

Graph Attribute Embedding via Riemannian Submersion Learning

Haifeng Zhao⁴ Antonio Robles-Kelly^{1,2,3} Jun Zhou^{1,2,3}

Jianfeng Lu⁴ Jing-Yu Yang⁴

¹ NICTA*, Locked Bag 8001, Canberra ACT 2601, Australia

² The Australian National University, Canberra ACT 0200, Australia

³ UNSW@ADFA, Canberra, ACT 2600, Australia

⁴ Nanjing University of Science and Technology, Nanjing 210094, China

Abstract

In this paper, we tackle the problem of embedding a set of relational structures into a metric space for purposes of matching and categorisation. To this end, we view the problem from a Riemannian perspective and make use of the concepts of charts on the manifold to define the embedding as a mixture of class-specific submersions. Formulated in this manner, the mixture weights are recovered using a probability density estimation on the embedded graph node coordinates. Further, we recover these class-specific submersions making use of an iterative trust-region method so as to minimise the L2 norm between the hard limit of the graph-vertex posterior probabilities and their estimated values. The method presented here is quite general in nature and allows tasks such as matching, categorisation and retrieval. We show results on graph matching, shape categorisation and digit classification on synthetic data, the MNIST dataset and the MPEG-7 database.

Keywords: Graph embedding; Riemannian geometry; relational matching

*NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

1 Introduction

The problem of embedding relational structures onto a manifold arises in parameterization of three-dimensional data [1], multidimensional scaling (MDS)[2], graph drawing [3] and structural graph matching [4]. In the pattern analysis community, there has recently been renewed interest in using embedding methods motivated by graph theory for representation, classification, retrieval or visualisation.

The aim along these lines is to recover a mapping of the relational structure under study so as to minimise a measure of distortion. An option here is to perform graph interpolation by a hyperbolic surface which has the same pattern of geodesic, i.e. internode, distances as the graph [5]. Collectively, these methods are sometimes referred to as manifold learning theory. In [6], a manifold is constructed whose triangulation is the simplicial complex of the graph. Other methods, such as ISOMAP [7], focus on the recovery of a lower-dimensional embedding which is quasi-isometric in nature. Related algorithms include locally linear embedding [8], which is a variant of PCA that restricts the complexity of the input data using a nearest neighbor graph. Belkin and Nigoyi [9] present a Laplacian eigenmap which constructs an adjacency weight matrix for the data-points and projects the data onto the principal eigenvectors of the associated Laplacian matrix.

Embedding methods can also be used to transform the relational-matching problem into one of point-pattern matching in a high-dimensional space. The problem is to find matches between pairs of point sets when there is noise, geometric distortion and structural corruption. The problem arises in shape analysis [10], motion analysis [11] and stereo reconstruction [12]. As a result, there is a considerable literature on the problem, and many contrasting approaches, including search [13] and optimisation [14] have been attempted. However, the main challenge in graph matching is how to deal with differences in node and edge structure.

Further, the main argument levelled against the work above is that they adopt a heuristic approach to the relational matching problem by using a goal-directed graph similarity measure. To overcome this problem, several authors have adopted a more general approach using ideas from information and probability theory. For instance, Wong and You [15] defined an entropic graph-distance for structural graph matching. Christmas, Kittler and Petrou [16] have shown how a re-

laxation labeling approach can be employed to perform matching using pairwise attributes whose distribution is modeled by a Gaussian. Wilson and Hancock [17] have used a maximum *a posteriori* (MAP) estimation framework to accomplish purely structural graph matching. Recently, Caetano *et al.* [18] have proposed a method to estimate the compatibility functions for purposes of learning graph matching.

One of the most elegant recent approaches to the graph matching problem has been to use graph spectral methods [19], which exploits the information conveyed by the graph spectra, i.e. the eigenvalues and eigenvectors of the adjacency matrix or the Combinatorial Laplacian. There have been successful attempts to use spectral methods for both structural graph matching [20] and point pattern matching [21, 12]. For instance, Umeyama [20] has developed a method for finding the permutation matrix which best matches pairs of weighted graphs of the same size, by using a singular value decomposition of the adjacency matrices. Scott and Longuet-Higgins [21], on the other hand, align point-sets by performing singular value decomposition on a point association weight matrix. Shapiro and Brady [12] have reported a correspondence method which relies on measuring the similarity of the eigenvectors of a Gaussian point-proximity matrix. Although Umeyama's algorithm [20] is elegant in its formulation and can deal with both weighted or unweighted graphs, it can not be applied to graphs which contain different numbers of nodes and for weighted graphs the method is susceptible to weight errors. One way to overcome these problems is to cast the problem of recovering correspondences in a statistical setting using the EM algorithm [22].

Spectral methods can be viewed as embedding the nodes of a graph in a space spanned by the eigenvectors of the adjacency matrix. In the case of Umeyama's algorithm [20], matching is effected by finding the transformation matrix that best aligns the embedded points. The Shapiro and Brady [12] algorithm finds correspondences by seeking the closest embedded points. Kosinov and Caelli [23] have improved this method by allowing for scaling in the eigenspace. Robles-Kelly and Hancock [?] make use of the relationship between the Laplace-Beltrami operator and the graph Laplacian so as to embed a graph onto a Riemannian manifold.

An alternative approach to inexact graph matching is that of computing a metric that reflects the overall cost of the set of edit operations required to transform one graph into another. The idea is to measure the similarity of graphs by counting the number of graph edit operations (i.e.

node, edge insertions and deletions) required to transform a graph into another. The concept of string edit distance has been extended to graphs by Sanfeliu and Fu [24], Eshera and Fu [25] and Shapiro and Haralick [26]. These methods pose the problem of computing a metric for measuring the similarity between two graphs as that of finding a common underlying topological structure through graph edit operations. This is closely related to the problem of measuring structural affinity using the maximum common subgraph. Myers, Wilson and Hancock [27] have shown how the method in [17] can be rendered more effective making use of the Levenshtein string edit distance. Furthermore, Rice, Bunke and Natker [28] showed the relationship between a particular set of cost functions and the longest common subsequence for a pair of strings. More recently, Sebastian and Kimia [29] have used a distance metric analogous to the string edit distance to perform object recognition from a dataset of shock graphs.

The maximum common subgraph is an important subject in graph theory and has been used to solve relational matching problems [30, 31]. The maximum common subgraph can be used to depict common topology and, as a result, it is a powerful tool for graph matching and object recognition. Unfortunately, the maximum common subgraph is not unique and does not depend on the index of the nodes of the graph. Moreover, the computation of the maximum common subgraph is an NP-complete problem. As a result, conventional algorithms for computing the maximum common subgraph have an exponential computational cost. Although these algorithms are simple to understand and code, their computational cost is high due to the fact they are usually based on maximal clique detection [32] or backtracking [33]. To overcome this problem, practical algorithms for computing the maximum common subgraph often rely upon approximate constraints to overcome the exponential complexity problem and achieve polynomial cost. For instance, Messmer and Bunke [34] have proposed a graph-error correction algorithm that runs in polynomial time.

In this paper, we aim at presenting a method to embed the graph vertex-attributes into a space where distances between nodes correspond the structural differences between graphs. This can be viewed as a learning process in which the aim of computation is the recovery of a set of linear operators, which map the node-set of a graph onto a metric space. Viewed from a Riemannian perspective, the embedding is given by a mixture of submersions governed by a kernel density estimation process. This leads to a formulation of the problem where the embedding space is

spanned by an operator which minimises the L2 norm between the hard limit of the graph-vertex posterior probabilities and their value estimated from class-specific information. The embedding presented here permits the use of metrics in the target space for purposes of categorisation and relational matching tasks. Moreover, it provides a link between structural and statistical pattern recognition techniques through differential geometry.

Thus, the formulation presented here has a number of advantages. Firstly, it permits matching and categorisation tasks to be performed using simple point-pattern matching methods in a unified framework by embedding graph nodes into a metric space. Secondly, we draw on the properties of operators on Riemannian manifolds [35]. This implies that the embedding is linked directly to the geometry of the underlying graph class. Moreover, the work presented here provides a means to graph embedding through Riemannian geometry and optimisation on manifolds which employ invariant subspaces of matrices [?]. Finally, the optimisation method presented here is quite general in nature. The method is devoid of parameters and can be considered as a trust-region method over a group of invariant matrix manifolds.

2 Riemannian Geometry

In this section, we provide the theoretical basis for our graph embedding method. Consider a set of graphs Γ partitioned into a set of classes $\Omega = \{\omega_1, \omega_2, \dots, \omega_{|\Omega|}\}$. Here, we develop a method to employ class-specific information to embed the graphs in Γ into a manifold using Riemannian submersions. This can be viewed as a learning process, in which the embedding is recovered from labeled, i.e. categorised, graphs. Thus, as input to the training step, the method takes a set of graphs whose nodes attributes are given by a set of vectors, and delivers, at output, a model which can be used for embedding graphs which are not in the training set.

This section is organised as follows. We commence by providing the necessary background on Riemannian manifolds. We then formulate the embedding as a mixture of Riemannian submersions arising from the set of charts corresponding to a set of submanifolds for the clusters in Ω . With this formulation at hand, we turn our attention to the recovery of the submersions themselves by optimising a cost function based upon the L2 norm on the posterior probabilities for the graph-

vertices given the class information.

2.1 Riemannian Manifolds

Recall that our aim in this paper is to embed a graph into a Riemannian manifold whose submersions are class-specific. This implies that we have to first abstract a graph into a manifold. Secondly, we have to define the submersions such that they benefit from the label information and the Riemannian characterisation of the graph set.

To commence, let $G = (V, E, W)$ denote a weighted graph with index-set V , edge-set $E = \{(u, v) | (u, v) \in V \times V, u \neq v\}$ and edge-weight $W : E \rightarrow [0, 1]$. Here, we view the nodes in the graph as a vector space on the manifold \mathcal{M} . Consider an n -dimensional differentiable manifold \mathcal{M} . For any point $p \in \mathcal{M}$, let \mathcal{M}_p denote the tangent space of \mathcal{M} at p . Further, let X be a differentiable vector field in \mathbb{R}^n such that $X = \sum_{i=1}^n x^i \partial_i$, where x^i is the i th element of the vector X on the chart $\eta = \sum_{i=1}^n \eta^i \partial_i|_p$ with coordinates η^i , and $e = \{e_1, e_2, \dots, e_n\}$ is the natural basis $(\mathbb{R}^n)_p$, i.e. the natural basis of \mathbb{R}^n at $p \in \mathcal{M}$. In the equations above, the symbol ∂_i has been defined so as to be consistent with both the notion of a chart in Riemannian geometry and the natural basis e . To provide a definition of ∂_i , we turn our attention to the natural identification $\mathfrak{S}_p : \mathbb{R}^n \mapsto (\mathbb{R}^n)_p$ of the tangent space at p , i.e. $(\mathbb{R}^n)_p$, onto \mathbb{R}^n . For the natural basis, the chart is then given by the identity map such that $\partial_i|_p = \mathfrak{S}_p e_i$.

The advantage of the definitions above is that they relate the concepts of a vector field, the tangent bundle and the notion of chart so as to study graphs, i.e. objects, associated with a submanifold $\mathcal{U} \in \mathcal{M}$ through a bijection $\varphi(\mathcal{U})$. Moreover, we can now see the collection of charts for every cluster in the graph set as an atlas \mathcal{A} of \mathcal{M} such that $\varphi(\mathcal{U}_{\omega_i}) \circ \varphi(\mathcal{U}_{\omega_j})^{-1} : \mathbb{R}^n \mapsto \mathbb{R}^n$, where we have written \mathcal{U}_{ω_i} to imply that the submanifold \mathcal{U}_{ω_i} corresponds to the i^{th} cluster ω_i in the graph set Γ . In Figure 1, we use a manifold \mathcal{M} in \mathbb{R}^3 to provide some intuition on the formalism above. Note that the bijection $\varphi(\mathcal{U})_{\omega_i}$ maps the submanifold \mathcal{U}_{ω_i} onto the plane spanned by the natural basis e , which, in this case, corresponds to \mathbb{R}^2 .

In the following section we elaborate further on how this formulation permits viewing the graph nodes as a vector field on a manifold which can be submerged into a Riemannian space. This pro-

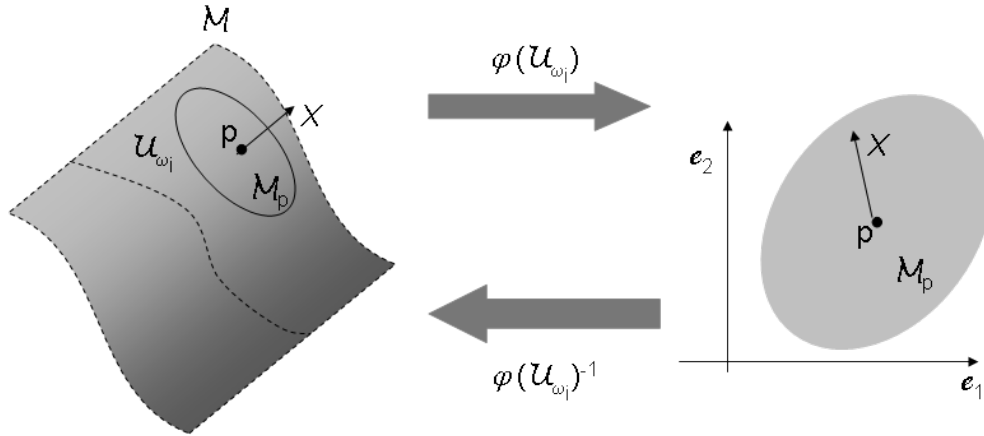


Figure 1: Relationship between a manifold \mathcal{M} , the bijection $\varphi(\mathcal{U})_{\omega_i}$ on the submanifold \mathcal{U}_{ω_i} for a vector X and the tangent space \mathcal{M}_p with respect to the natural basis e .

vides the flexibility and formalism of the tools and techniques available from the field of differential geometry and, moreover, information geometry [?].

2.2 Embeddings, Submersions and Charts

Note that the idea of chart endows the graph abstracted into a vector space X with a manifold structure. This manifold structure is bestowed upon the vector space X and, therefore, the topology of the graph G has to be used to recover the vector field in such a manner that it encodes structural information. In our experiments, we have used vertex-neighbourhood information to recover the attributes used by our embedding. A more principled use of neighborhoods and cliques may be formalised through the application of branched coverings [?], which operate on vertex neighbourhoods and generalise to arbitrary dimensions in a straightforward manner.

For now, we leave open the manner in which the graph-vertices are represented by the vector space X . As mentioned above, we will examine this further in the Experiments section. Note that we can use the idea of bijection between the natural basis and \mathcal{M} and a submersion $F : \mathcal{M} \mapsto \mathcal{U}_{\omega_i}$ to embed the graphs in the set Γ . We can then view the F as a transformation matrix such that $Y = FX$. Thus, given the coordinates in \mathfrak{R}^n for all the nodes in the graph-set Γ the purpose is to find a transformation matrix $F \in \mathfrak{R}^{n \times d}$ that projects the input vectors for the nodes in the graph

onto the submanifold \mathcal{U}_{ω_i} such that $Y = FX$ belongs to a lower dimensional space $\mathfrak{R}^d (d \leq n)$ maximising the separation between classes in Ω and the affinity within classes.

Following the chart and atlas concepts presented above, there must be a smooth transition between charts across the submanifolds \mathcal{U}_{ω_i} . Moreover, consider the posterior probability $P(G | \omega_i)$ of a graph in Γ belonging to the cluster ω_i . Ideally, if a graph belongs to a given cluster, the posterior $P(G | \omega_i)$ should be unity if and only if the graph belongs to the class ω_i , i.e.

$$P(X | G \in \omega_i) = \begin{cases} 1 & \text{if } G \in \omega_i \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

By relaxing the hard limit above for $P(X | G \in \omega_i) \in [0, 1]$, we can write the embedding for the nodes in G as a mixture of the form

$$Y^* = \sum_{i=1}^{|\Omega|} P(X | G \in \omega_i) F_{\omega_i} X \quad (2)$$

where, as before, we have written F_{ω_i} to imply that the submersion corresponds to the cluster ω_i .

Thus, for the atlas, the submersion is then given by the mixture

$$F^* = \sum_{i=1}^{|\Omega|} P(X | G \in \omega_i) F_{\omega_i} \quad (3)$$

In Figure 2, we illustrate the geometric meaning of our embedding approach. In the figure, we show a manifold $\mathcal{M} \in \mathfrak{R}^3$ akin to that in Figure 1. Here, the submersions F_{ω_i} and F_{ω_j} correspond to two clusters in the graph set Γ . Note that, despite these being mutually independent, the tangent bundle \mathcal{M}_p “overlaps” the submanifolds \mathcal{U}_{ω_i} and \mathcal{U}_{ω_j} . Following the figure, we can view the posterior probabilities $\alpha_i(Y) = P(X | G \in \omega_i)$ as weights that govern the contributions of the submersions to the node embedding Y^* . This is consistent with our premise above regarding the smooth transition between the tangent spaces of \mathcal{U}_{ω_i} and \mathcal{U}_{ω_j} . Note that the embedding thus hinges in the computation of both, the submersions and the posteriors. For the remainder of the section, we elaborate on the computation of the posterior probabilities. In the following section we show how the recovery of the submersions can be effected through an optimisation whose aim of computation is to maximise separation between classes and affinity within clusters.

Recall that, since label information is at hand, we have at our disposal the true values, i.e. the hard limits, of the posterior probabilities. Furthermore, the posterior probability density functions

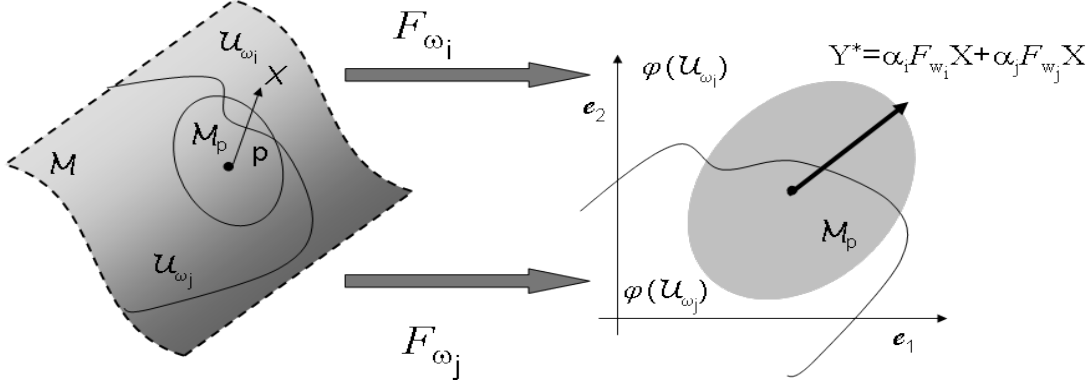


Figure 2: Geometric interpretation of our embedding approach.

can be estimated from the data using Kernel Density Estimation (KDE) [36]. This suggests an optimisation approach to the problem in which the posterior probabilities can be estimated from the submerged graph nodes while the submersions themselves can be optimised upon. Thus, from now on, we use the shorthand $\alpha_i(Y)$ to denote the posterior probabilities given by

$$\alpha_i(Y) = P(X | G \in \omega_i) = \frac{KDE_i[F_{\omega_i} X]}{\sum_{k=1}^{|\Omega|} KDE_k[F_{\omega_k} X]} \quad (4)$$

where $KDE_i[\cdot]$ is the evaluation of the Kernel Density Estimator trained using the vectors $Y = F_{\omega_i} X$ corresponding to the nodes in those graphs $G \in \omega_i$.

2.2.1 Optimisation

Now, we turn our attention to the recovery of the submersion F_{ω_i} . To this end, we make use of the hard limits of the posterior $P(X | G \in \omega_i)$ so as to formulate a cost function which we can then optimise accordingly. Recall that, if the graph G is in the cluster ω_i , the posterior should be unity and zero otherwise. Thus we can define the cost function as the L2 norm expressed as follows

$$\mathcal{D} = \sum_{i=1}^{|\Omega|} \sum_{\substack{V \in G \\ G \in \omega_i}} |1 - \alpha_i(Y)|^2 + a \|F_{\omega_i}\| = \sum_{i=1}^{|\Omega|} \sum_{\substack{V \in G \\ G \in \omega_i}} g(Y, \xi)^2 \quad (5)$$

where a is a regularisation constant.

In the cost function above, the first term is zero when the posterior given by $\alpha_i(Y)$ is unity. That is, the term tends to zero as the posterior is closer to its hard limit. The second term is a regularisation one which encourages sparseness upon the submersions F_{ω_i} . As a result, the cost

function is minimum when the posteriors achieve their hard limit values making use of sparse submersions.

With the cost function at hand, we now turn our attention to its extremisation. For the minimisation of the target function we have used a manifold-based variant of the Levenberg Marquardt [37] approach. The Levenberg-Marquardt Algorithm (LMA) is an iterative trust region procedure [38] which provides a numerical solution to the problem of minimising a function over a space of parameters. For purposes of minimising the cost function, we commence by writing the cost function above in terms of the parameter set, which in this case corresponds to the entries of the submersions F_{ω_i} . Let $\xi = [VEC(F_{\omega_1}), VEC(F_{\omega_2}), \dots, VEC(F_{\omega_{|\Omega|}})]^T$ be the parameter vector at iteration t , where $VEC(\cdot) : \mathfrak{R}^{n \times d} \mapsto \mathfrak{R}^{nd}$ is an operator that takes a matrix as input and delivers a vector as output. At each iteration, the new estimate of the parameter set is defined as $\xi + \delta$, where δ is an increment in the parameter space and ξ is the current estimate of the transformation parameters. To determine the value of δ , let $g(Y, \xi) = \sqrt{|1 - \alpha_i(Y)|^2 + a\|F_{\omega_i}\|}$ be the posterior probability evaluated at time t approximated using a Taylor series such that

$$g(Y, \xi + \delta) \approx \sqrt{|1 - \alpha_i(Y)|^2 + a\|F_{\omega_i}\|} + \mathcal{J}(Y)\delta \quad (6)$$

where $\mathcal{J}(Y)$ is the Jacobian of $\frac{\partial g(Y, \xi + \delta)}{\partial \xi}$.

The set of equations that need to be solved for δ is obtained by equating to zero the derivative with respect to δ of the equation resulting from substituting Equation 6 into the cost function \mathcal{D} . Let the matrix \mathbf{J} be comprised by the entries $\frac{\partial g(Y, \xi + \delta)}{\partial \xi}$, i.e. the element indexed k, i of the matrix \mathbf{J} is given by the derivative of $\alpha_i(Y)$ with respect to the k^{th} element of the vector ξ for every graph-node in the set Γ . We can write the resulting equations in compact form as follows

$$(\mathbf{J}^T \mathbf{J})\delta = \mathbf{J}^T[\mathbf{G}^* - \mathbf{G}(\xi)] \quad (7)$$

where $\mathbf{G}(\xi)$ is a vector whose elements correspond to the values $g(Y, \xi)$ and \mathbf{G}^* is a vector whose elements correspond to the extreme values of the function $g(\cdot)$ for every X in those graphs in Γ such that $|\mathbf{G}^* - \mathbf{G}(\xi)|$ becomes minimal.

Recall that Levenberg [?] introduced a ‘‘damped’’ analogue of the equation above given by

$$(\mathbf{J}^T \mathbf{J} + \beta \mathbf{I})\delta = \mathbf{J}^T[\mathbf{G}^* - \mathbf{G}(\xi)] \quad (8)$$

β is damping factor adjusted at each iteration and \mathbf{I} is the identity matrix. In [37], the identity matrix \mathbf{I} above is substituted with the diagonal of the Hessian matrix $\mathbf{J}^T \mathbf{J}$ thus yielding

$$(\mathbf{J}^T \mathbf{J} + \beta \text{diag}[\mathbf{J}^T \mathbf{J}])\delta = \mathbf{J}^T[\mathbf{G}^* - \mathbf{G}(\xi)] \quad (9)$$

where $\text{diag}[\cdot]$ is an operator that yields a diagonal matrix.

Note that, within the trust region, the minimiser of the cost function is the solution of the Gauss-Newton Equation [38]. Since we have formulated the approach in terms of matrix manifolds [?], we can employ an approximate Hessian, which is positive definite making use of the gradient of the function such that

$$(\mathbf{J}^T \mathbf{J} + \mathbf{E})\mathcal{X}_t = -\nabla \mathbf{G}(\xi) \quad (10)$$

where \mathcal{X}_t is an iterate on the tangent bundle of the matrix manifold and \mathbf{E} is an operator chosen such that the convergence of the Newton iteration is superlinear, i.e.

$$\lim_{t \rightarrow \infty} \frac{|g(Y, \xi - \delta) + 1|}{|g(Y, \xi)|} = 0 \quad (11)$$

It can be shown that if $|\mathbf{E}| < 1$, then $(\mathbf{I} - \mathbf{E})^{-1}$ exists and satisfies the following relation

$$|(\mathbf{I} - \mathbf{E})^{-1}| \geq \frac{1}{1 - |\mathbf{E}|} \quad (12)$$

We refer the reader to [?] for the proof of the relations above.

In general, we can view ∇ as a connection on the matrix manifold. As a result, Equation 10 can be rewritten as

$$\mathbf{A}\mathcal{X}_t = -\zeta \quad (13)$$

where $\mathbf{A} = \mathbf{J}^T \mathbf{J} + \mathbf{E}$ and the entries of the vector ζ can be viewed as an object that connect tangent spaces on the manifold over the vector field spanned by \mathcal{X}_t at the current estimate of ξ , i.e. $\zeta_\xi = \nabla_{\mathcal{X}_t} \zeta$. This tells that, by construction, we have constrained the solution of the Newton iteration to a retraction \mathcal{R}_ξ of \mathcal{X}_t from the tangent bundle to the manifold at each iterate \mathcal{X}_t such that the vector ζ is parallel transported along \mathcal{X}_t .

Moreover, we can view Equation 13 as an eigenvalue problem on the matrix \mathbf{A} and aim at recovering the iterate such that the gradient of $\mathbf{G}(\xi)$ is maximum. This procedure can be interpreted as a gradient descent on the manifold. Note that, from the relations in Equations 11 and 12, we

can set $\mathbf{E} = \emptyset$, i.e. a matrix whose entries are all nil, and make $\mathbf{A} = \mathbf{J}^T \mathbf{J}$. Thus, let τ by the eigenvector corresponding to the leading eigenvalue ζ of \mathbf{A} , following the developments above, we can compute the current connection $\zeta = \zeta \tau$ and set the natural basis e on the manifold so as to coincide with the current iterate on the tangent bundle of \mathcal{M} . Thus, we have

$$\zeta = -(\mathbf{J}^T \mathbf{J} + \beta \text{diag}[\mathbf{J}^T \mathbf{J}]) \bullet e \quad (14)$$

The increment δ can be recovered by substituting Equation 14 into Equation 9, which yields

$$\begin{aligned} \delta &= -\frac{1}{\zeta} \bullet (\mathbf{J}^T [\mathbf{G}^* - \mathbf{G}(\xi)]) \\ &= -\frac{1}{\zeta} \bullet (\mathbf{J}^T [\mathbf{G}(\xi)]) \end{aligned} \quad (15)$$

where the last line above proceeds from the fact that the extreme values of $g(\cdot)$ are zero, i.e. we aim at minimising \mathcal{D} , and \bullet denotes the Hadamard (entry-wise) product between the vectors $\frac{1}{\zeta}$ and $\mathbf{J}^T [\mathbf{G}^* - \mathbf{G}(\xi)]$. Note that, in Equation 16, the computation of the increment δ is devoid of the damping factor β in Equation 8 with a superlinear convergence rate.

It is worth mentioning that no assumptions on the extreme values of the function $g(\cdot)$ were made in the derivation above until the second line of Equation 16 was reached. This is important since it implies that the method above can be employed for other tasks requiring non-linear least squares optimisers. Moreover, the method above can be viewed as an LMA one where the damping factor has been absorbed into the eigenvalue formulation in Equation 13. Thus, the method combines the benefits of a damped trust-region approach being devoid of parameters.

3 Discussion and Implementation Issues

In the previous sections, we have presented the theoretical foundation for the embedding method developed here. In this section, we turn our attention to the implementation issues regarding the method and provide further discussion.

In Figure 3 we show the step sequence of the algorithm. The algorithm takes the set of graphs Γ and clusters Ω as input and returns the submersions F_{ω_i} . In the pseudocode below, $MAT(\cdot)$ is an operator that delivers a matrix from a vector being the inverse of the $VEC(\cdot)$ operation

```

1: Input  $\Gamma, \Omega, a, \epsilon$ 
2: Initialise  $F_{\omega_i} \forall \omega_i \in \Omega$ 
3: Train the Kernel Density estimator  $KDE_i[\cdot]$  for every cluster  $\omega_i$  in  $\Omega$ 
4: Set  $t = 0$  and Compute  $\mathcal{D}_0 = \sum_{i=1}^{|\Omega|} \sum_{\substack{V \in G \\ G \in \omega_i}} |1 - \alpha_i(Y)|^2 + a \|F_{\omega_i}\|$ 
5: Do
6:   Compute  $Y = F_{\omega_i} X$  for all the vertices in the graph-set  $\Gamma$ 
7:   For  $i = \{1, 2, \dots, |\Omega|\}$ 
8:     Compute the matrix  $\mathbf{J}_i$  using finite differences
9:     Compute  $\zeta$  by performing and eigendecomposition on  $\mathbf{A} = \mathbf{J}_i^T \mathbf{J}_i$ 
10:    Compute the vector  $\mathbf{B} = (\mathbf{J}_i^T [\mathbf{1} - \mathbf{G}_i(\xi)])$ 
11:    Compute  $\xi_i |_{t+1} = VEC(F_{\omega_i})$  such that the  $j^{th}$  element of  $\xi_i |_{t+1}$  is given by
12:      
$$\xi_i^j |_{t+1} = \xi_i^j |_t - \frac{\mathbf{B}^j}{\zeta^j}$$

13:    End For
14:    Set  $t = t + 1$ 
15:    Train the Kernel Density estimator  $KDE_i[\cdot]$  for every cluster  $\omega_i$  in  $\Omega$ .
16:    Compute  $\mathcal{D}_t = \sum_{i=1}^{|\Omega|} \sum_{\substack{V \in G \\ G \in \omega_i}} |1 - \alpha_i(Y)|^2 + a MAT(\xi_i |_t)$ 
17: While  $|\mathcal{D}_t - \mathcal{D}_{t-1}| \geq \epsilon$  or maximum number of iterations has not been reached
18: Return  $F_{\omega_i} = MAT(\xi_i |_t) \forall \omega_i \in \Omega$ 

```

Figure 3: Pseudocode for the recovery of the submersions F_{ω_i} used in our algorithm.

introduced in Section 2.2.1. Also, we have modified slightly the method as presented previously. We note that, from Equation 5, we can conclude that minimising the squared difference for each cluster over Ω is equivalent to extrimising the cost function \mathcal{D} . Thus, in order to reduce the memory requirements on the optimisation and the computational cost on the eigendecomposition associated with the optimisation, we have introduced a loop over Ω in Line 7. As a result, we have indexed the matrices \mathbf{J} and $\mathbf{G}_i(\xi)$ in Lines 7-13 to the i^{th} cluster. Similarly, we have written ξ_i to imply that the parameter vector corresponds to the cluster ω_i and, for the sake of consistency with the notation used in the previous sections, we have used superscripts to indicate vector-element indexes.

It is worth noting that, being a trust region method, the algorithm in Figure 3 is sensitive to initialisation. To overcome this drawback, in the initialisation step in Line 2, we have used the set of submersions F_{ω_i} which correspond to the linear subspace projection that best separates every class from the others. One of the most popular methods for recovering these linear combinations in supervised learning is Linear Discriminant Analysis (LDA). Here, a transformation matrix $F_{\omega_i} \in \mathfrak{R}^{n \times d}$ is determined so as to maximise the *Fisher criterion* given by

$$\begin{aligned}
\mathcal{H}(F_{\omega_i}^T) &= \text{tr}((F_{\omega_i} \mathbf{S}_w F_{\omega_i}^T)^{-1} (F_{\omega_i} \mathbf{S}_b F_{\omega_i}^T)) \\
\mathbf{S}_w &= \sum_{\substack{X \in G \\ G \in \omega_i}} (X - \mathbf{m}_i)(X - \mathbf{m}_i)^T \\
\mathbf{S}_b &= (\mathbf{m}_i - \bar{\mathbf{m}})(\mathbf{m}_i - \bar{\mathbf{m}})^T \\
&= \sum_{j=1}^{|\Omega|} p_j (\mathbf{m}_i - \mathbf{m}_j)(\mathbf{m}_i - \mathbf{m}_j)^T
\end{aligned} \tag{16}$$

where $|\Omega|$ is the number of classes, \mathbf{m}_j and $\bar{\mathbf{m}}$ are the class mean and sample mean vectors, respectively, and $p_i = P(\omega_i)$ is the prior probability of class ω_i given by the contribution of the i^{th} class to the sample set $\mathcal{X} = \{X\}_{G \in \Gamma}$. Also, in the equations above, \mathbf{S}_w and \mathbf{S}_b represent the within and between-class scatter matrices. Note that the matrix \mathbf{S}_w can be regarded as the average class-specific covariance, whereas \mathbf{S}_b can be viewed as the mean distance between all different classes. Thus, the purpose of Equation 16 is to maximise the between-class scatter while preserving within-class dispersion in the transformed feature space. This is effected by affine projecting the inverse intraclass covariance matrix and solving the generalised eigenvalue problem $\mathbf{S}_b F_{\omega_i}^T = \lambda \mathbf{S}_w F_{\omega_i}^T$. As the rank of \mathbf{S}_b is $|\Omega| - 1$, the solution, for the $|\Omega|$ classes, is obtained by taking the eigenvectors corresponding to the largest $|\Omega| - 1$ eigenvalues of $\mathbf{S}_w^{-1} \mathbf{S}_b$ [39].

As presented above, the optimisation implies that the posterior $\alpha_i(Y)$ would be evaluated at every iteration t . This, in turn, assumes the method trains the KDE at every iteration. This is as the submersion F_{ω_i} is updated at every t and, hence, the model for the KDE varies accordingly. In our implementation, we have used a Normal, i.e. Gaussian, kernel.

Note that Equations 16 and 14 employ the natural basis e so as to coincide with the iterates on the tangent bundle of the manifold \mathcal{M} . Since the elements of the natural basis are, by definition, orthogonal with respect to one another [?], we can view the vector Y as a projection of the vector

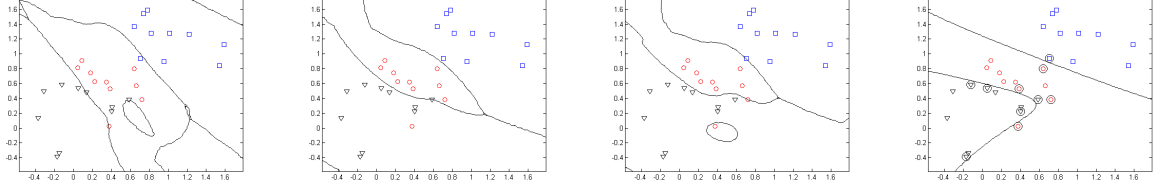


Figure 4: First, second and third panels: Posterior boundaries for a sample synthetic dataset at the first, tenth and last (twentieth) iterations of the optimisation method; Right-most panel: Decision boundaries for an SVM classifier.

X onto an orthogonal basis induced by the submersion F_{ω_i} . Moreover, as a result of our choice of a Gaussian kernel, the use of singletons can be interpreted as the equivalent of applying multivariate KDE to the principal components of a subspace-projected vector space [?]. Note that here, the scaling step of PCA is not applied and, therefore, the variances across the coordinates for the vectors Y are not equal. It is worth noting in passing that, for small sample sizes, this may result in kernel bandwidth problems or biased results. Thus, for small datasets, it may be preferable to effect the transformation directly prior to the processing of the data or to directly apply multivariate estimation techniques [?].

In our experiments, we have used datasets of medium and large sample sizes and, therefore, we train the estimators on the set of singleton fields. We can view these singletons as independent probabilities themselves and write

$$KDE_i[Y] = \prod_{k=1}^d kde[y^k] \quad (17)$$

where $kde[y^k]$ is the evaluation of the one-dimensional estimator trained on the k^{th} coordinate of the vectors $Y = F_{\omega_i}X = \sum_{k=1}^d y^k \partial_k$ corresponding to those graphs $G \in \omega_i$, where we have used the notation introduced in Section 2.1 for the sake of consistency.

Further, in Equation 17 we have assumed independence over the posterior probabilities in Equation 4. Indeed, we can write

$$\alpha_i(Y) = \frac{KDE_i[F_{\omega_i}X]}{\sum_{k=1}^{|\Omega|} KDE_k[F_{\omega_k}X]} = \frac{1}{\mathcal{Z}} \prod_{k=1}^d kde[y^k] \quad (18)$$

where $\mathcal{Z} = \sum_{k=1}^{|\Omega|} KDE_k[F_{\omega_k}X]$ is a partition function. It is worth noting in passing that this is

reminiscent of Gibbs fields [40] which can be modelled through Markovian processes in undirected graphs. This would also permit the application of our method to non-attributed, but rather weighted graphs by, for instance, using the edge-weights as the potentials of a Markov Random Field. This treatment has been employed for purposes of segmentation [?] and more generally in [?].

The use of singletons is practically useful, since, from the implementation point of view, we can use a univariate estimation method. This avoids the burden of dimensionality commonly found in multivariate KDE implementations. Thus, at each iteration t , we train a set of $|\Omega| \times d$ one-dimensional estimators and re-evaluate the posteriors so as to compute the values $\alpha_i(Y) | t$. This yields $|\Omega|$ submersions F_{ω_i} such that we can embed the graph nodes using Equation 2. The method is, indeed, quite general since it opens-up the possibility of performing a MAP classification based upon the posterior values by assigning a label to the graph by majority voting on the posteriors corresponding to its nodes. We can tackle the matching task by using Euclidean distances on the embedded nodes.

To illustrate the effects of the optimisation on the embedding and the posteriors, in Figure 4, we show a sample set of two-dimensional points which correspond to three classes. Each of these points have been plotted with a different marker, i.e. triangles, squares or circles, depending on their corresponding classes. In the panels, the boundaries correspond to those curves where the function $\chi = \{i | \max_i \{\alpha_i(Y)\}\}$ changes value. The left-hand panel shows the points and the posterior boundaries at the first iteration of the optimisation algorithm, i.e. $t = 1$. The two middle panels show the boundaries at $t = 10$ and convergence ($t = 20$), respectively. For the sake of comparison, the right-most panel shows the decision boundaries yielded by an SVM with an RBF kernel and parameters chosen via cross validation.

From the panels, we can see that the posterior boundaries between classes are highly non-linear. This is an important observation since the embedding is a mixture of submersions which can be viewed as linear operators. Nonetheless, the posteriors are estimated using a KDE process which is discriminative and non-linear in nature.

4 Experiments

Now we turn our attention to the applicability of the embedding method to relational matching, shape categorisation and object classification tasks. To this end, we have used three datasets. The first of these is a synthetic one which comprises graphs of different sizes, i.e. 50, 100, 150 and 200 nodes. The second set is the MNIST handwritten digit database [41]. The third dataset is given by the MPEG-7 CE-Shape-1 shape database, which contains 1400 binary shapes of 70 different classes with 20 images in each category. In all our experiments we have set $a = 0.01$ for the cost function in Equation 5. It is worth noting in passing that, in our experiments, the method was quite robust to the value of a .

4.1 Synthetic Graphs

We commence by performing experiments on synthetic data. To this end, we have generated graphs making use of randomly distributed point-sets of different sizes. These two-dimensional point-sets comprise the nodes of our synthetically generated data. We have computed two kinds of relational structures whose edges are given by the complete graphs and Delaunay triangulations [36] of the point-clouds under study. For each point-set order, i.e. 50, 100, 150 and 200, we have generated 20 point-clouds. Here, for purposes of comparison, we perform graph matching and compare our results with those yielded by the graduated assignment method of Gold and Rangarajan [14], the discrete relaxation algorithm in [42] and the graph seriation method in [43].

For the all-connected graphs, we use vectors X whose entries are given by the normalised histogram of pairwise Euclidean distances between the node of reference and the rest of the nodes in the graph. That is, given the node u , its corresponding vector X is given by the histogram of Euclidean distances $d(u, v)$ between the point-coordinates of u and those of $v \in V$. For the Delaunay triangulations, the vector X for the node u is given by the first k pairwise Euclidean distances $d(u, v)$, sorted in ascending rank order.

Note that the Delaunay graphs, together with other point neighbourhood graphs, such as the k -nearest neighbour graph, Gabriel graph and the relative neighbourhood graph, is sensitive to node deletion errors [44]. This implies that, for purposes of providing a sensitivity analysis on the

embedding method presented here, Delaunay graphs provide a means to evaluate the dependency of the embedding on the structural information and its robustness to change due to noise corruption and node deletion. Moreover, we have formulated the method in such manner that the embeddings are effected so as to operate on the vectors X , which are used to abstract the graph-nodes into a vector field. Hence, the effects of noise corruption or deletion on the all-connected relational structures should be much less severe than that exhibited by the Delaunay triangulations. This is due to the fact that the histograms used as the X vectors for the nodes of the all connected graphs can be viewed as a discrete approximation of the pairwise distance distribution for the points in G . This is in contrast with the Delaunay triangulations, where the vectors X have been formulated so as to reflect the structure of the edge-set rather than the pairwise distances for the node-set.

In our sensitivity study, we examine the effects of random node-deletions and the consequences of adding zero-mean Gaussian noise with increasing variance to the point-coordinates before the Delaunay triangulation computation. Thus, in the case of the node deletion experiments shown here, we have randomly deleted, in intervals of 5%, up to 50% of the graph nodes, i.e. points in the cloud. We have done this over 10 trials for each 5% interval, which yielded 2000 graphs for each of the four graph-sizes above, i.e. 8000 point-clouds in total. In the case of the Delaunay triangulations, these are computed after node deletion is effected.

To evaluate the effects of noise corruption, we have added zero-mean Gaussian noise of increasing variance to the 2D-coordinates of the point-sets used in our experiments. As for our node-deletion sensitivity study, we have computed the Delaunay triangulations after the Gaussian noise has been added. Here, we have increased the variance in steps of 0.05 up to 0.5 over 10 trials, i.e. $10 \times 20 = 200$ graphs for each of the point-sets under study, which as for the node-deleted graphs, yields 8000 graphs in total.

For the computation of the vector fields, we have used 10 histogram bins (all connected graphs) and $k = 5$ (Delaunay triangulations). For both graph types, i.e. all-connected and Delaunay triangulations, we have computed 20 submersions using the graphs recovered from the point-clouds devoid of deletions. We use these deletions so as to embed the remaining graphs into a space in which matching can be effected using a nearest-neighbour approach based upon Euclidean distances between the model and the data graphs, i.e. the one from which the submersion was com-

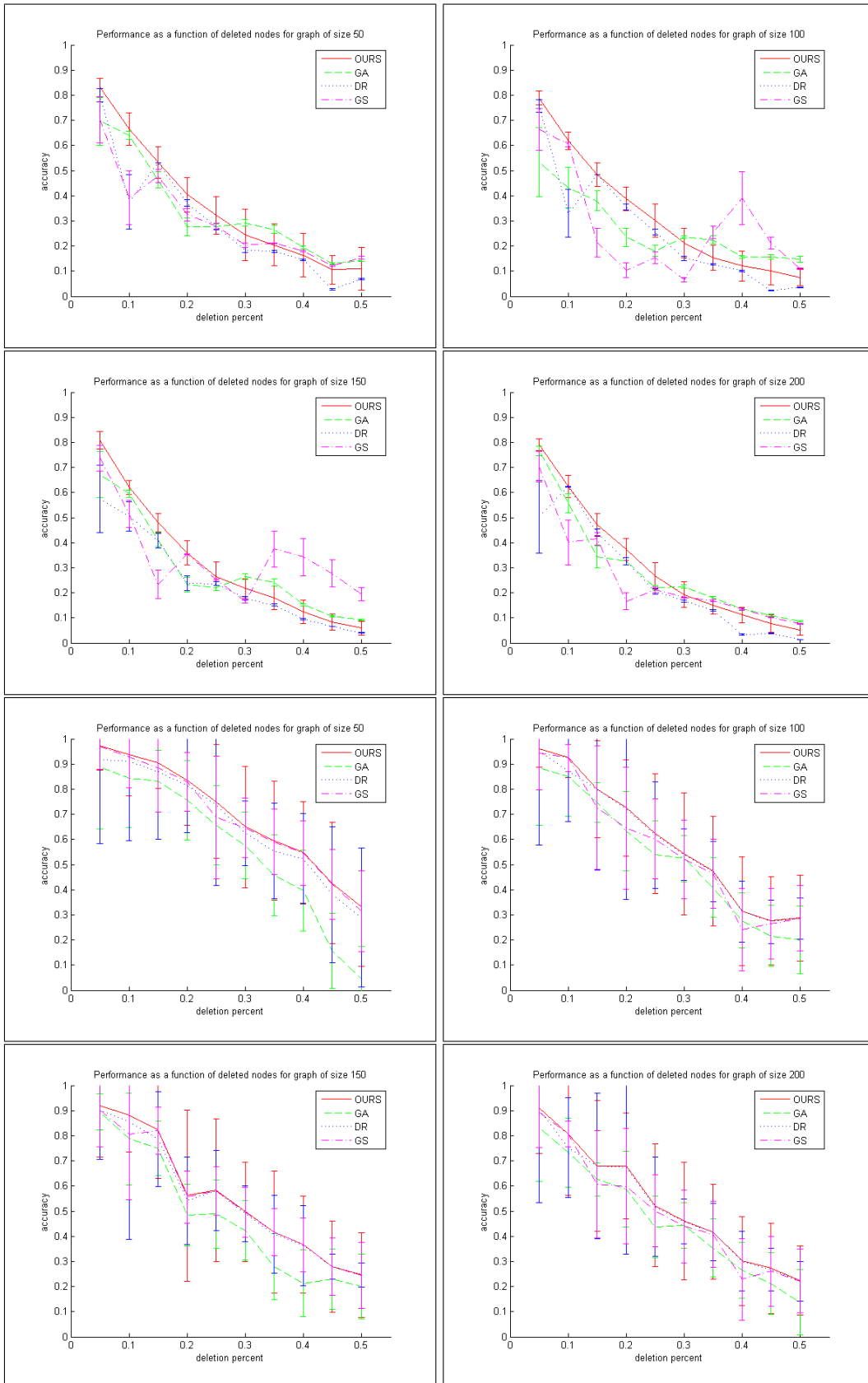


Figure 5: Matching performance as a function of proportion of nodes randomly deleted. Top-most rows: Results yielded for Delaunay triangulations of 50, 100, 150 and 200 nodes; Third and fourth rows: Results for all-connected graphs comprised by 50, 100, 150 and 200 nodes.

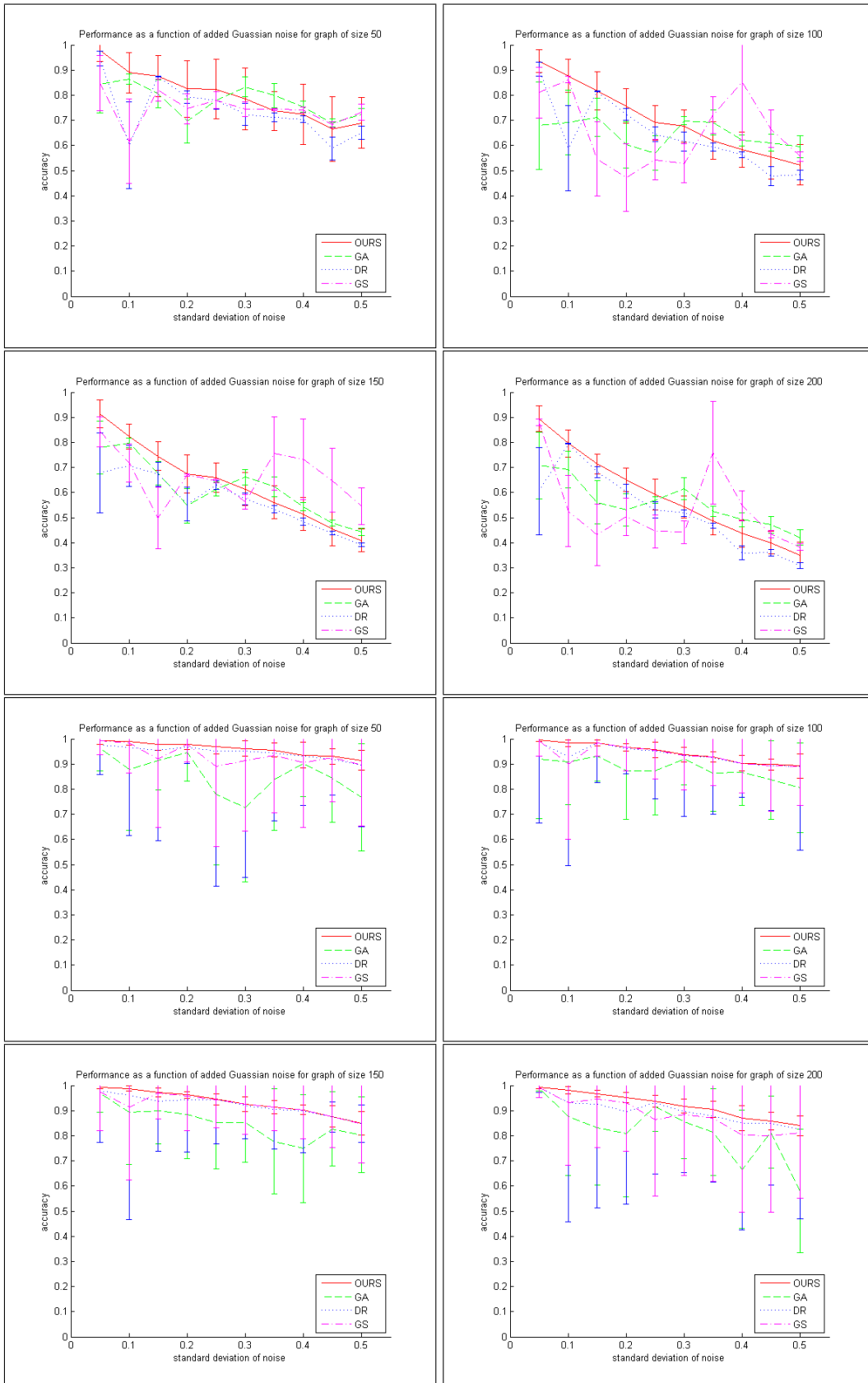


Figure 6: Matching performance as a function of added Gaussian noise variance. Top-most rows: Results yielded for Delaunay triangulations of 50, 100, 150 and 200 nodes; Third and fourth rows: Results for all-connected graphs comprised by 50, 100, 150 and 200 nodes.

puted and the relational structure we aim at matching against the model. Thus, a node v in the model graph G_M is a match to a node u in the data graph G_D such that $\operatorname{argmin}_{u \in G_D} \{d(u, v)\} = \operatorname{argmin}_{u \in G_D} \{(Y_u - Y_v)^2\}$, where the vector Y_u corresponds to the submerged, i.e. embedded, vector X corresponding to the node u . This is, u and v match if and only if they are nearest neighbours in the submersion corresponding to the model graph G_M .

In Figure 5 we show the plots for the accuracy, in percentage of correctly matched nodes, as a function of percentage of deleted nodes for the four graph sizes in our dataset. In the panels, the error-bars account for the standard deviation across each of the ten graphs for every node-deletion proportion from 0.05 to 0.5, with each line-style denoting a different approach, i.e. our method, the graduated assignment (GA) algorithm, the discrete relaxation (DR) approach and the graph seriation (GS) method. In the figure, the two top-most rows correspond to the Delaunay triangulations, whereas the third and fourth rows show the accuracy plots for the all-connected graphs.

In Figure 6, we repeat the sequence above for the noise-corrupted graphs. Note that, from the plots in Figures 5 and 6, the method provides a margin of improvement over the alternatives for low and moderate amounts of corruption when used to match both, the Delaunay triangulations and the all-connected graphs. Our embedding method is comparable to the alternatives even for large amounts of noise corruption or proportion of nodes deleted, being more stable than the alternatives. This is particularly evident as compared to the graph seriation (GA) method, which delivers better results in average with an increasing standard deviation. This illustrates the ability of the embedding to capture structural information. Moreover, when all-connected graphs are used, the sensitivity to noise corruption or node deletion is low. This is due to the fact that the X vectors have been constructed making use of a pairwise histogram as an alternative to the rank ordering of the pairwise distances for those nodes comprising cliques in the Delaunay triangulations. This sensitivity differences between the Delaunay triangulations and the all-connected graphs are in accordance with the developments in [44].

4.2 Digit Classification

Now we illustrate the ability of our method to embed graph attributes extracted from a real-world dataset. To this end, we show results on digit classification making use of the MNIST dataset. The MNIST database which contains a training set of 60,000 gray-scale images of handwritten digits from 0 to 9, and a testing set of 10,000 images. Sample handwritten digits in the dataset are shown in Figure 7. The image sizes are 28×28 . We have chosen the dataset due to its size and its widespread use in the literature. We stress in passing that the aim in this section is not to develop an OCR algorithm but rather to provide a quantitative study on a large, real-world dataset comparing our embedding approach with other subspace projection methods. Note that there are a wide variety of OCR techniques that have been applied to the dataset. As these are out of the scope of this paper, for a detailed list we refer the reader to the MNIST website ¹.

As mentioned earlier, one of the advantages of the strategy adopted here is that it provides a means to connect structural techniques with statistical pattern recognition methods through a general method for matching, categorisation and classification. As a result, we focus on the comparison of our method to other embedding techniques. These are Linear Discriminant Analysis (LDA) [39], the Maximum Margin Criterion (MMC) [?] and Nonparametric Discriminant Analysis (NDA) [39]. Note that these are subspace projection methods that employ linear transformations so as to recover the optimal projection direction for the classification task at hand. Here, classification using the alternatives is based on a feature space with dimensionality reduced to 9, i.e. the number of classes minus 1. To compare the methods on an equal basis, for all the alternatives we have used a 1-nearest-neighbour classifier [45].

For our digit classification experiments we have used a codebook-based approach for recovering the vectors X . To recover the feature vectors, we commence by binarising the digits using the thresholding method of Otsu [46]. With the binary imagery at hand, we have recovered frequency pairwise-distance histograms between pixels on the digit contours. With these histograms at hand, we have computed a codebook via k -means clustering. This codebook is used to compute codewords which are then concatenated to the projection histograms [?] with respect to the upper,

¹<http://yann.lecun.com/exdb/mnist/>

lower, left and right edges of the image under consideration. These codewords are of dimensionality $k = 200$, whereas the each projection histogram has 28 entries. Thus, our vectors X have a dimensionality of 312, i.e. $28 \times 4 + 200$. Here, the submersions used are such that they embed the digits into a space whose dimensionality is 20. This dimensionality was chosen as a good tradeoff between computational complexity and performance.

For purposes of classification, we note that we can use the posteriors given by α_{ω_i} in conjunction with the cluster structure in the MNIST database so as to search for the most similar object in a manner akin to that in [47]. The idea is to find the digit-classes which are most similar to the testing imagery. This can be viewed as a query in which the graph within the database-classes that is most similar to the query is the one that is retrieved. The search process is as follows. First, we compute the set of posteriors for the testing digit for each class in the dataset. The classes with the most similar digits are expected to have larger posteriors in comparison with the others. Thus, we select the first 3 classes, i.e. digit groups, in the training set which correspond to those posteriors α_{ω_i} that are the largest in rank amongst those computed for the testing raster scan. With the candidate classes at hand, we search the training samples in each class so as to find the one that is most similar to the testing digit. The testing raster scan then belongs to the class whose item, i.e. digit, corresponds to the minimum pairwise Euclidean distance. It must be stressed that this can also be viewed as a simple recall strategy which illustrates how the embedding method presented here can be used for purposes of retrieval in a broader sense. The information retrieval literature contains more principled and more efficient alternatives which should be used if the example given here is scaled to very large databases of thousands or tens of thousands of images.

Results for the MNIST dataset are shown in Table 1. From the experiments, we can conclude that the proposed embedding algorithm provides better performance than the alternatives, followed by NDA. Note that, as mentioned earlier, the method permits classification and matching tasks to



Figure 7: Sample images from the MNIST handwritten digit database.

Classification Method	Our Method	LDA	MMC	NDA
Classification Rate	96.12%	87.52%	86.92%	88.39%

Table 1: Percentage of correctly classified digits for our method as compared with the alternatives.

be effected in a metric space allowing for a wide variety of features to be used as vectors X . This is important since, in our experiments, we have not preprocessed the data or performed any stabilisation operation on the digits. These tasks can be easily introduced into our method without any loss of generality. Moreover, despite the fact that our testing strategy does not use intensive search as that for the alternatives, we still obtain a performance improvement for more than 8% as compared with non-linear LDA [39] and maximum margin criterion [?].

Note that, despite the X vectors are of dimensionality 312, our approach is quite robust to changes in dimensionality and, due to the use of singletons for the KDE estimators, is not overly affected by the curse of dimensionality. Moreover, our descent method is based upon an eigenvalue estimation step for which computational methods exist so as to deal with very large dimensions, i.e. thousands of dimensions. This is consistent with our experiments, where we have not noticed serious drops in performance as a result of employing larger dimensionalities in our frequency histograms.

4.3 Shape Categorisation and Matching

Now, we turn our attention to shape categorisation. To this end, we use the MPEG-7 CE-1 Part B database [?]. The database contains 1400 binary shapes organised in 70 classes, each comprised of 20 images. In Figure 8, we show some example shapes for the MPEG-7 dataset. For the MPEG-7 CE-Shape-1 shape database, we employ the histogram of pairwise distances in the embedding space to define the vectors X . We do this in an analogue manner to that used in the previous sections for matching all-connected synthetic graphs and MNIST digits. We construct a graph for each shape, where the pixels on the contour are the nodes and the X vectors are given by the pairwise-distance frequency histograms between adjacent vertices, i.e. points on the contour.

The submersions for our method, i.e. one for each of the 70 shape categories in the dataset, have

been recovered making use of these frequency histograms. Here, we have sampled 1 in every 10 pixels on the shape contours. With the sample contour pixels at hand, we build a fully connected graph whose edge-weights are given by the Euclidean distances on the image plane between each pair of pixel locations. Thus, the entries of the weight matrix for the graph correspond to the pairwise distances between the image-coordinates for every pair of vertices in the graph. The weight matrix is then normalised to unity so as to have every weight in the graph in the interval $[0, 1]$. The vectors X are given by the frequency histograms of these distances for every node. That is, for the i^{th} vertex in G , the vector X is given by the histogram for the edge-weights for the pairs centered at the node indexed i . In our experiments, we have used 10 bins for the frequency histogram computation.

Once the embedding is at hand, we can perform relational matching by viewing the node matching task as a point-matching one in the embedding space. To this end, we have used the coordinates $Y = F_{\omega_i} X$ in the target space so as to compute distances between nodes in the graphs to be matched. The correspondences are deemed to be the nearest neighbours for the vertex embeddings for each of the graphs under study. That is, the vertex in the data graph is such that the Euclidean distance between the corresponding embedded coordinate vectors is minimum for all the nodes in the candidate graph. In Figure 9 we show sample matching results for our method and the alternatives on two pairs of shapes. From top-to-bottom, we show the results yielded by our method, the graduated assignment algorithm [14], discrete relaxation [42] and the seriation method in [43]. Note that our method can cope with shape deformations of the shape contours. Moreover, qualitatively, the results obtained making use of the embedding presented here show less mis-assignments than those recovered using the alternatives.

We now examine the utility of our method for purposes of shape categorisation. For our method, categorisation is effected making use of a codebook-based method akin to the bag of words approaches in computer vision [48]. Here we have used the pairwise distances between pixels on the shape contour so as to recover a frequency histogram which can then be used to recover a codebook via k -means clustering. With the codebook at hand, the shapes under study can be represented by codewords which we then embed into a metric space using our method. Thus, the vectors X are given by these codewords. Here, we have set $k = 200$. Recall that, in previous sections, we intro-

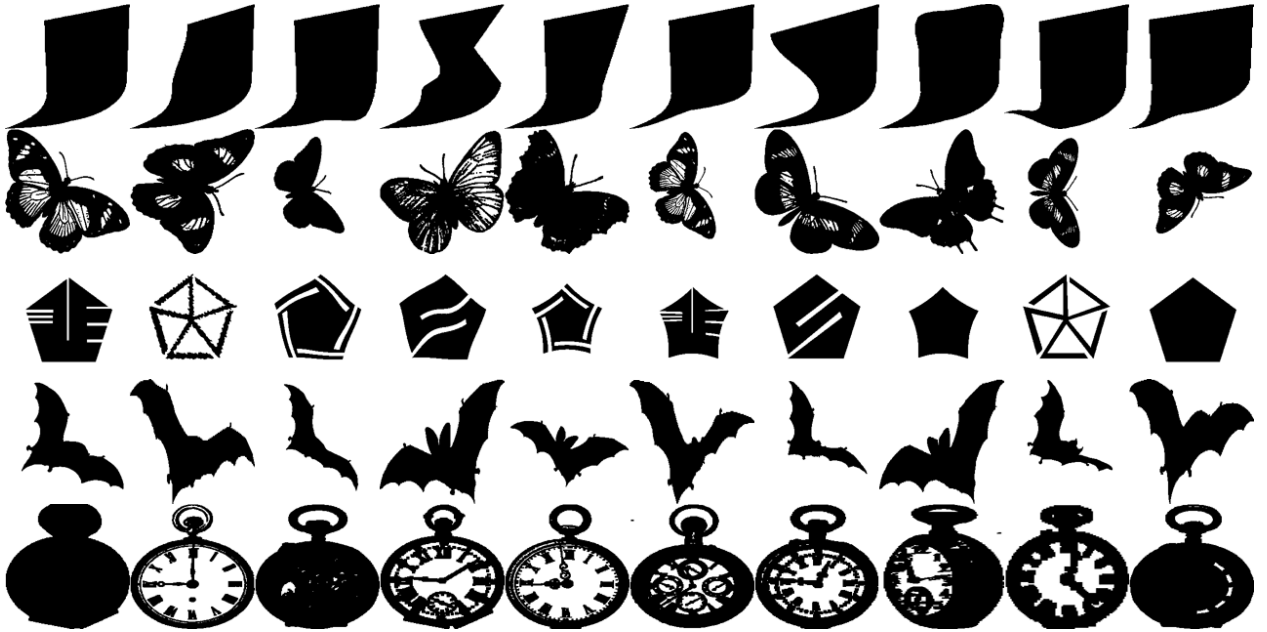


Figure 8: Samples images from the MPEG-7 shape databases used in our experiments.

duced the variables $\alpha_i(\cdot)$ as posterior probabilities for a graph in the i^{th} class ω_i in the database. Note that we can use this posteriors in a number of ways. These can be viewed as class indicator variables, such as those employed in [49]. Other options include taking a classifier ensemble approach such as boosting [50], where each of the variables $\alpha_i(\cdot)$ can be combined making use of a purposely recovered rule. Here, we take a simple majority voting approach, where a graph belongs to the cluster ω_i if the majority of its nodes have a value of $\alpha_i(\cdot)$ such that $\alpha_i(\cdot) > \alpha_j(\cdot)$ for any cluster indexed $j \neq i$. This approach can be viewed as a MAP classification with a uniform prior over the graph vertex-set.

For the sake of comparing our results with alternatives elsewhere in the literature, we also show classification rates making use of two methods specifically designed for purposes of shape classification. These are the shape and skeletal contexts in [?] and [51], respectively. Once the shape and skeletal contexts are at hand, we train two one-versus-all SVM classifiers whose parameters have been selected through ten-fold cross validation. For our shape categorisation experiments, we have divided the graphs in the dataset into a training and a testing set. Each of these contains half of the graphs in each dataset. This is, we have used 700 randomly selected graphs for training and the remaining 700 for testing. We have done this ten times, applying our submersion method and the

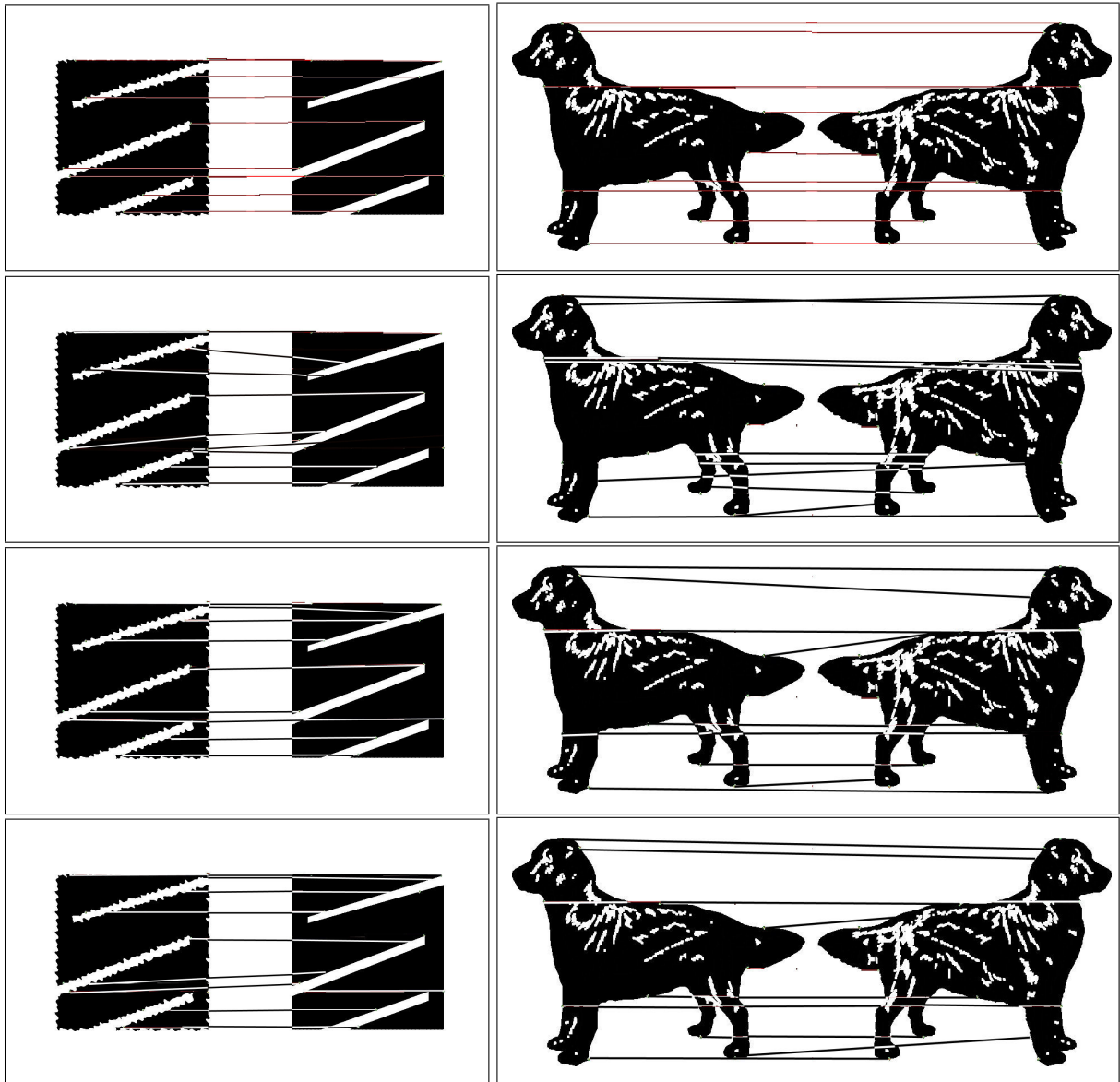


Figure 9: Example matching results for two pairs of sample shapes in the MPEG-7 dataset. From top to bottom: Results yielded by our embedding method, graduated assignment, discrete relaxation and graph seriation.

alternatives to each of these training and testing sets.

The categorisation results are shown in Table 2. In the table, we show the mean and variance for the percentage of correctly classified shapes in the dataset. Despite the basic strategy taken for the construction of our graphs, which contrasts with the specialised developments in the alternatives, our method still provides a margin of improvement with respect to the method in [?] and that in [51]. Moreover, our method delivers a lower variance, which indicates that the algorithm is more

Categorisation Method	Our Method	Shape Context [?]	Skeletal Context [51]
Classification Rate	81.57±0.79%	77.55±2.39 %	79.91±1.78%

Table 2: Shape categorisation results on the MPEG-7 dataset.

stable to training set variation than the alternatives.

5 Conclusions

We have presented a novel method to embed a graph into a metric space making use of class-specific information. To this end, we have cast the problem into a Riemannian setting, so that the embedding can be formulated as a mixture of class-specific submersions. The mixture weights are calculated using a probability density estimation and the submersions themselves are refined by iteratively minimising the differences between the graph-vertex posterior probabilities and the estimated values from the class information. The results show that the proposed method is effective in addressing the graph matching problem and can be employed to perform object categorisation and shape classification tasks. Moreover, the method is quite general and could use more sophisticated features to get further improvement on the results presented here. Other possibility is using classifier fusion methods to combine the posteriors yielded by our algorithm.

References

- [1] C. Gotsman, X. Gu, and A. Sheffer. Fundamentals of spherical parameterization for 3D meshes. *ACM Trans. on Graphics*, 22(3):358–363, 2003.
- [2] I. Borg and P. Groenen. *Modern Multidimensional Scaling, Theory and Applications*. Springer Series in Statistics. Springer, 1997.
- [3] G. Di Battista, P. Eades, R. Tamassia, and I. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall, 1998.

- [4] M. F. Demirci, A. Shokoufandeh, S. Dickinson, Y. Keselman, and L. Bretzner. Many-to-many feature matching using spherical coding of directed graphs. In *ECCV04*, pages I: 322–335, 2004.
- [5] H. Busemann. *The geometry of geodesics*. Academic Press, 1955.
- [6] A. Ranicki. *Algebraic l-theory and topological manifolds*. Cambridge University Press, 1955.
- [7] J. B. Tenenbaum, V. de Silva, and J. C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000.
- [8] S. T. Roweis and L. K. Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290:2323–2326, 2000.
- [9] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Neural Information Processing Systems*, number 14, pages 634–640, 2002.
- [10] T. Cootes, C. Taylor, D. Cooper, and J. Graham. Active shape models - their training and application. *Computer Vision and Image Understanding*, 61(1):38–59, 1995.
- [11] P. Torr and D. W. Murray. The development and comparison of robust methods for estimating the fundamental matrix. *International Journal of Computer Vision*, 24:271–300, 1997.
- [12] L. Shapiro and J. M. Brady. Feature-based correspondance - an eigenvector approach. *Image and Vision Computing*, 10:283–288, 1992.
- [13] S. Ullman. Filling in the gaps: the shape of subjective contours and a model for their generation. *Biological Cybernetics*, 25:1–6, 1976.
- [14] S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *PAMI*, 18(4):377–388, April 1996.
- [15] A. K. C. Wong and M. You. Entropy and distance of random graphs with application to structural pattern recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 7:599–609, 1985.
- [16] W. J. Christmas, J. Kittler, and M. Petrou. Structural matching in computer vision using probabilistic relaxation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(8):749–764, 1995.

- [17] 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.
- [18] T. Caetano, L. Cheng, Q. Le, and A. Smola. Learning graph matching. In *Proceedings of the 11th International Conference on Computer Vision*, pages 14–21, 2007.
- [19] Fan R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.
- [20] S. Umeyama. An eigen decomposition approach to weighted graph matching problems. *PAMI*, 10(5):695–703, September 1988.
- [21] 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.
- [22] 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.
- [23] T.M. Caelli and S. Kosinov. An eigenspace projection clustering method for inexact graph matching. *PAMI*, 26(4):515–519, April 2004.
- [24] 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.
- [25] 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.
- [26] L. G. Shapiro and R. M. Haralick. A metric for comparing relational descriptions. *IEEE Trans on Pattern Analysis and Machine Intelligence*, pages 90–94, 1985.
- [27] R. Myers, R. C. Wilson, and E. R. Hancock. Bayesian graph edit distance. *PAMI*, 22(6):628–635, June 2000.
- [28] S. V. Rice, H. Bunke, and T. A. Nartker. Classes of cost functions for string edit distance. *Algorithmica*, 18:271–280, 1997.
- [29] T. B. Sebastian, P. N. Klein, and B. B. Kimia. Shock-based indexing into large shape databases. In *European Conference on Computer Vision*, volume 3, pages 731–746, 2002.

- [30] R. Horaud and T. Skordas. Stereo correspondence through feature grouping and maximal cliques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1168–1180, 1989.
- [31] R. Levinson. Pattern associativity and the retrieval of semantic networks. *Comput. Math. Appl.*, 23:573–600, 1992.
- [32] G. Levi. A note on the derivation of maximal common subgraphs of two directed or undirected graphs. *Calcols*, 9:341–354, 1972.
- [33] J. J. McGregor. Backtrack search algorithms and the maximal common subgraph problem. *Software Practice and Experience*, 12:23–34, 1982.
- [34] B. T. Messmer and H. Bunke. A decision tree approach to graph and subgraph isomorphism detection. *Pattern Recognition*, 32(12):1979–1998, December 1999.
- [35] I. Chavel. *Riemannian Geometry: A Modern Introduction*. Cambridge University Press, 1995.
- [36] R. O. Duda and P. E. Hart. *Pattern Classification*. Wiley, 2000.
- [37] D. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11:431–441, 1963.
- [38] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 2000.
- [39] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, second edition, 1990.
- [40] P. Bremaud. *Markov Chains, Gibbs Fields, Monte Carlo Simulation and Queues*. Springer, 2001.
- [41] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*, volume 86, pages 2278–2324, 1998.
- [42] E. R. Hancock and J. V. Kittler. Discrete relaxation. *Pattern Recognition*, 23:711–733, 1990.
- [43] A. Robles-Kelly and E. R. Hancock. Graph edit distance from spectral seriation. *Trans. on Pattern Analysis and Machine Intelligence*, 27(3):365–378, 2005.
- [44] M. Tuceryan and T. Chorzempa. Relative sensitivity of a family of closest-point graphs in computer vision applications. *Pattern Recognition*, 24(5):361–373, 1991.
- [45] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, 1973.

- [46] N. Otsu. A thresholding selection method from gray-level histograms. *IEEE Trans. on Systems, Man, and Cybernetics*, 9(1):62–66, 1979.
- [47] Z-H. Zhou and J-M. Xu. On the relation between multi-instance learning and semi-supervised learning. In *Intl. Conf. Machine Learning*, pages 1167–1174, 2007.
- [48] L. Fei-Fei and P. Perona. A bayesian hierarchical model for learning natural scene categories. In *Comp. Vision and Pattern Recognition*, pages II:524–531, 2005.
- [49] A. Robles-Kelly and E. R. Hancock. An expectation-maximisation framework for segmentation and grouping. *Image and Vision Computing*, 20(9-10):725–738, 2002.
- [50] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.
- [51] J. Xie and M. Shah P. Heng. Shape matching and modeling using skeletal context. *Pattern Recognition*, 41(5):1756–1767, 2008.