Spectral Generative Models for Graphs

David White and Richard C. Wilson
Department of Computer Science
University of York
Heslington, York, UK
wilson@cs.york.ac.uk

Abstract

Generative models are well known in the domain of statistical pattern recognition. Typically, they describe the probability distribution of patterns in a vector space. The individual patterns are defined by vectors and so the individual features of the pattern are well defined. In contrast, very little work has been done with generative models of graphs because graphs do not have a straightforward vectorial representation. Because of this, simple statistical quantities such as mean and variance are difficult to define for a group of graphs. While we can define statistical quantities of individual edges, it is not so straightforward to define how sets of edges in graphs are related.

In this paper we examine the problem of creating generative distributions over sets of graphs. We use the spectral representation of the graphs to construct a dual vector space for the graphs. The spectral decomposition of a graph can be used to extract information about the relationship of edges and parts in a graph. Distributions are then defined on the vector spaces and used to generate new samples. Finally, these points must be used to reconstruct the sampled graph.

1 Introduction

Generative models are well known in the domain of statistical pattern recognition. Typically, they describe the probability distribution of patterns in a vector space. The individual patterns are defined by vectors and so the individual features of the pattern are well defined. Graphs are not naturally represented in a vector space since there is no natural labelling of the vertices of the graphs - different labellings lead to different representations of the graph structure.

The key problem in utilising graph representations lies in measuring their structural similarity. This is a difficult problem because there is no explicit labelling of the parts, and typically correspondences must be established before similarity can be assessed. As an example, Sanfeliu and Fu[10] employed the concept of graph edit distance, giving separate edit costs for relabeling, insertion and deletion on both nodes and edges. A search is necessary to locate the set of operations which have minimal cost. More recently, Bunke[1, 2] has established a relationship between the minimum graph edit distance and the size of the maximum common subgraph. The graph edit distance provides a well defined way of measuring the similarity of two graphs.

Spectral graph theory provides another approach to the problem of graph similarity[3]. Eigenvector methods have been used for grouping via pairwise clustering. Examples include Shi and Malik's [12] iterative normalised cut method which uses the Fiedler (i.e. second) eigenvector for image segmentation and Sarkar and Boyer's use of the leading eigenvector of the weighted adjacency matrix [11]. The spectrum therefore reveals information about the relationships between parts of the graph. Graph spectral methods have also been used for correspondence analysis. For example, Umeyamas method[14] allows the matching of two graphs of equal size by using the eigendecompositions of the adjacency matrices. Ferrer et al[4] have used the spectral decomposition to match graphs in a intermediate step to finding the graph median.

The problem of representing a set of graphs via a representative has been addressed by Jiang et al[5, 6], who define the median of a set of graphs to be the graph which has the minimum sum-of-distances to all the graphs in the set. This definition is equivalent in a sense to the mean of a set of points. The computation of such median graphs is computationally expensive and Jiang et al employ a genetic algorithm to locate median graphs. Ferrer et al[4] use a spectral method to compute correspondences between the eigendecompositions of graphs, and then update the median graph using these correspondences. The updates are performed sequentially.

While the construction of generative probabilistic models for vectorial data is well know, the problem of defining

distributions of graphs and generating new examples has recieved little attention. One approach to the problem is to find a vectorial representation of the graph and then define a distribution on the new representation. Of course, this also creates the additional problem of reconstructing a new graph from the vector space. There are a number of methods in the literature for representing graphs in a vector space. Kosinov and Caelli[7] have used properties of the spectral decomposition to represent graphs and Shokoufandeh et al[13] has used eigenvalues of shock graphs to index shapes. Wilson and Hancock have shown[16] how permutation invariant polynomials can be used to derive features which describe graphs and make full use of the available spectral information. However, it is difficult to reconstruct graphs from these representations

Luo, Wilson and Hancock[9] directly exploit the adjacent matrix by converting it into a long-vector. An initial correspondence step is used to align the adjacency matrix and the long-vectors are analysed using the eigenmodes. Xiao and Hancock[17] have used the eigenvalues and eigenvectors of the heat kernel to construct the requisite vectors before constructing a Normal distribution in the vector space. While both of these methods construct generative models over the vector space, neither is able to produce reconstructed graphs satisfactorily, for reasons which will become apparent later in this paper.

The outline of the paper is as follows; In section 2 we describe various matrix representations of graphs, which are key to the reconstruction process and the derived spectral representation. In section 3, we construct a generative model. In section 4 we explain how the graphs are reconstructed. Finally, we provide some experimental analysis.

2 Matrix representations of graphs

The graphs under consideration here are undirected graphs. The methods employed here also apply to weighted graphs, but the case of a weighted graph we need not ultimately concern ourselves with recovering discrete edges. We denote a graph by $G = (V, E)$ where V is the set of nodes and $E \subseteq V \times V$ is the set of edges. The degree of a vertex u is the number of edges incident on the vertex u and is denoted d_u. A *matrix representation* of the graph is a $|V|$ by $|V|$ matrix $\mathbf{X}$, such that an element X_{ij} of this matrix represents some property of the pair of vertices i and j. Diagonal elements X_{ii} encode information about the vertex i only. A simple example is the adjacency matrix A, where A_{ij} is 1 when there is an edge between i and j, and zero otherwise. There are a number of alternative representations which we discuss below. Since the order of vertices in the graph does not matter, if we permute the indices i associated with the graph vertices then the graph remains the same. If $\mathbf{P}$ is the permutation matrix which re-orders the vertices, then

$$\mathbf{X}' = \mathbf{P}\mathbf{X}\mathbf{P}^T$$

represents the same graph as $\mathbf{X}$.

2.1 Standard Graph Representations

The most basic matrix representation of a graph is using the *adjacency matrix* A for the graph. This matrix is given by

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

Clearly if the graph is undirected, the matrix A is symmetric. As a consequence, the eigenvalues of A are real. These eigenvalues may be positive, negative or zero and the sum of the eigenvalues is zero. The eigenvalues may be ordered by their magnitude and collected into a vector which describes the graph spectrum.

In some applications, it is useful to have a positive semidefinite matrix representation of the graph. This may be achieved by using the *Laplacian*. We first construct the diagonal degree matrix $\mathbf{D}$, whose diagonal elements are given by the node degrees $D(u, u) = d_u$. From the degree matrix and the adjacency matrix we then can construct the standard combinatorial Laplacian matrix

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \quad (2)$$

i.e. the degree matrix minus the adjacency matrix. The Laplacian has at least one zero eigenvalue, and the number of such eigenvalues is equal to the number of disjoint parts in the graph. The *signless Laplacian* is a modification of the Laplacian which contains only positive entries[15]:

$$|\mathbf{L}| = \mathbf{D} + \mathbf{A} \quad (3)$$

The normalised Laplacian is a scaled version of the Laplacian

$$\mathcal{L} = \mathbf{I} - \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}} \quad (4)$$

which has eigenvalues $0 \leq \lambda \leq 2$. Finally, the commute time matrix $\mathbf{C}$ has also been used as a graph representation. The commute time between two vertices u and v is the expected time for a random walk on the graph to travel from u to v and back, which we denote by C_{uv}. The commute time is related to the spectral decomposition of the normalised Laplacian [8].

2.2 Spectral Representations

The *spectral representation* of the graph is obtained from the matrix representation using the eigendecomposition. Let $\mathbf{X}$ be the matrix representation in question. Then the eigendecomposition is $\mathbf{X} = \Phi\Lambda\Phi^T$ where $\Lambda =$

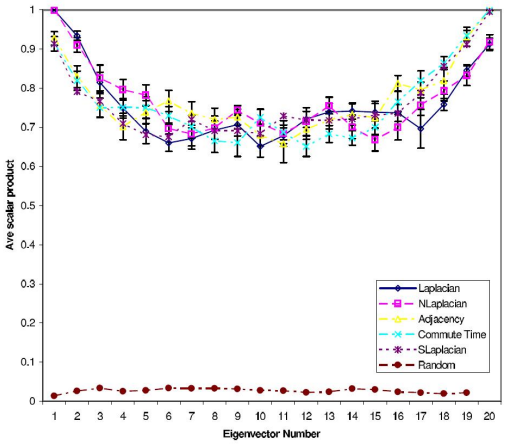
$\mathrm{diag}(\lambda_1, \lambda_2, ..., \lambda_{|V|})$ is the diagonal matrix with the ordered eigenvalues as elements and $\Phi = (\phi_1|\phi_2|....|\phi_{|V|})$ is the matrix with the ordered eigenvectors as columns. The spectrum is the set of eigenvalues

$$s = \{\lambda_1, \lambda_2, ..., \lambda_{|V|}\}$$

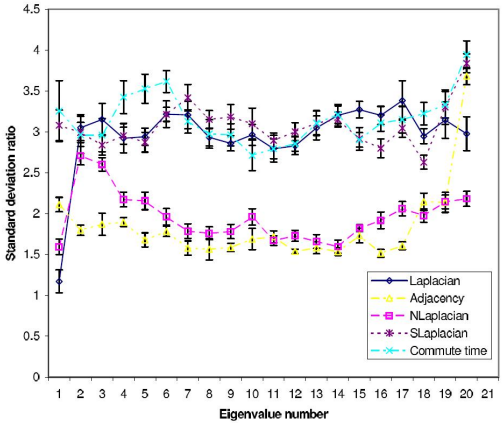
The spectral representation is the pair $\{\Phi, \Lambda\}$ which completely describe the graph, since the original adjacency matrix can be constructed from the spectral representation. If $\mathbf{X}$ is positive semi-definite (and so has positive or zero eigenvalues), then we can combine these matrices to define the spectral representation by a single matrix $\Phi\Lambda^{\frac{1}{2}}$.

The spectral representation is not however a unique description of the graph. If we reorder the vertices of G, then although the graph is unchanged, we obtain a different matrix and therefore a different spectral representation. The spectral representation $\{\mathbf{P}\Phi, \Lambda\}$ therefore represents the same graph for any permutation matrix $\mathbf{P}$. In addition if there are repeated eigenvalues in the spectrum, then the eigendecomposition is not uniquely defined.

Given that there are many matrix representations of a graph, the question remains which we should use. There seems no apriori reason why one particular representation should be favourable, but in order to statistically model graphs we would like similar graphs to have similar spectral representations. Therefore, in order to answer the question of which representation to use, we have compared the similarity of the spectral representations of related and unrelated graphs. The unrelated graphs have 20 vertices and are created by randomly selecting edges with probability 1/2. The related graphs are generated from a random seed graph by corrupting edges with probability 0.015, leading to an average dissimilarity of 2.85 edges. Figure 1a measures the similarity of the eigenvectors of related graphs by the average scalar product between the eigenvectors. The eigenvalues of the random and related graphs have the same mean, so we measure the similarity by the ratio of standard deviation for the related and unrelated classes of graph (Figure 1b). The results show that the representation makes no difference to the eigenvectors, but the eigenvalues are clearly more consistent for the Laplacian, commute time and the signless Laplacian. Therefore we choose to use the Laplacian representation.



a) Similarity of Eigenvectors



b) Similarity of Eigenvalues

Figure 1. The similarity of spectral decompositions of various matrix representations

3 A Generative Model

The spectral representation is an interesting one in terms of mixing graphs for a number of reasons. Firstly, part of the correspondence problem is solved in the spectral representation; the columns of Φ are ordered by the eigenvector magnitude which is not affected by the vertex labelling. However, the rows are still permuted when we change the graph indexing. Secondly, in the Laplacian and related matrices, structures of different scales in the graph are associated with eigenvalues of different magnitudes. It is well known that the Fiedler vector of the normalised Laplacian can be used to partition the graph into parts, whereas the principle eigenvector represents global structure in the graph. As a result, it is possible to mix different scales separately using the spectral representation.

Before proceeding, we must first align the rows of Φ so that they are in the same order. This is necessary so that the corresponding elements of our vector space are the same over all graphs in the sample set. This alignment step is not the focus of this paper, so we assume that our graph set has been pre-aligned using one of the many alignment methods in the literature. In particular we propose the use of a spectral graph matching method such as Umeyama's method[14] or a variant such as that of Ferrer et al[4] since

we already have the spectral decomposition to hand.

It is well known that the eigenvectors of the decomposition of a matrix are sign-ambiguous. In other words, the eigenvectors are recovered up to a sign factor of ± 1. It is necessary to determine these factors if we are to correctly mix the corresponding eigenmodes. Our method is based on identifying the largest component of an eigenvector and correcting the sign based on that coordinate. Given a set of spectral matrices $\{\Phi_1, \Phi_2, ..., \Phi_m\}$, let ϕ_{ij} be the jth eigenvector (mode) from Φ_i. The kth component of this eigenvector can then be denoted ϕ_{ijk}. We find the largest magnitude component for mode j from

$$l_j = \arg\max_k \sum_i |\phi_{ijk}|$$

We then correct the sign of the eigenvectors by ensuring that component l_j is positive for mode j in all the spectral matrices.

Once aligned, we construct two vectors from the spectral decomposition. The first vector is simply the vector of (ordered) eigenvalues:

$$\mathbf{e}(G) = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \dots \\ \lambda_n \end{pmatrix} \tag{5}$$

The second vector is the long-vector of the eigenvector matrix:

$$\mathbf{z}(G) = \begin{pmatrix} \phi_{1,1} \\ \phi_{2,1} \\ \dots \\ \phi_{nn} \end{pmatrix} \tag{6}$$

In Xiao and Hancock[17] these components were combined by considering the long-vector of $\Phi\Lambda^{\frac{1}{2}}$ derived from the heat kernel. This long-vector contains all the information to reconstruct the graph in a single vector; the eigenvalues are encoded in the lengths of the vectors. This will not lead to a satisfactory vector space, which we can see by considering the mixing of two vectors from two slightly different graphs. If we average these vectors and they are not pointing in the same direction, the resultant length will be shorter than the lengths of the originals. Since the eigenvalue is encoded in the length, this results in smaller eigenvalues and unsatisfactory reconstructions. By employing two separate models we can correct the length of the eigenvectors.

Given the dual vector space representation of the graphs, we construct a distribution over the sample set of graphs. To do this we compute the PCA decomposition for both the eigenvalues and eigenvectors of the sample graphs. We commence by computing the mean and covariance for each:

$$\overline{\mathbf{e}} = \frac{1}{k}\sum_{i=1}^{k} \mathbf{e}(G_i) \tag{7}$$

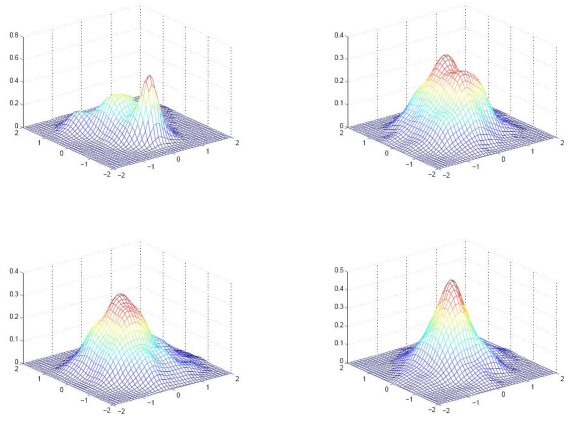


Figure 2. The kernel density estimations for the eigenvectors of the sample graphs. The graphs have (top left) 10 nodes, (top right) 20 nodes, (bottom left) 30 nodes and (bottom right) 40 nodes.

$$\overline{\mathbf{z}} = \frac{1}{k}\sum_{i=1}^{k} \mathbf{z}(G_i) \tag{8}$$

We then construct separate covariance matrices for the two sets $\mathbf{C}_e$ and $\mathbf{C}_z$

With the covariance matrices to hand, we perform the eigendecomposition on both $\mathbf{C}_e$ and $\mathbf{C}_z$. This results in four matrices $\mathbf{\Lambda}_e$ and $\mathbf{\Phi}_e$ (representing the eigenvalues and eigenvectors of $\mathbf{C}_e$) and $\mathbf{\Lambda}_z$ and $\mathbf{\Phi}_z$ (representing the eigenvalues and eigenvectors of $\mathbf{C}_z$).

We now have the information needed about the distribution of the eigenvalues and eigenvectors of the graphs in the sample set. For example, the matrix $\mathbf{\Lambda}_z$ describes the variance of each component in $\mathbf{\Phi}_z$ and the magnitude of the values in $\mathbf{\Lambda}_z$ shows the relative importance of the associated component.

For the generative step of our method to work it is necessary to show that the distribution is normal, clearly the eigenvalues are normally distributed but this is not as clear for the eigenvectors. To show that the eigenvectors are also normally distributed we take the zero-mean long eigenvector vector $\mathbf{z}'$ for each graph and multiply it with the with the two principle components in $\mathbf{\Phi}_z$ resulting in a point in 2d space. We can visualize this set of points using kernel density estimation and see if the surface formed is close to the normal distribution. For graphs below 10 nodes the surface is far from normal, however as the number of nodes in the sample graphs increases the surface becomes closer to the normal distribution (see Figure 2).

4 Generation and Reconstruction

With the distribution of the sample graphs defined, we can sample from it to generate new graphs. A vector for the eigenvalues and eigenvectors of the new graph is generated by sampling from the normal distributions defined on the original graph sets. Parameter vectors $\mathbf{b}_e$ and $\mathbf{b}_z$ are generated (for the eigenvalues and eigenvectors respectively).

$$\hat{\mathbf{e}} = \overline{\mathbf{e}} + \mathbf{\Phi}_e \mathbf{b}_e \tag{9}$$

$$\hat{\mathbf{z}} = \overline{\mathbf{z}} + \mathbf{\Phi}_z \mathbf{b}_z \tag{10}$$

Each element of the parameter vector is computed by sampling from the normal distribution with zero mean and variance determined by the diagonal values in $\mathbf{\Lambda}_e$ for the eigenvalues and $\mathbf{\Lambda}_z$ for the eigenvectors. If we define a random number generator for the normal distribution as $\mathcal{N}(Mean, Variance)$, then the parameter vectors are computed as such:

$$\mathbf{b}_e(i) = \mathcal{N}(0, \mathbf{\Lambda}_e(i,i)) \tag{11}$$

$$\mathbf{b}_z(i) = \mathcal{N}(0, \mathbf{\Lambda}_z(i,i)) \tag{12}$$

The new vectors must be converted back into matrix form:

$$\hat{\mathbf{\Lambda}} = \mathrm{diag}(\hat{\mathbf{e}}_1, \hat{\mathbf{e}}_2, \cdots, \hat{\mathbf{e}}_n) \tag{13}$$

$$\hat{\mathbf{\Phi}} = \begin{pmatrix} \hat{\mathbf{z}}_1 & \hat{\mathbf{z}}_{n+1} & \cdots & \hat{\mathbf{z}}_{n^2-n+1} \\ \vdots & \vdots & & \vdots \\ \hat{\mathbf{z}}_n & \hat{\mathbf{z}}_{2n} & \cdots & \hat{\mathbf{z}}_{n^2} \end{pmatrix} \tag{14}$$

In general the new set of eigenvectors that have been generated will not be orthogonal and therefore do not represent an eigendecomposition. We correct them by projecting back onto an orthogonal matrix:

$$\hat{\mathbf{\Phi}}_O = (\hat{\mathbf{\Phi}}\hat{\mathbf{\Phi}}^T)^{-\frac{1}{2}}\hat{\mathbf{\Phi}} \tag{15}$$

The eigenvectors and eigenvalues are combined to produce the Laplacian matrix for the new graph:

$$\hat{\mathbf{L}} = \hat{\mathbf{\Phi}}_O \hat{\mathbf{\Lambda}} \hat{\mathbf{\Phi}}_O^T \tag{16}$$

The generated matrix will be close to a "correct" Laplacian matrix of a graph due to the separate eigenvalue/eigenvector distributions and the othogonization of the generated eigenvectors. However, it will not be an exact Laplacian due to the continuous nature of the normal distribution in contrast to the discrete graph representation. We must therefore perform a final recovery step to obtain an accurate matrix representation of the new graph. We choose to do this through thresholding. An exact value cannot be set for all cases since it depends on the average connectivity of the sample graphs. However, this is quite simple to compute by observing the average value of the off-diagonal elements in the normalized Laplacian matrices of the sample graphs. The values in the off-diagonal of the Laplacian are negative and hence the elements in $\hat{\mathbf{L}}$ are only recorded as an edge if they are *lower* than the threshold θ. We recover the Laplacian back to an adjacency matrix $\hat{\mathbf{A}}$:

$$\hat{\mathbf{A}}(u,v) = \begin{cases} 0 & \text{if } u = v \\ 1 & \text{if } \hat{\mathbf{L}}(u,v) < \theta \\ 0 & \text{otherwise} \end{cases} \tag{17}$$

5 Experimental results

In this section we detail the results of testing our method on synthetic data. We commence by generating a random *reference graph*. We then perturb this graph by performing a random number of edit operations on the edges (either inserting a new edge or deleting an edge present in the reference graph). This is repeated to produce the sample set of graphs. Each potential edge in the graph is inserted or deleted with a probability of 0.05%.

With the sample set to hand we create the generative model. Newly generated adjacency matrices are checked to see if they appear in the set of sample graphs, that is to see if we are generating new graphs or just reproducing graphs in the sample set. If we generate 100 new graphs from a sample set of 1000 graphs of 10 nodes each then about 95% of the new graphs do not appear in the sample set.

To confirm that our generated graphs fall correctly within the distribution we perform PCA on the generated graphs and those in the sample set. The results of this are in figure 3, which shows the generated graphs conforming to the distribution of the sample graphs.

Figure 4 shows the generated adjacency matrices in graphical form. Each 1 in the adjacency matrix is depicted as a filled box and each 0 as a white box. The reference graph is shown at the top and the generated graphs are shown below. The generated graphs in this figure are only ones that do not appear in the sample set. They are ordered from left to right by the number of edit operations they differ from the reference graph (the top left graph has the least edit operations, the bottom right graph the most edit operations).

6 Conclusions

In this paper we have shown how to construct a probabilistic model of the distribution of a set of graphs which can be used to generate new examples from the set. This is achieved by constructing two vector spaces to represent the graph. This separation of eigenvalue and eigenvector model is vital because of the vastly different properties of these elements. Vectors generated from the distribution can then

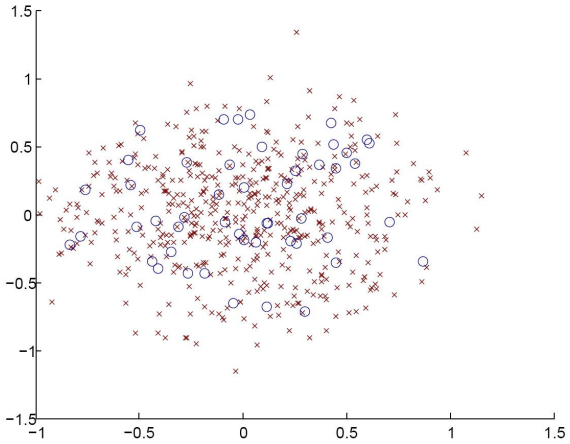


Figure 3. The PCA 2D plot showing the distribution of the sample graphs (marked with crosses) and the generated graphs (marked with circles).

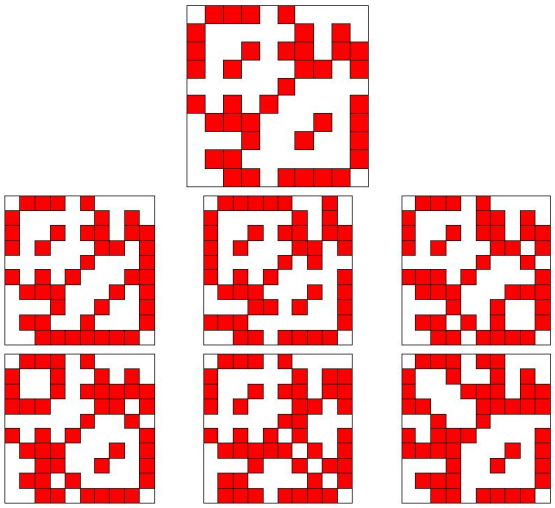


Figure 4. The reference graph (top) and 6 generated graphs that were not in the sample set.

be reconstructed into graphs after appropriate conditioning. We show that the generated graphs do in fact form part of the original distribution, and lie within the original cluster or graphs. Furthermore the generated graphs are novel and nearly always different from graphs in the sample set, but are still close in terms of edit distance.

References

[1] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18:689–694, 1997.

[2] H. Bunke. Error correcting graph matching: On the influence of the underlying cost function. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21:917–922, 1999.

[3] F. R. K. Chung. *Spectral Graph Theory*. AMS, 1997.

[4] M. Ferrer, F. Serratosa, and A. Sanfeliu. Synthesis of median spectral graph. In *IbPRIA*, pages 139–146, 2005.

[5] X. Jiang, A. Münger, and H. Bunke. Computing the generalized median of a set of graphs. In *Proc. 2nd IAPR workshop on Graph Based representations*, pages 115–124, 1999.

[6] X. Jiang, A. Münger, and H. Bunke. On median graphs: Properties, algorithms and applications. *IEEE Transations on Pattern Analysis and Machine Intelligence*, 23(10):1144–1151, 2001.

[7] S. Kosinov and T. Caelli. Inexact multisubgraph matching using graph eigenspace and clustering models. *Structural, Syntatic and Statistical Pattern Recognition, LNCS*, 2396:133–142, 2002.

[8] L. Lovász. Random walks on graphs: A survey. In *Combinatorics, Paul Erdõs is Eighty, Vol. 2*. János Bolyai Mathematical Society, Budapest, 1996.

[9] B. Luo, R. C. Wilson, and E. R. Hancock. A linear generative model for graph structure. In *Graph-based Representations in Pattern Recognition*, 2005.

[10] A. Sanfeliu and K. S. Fu. A distance measure between attributed relational graphs for pattern-recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 13(3):353–362, 1983.

[11] S. Sarkar and K. L. Boyer. Preceptual organization in computer vision. *IEEE Trans. Systems, Man and Cybernetics*, 23:382–399, 1993.

[12] J. Shi and J. Malik. Normalized cuts and image segmentation. *CVPR*, pages 731–737, 1997.

[13] A. Shokoufandeh, S. Dickinson, K. Siddiqi, and S. Zucker. Indexing using a spectral coding of topological structure. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition*, number 491–497, 1999.

[14] S. Umeyama. An eigendecompostion approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):695–703, 1988.

[15] E. R. van Dam and W. H. Haemers. Which graphs are determined by their spectrum? *Linear Algebra and its Applications*, 356:241–272, 2003.

[16] R. C. Wilson, E. R. Hancock, and B. Luo. Pattern vectors from algebraic graph theory. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(7):1112–1125, 2005.

[17] B. Xiao and E. R. Hancock. A spectral generative model for graph structure. In *International Workshop on Structural and Syntactic Pattern Recognition*, 2006.