International Journal of Pattern Recognition and Artificial Intelligence

String Edit Distance, Random Walks and Graph Matching

Antonio Robles-Kelly *and Edwin R. Hancock

Department of Computer Science
University of York
York, Y01 5DD, UK
{*arobkell,erh*}*@cs.york.ac.uk*

This paper shows how the eigenstructure of the adjacency matrix can be used for the purposes of robust graph-matching. We commence from the observation that the leading eigenvector of a transition probability matrix is the steady state of the associated Markov chain. When the transition matrix is the normalised adjacency matrix of a graph, then the leading eigenvector gives the sequence of nodes of the steady state random walk on the graph. We use this property to convert the nodes in a graph into a string where the node-order is given by the sequence of nodes visited in the random walk. We match graphs represented in this way, by finding the sequence of string edit operations which minimise edit distance.

1. Introduction

Graph-matching is a task of pivotal importance in high-level vision since it provides a means by which abstract pictorial descriptions can be matched to one-another. Unfortunately, since the process of eliciting graph structures from raw image data is a task of some fragility due to noise and the limited effectiveness of the available segmentation algorithms, graph-matching is invariably approached by inexact means [18,16]. The search for a robust means of inexact graph-matching has been the focus of sustained activity over the last two decades. Early work drew heavily on ideas from structural pattern recognition and revolved around extending the concept of string edit distance to graphs [16,3]. More recent progress has centred around the use of powerful optimisation and probabilistic methods, with the aim of rendering the graph matching process robust to structural error.

Despite proving effective, these methods lack the elegance of the matrix representation first used by Ullman in his work on subgraph isomorphism [21]. The task of posing the inexact graph matching problem in a matrix setting has proved to be an elusive one. This is disappointing since a rich set of potential tools are available from the field of mathematics referred to as spectral graph theory. This is the term given to a family of techniques that aim to characterise the global structural properties of graphs using the eigenvalues and eigenvectors of the adjacency ma-

*Corresponding author

trix [2]. In the computer vision literature there have been a number of attempts to use spectral properties for graph-matching, object recognition and image segmentation. Umeyama has an eigendecomposition method that matches graphs of the same size [22]. Borrowing ideas from structural chemistry, Scott and Longuet-Higgins were among the first to use spectral methods for correspondence analysis [17]. They showed how to recover correspondences via singular value decomposition on the point association matrix between different images. In keeping more closely with the spirit of spectral graph theory, yet seemingly unaware of the related literature, Shapiro and Brady [19] developed an extension of the Scott and Longuet-Higgins method, in which point sets are matched by comparing the eigenvectors of the point proximity matrix. Here the proximity matrix is constructed by computing the Gaussian weighted distance between points. The eigenvectors of the proximity matrices can be viewed as the basis vectors of an orthogonal transformation on the original point identities. In other words, the components of the eigenvectors represent mixing angles for the transformed points. Matching between different point-sets is effected by comparing the pattern of eigenvectors in different images. Kosinov and Caelli [7] have recently reported a graph-matching method which is closely related to that of Shapiro and Brady. Their method involves projecting the graph onto the leading eigenvectors of the adjacency matrix, and matching can be realised as a rotation of the projections. Shapiro and Brady's method can be viewed as operating in the attribute domain rather than the structural domain. Horaud and Sossa [6] have adopted a purely structural approach to the recognition of line-drawings. Their representation is based on the immanental polynomials for the Laplacian matrix of the line-connectivity graph. By comparing the coefficients of the polynomials, they are able to index into a large data-base of line-drawings. Shokoufandeh, Dickinson and Siddiqi [20] have shown how graphs can be encoded using local topological spectra for shape recognition from large data-bases.

In a recent paper Luo and Hancock [11] have returned to the method of Umeyama and have shown how it can be rendered robust to differences in graph-size and structural errors. Commencing from a Bernoulli distribution for the correspondence errors, they develop an expectation-maximisation algorithm for graph-matching. Correspondences are recovered in the M or maximisation step of the algorithm by performing singular value decomposition on the weighted product of the adjacency matrices for the graphs being matched. The correspondence weight matrix is updated in the E or expectation step. However, since it is iterative the method is relatively slow and is potentially sensitive to initialisation.

The aim in this paper is to investigate whether the eigenstructure of the adjacency matrix can be used to match graphs using a search method rather than by iteration. To do this we draw on the theory of Markov chains. We consider a Markov chain whose transition probability matrix is the normalised edge-weight matrix for a graph. The steady-state random walk for the Markov chain on the graph is given by the leading eigenvector of the transition probability, i.e. edge weight, matrix.

Hence, by considering the order of the nodes defined by the leading eigenvector, we are able to convert the graph into a string. This opens up the possibility of performing graph matching by using string alignment to minimise the Levenshtein or edit distance [8,24]. We can follow Wagner [24] and use dynamic programming to evaluate the edit distance between strings and hence recover correspondences [24]. It is worth stressing that although there been attempts to extend the string edit idea to trees and graphs [25,13,16,18], there is considerable current effort aimed at putting the underlying methodology on a rigourous footing. For instance, Bunke has demonstrated the relationship between graph edit distance and the size of the maximum common subgraph [1]. Taking a Bayesian perspective, Myers, Wilson and Hancock [12] have shown how to construct a probability distribution for local graph-edit distance and have used this distribution for maximum likelihood inexact graph-matching.

The outline of the paper is as follows. In Section 2 we describe the relationship between the steady state random walk on a graph and the leading eigenvector of the edge transition weight matrix. Section 3 explains how the serial ordering of the nodes in the walk may be used to convert the nodes of the graphs to a string order and how the strings may be matched so as to minimise string edit distance. Section 4 presents experiments on real-world and synthetic data. Finally, Section 5 offers some conclusions and identifies directions for future work.

2. Random Walks on Graphs

Our aim in this paper is to find a serial ordering of the nodes in a graph that can be used to convert graphs to strings, so that graph-matching can be effected using standard string matching techniques. To define the string order, we make use of the steady-state random walk on the graph. The transition probability matrix for the random walk is found by normalising the adjacency matrix. In the steady state the probability of visiting nodes is related to the leading or left eigenvector of the adjacency matrix. In its raw form the sequence defined by the serial order of the steady-state site probabilities is not suitable as a string representation of the graph. The reason for this is that the steady state random walk is not an edge connected path. Hence, we can not exploit edge constraints when matching the strings. Without these constraints the sites in the strings have no syntactic structure and the matching process must rely on attributes alone. To overcome this problem, we use a constraint filtering technique to search for an edge connected path using the steady state site probabilities. In passing, it is important to note that there are several alternative methods that can be used to characterise the steady state random walk using the eigenvectors of the adjacency matrix. Since we are interested in the steady state site probabilities, here we symmetrise the normalised transition probability matrix and make use of the eigenvalue-eigenvector expansion. However, there are more direct treatments which involve performing singular value decomposition on the asymmetric transition probability matrix, and using the matrix of left eigenvectors.

To commence, consider the graph $G = (V, E)$ with node index-set V and edge-set $E = \{(i, j)|(i, j) \in V \times V, i \neq j\}$. Associated with the graph is an adjacency matrix A whose elements are defined as follows

$$A(i,j) = \begin{cases} 1 & \text{if } (i,j) \in E \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Our aim is to assign the nodes of the graph to a sequence order which follows the steady state random walk on the graph. To pursue this analysis, we compute a transition probability matrix P from the adjacency matrix. The elements of this matrix are given by

$$P(i,j) = \frac{A(i,j)}{\sum_{j \in V} A(i,j)} \tag{2}$$

To pursue our analysis we intend to exploit the relationship between the leading eigenvector of the transition probability matrix and the steady state random walk on the graph. However, the matrix P is not symmetric and before we can commence our spectral analysis we must therefore convert it into a symmetric form so that we can perform an eigenvector expansion. To do this we first compute the diagonal degree matrix D, whose elements are

$$D(i,j) = \begin{cases} \frac{1}{d(i)} = \frac{1}{\sum_{j=1}^{|V|} P(i,j)} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Hence, in matrix form $P = DA$. The symmetric version of the matrix P is

$$W = D^{-\frac{1}{2}} P D^{\frac{1}{2}} = D^{\frac{1}{2}} A D^{\frac{1}{2}}$$

By normalising the matrix P in this way, we arrive at a normalised transition matrix W that depends on the degree of the nodes in the graph. As we demonstrate later, the probability of visiting nodes in the steady state random walk is controlled by the degree of the nodes in the graph. The spectral expansion for the symmetric transition matrix W is

$$W = \sum_{i=1}^{|V|} \lambda_i \vec{\phi}_i \vec{\phi}_i^T$$

where λ_i are the eigenvalues of W and $\vec{\phi}_i$ are the corresponding eigenvectors of unit length.

We are interested in random walks on the graph G. Let the sequence of nodes visited by the random walk be $X =< j_1, ..., j_{|V} >$. Suppose that the transition probability associated with the single step move between the nodes indexed j_l and j_m is $P(j_l, j_m)$. If the random walk can be represented by a Markov chain with transition matrix P, then the probability of visiting the nodes in the sequence is

$$P_X = P(j_1) \prod_{l=1}^{|V|} P(j_{l+1}, j_l)$$

Further, let $Q_t(i)$ be the probability of visiting the node indexed i after t-steps of the random walk and let $Q_t = (Q_t(1), Q_t(2), ...)^T$ be the state vector of site visitation probabilities at time t. After t time steps $Q_t = P^t Q_0$. As a result, after t applications of the Markov transition probability matrix

$$P^t = D^{\frac{1}{2}} W^t D^{-\frac{1}{2}}$$

Substituting the spectral expansion of the matrix W, we find

$$P^t = \sum_{i=1}^{|V|} \lambda_i^t D^{\frac{1}{2}} \vec{\phi}_i \vec{\phi}_i^T D^{-\frac{1}{2}}$$

Since the elements in individual rows and columns of the matrix W sum to unity, then the leading eigenvalue is unity, i.e. $\lambda_1 = 1$. Furthermore, from spectral graph theory [2] provided that the graph G is not a bipartite graph, then the smallest eigenvalue $\lambda_{|V|} > -1$. As a result, when the Markov chain approaches its steady state, i.e. $t \rightarrow \infty$, then all but the first term in the above series become negligible. Hence,

$$\lim_{t \rightarrow \infty} P^t = D^{\frac{1}{2}} \vec{\phi}_1 \vec{\phi}_1^T D^{-\frac{1}{2}}$$

This establishes that the leading eigenvector of the transition probability matrix determines the steady state of the Markov chain. It is also important to note that the equilibrium equation for the Markov process is $Q_s = PQ_s$, where Q_s is the vector of steady-state site visitation probabilities. Hence, since the leading eigenvalue of P is unity, then it follows that Q_s is the leading eigenvector of P. For a more complete proof of this result see the book by Varga [23] or the review of Lovasz [9].

We aim to visit the nodes in the graph in the order of their steady-state state probabilities. Suppose that the initial state vector for the sites is uniform, i.e.

$$Q_0 = (\frac{1}{|V|}, ..., \frac{1}{|V|})^T$$

We can hence write

$$Q_s = \lim_{t \rightarrow \infty} P^t Q_0$$

As a result the steady-state probability of visiting the node i is

$$Q_s(i) = \frac{1}{|V|} \sum_{j=1}^{|V|} \sqrt{\frac{d(j)}{d(i)}} \phi_1(i) \phi_1(j) = \frac{1}{|V|} \frac{\phi_1(i)}{\sqrt{d(i)}} \sum_{j=1}^{|V|} \sqrt{d(j) \phi_1(j)}$$

Since the summation appearing above is the same for all nodes, the probability rank order is determined by the quantity $\phi^*(i) = \frac{\phi_1(i)}{\sqrt{d(i)}}$. Hence, it is the scaled leading eigenvector

$$\vec{\phi}^* = D^{\frac{1}{2}} \phi_1 = (\frac{\phi_1(1)}{\sqrt{d(1)}}, ..., \frac{\phi_1(|V|)}{\sqrt{d(|V|}})^T$$

that determines the probability rank order of the sites in the steady state random walk.

Our aim is to use the sequence of nodes given by this rank order to define a serial ordering for the nodes in the graph. If we visit the nodes of the graph in the order defined by the magnitudes of the co-efficients of the leading eigenvector of the transition probability matrix, then the path is the steady state of the Markov chain. In this paper we aim to exploit this property to impose a string ordering on the nodes of a graph, and to use this string ordering property for matching the nodes in different graphs by minimising string edit distance. Unfortunately, the path followed by the steady state random walk is not edge-connected. Hence, we need a means of placing the nodes in a serial order in which edge constraints are preserved using the elements of the scaled leading eigenvector $\vec{\phi}^*$.

To do this we commence from the node associated with the largest component of ϕ^*, i.e. the largest site probability. We then sort the elements of the leading eigenvector such that they are both in the decreasing magnitude order of the co-efficients of the eigenvector, and satisfy edge connectivity constraints on the graph. The procedure is a recursive one that proceeds as follows. At each iteration, we maintain a list of nodes visited. At iteration k let the list of nodes be denoted by $\mathcal{L}_k$. Initially, $\mathcal{L}_0 = j_o$ where $j_0 = \arg\max_j \phi^*(j)$, i.e. j_0 is the component of ϕ^* with the largest magnitude. Next, we search through the set of first neighbours $\mathcal{N}_{j_0} = \{k|(j_0, k) \in E\}$ of j_o to find the node associated with the largest remaining component of ϕ^*. The second element in the list is $j_1 = \arg\max_{l \in \mathcal{N}_{j_0}} \phi^*(l)$. The node index j_1 is appended to the list of nodes visited and the result is $\mathcal{L}_1$. In the kth (general) step of the algorithm we are at the node indexed j_k and the list of nodes visited by the path so far is $\mathcal{L}_k$. We search through those first-neighbours of j_k that have not already been traversed by the path. The set of nodes is $C_k = \{l|l \in \mathcal{N}_{j_k} \wedge l \notin \mathcal{L}_k\}$. The next site to be appended to the path list is therefore $j_{k+1} = \arg\max_{l \in C_k} \phi^*(l)$. This process is repeated until no further moves can be made. This occurs when $C_k = \emptyset$ and we denote the index of the termination of the path by T. The serial ordering of the nodes of the graph X is given by the ordered list or string of nodes indices $\mathcal{L}_T$.

In practice we will be interested in finding the edit distance for a pair of graphs $G_M = (V_M, E_M)$ and $G_D = (V_D, E_D)$. The leading eigenvectors of the corresponding normalised transition matrices W_M and W_D are respectively ϕ_M^* and ϕ_D^*. The string representations of the graph G_M is denoted by X and that for the graph G_D by Y.

We augment, the information provided by the leading eigenvectors, with morphological information conveyed by the degree of the nodes in the two graphs. We establish the morphological affinity $\beta_{i,j}$ of nodes $i \in V_D$ and $j \in V_M$ using their degree ratio. Specifically, the morphological affinity of the nodes is taken to be

$$\beta_{i,j} = \exp\left(-\frac{\max(d(i), d(j)) - \min(d(i), d(j))}{\max(d(i), d(j))}\right) \tag{4}$$

If the degree ratio is one then the affinity measure is maximum. If the ratio is small (i.e. $\beta_{i,j} << 1$) then the affinity is zero. Of course, if we were working with directed graphs, then a it would be possible to define a finer measure which distinguishes between the number of incoming and outgoing edges.

3. Edit Distance

We are interested in computing the edit distance between the graphs $G_M = (V_M, E_M)$ referred to as the model graph and the graph $G_D = (V_D, E_D)$ referred to as the data-graph. The serial orderings of the nodes of the two graphs are denoted by $X = \{x_1, x_2, ..., x_{|V_M|}\}$ for the model graph and $Y = \{y_1, y_2, ..., y_{|V_D|}\}$ for the data graph. These two strings are used to index the rows and column of an edit lattice. The rows of the lattice are indexed using the data-graph string, while the columns are indexed using the model-graph string. To allow for differences in the sizes of the graphs we introduce a null symbol ϵ which can be used to pad the strings. We pose the problem of computing the edit distance as that of finding a path $\Gamma =< \gamma_1, \gamma_2, ...\gamma_k, ..., \gamma_L >$ through the lattice. Each element $\gamma_k \in (V_D \cup \epsilon) \times (V_M \cup \epsilon)$ of the edit path is a Cartesian pair. We constrain the path to be connected on the edit lattice. In particular, the transition on the edit lattice from the state γ_k to the state γ_{k+1} is constrained to move in a direction that is increasing and connected in the horizontal, vertical or diagonal direction on the lattice. The diagonal transition corresponds to the match of an edge of the data graph to an edge of the model graph. A horizontal transition means that the data-graph index is not incremented, and this corresponds to the case where the traversed nodes of the model graph are null-matched. Similarly when a vertical transition is made, then the traversed nodes of the data-graph are null-matched.

Suppose that $\gamma_k = (a, b)$ and $\gamma_{k+1} = (c, d)$ represent adjacent states in the edit path between the strings X and Y. According to the classical approach, the cost of the edit path is given by

$$d(X, Y) = C(\Gamma) = \sum_{\gamma_k \in \Gamma} \eta(\gamma_k \rightarrow \gamma_{k+1}) \tag{5}$$

where $\eta(\gamma_k \rightarrow \gamma_{k+1})$ is the cost of the transition between the states $\gamma_k = (a, b)$ and $\gamma_{k+1} = (c, d)$. The optimal edit path is the one that minimises the edit distance between string, and satisfies the condition

$$\Gamma^* = \arg\min_{\Gamma} C(\Gamma) \tag{6}$$

and hence the edit distance is $d(X, Y) = C(\Gamma^*)$. Classically, the optimal edit sequence may be found using Dijkstra's algorithm or by using the quadratic programming method of Wagner [24].

Since, we commenced with a probabilistic characterisation of the matching problem using Markov chains, we define the elementary edit cost to be the negative logarithm of the transition probability for the edit operation. Hence,

$$\eta((a,b) \rightarrow (c,d)) = -\ln P((a,b) \rightarrow (c,d)) \tag{7}$$

We adopt a simple model of the transition probability. The probability is a product of the node similarity weight, and the edge probabilities. Hence we write

$$P((a,b) \rightarrow (c,d)) = \beta_{a,b}\beta_{c,d}R_D(a,c)R_M(b,d) \tag{8}$$

where R_D and R_M are matrices of compatibility weights. The elements of the matrices are assigned according to the following distribution rule

$$R_D(a,c) = \begin{cases} \hat{P}_D(a,c) & if(a,c) \in E_D \\ P_\epsilon & \text{if } a = \epsilon \text{ or } c = \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

where E_D is the edge set of the data-graph, W_D is the associated normalised transition probability matrix and P_ϵ is the probability associated with a match to the null symbol ϵ. In practice, P_ϵ can be estimated using the size difference in the strings by setting

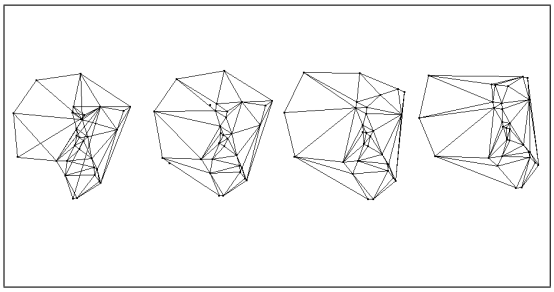
$$P_\epsilon = \frac{2|(|V_M| - |V_D|)|}{|V_M| + |V_D|}$$

The compatibility weight is hence zero if either the symbol pair (a,c) is unconnected by an edge of the data-graph, or the symbol pair (b,d) is unconnected by a model graph edge. As a result, edit operations which violate edge consistency on adjacent nodes in the strings are discouraged.

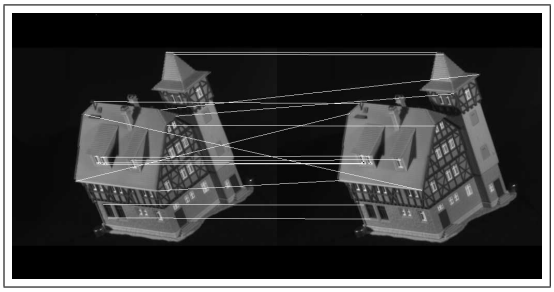
The optimal set of correspondences between the two sequences of nodes is found by minimising the string edit distance. The optimal sequence of correspondence Γ^* is found using Dijkstra's algorithm.

4. Experiments

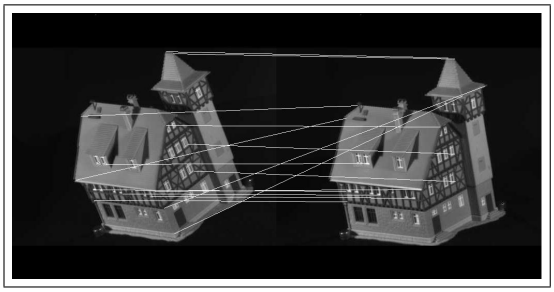
We have conducted some experiments with the CMU house sequence. This sequence consists of a series of images of a model house which have been captured from different viewpoints. To construct graphs for the purposes of matching, we have first extracted corners from the images using the corner detector of Luo, Cross and Hancock [10]. The graphs used in our experiments are the Delaunay triangulations of these points. The Delaunay triangulations of the example images are shown in Figure 1a. We have matched pairs of graphs representing increasingly different views of the model house. To do this, we have matched the first image in the sequence, with each of the subsequent images. In Figure 1 b, c and d we show the sequence of correspondence matches. In each case the left-hand graph contains 34 nodes, while the right-hand graphs contain 30, 32 and 34 nodes. From the Delaunay graphs it is clear that there are significant structural differences in the graphs. The numbers of



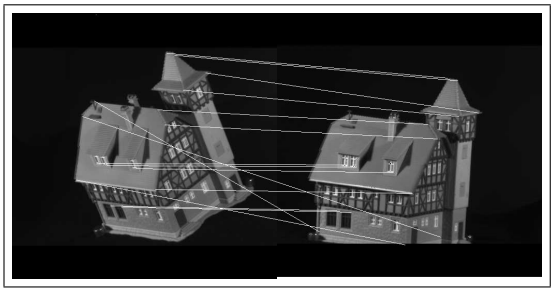
(a)



(b)



(c)



(d)

Fig. 1. Delaunay triangulations and sequence of correspondences

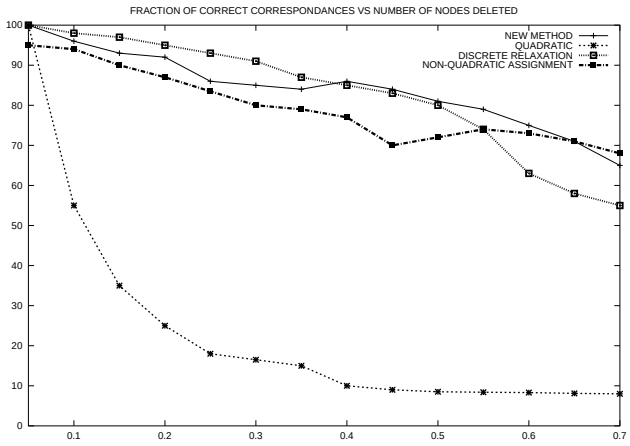


Fig. 2. Sensitivity study results.

correctly matched nodes in the sequence are respectively 29, 24 and 20 nodes. By comparison, the more complicated iterative EM algorithm of Luo and Hancock [11] gives 29, 23 and 11 correct correspondences. As the difference in viewing direction increases, the fraction of correct correspondences decreases from 80% for the closest pair of images to 60% for the most distant pair of images.

We have conducted some comparison with a number of alternative algorithms. The first of these share with our method the feature of using matrix factorisation to locate correspondences and have been reported by Umeyama [22] and Shapiro and Brady [19]. Since these two algorithms can not operate with graphs of different size, we have taken pairs of graphs with identical numbers of nodes from the CMU sequence; these are the second and fourth images which both contain 32 nodes. Here the Umeyama method and the Shapiro and Brady method both give 6 correct correspondences, while both the Luo and Hancock [11] method and our own give 22 correct correspondences.

Finally, we have conducted some experiments with synthetic data to measure the sensitivity of our matching method to structural differences in the graphs and to provide comparison with alternatives. Here we have generated random point-sets and have constructed their Delaunay graphs. We have simulated the effects of structural errors by randomly deleting nodes and re-triangulating the remaining point-set. Three algorithms are compared with the new method. The first of these is the Wilson and Hancock discrete relaxation scheme [26]. This assigns discrete matches to the nodes, and adjust the configuration of matches to maximise a probabilistic consistency criterion. The second method is Gold and Rangarajan's [5] soft-assign method, which uses a mean-field method to update a set of continuous assignment variables so as to optimise a quadratic consistency criterion. Thirdly, there is method of Finch, Wilson and Hancock [4] which is similar to the Gold and

Rangarajan method, except that the consistency criterion is exponential in nature.

In Figure 2 we show the fraction of correct correspondences as a function of the fraction of deleted nodes. For all four methods, as the fraction of nodes deleted increases then so the fraction of correct correspondences decreases. The performance curve for our new method is marked as "New Method". Also shown on the plot are performance curves for the Wilson and Hancock discrete relaxation scheme, [26], the Gold and Rangarajan [5] quadratic assignment method and the Finch, Wilson and Hancock [4] non-quadratic assignment method. The poorest performance is delivered by the Gold and Rangarajan method, which drops of very rapidly with increasing levels of corruption. Our method gives performance that is significantly better than the Gold and Rangarajan method, and intermediate in performance between the discrete relaxation and non-quadratic assignment methods. Both of the competitive methods, achieve robustness by compounding a series of exponential functions, and are hence computationally demanding, In fact, in the worst case the complexity Wilson and Hancock method can grow exponentially with the degree difference between nodes.

5. Conclusions

The work reported in this paper provides a synthesis of ideas from spectral graph-theory and structural pattern recognition. We use the result from spectral graph theory that the steady state random walk on a graph is given by the leading eigenvector of the adjacency matrix. This allows us to provide a string ordering of the nodes in different graphs. We match the resulting string representations by minimising edit distance. The edit costs needed are computed using a simple probabilistic model of the edit transitions which is designed to preserve the edge order on the correspondences.

The ideas are relatively preliminary and there are clearly a number of ways in which the work presented in this paper can be developed. First, there is no guarantee that the serial ordering of the nodes is edge connected. To overcome this problem, in a companion paper we have recently developed a graph-spectral method for recovering a serial ordering which is maximally edge connected [14]. This method has also been applied to detect curvature minimising paths through fields of surface normals for the purposes of surface height recovery [15]. Second, there is considerable scope for placing some of the heuristic elements of our current method on a more rigorous footing. In the companion paper [14], we have posed the estimation of edit costs in a maximum a posteriori probability setting. This more principled framework will allow a deeper analysis of the method, and some of its theoretical properties to be better understood. For instance, it would be interesting to relate the edit distances to the structural properties of the graphs under study. One interesting question, is that of what value the distance acquires when one graph is subgraph isomorphic with a second. Finally, the serial orderings of the nodes of graphs may be used for tasks other than matching. For instance, they may be used for graph

clustering or embedding graphs in vector-spaces. This latter endeavour may be of particular importance, since it would allow the apparatus of conventional pattern recognition to be applied to graphs in vector form. Hence, the elusive task of learning relational structures may come within grasp.

References

1. H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(8):689–694, 1997.
2. Fan R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.
3. M. A. Eshera and K. S. Fu. A graph distance measure for image analysis. *IEEE Transactions on Systems, Man and Cybernetics*, 14:398–407, 1984.
4. A. M. Finch, R. C. Wilson, and E. R. Hancock. An energy function and continuous edit process for graph matching. *Neural Computation*, 10(7):1873–1894, 1998.
5. S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *PAMI*, 18(4):377–388, April 1996.
6. R. Horaud and H. Sossa. Polyhedral object recognition by indexing. *Pattern Recognition*, 28(12):1855–1870, 1995.
7. S. Kosinov and T. Caelli. Inexact multisubgraph matching using graph eigenspace and clustering models. In *Structural, Syntactic and Statistical Pattern Recognition*, number 2396 in LNCS, pages 133–142, 2002.
8. V. I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys. Dokl.*, 6:707–710, 1966.
9. L. Lovász. Random walks on graphs: a survey. *Bolyai Society Mathematical Studies*, 2(2):1–46, 1993.
10. Bin Luo and E. R. Hancock. Procrustes alignment with the EM Algorithm. In *8th International Conference on Computer Analysis of Images and Image Patterns*, pages 623–631, 1999.
11. Bin Luo and E. R. Hancock. Structural graph matching using the EM algorithm and singular value decomposition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 23(10):1120–1136, 2001.
12. R. Myers, R. C. Wilson, and E. R. Hancock. Bayesian graph edit distance. *PAMI*, 22(6):628–635, June 2000.
13. B. J. Oommen and K. Zhang. The normalized string editing problem revisited. *PAMI*, 18(6):669–672, June 1996.
14. A. Robles-Kelly and E. R. Hancock. Graph edit distance from spectral seriation. Submitted to IEEE Trans. on Pattern Analysis and Machine Intelligence, 2003.
15. A. Robles-Kelly and E. R. Hancock. A graph-spectral approach to shape-from-shading. *IEEE Trans. on Image Processing*, To Appear, 2004.
16. A. Sanfeliu and K. S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man and Cybernetics*, 13:353–362, 1983.
17. G. Scott and H. Longuet-Higgins. An algorithm for associating the features of two images. In *Proceedings of the Royal Society of London*, number 244 in B, pages 21–26, 1991.
18. L. G. Shapiro and R. M. Haralick. Relational models for scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 4:595–602, 1982.
19. L. S. Shapiro and J. M. Brady. A modal approach to feature-based correspondence. In *British Machine Vision Conference*, pages 78–85, 1991.
20. A. Shokoufandeh, S. J. Dickinson, K. Siddiqi, and S. W. Zucker. Indexing using a

spectral encoding of topological structure. In *Proceedings of the Computer Vision and Pattern Recognition*, pages 491–497, 1998.
21. S. Ullman. Filling in the gaps: the shape of subjective contours and a model for their generation. *Biological Cybernetics*, 25:1–6, 1976.
22. S. Umeyama. An eigen decomposition approach to weighted graph matching problems. *PAMI*, 10(5):695–703, September 1988.
23. R. S. Varga. *Matrix Iterative Analysis*. Springer, second edition, 2000.
24. R. A. Wagner and M. J. Fisher. The string-to-string correction problem. *Journal of the ACM*, 21(1):168–173, 1974.
25. J. T. L. Wang, B. A. Shapiro, D. Shasha, K. Zhang, and K. M. Currey. An algorithm for finding the largest approximately common substructures of two trees. *PAMI*, 20(8):889–895, August 1998.
26. R.C. Wilson and E. R. Hancock. Structural matching by discrete relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):634–648, June 1997.