4}$$

To demonstrate the results, we have simulated the heat equation on a graph with 5 vertices and 7 edges, where the initial condition is a Gaussian heat packet on a single edge of the graph. Figure 7.1 shows the evolution of the heat packet for the times $t = 0$, $t = 0.01$, $t = 0.05$ and $t = 0.1$ $t = 1$ and $t = 5$.

7.2 Three-dimensional shape descriptors

In this section, we define shape descriptors for the three-dimensional shapes that uses the eigenvalues and eigenfunctions of the edge-based Laplacian and a solution of the edge-based heat equation. We define local signatures and show their applications in three-dimensional shape segmentation and correspondence matching. We also define a global signature for the purpose of three-dimensional shape classification.

7.2.1 Local descriptor

In this section, we define local signature for every point on the mesh. This point could be any vertex of the mesh or it could be a point on any edge of the mesh. Since the mesh approximates the bounding surface of the three-dimensional shape, therefore this point is defined on the surface of the shape. In the remainder of this chapter, we will use the term p to refer to a point on the mesh (and therefore on the surface of the three-dimensional shape).

To demonstrate the effectiveness of the eigenfunctions of the edge-based Laplacian, we commence by defining an *edge-based global point signature* (EGPS) which is similar to the *global point signature* (GPS), defined by Rustamov [67]. Given (ω^2, ϕ), the eigenpairs of the edge-based Laplacian of the graph, the EGPS for a point p on the surface of the three-dimensional shape is an infinite dimensional feature vector, defined as

$$\text{EGPS}(p) = \left(\frac{1}{\omega_1}\phi_1(p), \frac{1}{\omega_2}\phi_2(p), \frac{1}{\omega_3}\phi_3(p), ... \right).$$

Here $\omega_1^2, \omega_2^2, \omega_3^2, ...$ are the smallest positive eigenvalues, while $\phi_1, \phi_2, \phi_3, ...$ are the corresponding eigenfunctions. In the experimental section, we will show that, although the EGPS can be used for shape segmentation, it is not invariant under bilateral symmetries and cannot be used for shape matching. Note that the point p in EGPS can represent the vertices as well as any point on the edges of the mesh. Therefore EGPS is different than GPS, in the sense that GPS is defined for the vertices of the mesh only.

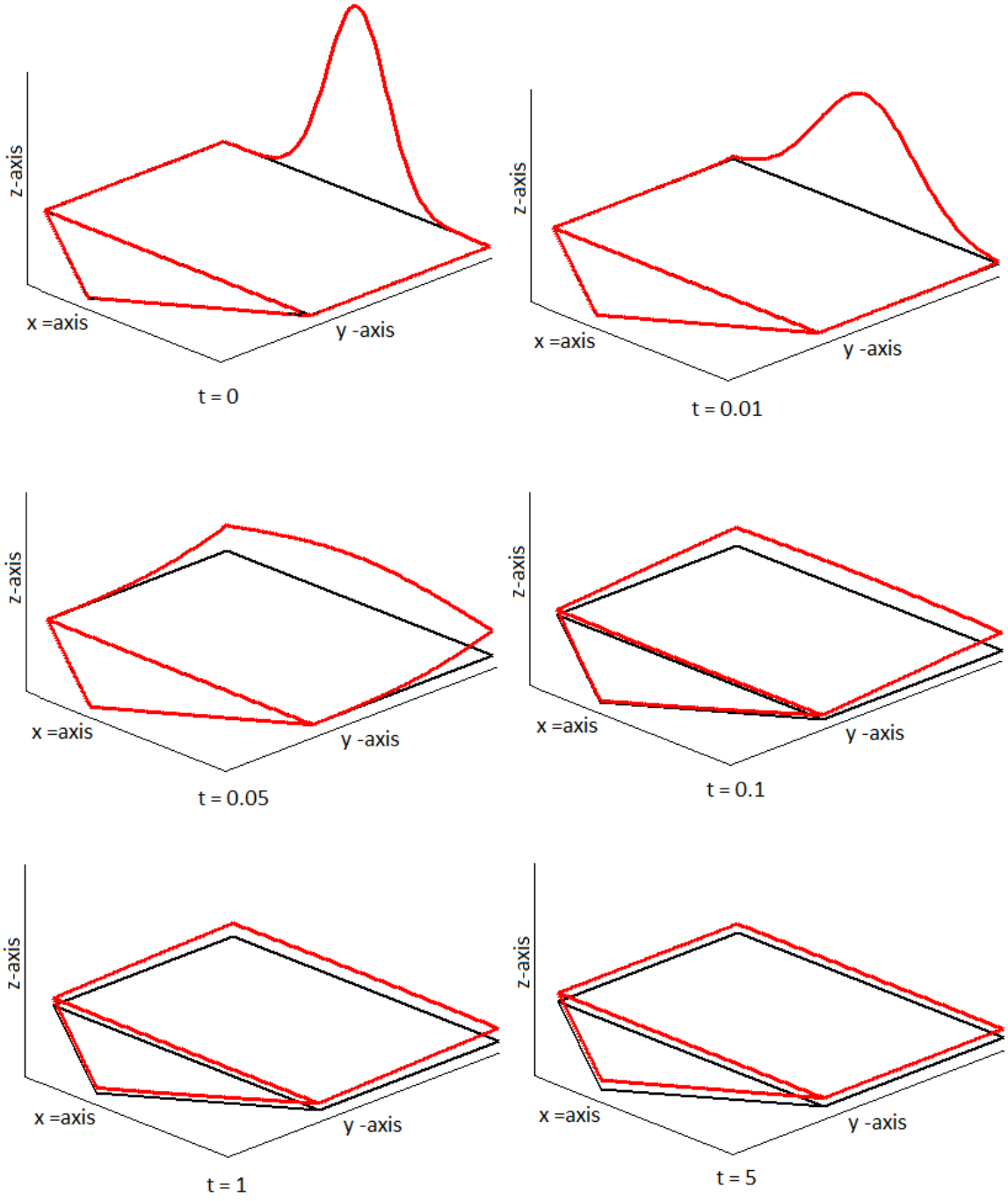


Figure 7.1: Solution of the heat equation on a graph with 6 vertices and 8 edges

To have a more robust representation, we define both local and global shape descriptors, which are based on the fundamental solution of the heat diffusion process, called the heat kernel. The heat kernel, H_t, has the following eigen-decomposition:

$$H_t(x, y) = \sum_{i=0}^{\infty} e^{-\omega^2 t} \phi(x)\phi(y), \tag{7.5}$$

where (ϕ, ω^2) are the eigenpairs of the edge-based Laplacian.

The local signature is defined by sampling the self-heat of vertices and edges over time which can be computed from the heat kernel. Given a point p on the mesh of the three-dimensional shape, its *edge-based heat kernel signature* (EHKS) is a feature vector in k-dimensional feature space, given as:

$$\text{EHKS}(p) = [H_{t_0}(p, p), H_{t_1}(p, p), ..., H_{t_{k-1}}(p, p)]. \tag{7.6}$$

To make EHKS scale invariant, we normalize the EHKS by scaling each H_t by $\int_M k_t(p, p)dp$ [77]. The quantity $\int_M k_t(p, p)dp$ is also called the heat kernel trace, and can be computed as $\sum_i e^{-\omega_i^2 t}$.

7.2.2 Global descriptor

To extend our method to the problem of shape classification we define a *global edge-based heat kernel signature* (GEHKS) for the three-dimensional shape, which is based on the local EHKS of the vertices of the mesh that approximates the bounding surface of the shape. Our approach of defining a global signature for the shape is closely related to the approach of [14]. Given a mesh with n vertices that represents a shape S, we define its global edge-based heat kernel signature as

$$\text{GEHKS}(S) = \text{hist}\left(\text{EHKS}(v_1), \text{EHKS}(v_2), ..., \text{EHKS}(v_n)\right), \tag{7.7}$$

where hist(.) is the histogram operator, and $v_1, v_2, ..., v_n$ represent the vertices of the mesh. Since the GEHKS is defined on small and large values of t, it encodes both the local and the global information about the shape.

7.2.3 Discrete settings

A three-dimensional shape can be conveniently represented by a mesh that approximates the bounding surface of the three-dimensional shape (see Figure 7.2). Therefore to find the corresponding edge-based Laplacian we need to find the adjacency matrix of the mesh.

The simplest way to define the adjacency matrix is to use the unweighted (0-1) or the weighted (distance or proximity) matrix. However, such representations are sensitive to the regularity of the particular triangulation and give little information about the shape itself. In this section, we propose a new method for constructing the adjacency matrix of the mesh that captures its geometric and topological properties.

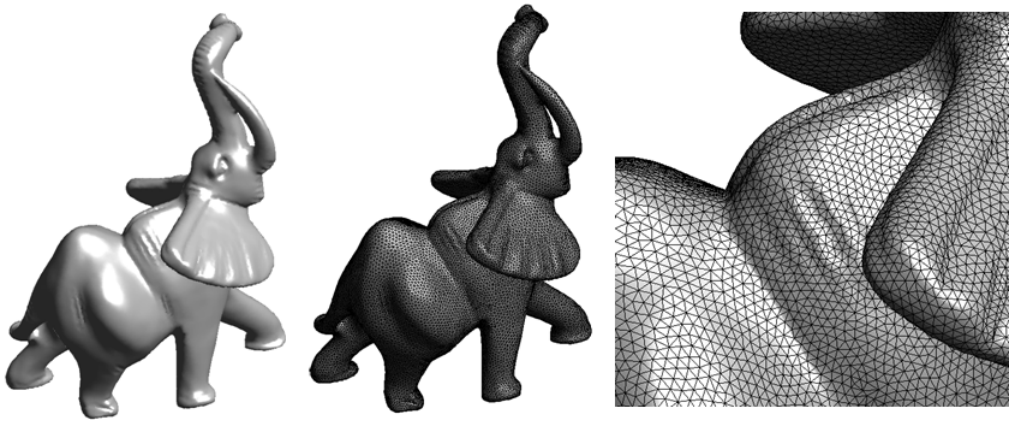


Figure 7.2: Mesh representation of a three-dimensional shape of an elephant

Most of the techniques [2,67,77] for characterizing points on non rigid three-dimensional shapes use the eigenpairs of the Laplace-Beltrami operator (discrete Laplacian). It is important to note the difference between a combinatorial Laplacian and the discrete one. The combinatorial Laplacian is related to the mesh and it does not contain significant information concerning the shape itself. The discrete Laplacian (or Laplace-Beltrami operator) is specifically designed to capture the geometric and topological properties of the underlying surface. Many schemes have been proposed to construct the discrete Laplacian that estimates the Laplace-Beltrami operator [2, 67, 77]. Most of these schemes use the so called cotangent scheme, which was originally introduced in [52].

To estimate the EHKS from a mesh, we construct the adjacency matrix in a way that is consistent with the Laplace-Beltrami operator and that captures the geometric as well as the topological properties of the shape. This can be done by using the cotangent scheme which uses the angle information between the edges and area around each vertex (see Figure 7.3).

Let M is a matrix whose $(i,j)th$ entry is defined as

$$M(i,j) = \begin{cases} \frac{\cot \alpha_{ij} + \cot \beta ij}{2}, & \text{if } (i,j) \in E; \\ 0, & \text{otherwise.} \end{cases} \tag{7.8}$$

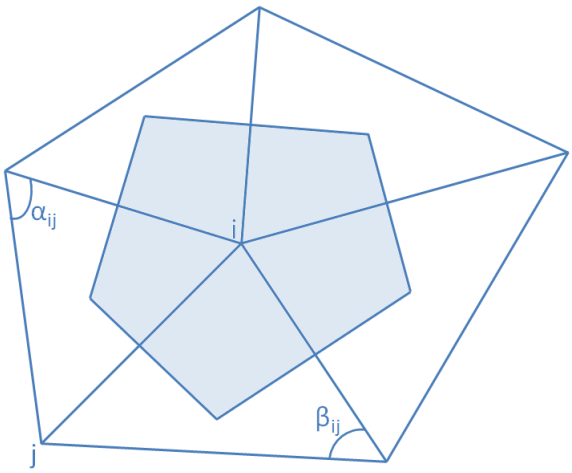


Figure 7.3: Angles and the area appearing in the adjacency matrix

where α_{ij} and β_{ij} are the angles opposite to the edge (i, j), as shown in Figure 7.3. Let S be a diagonal matrix whose ith diagonal entry is the area associated with the triangles abutting the vertex i. We define the symmetric adjacency matrix as $A = S^{1/2}MS^{1/2}$. The $(i, j)th$ entry of the adjacency matrix, in terms of the elements of the matrices M and S, is given as follows:

$$A(i,j) = \begin{cases} \sqrt{S(i,i)S(j,j)}M(i,j), & \text{if } (i,j) \in E; \\ 0, & \text{otherwise.} \end{cases} \tag{7.9}$$

The matrix defined above not only captures more information about the geometric and topological properties of the shape, but it also minimizes the dependence of the adjacency matrix on the mesh.

7.3 Experiments

In this section, we will present both the qualitative and the quantitative analysis of the proposed edge-based heat kernel signature. We perform our experiments on a subset of SHREC 2010 dataset that contains 10 different shapes, each with 20 different non-rigid deformations. Figure 7.4 shows some of these shapes. To find the edge-based eigenpairs, we first construct the adjacency matrix, as described in the previous section. We compute the area associated with each vertex using the method proposed in [88]. We then find the eigenpairs of the normalized adjacency matrix.

To find the edge-based heat kernel signature for shape, we compute first 300 smallest eigenvalues and corresponding eigenvectors using the **eigs** routine in Matlab, which is

used to solve the sparse eigenvalue problem. We compute the scaled EHKS by uniformly sampling 100 points for different values of t over the time interval $[t_{min}, t_{max}]$ where $t_{min} = 4 \ln 10/\lambda_{300}$ and $t_{max} = 4 \ln 10/\lambda_2$ [77].

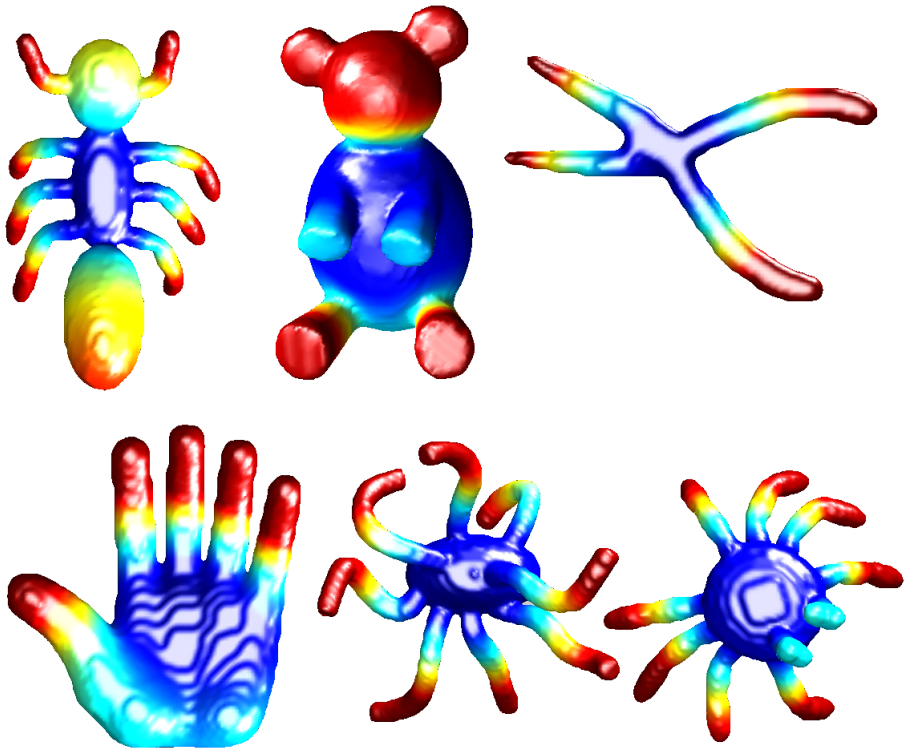


Figure 7.4: The SHREC 2010 database of shapes

7.3.1 Edge-based eigenfunctions for segmentation

We commence by analysing the eigenvalues and eigenfunctions of the edge-based Laplacian. Figure 7.5 shows the values of three different edge-based eigenfunctions (corresponding to the second, fifth, and seventh smallest eigenvalues) on the vertices of the mesh of a three-dimensional shape of an elephant. Here different colours represent different values of the eigenfunction. It is clear from the figure that these eigenfunctions can be used to segment parts of the shape.

To demonstrate the effectiveness of the edge-based eigenpairs for segmentation of the different parts of a three-dimensional shape, we have selected a three-dimensional shape of pliers and its two different deformations. We have computed the EGPS of every vertex of the mesh of each shape. The feature vector is approximated by taking the first 300 smallest non-zero eigenvalues and the corresponding eigenfunctions. Next we have applied the k-mean clustering on the EGPS coordinates, with $k = 5$. Figure 7.6 shows the clustering

Figure 7.5: Eigenfunctions corresponding to some of the small eigenvalues

results of EGPS coordinates.

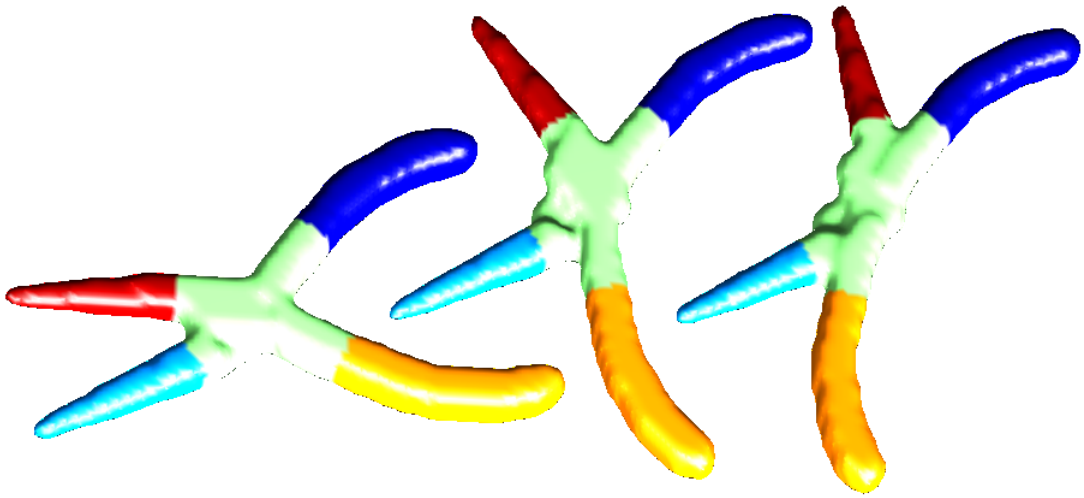


Figure 7.6: Shape segmentation using EGPS

These results suggest that the eigenpairs of the Edge-based Laplacian can be used to extract meaningful segments of a three-dimensional shape. Note that, although the EGPS can be used for shape segmentation, it is not invariant under the bilateral symmetries. For this reason both the GPS and the EGPS are not very effective for correspondence matching problem. In next section, we will show that EHKS, on the other hand, is not only invariant under different poses but it is also invariant under bilateral symmetries of the shape.

7.3.2 Segmentation using EHKS

In this section, we use the EHKS for segmenting different parts of a three-dimensional shape. We commence by computing the EHKS for every vertex of the mesh of six different deformations of a human body. Figure 7.7 shows the results, where different colours represent different values of the EHKS. It is clear from the figure that the EHKS is stable across different deformations of the shape. The results also suggest that EHKS is not only invariant across different deformations, but it is also invariant under bilateral symmetries of the shape.

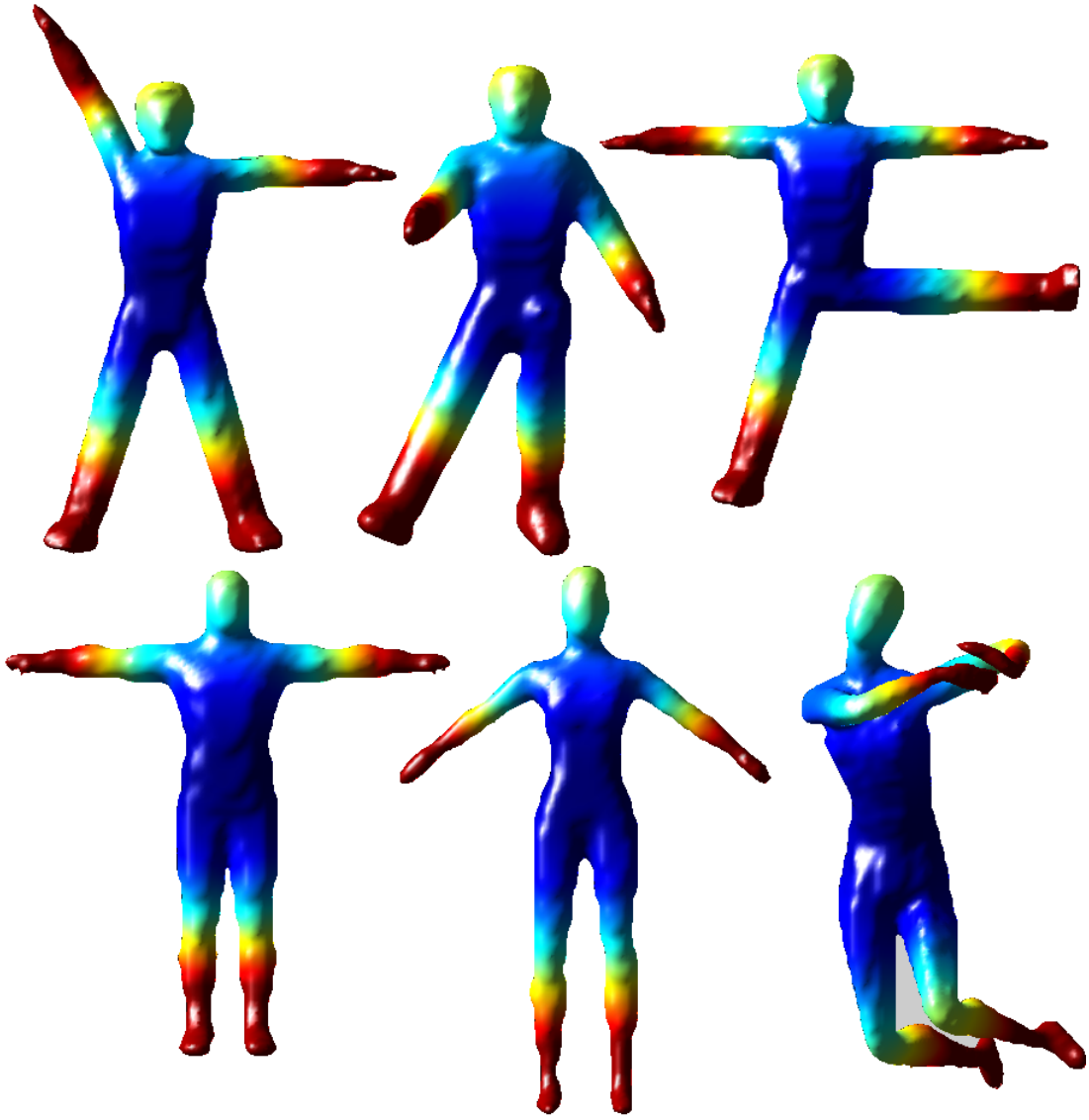
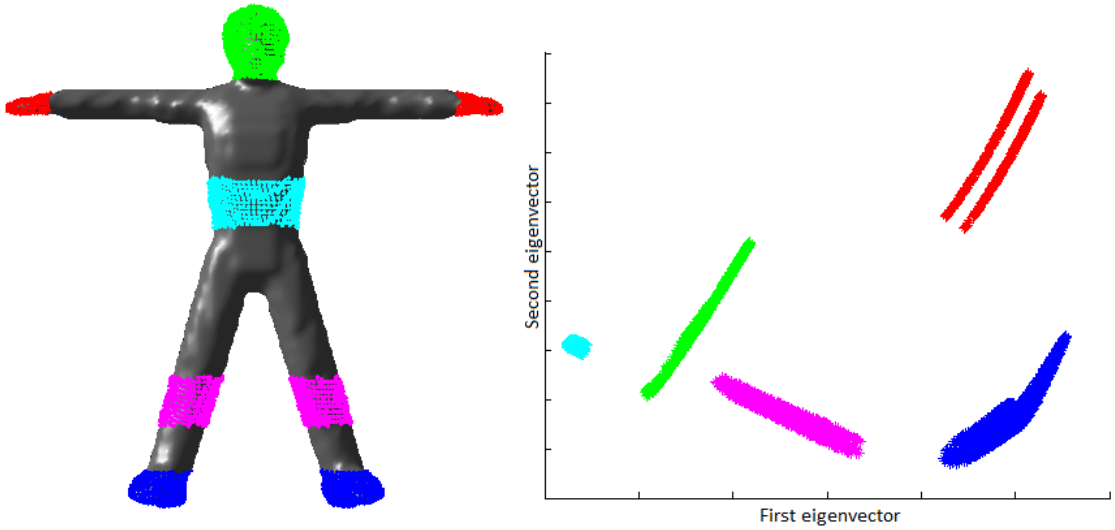


Figure 7.7: EHKS for six shapes

To show the stability of the proposed method for segmentation, we illustrate the

method on the problems of segmenting and classifying parts of a human body using EHKS. For this purpose, we perform three different experiments. In our first experiment, we select all the vertices of the mesh of five different parts of a human body and compute their EHKS. We embed the resulting feature vectors in a lower-dimensional feature space by performing principal component analysis (PCA) [33]. The projection of EHKS on the first two principal components is shown in Figure 7.8, which suggests that the EHKS can be used to distinguish between the different classes of features of the same shape.



(a) A human body (b) Embedding using EHKS

Figure 7.8: Embedding results of different parts of a human body in a two-dimensional Euclidean space

In our next experiment, we show that EHKS is consistent across different deformations of the shape. For this purpose, we select points (only vertices of the mesh) on the hands, the feet, and the head of 15 different poses a human body and compute their EHKS. To visualize the results, we apply PCA on the resulting signatures and embed them in a three-dimensional space. Figure 7.9 shows that not only the EHKS can distinguish between different classes of features such as the hands, the feet, and the head, but can also distinguish classes of features of different shapes.

In our final experiment, we demonstrate the effectiveness of the proposed method for the purpose of shape clustering and compare it with the wave kernel signature. For this purpose, we select a three-dimensional shape and compute EHKS of every vertex of the mesh that approximates the shape. Next we apply k-mean clustering on the resulting signatures. We perform this experiment on three different three-dimensional shapes (i.e.,

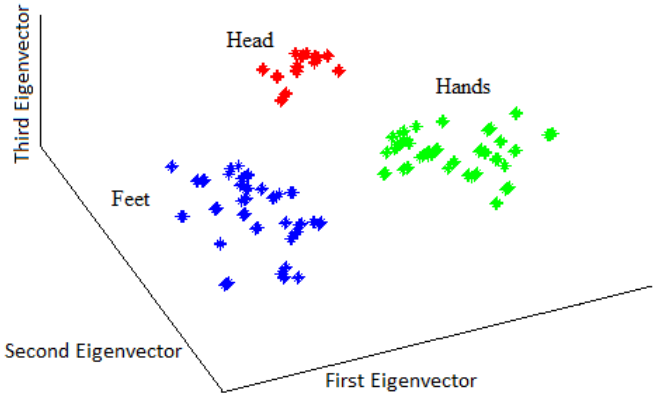
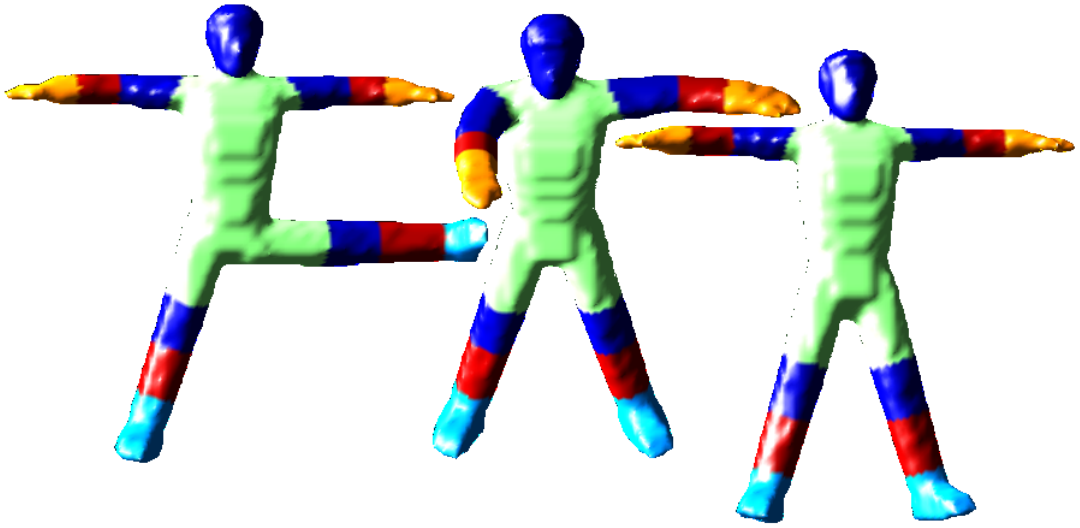


Figure 7.9: Feature points segmentation

the shape of a man, pliers, and an ant), each with three different deformations. Figure 7.10(a), Figure 7.11(a) and Figure 7.12(a) show the segmentation results of the EHKS of the shape of a man, pliers, and an ant and their deformations respectively. Results show that not only the EHKS can find the meaningful segments of the shape but it is also consistent across different deformations of the shape. To compare these results, we have also computed the WKS for the same shapes and their deformations. Figure 7.10(b), Figure 7.11(b) and Figure 7.12(b) show the segmentation results of the WKS. The results show that, although the WKS can segment the parts of a shape, it is not consistent across different deformations of the shape.

7.3.3 Matching using EHKS

In this section, we evaluate the performance of the proposed EHKS for the purpose of correspondence matching. To commence we select three different three-dimensional shapes (i.e., the shape of a human body, an ant, and glasses). We also select a deformed shape corresponding to each of these shapes. For each three-dimensional shape we randomly select a point (vertex) on five different parts of the shape and compute EHKS for each of these points. We also compute the EHKS of every vertex of the mesh that approximates the deformed shape. Next we compute the Euclidean distance of the feature descriptor of selected points on each shape with the feature descriptors of each of the vertices on the corresponding deformed shape. Figure 7.13 shows the first 50 best matches of each of the points on the shape with the points on deformed shape. Results show that EHKS is highly robust across different deformations of the shape. We perform a similar experiment for WKS (Figure 7.13). Results suggest that EHKS is more robust and stable as compared



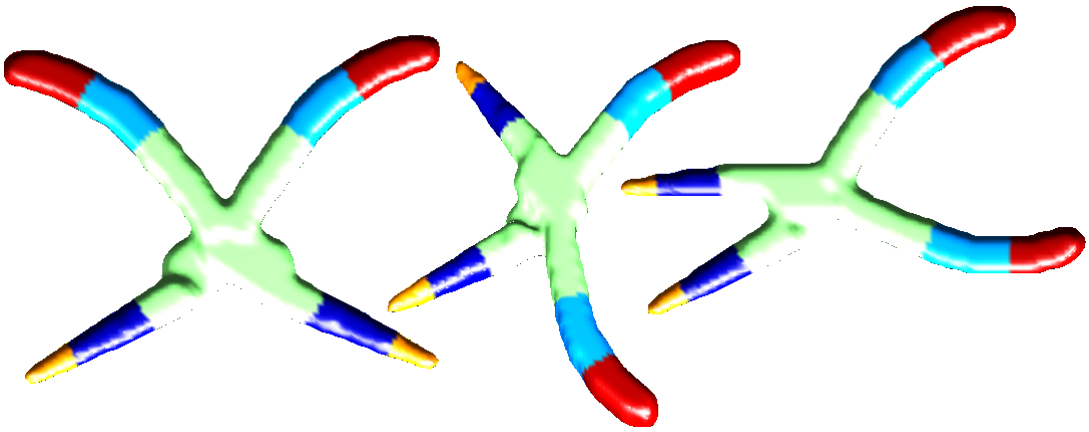
(a) Clustering using EHKS

(b) Clustering using WKS

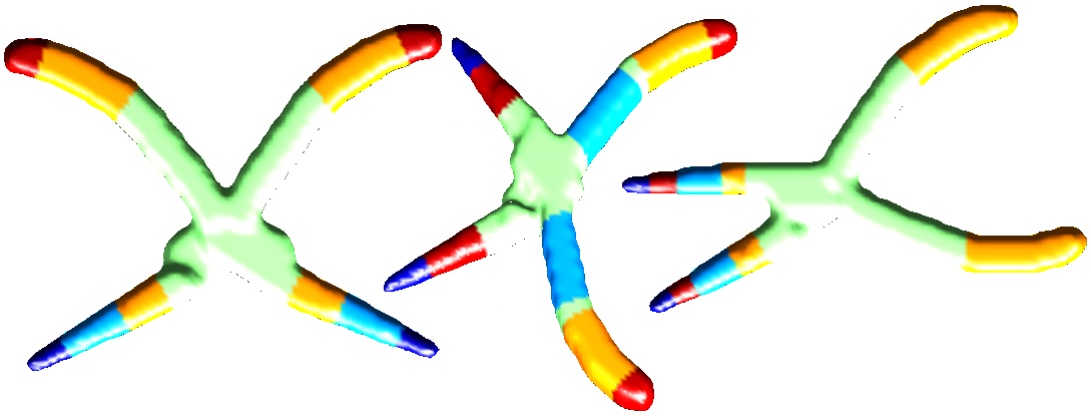
Figure 7.10: Comparison of clustering results of a human body

to WKS. Note that some of the mismatches with WKS are due to bilateral symmetries of the shape.

To compare the performance of EHKS with WKS, we select a three-dimensional shape of a human body and its deformed shape. We randomly select a point (vertex) on 10 different parts of the shape and find the best match for each vertex of the mesh of the deformed shape using the EHKS as well as the WKS. We repeat this experiment for five times and count the number of best matches for both cases. The number of successful matches for both methods are reported in Table 7.1, which suggests that EHKS gives



(a) Clustering using EHKS



(b) Clustering using WKS

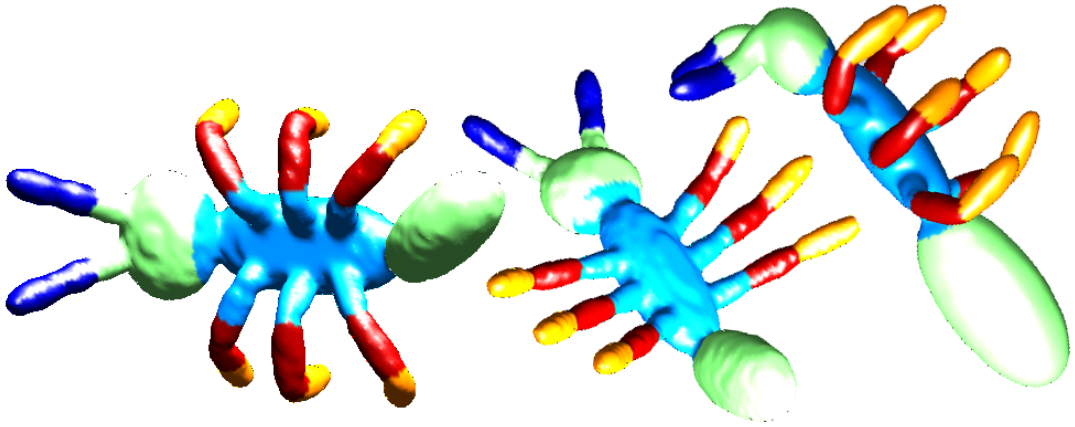
Figure 7.11: Comparison of clustering results of pliers

better performance results compare to WKS.

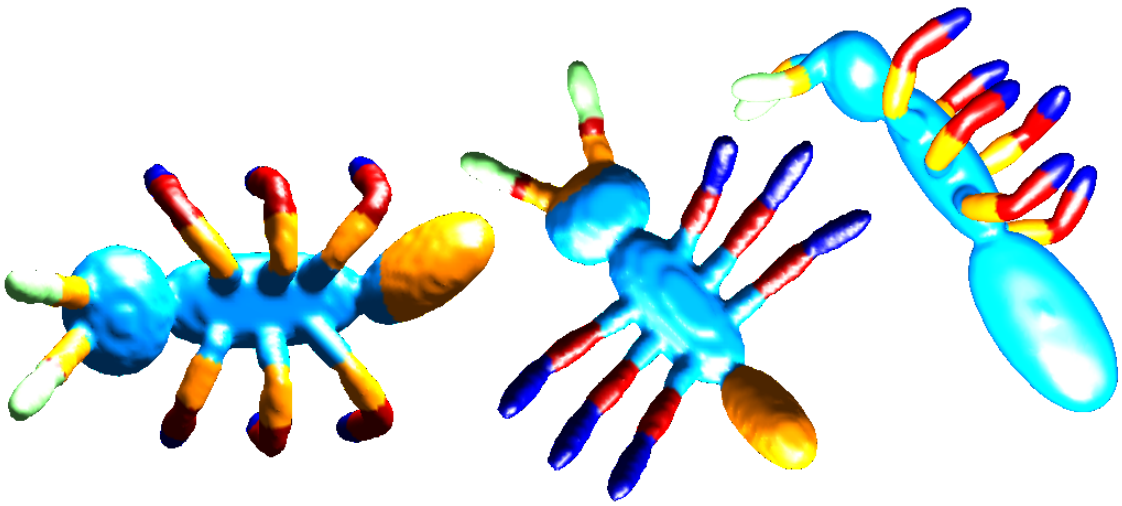
In our final experiment, we demonstrate the effectiveness of EHKS for dense graph matching. For this purpose we select the three-dimensional shape of a human body and compute the EHKS of every point (vertex) on one of its feet. Next we select a deformed shape of the selected shape and compute the EHKS of every vertex of the mesh of the shape. Figure 7.14 shows the best match for points on the selected shape to its deformed shape. The results suggest that the proposed signature is also suitable for the dense matching problem.

7.3.4 Stability analysis of EHKS

In our next experiment, we show the stability of EHKS under controlled noise. For this purpose we take three-dimensional shapes of a human body and a bear and their deformed



(a) Clustering using EHKS

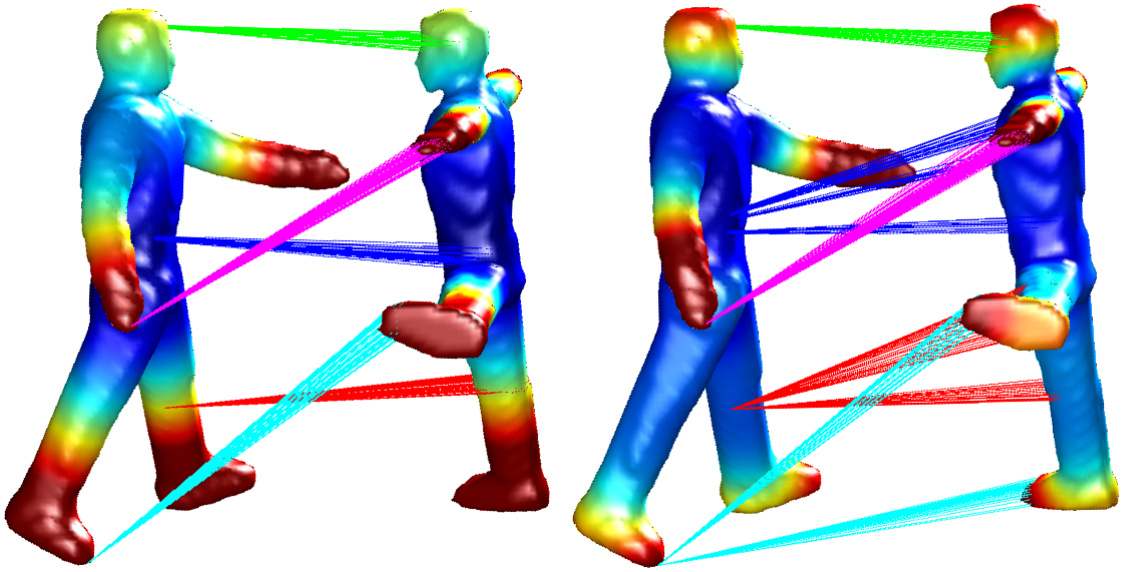


(b) Clustering using WKS

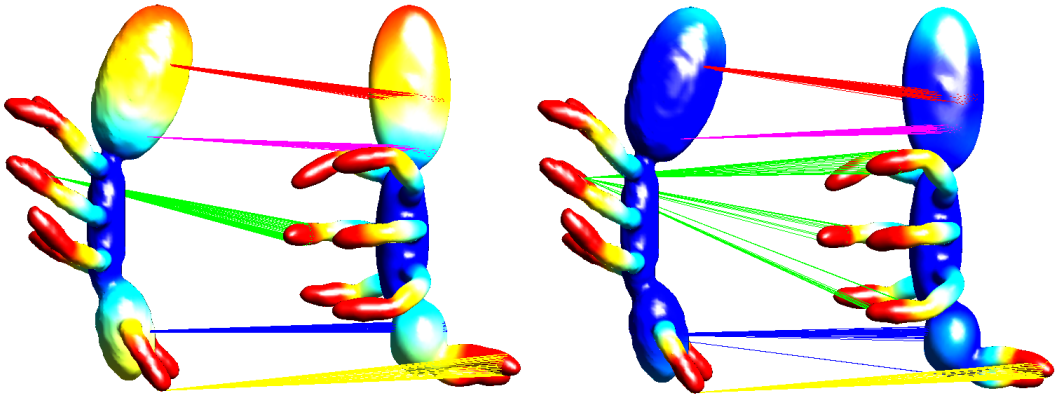
Figure 7.12: Comparison of clustering results of an ant

shapes. We add controlled Gaussian noise to the deformed shapes with mean $\mu = 0$, and standard deviation $\sigma = 0.3$. We randomly select a point (vertex) on three different parts of each of the given shapes and compute their EHKS. Next we compute EHKS for each vertex on the deformed shapes, corrupted by the Gaussian noise. We compute the Euclidean distance of the feature descriptors of the selected vertices with feature descriptors of each vertex on the deformed shape. In Figure 7.15, the lines between shapes show the first 50 best matches of each of the three vertices on the given shape with the points on deformed shape. Results show that the proposed method is robust under controlled Gaussian noise.

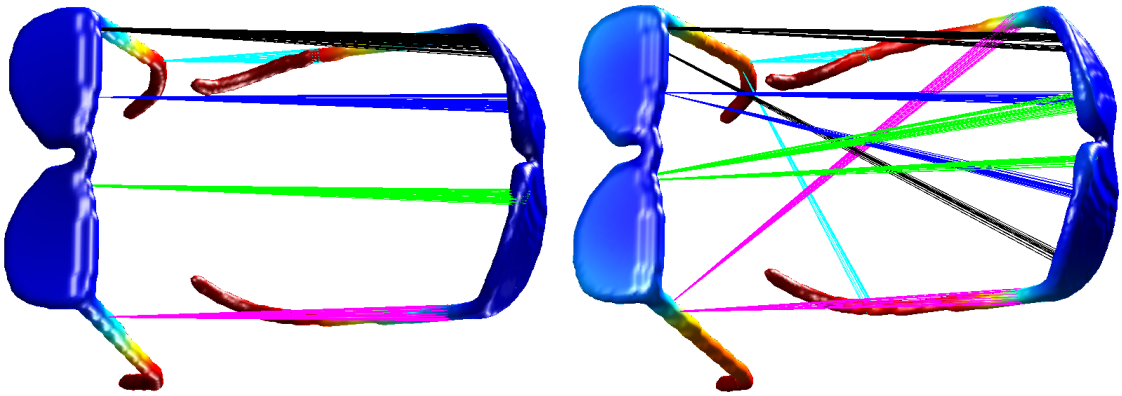
To demonstrate the usefulness of the proposed adjacency matrix, we compare the performance of the EHKS using different adjacency matrices. We select a three-dimensional



(a) EHKS (left) and WKS (right)



(b) EHKS (left) and WKS (right)



(c) EHKS (left) and WKS (right)

Figure 7.13: Comparison of correspondence matching

Table 7.1: Number of best matches

	1	2	3	4	5
EHKS	8	8	8	9	8
WKS	6	8	7	7	8

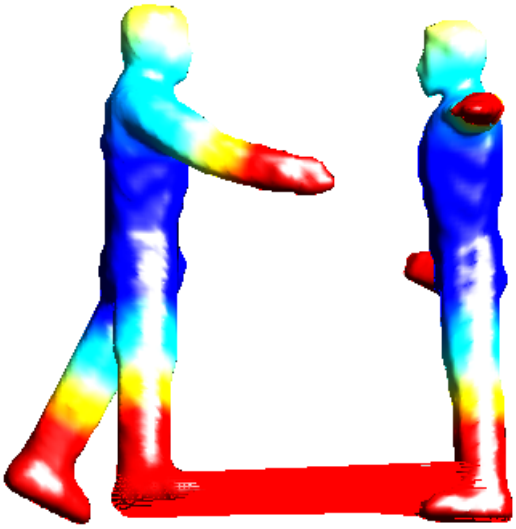


Figure 7.14: Dense point matching using EHKS

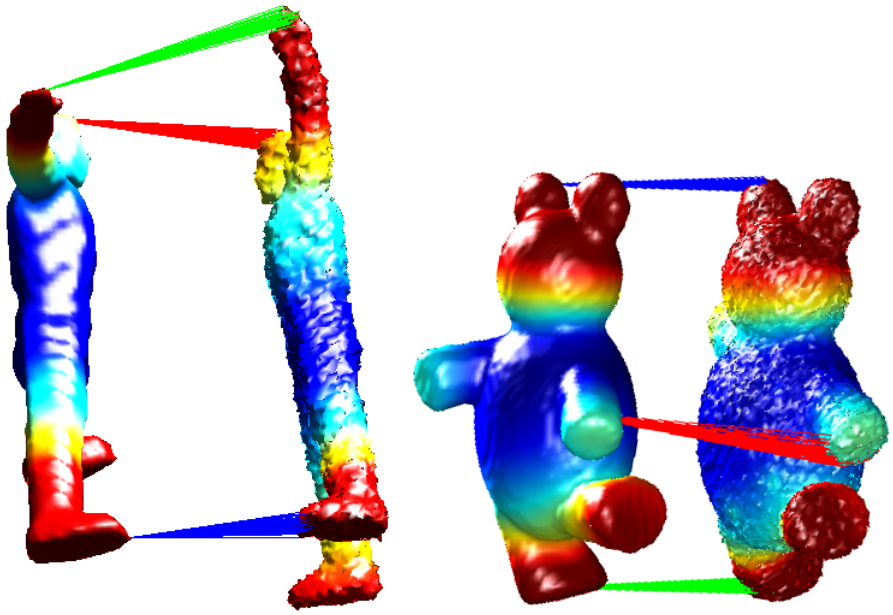
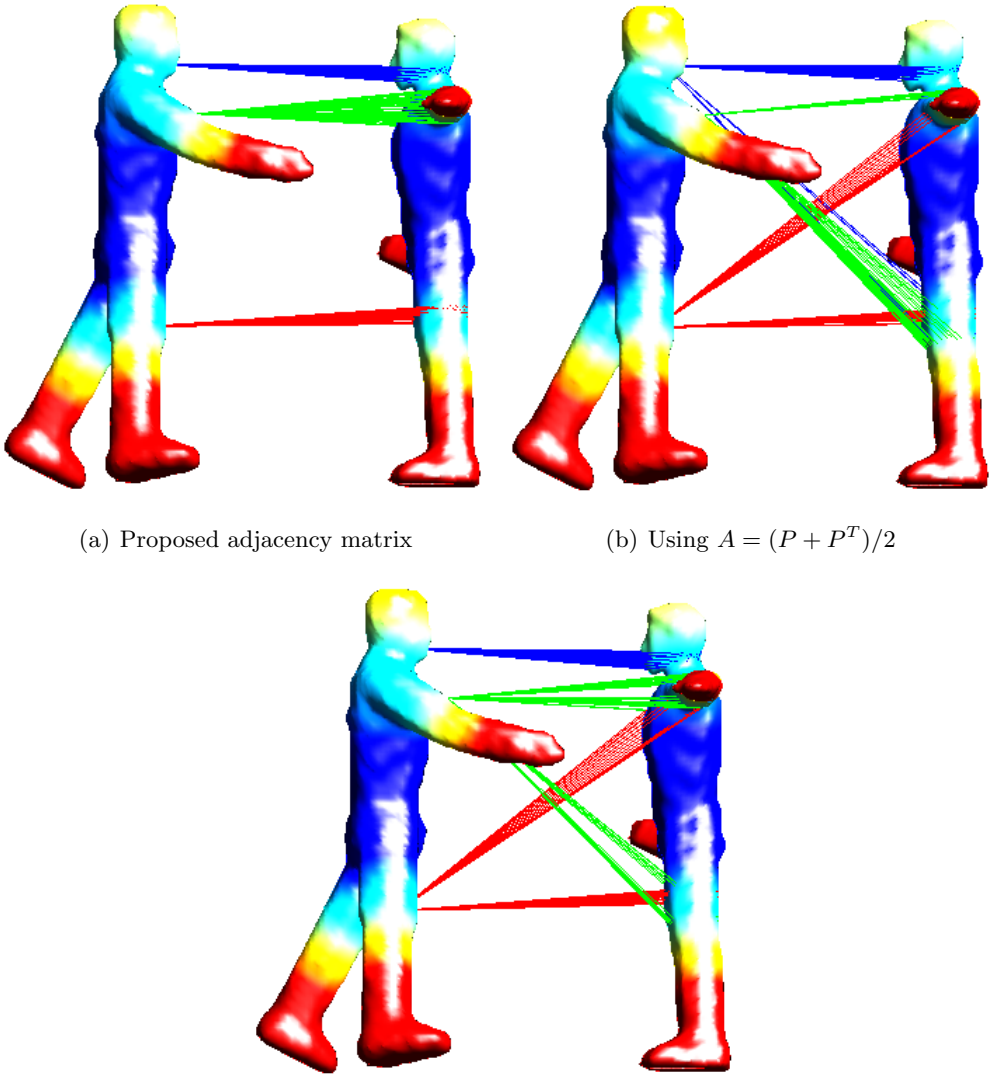


Figure 7.15: Robustness of EHKS under noise

shape of a human body and its deformation. Next we randomly select a point (vertex) on three different parts of the shape and, for each point, find the first 50 best matches on the deformed shape. Figure 7.16 shows the results of EHKS when computed from the proposed matrix (Figure 7.16(a)), the matrix $A = \left(\frac{P+P^T}{2}\right)$ where $P = S^{-1}M$ (Figure 7.16(b)), and the symmetric matrix M that uses the angle information only (Figure 7.16(c)). Results shows that EHKS constructed using the proposed adjacency matrix is more stable than the alternatives.



(a) Proposed adjacency matrix

(b) Using $A = (P + P^T)/2$

(c) Symmetric matrix M

Figure 7.16: Comparison of different adjacency matrices

7.3.5 Classification using GEHKS

In our final experiment, we show the applications of the GEHKS for shape classification. For this purpose, we select three-dimensional shapes of an ant, pliers and an octopus from SHREC 2010 dataset with all of their deformations, and compute GEHKS for each of these shapes. To visualize the results, we apply PCA on these GEHKS and embed them in a three-dimensional space. Figure 7.17 shows that the proposed method can be useful for clustering different shapes. To compare the accuracy of the proposed method we perform a similar experiment with WKS and compute the rand indices for both methods. The accuracy of the proposed method was 0.8514 while that of the WKS was 0.7893. These results show that the EHKS is more informative than the WKS, and gives higher performance for shape classification.

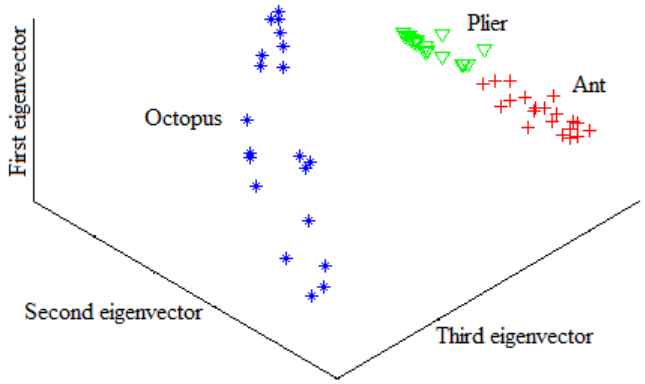


Figure 7.17: Classification of different shapes

7.4 Timing analysis

The running time of computing the EHKS for a vertex of the mesh is dominated by the eigen-decomposition of the adjacency matrix of the mesh. The worst-case running time of computing these eigenfunction is $O(|V|^3)$. Computing the adjacency matrix requires $O(|E|)$ time. Once the eigenpairs of the edge-based Laplacian are known, the EHKS can be quickly computed and its running time depends upon the number of eigenvalues that we are using to approximate EHKS (in our case it is 300). Note that to compute the EHKS of a vertex of the mesh, we don't need to find the adjacency matrix of the oriented line graph. This is due to the fact that edge-interior eigenfunctions are zero on the vertices of the graph. Hence, the worst-case time of computing EHKS for the vertices of the mesh

is $O(|V|^3)$. In practice the execution time can be accelerated by using the **eigs** routine of matlab. This is due to the fact that the mesh representation of the 3-D shape is usually very sparse and also we only need to find a fixed number of smallest eigenvalues and corresponding eigenvectors. Finally, to compute the GEHKS, we need to compute the EHKS for every vertex, and so the worst-case running time of computing GEHKS can be $O(|V|^4)$.

7.5 Summary

We have presented a method for analyzing three-dimensional non-rigid shapes, which is based on the heat equation defined over the edge-based Laplacian. Our novel contribution in this chapter is to define the adjacency matrix of the mesh (using the cotangent scheme and area around each vertex of a mesh) in a way that is robust under different shape deformations and which captures the geometric as well as the topological properties of the shape. We have computed the edge-based eigenvalues and eigefunctions of the shape using the proposed adjacency matrix. Our main contribution in this chapter is to define pose invariant shape signatures using the edge-based heat equation. Experimental results show that the proposed signatures can be used for clustering, correspondence matching and classifying three-dimensional shapes with higher accuracy.

Chapter 8

Conclusion

This chapter summarises the main contributions of the thesis and discusses its limitations and possible future directions for research.

8.1 Contributions

Our goal in this thesis is to define invariants for a graph that can be used to efficiently characterize the graph. We have presented applications of proposed methods for the purpose of graph characterization and three-dimensional shape matching, clustering and classification.

Our first contribution in this thesis is to develop a novel and an efficient method for computing the coefficients of the reciprocal of the Ihara zeta function, called the Ihara coefficients. The Ihara zeta function is determined by frequencies of the prime cycles of the graph. It has proved to be a powerful tool in the analysis of graph structures and has been successively used in pattern analysis and machine learning. However, despite its attractions as a compact representation of graph structure, applications of the Ihara zeta function have been limited due to the computational overheads required to calculate it. In this thesis we have developed methods to efficiently compute Ihara coefficients. This is accomplished by first establishing a relationship between the Ihara coefficients and the Bell polynomials. Then using a recursive formulation for the complete Bell polynomials, we have shown how the set of all Ihara coefficients can be efficiently computed in $O(|E|^3)$ time and how the set of low order Ihara coefficients can be computed in $O(|V|^3)$ time. This relationship also helps us to understand the structure of each of the Ihara coefficients in terms of frequencies of prime cycles in the graph.

Our second contribution is to develop efficient methods for characterizing labelled graphs with higher accuracy. We have defined a graph kernel which is based on backtrackless walks on a graph. A graph kernel is a kernel function which gives similarity measure between two graphs. It computes the inner product of the two graphs by implicitly assuming the embedding of the graphs into a higher-dimensional feature space. To compute the kernel we have used the product graph formalism that can be used to find a measure of similarity between graphs without explicitly embedding the graph into a higher-dimensional feature space. The worst-case running time of the proposed graph kernel is $O(|V_\times|^3)$, which is the same as the worst-case running time of the kernel defined using random walks on a graph. Here $|V_\times|$ represents the size of the vertex set of the product graph. We have experimentally shown that, by avoiding tottering, the kernel defined using backtrackless walks provides a more richer representation of the graph than the kernel defined using random walks and can characterize the labelled graphs with higher accuracy.

We have also developed efficient methods for characterizing unlabelled graphs using feature vectors composed of Ihara coefficients and backtrackless walks on a graph. We have shown how these feature vectors can be computed in $O(|V|^3)$ in the worst-case. We have experimentally demonstrated that when the graph is cyclic then the feature vector composed of Ihara coefficients gives better classification results. However, when the graph has many branches then the feature vector constructed from backtrackless walks gives higher performance. Another advantage of using backtrackless walks and Ihara coefficients is that they are less prone to problems of failing to distinguish graphs due to cospectrality of the Laplacian or the adjacency matrix. This is due to the fact that the Ihara coefficients and backtrackless walks are based on the adjacency matrix of the oriented line graph, which is closely akin to the discrete time quantum walk on a graph [56]. We have experimentally shown that the Ihara coefficients and backtrackless walks can distinguish graphs which are cospectral with respect to their adjacency matrices as well as the adjacency matrices of their complements. On the other hand, random walks cannot distinguish such graphs.

The second part of the thesis discusses the geometric graphs and their applications in machine learning and three-dimensional computer vision. A geometric graph is a graph with a geometric realization that assigns real interval to every edge of the graph. Functions may therefore exist both at the vertices and on the interior of edges. This results in two part Laplacian, i.e., vertex-based Laplacian and edge-based Laplacian. One of our

contributions in this thesis is that we have established methods to explicitly compute the eigenfunctions of the edge-based Laplacian of the graph. This revealed a connection between the eigenpairs of the edge-based Laplacian of the graph with both the random walks and backtrackless walks on the graph. The eigenfunctions of the edge-based Laplacian of a graph are of two types, vertex-supported eigenfunctions and edge-interior eigenfunctions. We have shown that the vertex-supported eigenfunctions are related to the eigenvectors of the line graph, while the edge-interior eigenfunctions are related to the certain eigenvectors of the oriented line graph.

Once the complete set of eigenfunctions of the edge-based Laplacian is known, we can use it to implement other complex partial differential equations on a graph that would have no meaning if defined using the traditional vertex-based Laplacian. Our next contribution is to define a wave equation on the graph using the edge-based Laplacian and using its solution to define a robust signature, called the global wave packet signature (GWPS), for both weighted and unweighted graphs. The GWPS is defined using the amplitudes of a travelling wave on different edges of the graph. We have conducted experiments on both weighted and unweighted graphs. For each method we have made an experimental comparison with the alternative methods. Experimental results show the effectiveness of the GWPS.

Our final contribution in this thesis is to develop methods based on the edge-based Laplacian of a mesh for the three-dimensional shape analysis. One of our novel contributions is to define an adjacency matrix of the mesh that approximates the bounding surface of the three-dimensional shape. The adjacency matrix is defined using the cotangent scheme and area around each vertex of a mesh and it captures both the geometric and topological properties of the shape. Our second contribution is to define pose invariant shape signature using the edge-based heat equation which is defined using the edge-based eigenpairs computed from the proposed adjacency matrix. This is called the edge-based heat kernel signature (EHKS) and it can be used for characterizing points on the surface of a three-dimensional shape. The EHKS is robust under different shape deformations and therefore it can be used for correspondence matching between deformed shapes. It can also be used to find meaningful parts of a three-dimensional shape and is invariant under bilateral symmetries. Our final contribution in this chapter was to define a global signature, called the global edge-based heat kernel signature (GEHKS). The GEHKS is defined by binning the local EHKS, and it can be used for shape classification and shape

retrieval. We have performed numerous experiments to demonstrate the applications of the proposed signatures for the purpose of shape segmentation, shape matching, and shape classification.

8.2 Limitations and future work

Although the methods proposed in this thesis outperform the state-of-the-art methods, there are some limitations with the proposed methods. In this section we discuss some of the limitations and possible approaches for future directions of research.

The advantage of using backtrackless walks or Ihara coefficients over random walks is that they reduce tottering which gives a richer representation of the graph structure. However, although the problem of tottering is reduced, it is not completely avoided. This is because the walks of higher length or high order Ihara coefficients may introduce some noise in the structural representation of the graph. For example, a backtrackless walk of length 6 might correspond to a random walk on six different edges of a graph or a random walk on a triangle that traverses the triangle twice. For this reason high order backtrackless walks and high order Ihara coefficients are not suitable to distinguish graphs. One possible approach to further reduce this problem is to look at the coefficients of the Bartholdi zeta function, which is a generalization of the Ihara zeta function that keeps track of the number of "bumps" (or the number of times a walk totters). Other directions are to look at the coefficients of path zeta function and edge zeta function to compute more interesting invariants for graph representation.

Another advantage of backtrackless walks and Ihara coefficients over random walks is that they are more powerful in distinguishing non-isomorphic cospectral graphs. We have demonstrated that backtrackless walks and Ihara coefficients can distinguish non-isomorphic graphs, which are cospectral with respect to their adjacency matrices as well as the adjacency matrices of their complements. However, backtrackless walks or Ihara coefficients cannot distinguish strongly regular cospectral graphs. Emms et al. [20] have conjectured that such graphs can be distinguished by quantum walks on graphs. It is worth trying to see if we can extract invariants from the Bartholdi zeta function or the edge zeta function that can be used to distinguish such graphs.

Since the feature vector composed of Ihara coefficients uses low order Ihara coefficients to reduce the problem of tottering, it can fail to distinguish cyclic graphs that contain cycles of higher length only. For example if the girth of a graph is r, and we are using

a feature vector whose length is k, where $k < r$, then the feature vector will contain all zeros. The problem can be reduced by selecting a proper weighting constant and carefully selecting the length of the feature vector. However, if the graph contains cycles of mixed lengths, the feature vector composed of the Ihara coefficients might fail. It might be interesting to adopt a more sophisticated strategy to select the feature vector length and the weight associated with each component of the feature vector.

For our work related to backtrackless walks and Ihara coefficients, we need to develop more general framework that can be used to characterize a large families of graphs. One of the limitations of the work presented in this thesis is that we have proposed different invariants for different types of graphs. We have used the Ihara coefficients to characterize a graph when it represents a cyclic structure. Examples of such graphs are Delaunay triangulations. Since the Ihara coefficients do not apply to hierarchical structures (like trees), we propose to use backtrackless walks for graphs where branches are present. Examples of such graphs are graphs extracted from chemical compounds. However, some classes of graphs exhibit both kinds of structures. For example the relative neighbourhood graph, which is a subset of Delaunay triangulation, exhibits a cyclic structure with some branches present. Another possible future direction is to combine these two invariants in an interesting way and to find a single feature vector that keeps track of both the cyclic structure of the graph and the number of tree like structures in the graph. Such feature vectors can be used to characterize a large family of graphs, including the graphs that have many cycles and branches.

Although graph invariants defined using the edge-based Laplacian improve the accuracy of the traditional methods, one of the drawbacks of such invariants is that it requires us to find the eigenfunctions of the oriented line graph of the original graphs. The size of the oriented line graphs is $2|E|$, where $|E|$ represents the number of the edges in the graph. Therefore the time required to compute the edge-interior eigenfunctions is $O(|E|^3)$. Hence, if the graph is dense, the computation of the edge-based Laplacian can be expensive. We have already proposed methods in this thesis that computes graph kernels based on backtrackless walks and Ihara coefficients in $O(|V|^3)$ worst-case time that do not require us to convert the graph into oriented line graph. It is worth trying to see if we can develop methods for efficiently computing edge-interior eigenfunctions of the graph directly, without transforming the graph into the oriented line graph.

One of the limitations of the work presented in this thesis about the analysis of the

edge-based Laplacian is that it is limited to the case where edge lengths are assumed to be uniform. It might be interesting to extend the work to the graphs where the edge lengths may vary.

Finally, the methodologies developed in this thesis can be explored in various research areas such as biological networks, social networks and complex systems. Furthermore, the definition of edge-based Laplacian allows us to define other complex partial differential equations on graphs that have close meaning to equations in analysis.

List of Symbols

Γ	Set of graphs
G	Graph
V	Vertex set of a simple graph
E	Edge set of a simple graph
D	Directed graph
SDG	Symmetric directed graph
LG	Line graph of a graph
OL	Oriented line graph of a graph
$G_\times$	Product graph of two graphs
$V_\times$	vertex set of the product graph
$E_\times$	Edge set of the product graph
$\mathring{G}$	Graph that excludes the boundary vertices and edges that are incident with the boundary vertices
$\mathring{V}$	Vertex set of $\mathring{G}$
$\mathring{E}$	Edge set of $\mathring{G}$
A	Adjacency matrix of a graph
$\tilde{A}$	Row-normalized adjacency matrix of a graph
L	Laplacian matrix of a graph
T	Perron-Frobenius operator (adjacency matrix of the oriented line graph)
U	Adjacency matrix of the line graph
A_k	Matrix whose $(u, v)^{th}$ entry gives the number of backtrackless walks of length k starting at vertex u and ending at vertex v

$\zeta_G(u)$	Ihara zeta function of a graph G
$[C]$	Equivalence classes of prime cycles
N_m	Number of prime cycles of length m
$B_{n,k}$	Partial Bell polynomial
B_n	Complete Bell polynomial
c_n	n^{th} Ihara coefficient
$\overrightarrow{bw}$	Pattern vector composed of backtrackless walks
$\overrightarrow{rw}$	Pattern vector composed of random walks
$\overrightarrow{ic}$	Pattern vector composed of Ihara coefficients
κ	Graph kernel
ϵ_k	weight assigned to pattern vectors and graph kernels
$\mathcal{G}$	Geometric graph
∂G	Boundary of graph
Δ_E	Edge-based Laplacian
Δ_V	Vertex-based Laplacian
$n_{e,v}$	Outward-pointing unit vector along edge e at vertex v
$\mathcal{V}$	Vertex measure
$\mathcal{E}$	Edge measure
x_e	Standard edge coordinate
λ	Eigenvalue of the vertex-based Laplacian
g	Eigenvector of the vertex-based Laplacian
ω	Square root of the eigenvalue of the edge-based Laplacian
ϕ	Eigenfunction of the edge-based Laplacian

Abbreviations

EHKS Edge-based heat kernel signature

GEHKS Global edge-based heat kernel signature

HKS Heat kernel signature

GHKS Global heat kernel signature

WKS Wave kernel signature

GWKS Global wave kernel signature

GPS Global point signature

EGPS Edge-based global point signature

WPS Gaussian wave packet signature

GWPS Global gaussian wave packet signature

RW Random walk

BW Backtrackless walk

IC Ihara coefficient (The coefficient of the reciprocal of the Ihara zeta function)

PCA Principal component analysis

MDS Multidimensional scaling

SE Standard error

md2 Graph where the degree of each vertex is at least 2

DT Delaunay triangulation

GG Gabriel graphs

RNG Relative neighbourhood graphs

COIL Columbia object image library

Mutag Mutagenicity

References

[1] M. Aubry, U. Schlickewei, and D. Cremers. Pose-consistent 3d shape segmentation based on a quantum mechanical feature descriptor. *Proc. 33rd DAGM symposium, Frankfurt, Germany*, 2011.

[2] M. Aubry, U. Schlickewei, and D. Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. *Tech. rep., TU München, Germany*, 2011.

[3] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, pages 1373–1396, 2003.

[4] E. T. Bell. Partition polynomials. *The Annals of Math. Second Series*, 29(1/4):38–46, 1927.

[5] E. T. Bell. Exponential polynomials. *The Annals of Math. Second Series*, 35(2):258–277, 1934.

[6] H. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. Shindyalov, and P. E. Bourne. The protein data bank. *Nucleic Acids*, 28:235–242, 2000.

[7] D. Bonchev. *Chemical Graph Theory: Introduction and Fundamentals*. 1991.

[8] K. M. Borgwardt and H. Kriegel. Shortest-path kernels on graphs. *In Proceedings of 5th IEEE ICDM*, pages 74–81, 2005.

[9] K. M. Borgwardt, C. S. Ong, S. Sch oonauer, S. V. N. Vishwanathan, A. J. Smola, and H. Kriegel. Protein function prediction via graph kernels. *In Proceedings of Intelligent Systems in Molecular Biology (ISMB)*, 2005.

[10] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener. Graph structure in the web. *Computer Networks 33*, pages 309–320, 2000.

[11] A. M. Bronstein. Spectral descriptors for deformable shapes. *arXiv preprint arXiv:1110.5015*, 2011.

[12] B. P. Brooks. The coefficients of the characteristic polynomial in terms of the eigenvalues and the elements of an n×n matrix. *Applied Mathematics*, pages 511–515, 2006.

[13] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Lett. 18(8)*, pages 689–694, 1997.

[14] U. Castellani, P. Mirtuono, V. Murino, M. Bellani, G. Rambaldelli, M. Tansella, and P. Brambilla. A new shape diffusion descriptor for brain classification. *MICCAI*, pages 426–433, 2011.

[15] F. R. K. Chung. Spectral graph theory. *American Mathematical Society*, 1997.

[16] R. R. Coifman and S. Lafon. Diffusion maps. *Applied and Computational Harmonic Analysis*, 21(1):5 – 30, 2006.

[17] B. Delaunay. Sur la sphére vide, izvestia akademii nauk sssr, otdelenie matematicheskikh i estestvennykh nauk. pages 793–800, 1934.

[18] A. Elad and R. Kimmel. On bending invariant signatures for surfaces. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(10):1285–1295, 2003.

[19] H. ElGhawalby and E. R. Hancock. Graph embedding using an edge-based wave kernel. *SSPR/SPR*, 2010.

[20] D. Emms, S. Severini, R. C. Wilson, and E. R. Hancock. Coined quantum walks lift the cospectrality of graphs. *Pattern Recognition*, 2009.

[21] F. Escolano, E. R. Hancock, and M. A. Lozano. Heat diffusion: Thermodynamic depth complexity of networks. 2012.

[22] M. Fiedler. A property of eigenvectors of non-negative symmetric matrices and its application to graph theory. *Czechoslovak Mathematics Journal*, 1975.

[23] J. Friedman and J.P. Tillich. Calculus on graphs. *CoRR*, 2004.

[24] J. Friedman and J.P. Tillich. Wave equations for graphs and the edge based laplacian. *Pacific Journal of Mathematics*, pages 229–266, 2004.

[25] K. R. Gabriel and R. R. Sokal. A new statistical approach to geographic variation analysis. *Systematic Zoology*, pages 205–222, 1969.

[26] T. Gartner, P. Flach, and S. Wrobel. On graph kernels: Hardness results and efficient alternatives. *16 Annual Workshop on Kernel Machines. Heidelberg: Springer-Verlag*, 2003.

[27] C. D. Godsil and B. D. McKay. Constructing cospectral graphs. *Aequationes Mathematicae*, 25(1):257–268, 1982.

[28] C. Harris and M. Stephens. A combined corner and edge detector. *In Fourth Alvey Vision Conference, Manchester, UK*, pages 147–151, 1988.

[29] P. R. He, W. J. Zhang, and Q. Li. Some further development on the eigensystem approach for graph isomorphism detection. *Journal of the Franklin Institute*, 342(6):657 – 673, 2005.

[30] M. D. Horton. Ihara zeta functions of irregular graphs. In *Ph.D. thesis, University of California, San Diego*, 2006.

[31] T. Horváth, T. Gärtner, and S. Wrobel. Cyclic pattern kernels for predictive graph mining. *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery*, pages 158–167, 2004.

[32] N.E. Hurt. Mathematical physics of quantum wires and devices: From spectral resonances to anderson localization. *Math. and its Appl*, 2000.

[33] I. T. JOLLIFFE. Principal component analysis. *Springer-Verlag, New York*, 1986.

[34] S. Jouili and S. Tabbone. Graph embedding using constant shift embedding. In *ICPR Contests*, pages 83–92, 2010.

[35] H. Kashima and A. Inokuchi. Kernels for graph classification. *In ICDM Workshop on Active Mining*, 2003.

[36] R. S. Kondor and J. Lafferty. Diffusion kernels on graphs and other discrete structures. *In Proceedings of the ICML*, pages 315–322, 2002.

[37] M. Kotani and T. Sunada. Zeta function of finite graphs. *Journal of Mathematics, University of Tokyo 7(1)*, page 2000, 7-25.

[38] P. Kuchment. Quantum graphs: an introduction and a brief survey. 2008.

[39] J. Lafferty and G. Lebanon. Diffusion kernels on statistical manifolds. *J. Mach. Learn. Res.*, 6:129–163, 2005.

[40] B. Lévy. Laplace-beltrami eigenfunctions towards an algorithm that "understands" geometry. In *Shape Modeling International*, page 13, 2006.

[41] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. volume 1, pages 281–297. University of California Press, 1967.

[42] P. Mahé, N. Ueda, T. Akutsu, J. Perret, and J. Vert. Extensions of marginalized graph kernels. *In Proceedings of the 21st International Conference on Machine Learning*, 2004.

[43] A. Micheli. Neural network for graphs: A contextual constructive approach. *Neural Networks, IEEE Transactions on*, 20(3):498 –511, 2009.

[44] H. Murase and Nayar S. K. Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14(1):5–24, 1995.

[45] B. Nadler, S. Lafon, R. Coifman, and I. Kevrekidis. Diffusion maps, spectral clustering and eigenfunctions of fokker-planck opera-tors. *Advances in Neural Information Processing Systems*, pages 955–962, 2006.

[46] C. H. Nguyen and H. Mamitsuka. Discriminative graph embedding for label propagation. *Neural Networks, IEEE Transactions on*, 22(9):1395 –1405, 2011.

[47] R. Ohbuchi, T. Minamitani, and T. Takei. Shape-similarity search of 3d models by using enhanced shape functions. *International Journal of Computer Applications in Technology*, pages 70–85, 2005.

[48] R. OSADA, T. FUNKHOUSER, B. CHAZELLE, and D. DOBKIN. Shape distributions. *ACM Transactions on Graphics*, pages 807–832, 2002.

[49] M. Ovsjanikov, M. Ben-Chen, J. Solomon, A. Butscher, and L. Guibas. Functional maps: A flexible representation of maps between shapes. *ACM Trans. Graph.*, 31(4):30:1–30:11, 2012.

[50] E. Pekalska and R. Duin. *The Dissimilarity Representations for Pattern Recognition.* 2005.

[51] E. Pekalska, R. Duin, and P. Paclik. Prototype selection for dissimilarity-based classifiers. *Pattern Recognition*, pages 189–208, 2006.

[52] S. Pinkall, D. Juni, and K. Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics*, pages 15–36, 1993.

[53] J. Qiangrong, L. Hualan, and G. Yuan. Cycle kernel based on spanning tree. *In proc. of International Conference on Electrical and Control Engineering*, pages 656–659, 2010.

[54] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical Association*, 66(336):846–850, 1971.

[55] D. Raviv, M. M. Bronstein, A. M. Bronstein, and R. Kimmel. Volumetric heat kernel signatures. In *Proceedings of the ACM workshop on 3D object retrieval*, 2010.

[56] P. Ren, T. Aleksic, D. Emms, R. C. Wilson, and E. R. Hancock. Quantum walks, ihara zeta functions and cospectrality in regular graphs. *Quantum Information Processing*.

[57] P. Ren, T. Aleksic, R. C. Wilson, and E. R. Hancock. A polynomial characterization of hypergraphs using the Ihara zeta function. *Pattern Recognition*, 44:1941–1957, 2011.

[58] P. Ren, R. C. Wilson, and E. R. Hancock. Graph characteristics from the ihara zeta function. *Structural, Syntactic, and Statistical Pattern Recognition, Joint IAPR International Workshop, Springer*, pages 257–266, 2008.

[59] P. Ren, R. C. Wilson, and E. R. Hancock. Characteristic polynomial analysis on matrix representations of graphs. *Graph Based Representation in Pattern Recognition*, pages 243–252, 2009.

[60] P. Ren, R. C. Wilson, and E. R. Hancock. Hypergraphs, characteristic polynomials and the ihara zeta function. *Computer Analysis of Images and Patterns, 13th International Conference, CAIP*, pages 369–376, 2009.

[61] P. Ren, R. C. Wilson, and E. R. Hancock. Weighted graph characteristics from oriented line graph polynomials. *IEEE 12th International Conference on Computer Vision*, pages 2280–2287, 2009.

[62] P. Ren, R. C. Wilson, and E. R. Hancock. Graph characterization via Ihara coefficients. *IEEE Tran. on Neural Networks*, 22:233–245, 2011.

[63] M. Reuter, S. Biasotti, D. Giorgi, G. Patanè, and M. Spagnuolo. Discrete laplace-beltrami operators for shape analysis and segmentation. *Computers & Graphics*, 33(3):381–390, 2009.

[64] K. Riesen and H. Bunke. Approximate graph edit distance computation by means of bipartite graph matching. *Image Vision Comput.*, pages 950–959, 2009.

[65] K. Riesen, M. Neuhaus, and H. Bunke. Graph embedding in vector spaces by means of prototype selection. In *Graph-Based Representations in Pattern Recognition*, pages 383–393, 2007.

[66] J. Rubinstein and M. Schatzman. Variational problems on multiply connected thin strips.i. basic estimates and convergence of the laplacian spectrum. *Arch. Ration. Mech. Anal.*, page 271308, 2001.

[67] R.M. Rustamov. Laplace-beltrami eigenfunctions for deformation invariant shape representation. *Eurographics Symp. on Geom. Processing*, pages 225–233, 2007.

[68] S. V. Savchenko. The zeta function and Gibbs measures. *Russian Math. Surveys*, 48(1):189–190, 1993.

[69] B. Schölkopf and A. J. Smola. Learning with kernels. *Cambridge, MA: MIT Press.*

[70] G. Scott and C. K. Storm. The coefficients of the ihara zeta function. *Involve: A J. of Math.*, 1(2):217–233, 2008.

[71] J. A. Sethian. Fast marching methods. *SIAM Review*, 41:199–235, 1998.

[72] A. Setyadi and C. K. Storm. Enumeration of graphs with the same ihara zeta function. *Linear Algebra and its Applications*, 438(1):564 – 572, 2013.

[73] B. Spillmann, M. Neuhaus, H. Bunke, E. Pekalska, and R. Duin. Transforming strings to vector spaces using prototype selection. *Structural, Syntactic, and Statistical Pattern Recognition*, pages 287–296, 2006.

[74] A. Srinivasan, S. H. Muggleton, M. J. E. Sternberg, and R. D. King. Theories for mutagenicity: a study in first-order and feature-based induction. *Artificial Intelligence*, pages 277–299, 1996.

[75] H. M. Stark and A. A. Terras. Zeta functions of finite graphs and coverings. *Adv. in Math.*, 121:124–165, 1996.

[76] P. Suau, E. R. Hancock, and F. Escolano. Analysis of the schrödinger operator in the context of graph characterization. *Similarity based Pattern Recognition*, pages 190–203, 2013.

[77] J. Sun, M. Ovsjanikov, and L. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. *Comp. Grarph Forum*, pages 1383–1392, 2010.

[78] D. Tschumperlé. Fast anisotropic smoothing of multi-valued images using curvature-preserving pde's. *International Journal of Computer Vision*, pages 65–82, 2006.

[79] S. V. N. Vishwanathan, N. N. Schraudolph, R. Kondor, and K. M. Borgwardt. Graph kernels. *Journal of Machine Learning Research*, page 2010, 1201-1242.

[80] Y. Watanabe and K. Fukumizu. Graph zeta function in the Bethe free energy and loopy belief propagation. In *Neural Information Processing Systems*, pages 2017–2025, 2009.

[81] J. Weickert. *Anisotropic Diffusion in Image Processing.* 1998.

[82] R. C. Wilson, F. Aziz, and E. R. Hancock. Eigenfunctions of the edge-based laplacian on a graph. *Linear Algebra and its Applications*, 438(11):4183–4189, 2013.

[83] R. C. Wilson, E. R. Hancock, and B. Luo. Pattern vectors from algebraic graph theory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1112–1124, 2005.

[84] C. Wojciech and M. Ehler. Schrödinger eigenmaps for the analysis of bio-medical data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1274–1280, 2013.

[85] I. Xenarios, L. Salwinski, X. Duan, .P Higney, S. Kim, and D. Eisenberg. Dip, the database of interacting proteins: a research tool for studying cellualr networks of protein interactions. *NAR*, pages 303–305, 2002.

[86] B. Xiao, E. R. Hancock, and R. C. Wilson. Graph characteristics from the heat kernel trace. *Pattern Recognition*, pages 2589–2606, 2009.

[87] B. Xiao, E. R. Hancock, and R. C. Wilson. Geometric characterization and clustering of graphs using heat kernel embeddings. *Image Vision Comput.*, 28(6):1003–1021, 2010.

[88] G. Xu. Discrete laplace-beltrami operator on sphere and optimal spherical triangulations. *International Journal of Computational Geometry*, pages 75–93, 2006.

[89] X. Yao, D. Wei, C. Soden, M. Summers, and D. Beckett. Structure of the carboxy-terminal fragment of the apo-biotin carboxyl carrier subunit of escherichia coli acetyl-coa carboxylase. *Biochemistry*, 36:15089–15100, 1997.

[90] F. Zhang and E. R. Hancock. Graph spectral image smoothing using the heat kernel. *Pattern Recognition*, pages 3328–3342, 2008.

[91] D. Zhao and X. Tang. Cyclizing clusters via zeta function of a graph. In *Neural Information Processing Systems*, pages 1953–1960, 2008.