Graph Matching for Object Recognition

Marcus Brubaker
mbrubake@cs.toronto.edu

18th December 2003
# Contents

1 Introduction 4

2 Bi-partite Matching: View-Based Object Recognition Using Saliency Maps 7
   2.1 The Wavelet Transform in One Dimension 7
   2.2 Detecting Saliency in Two Dimensions 8
   2.3 The Saliency Map Graph 9
   2.4 Comparing Saliency Map Graphs Using Topological Similarity 9
   2.5 Comparing Saliency Map Graphs Using Geometric Similarity 10
   2.6 Experiments and Results 10

3 Graph-Edit Distance: Recognition of Shapes by Editing Shock Graphs 12
   3.1 Shock Graphs and Shock Transitions 12
   3.2 The Edit Distance 13
   3.3 The Edit Distance Algorithm 14
   3.4 Experiments and Results 15

4 Graduated Assignment: A Graduated Assignment Algorithm for Graph Matching 16
   4.1 Graph Matching Formulation 16
   4.2 Constraint Satisfaction 17
4.3 Error Minimization .............................................. 18
4.4 The Final Algorithm ............................................. 19
4.5 Extending the Algorithm to Attributed Relational Graphs ............ 20
4.6 Experiments and Results ........................................ 20

5 Graph Eigenspaces: On the Representation and Matching of Qualitative

Shape at Multiple Scales ............................................. 22
5.1 Blobs and Ridges .................................................. 22
5.2 Feature Graphs .................................................... 24
5.3 Topological Signature Vectors .................................... 25
5.4 Geometric Information .......................................... 25
5.5 Graph Matching ................................................... 26
5.6 Experiments and Results ........................................ 26

6 Conclusions ........................................................... 28
6.1 Analysis .............................................................. 28
6.2 Future Work ........................................................ 29

Bibliography ............................................................. 30
1 Introduction

Central to the problem of object recognition is comparing, or matching, two objects. Matching has been approached many ways over the years and recently the use of graphs as a representation of objects and shape has proved itself as a practical and elegant description for the purposes of matching.

An immediate question when using graph based methods is what kind of graph. In vision, one of the first graph like representations was the medial axis (or skeleton) shape transformation which was defined by Blum in [3]. Since then, many alternative definitions of a medial axis have been put forth but are largely theoretically equivalent. Unfortunately, skeletal graphs have significant inherent instabilities in the presence of noise. As a result attempts to use skeletal graphs for matching shapes, such as in [27], have generally required significant work to deal with these issues. Recently, however, efficient methods for calculating more stable variants of the medial axis have been found such as the one in [17, 16].

A derivative of the medial axis transformation, known as a shock graph, is based on the grass-fire analogy of medial axis formation where points of the medial axis are the shocks or singularities which arise in propagating a grass-fire wave from the boundaries of the shape. The properties of these shocks were described and enumerated in [23] and shock graphs are now a popular representation of shape for object recognition.

One limitation shared by skeletons and related structures is the prerequisite of segmentation. While many methods exist for image segmentation, it remains computationally
expensive and unstable in the general case. This reality has encouraged the development of other kinds of graph based image descriptors which don’t depend on specific feature extraction or segmentation. Examples of these descriptions include using Gabor wavelet jets as in [26], a Laplacian pyramid as in [4] and a multi-scale wavelet transform as in [22] which will be covered in more depth later.

Once an underlying representation for images has been chosen, the problem becomes that of matching instances of these representations. The more general problem of matching image features has frequently been mathematically formulated as that of matching Attributed Relational Graphs (ARGs) such as in [14] with colour clusters, [15] with contour approximations and [1] with regions. In [20] arbitrary ARGs are used to develop a project transformation invariant representation to facilitate matching using only unary and binary relations which was then applied to 3D object recognition.

Many techniques exist for graph matching. The starting point for numerous popular methods is transforming the problem into that of finding a maximal clique in some derived association graph as was originally proposed in [2] and has since been used in a variety of tasks and areas. Unfortunately, the maximal clique problem is known to be \( NP \)-complete [7] in the general case (though solvable in polynomial-time for certain classes of graphs) and the standard formulation fails to preserve any kind of ordering relation inherent in the graph. While the latter of these may seem trivial, many natural representations of images and image features contain significant orderings which must be taken into account when solving the matching problem.

In this paper I will survey a variety of recent results relating to this area. In Chapter 2 work done by Ali Shokoufandeh, Ivan Marsic and Sven Dickinson is covered which uses a multi-scale wavelet transform to produce a scale-space graph representation of image feature saliency which is then matched for topological and geometric similarity using a polynomial time, greedy algorithm.

Chapter 3 covers work by Thomas Sebastian, Philip Klein and Benjamin Kimia wherein
matching is considered in terms of the edit distance between graphs.

Chapter 4 reviews work by Steven Gold and Anand Rangarajan in which graph matching is formulated as a discrete optimization problem and continuation methods are applied to efficiently and effectively find optimal (or near optimal) solutions.

Chapter 5 summarizes work by Ali Shokoufandeh, Sven Dickinson, Clas Jonsson, Lars Bretzner and Tony Lindeberg which used scale-space blob graphs and an eigenspace representation of a graph to compare graph topology.
2 Bi-partite Matching: View-Based Object Recognition Using Saliency Maps

In this paper Shokoufandeh, Marsic, and Dickinson [22] introduce a graph based representation of salient image features known as a saliency map graph or SMG. This saliency map graph is generated by performing a multi-scale wavelet transform of the image. Two algorithms are then developed. The first is a greedy algorithm for matching nodes first at the highest scale and subsequently recursively matching the subgraphs of the previously matched nodes. This algorithm is then extended to take into account the geometric relations between nodes by estimating an affine transformation between features at similar scales.

2.1 The Wavelet Transform in One Dimension

A wavelet transform is used at multiple scales. Specifically, the dyadic wavelet transform $W$ of a function $f \in L^2(\mathbb{R})$ at scale $2^j$ and at a position $k$ is defined as:

$$(Wf)(j, k) = \langle f, \psi_{j,k} \rangle = 2^{-j/2} \int_{-\infty}^{+\infty} f(x) \overline{\psi(2^{-j}x - k)} dx$$

where the overline is the complex conjugate.

For each object in the image a scale $j$ is chosen which most efficiently encodes the
salient shape of the object. This scale is roughly one octave below the scale where an object’s wavelet response closely resembles the wavelet basis function. Computationally this determined by measuring the correlation between the wavelet transform and the basis function. The scale is chosen to be the finest scale which exceeds some threshold.

This 1D wavelet transform can be applied to two dimensional images by taking cross-sections of the image at various angles and performing the 1D transformation on the individual cross-sections.

2.2 Detecting Saliency in Two Dimensions

The algorithm that Shokoufandeh et al. use to detect salient image features breaks down into four steps. First a “wavelet pyramid” is formed by evaluating the 1D wavelet transform above in 16 different orientations \( \Theta = 0, 22.5, 45, \ldots, 337.5 \) degrees at \( \ell \) different scales. Then, using a quadrature pair of analyzing wavelet filters \( \Theta(s, x, y) \) and \( H^{\Theta}(s, x, y) \) the second step consists of calculating the local energies for each scale, orientation and image location which is defined by:

\[
E(\Theta, s, x, y) = [G^{\Theta}(s, x, y)]^2 + [H^{\Theta}(s, x, y)]^2
\]

For the third step the saliency maps are computed. Saliency at a given scale and point is defined in terms of local energy and the filter kernel \( \vartheta(\Theta, x, y) \) which is the sum of the squares of the impulse responses of the two analyzing filters \( G \) and \( H \). Specifically

\[
saliency(s, x, y) = \sum_{\Theta} [E(\Theta, s, x, y) + \vartheta(\Theta, x, y)]
\]

Then in the fourth step, moving from finer to coarser scale, for every location the first saliency map is selected for which there is a peak (local maximum) at that location which exceeds a preset threshold. The result is a cluster of oriented peaks for a specific
shape from which the shapes location is calculated as the centroid of the peaks and the
saliency of the shape is computed as the sum of the saliences of the peaks neighboring the
centroid. Then, as a cleanup step, non-maximal suppression is performed to eliminate
potentially overlapping salient shapes at each scale.

2.3 The Saliency Map Graph

The saliency map can then be represented as a hierarchical directed acyclic graph. Each
node in the graph represents a salient region in the image and contains information about
the regions location, size, saliency and scale level. The edges are directed from nodes
of coarser scale to nodes of finer scale and an edge between from node \( v_i \) representing
region \( r_i \) to node \( v_j \) representing region \( r_j \) exists if and only if region \( r_j \) is completely
contained with in \( r_i \).

2.4 Comparing Saliency Map Graphs Using Topological
Similarity

The basic algorithm presented in [22] runs in phases. Each phase corresponds to matching
the nodes of two saliency map graphs at a given level. Specifically, at a given phase \( i \)
when matching two SMGs \( G_1 = (V_1, E_1) \) and \( G_2 = (V_2, E_2) \) the method chooses nodes
\( A_i \) and \( B_i \) which are nodes at level \( i \) in \( G_1 \) and \( G_2 \) respectively. It then constructs a
weighted, bipartite graph \( G(A_i \cup B_i, E) \) with edge weights \( w(u, v) = |s(v) - s(u)|^3 \) for
each edge \( (u, v) \). Further, an edge \( (v, u) \) is in \( G \) if and only if either both \( u \) and \( v \)
don’t have parents or if they both share at least one matched parent. Then a maximum
cardinality, minimum weight matching \( M_i \) is found in \( G \).

The union of these matchings \( M \) constitutes a topological matching for graphs \( G_1 \)
and \( G_2 \). The quality of the match is specified by an error function \( \varepsilon \) whose detailed

\[^1\text{Where } s(u) \text{ is the saliency of node } u.\]
formulation will be omitted.

2.5 Comparing Saliency Map Graphs Using Geometric Similarity

The method above for matching is then extended to include the geometric information which is encoded in the SMG. This is done by making two modifications. First, before matching begins three nodes (or regions) are selected from each SMG which are used to calculate an affine transformation which best aligns the two graphs. \((r_1, r_2, r_3)\) are chosen from \(G_1\) and \((r'_1, r'_2, r'_3)\) are chosen from \(G_2\) such that the mapping \((r_i \rightarrow r'_i)\) satisfies the following conditions:

- \(r_i\) and \(r'_i\) have the same level in the SMG for all \(i\)
- \((r_i, r_j) \in E_1\) if and only if \((r'_i, r'_j) \in E_2\) for all \(i\) and \(j\)

Once these regions have been selected, a system constructed from the homogeneous 3D coordinates of the selected region centres is solved for an affine transform \(A\) which is then applied to all regions in \(G_1\) to produce the modified graph \(G'_1\).

Once this is done, the matching occurs as before between \(G'_1\) and \(G_2\) except that the weighting on an edge in the constructed bipartite graph is now defined as \(w(u, v) = d(u, v)^2\). The error of this resulting match is the defined as the sum of the distances between matched nodes, weighted by the nodes difference in saliency.

2.6 Experiments and Results

In testing the performance of their representation Shokoufandeh, Marsic, and Dickinson ran their method on the Columbia University COIL image database. They tested using both strict topological similarity as well as geometric similarities. The COIL database

\(^2\)Where \(d(u, v)\) is the Euclidean distance between the region centres.
contains 72 systematically selected views each of 20 different objects. The testing consisted of removing every other view for each object and matching the removed views against the remaining views. The match was selected which minimized the respective error function $\varepsilon$ and a match was considered a hit if the selected view was one of the two neighboring views.

For simple topological similarity, the method produced a hit 89% of the time. 8.4% of the time it missed but selected a view of the correct object. The remaining 2.6% of the time the wrong object was selected. Using geometric similarity, the results were considerably better, producing hits 96.6% of the time, missing but selecting the correct object 2.9% of the time and missing altogether only 0.5% of the time.

Some limited experimentation was done to analyze the performance of the method in the presence of occlusion with respect to viewpoint invariance, but nothing systematic.
3 Graph-Edit Distance: Recognition of Shapes by Editing Shock Graphs

In “Recognition of Shapes by Editing Shock Graphs” Sebastian, Klein, and Kimia [19] define the similarity of two shapes in terms of the cost of manipulating one shock graph into another. In order to reduce the number of potential manipulations they assume that equivalent shock graph topologies correspond to equivalent shapes. Further, they discretize the space of possible deformations to those corresponding to shock transitions which are points along a continues shape deformation path where the shock graph topology of a shape changes. These ideas are put together and a polynomial time algorithm for finding the globally optimal deformation sequence is tested.

3.1 Shock Graphs and Shock Transitions

As noted earlier in the introduction, the notion of shocks are based on the grass-fire analogy for medial axis formation. In this analogy the shape is imagined as a patch of grass and a fire is set simultaneously at its boundary. The medial axis is then the points or shocks where the fire self-intersects. Associated with each shock is a time of formation which is analogous to the values in the distance map formulation of the medial axis, that is, the time of formation of a shock point is a function of the shortest distance between that point and boundary of the shape.

Shock points have been categorized into four types and converted into rooted trees
(such as in [24]) and a complete grammar has been defined which characterizes all such
rooted trees in [23, 24]. Connected, or neighboring, shocks are clustered together by
type and become the nodes of a tree with an edge \((u, v)\) from node \(u\) to node \(v\) existing
if and only if the range of formation times for node \(u\) are later than those for node \(v\)
and the shocks associated with nodes \(u\) and \(v\) are connected. The effect of continuous
shape deformation on the discrete shock graph has been enumerated as six kinds of shock
transitions in [8].

3.2 The Edit Distance

Intuitively, edit distance is defined as the cost of editing one object to another. This
notion was originally used in string comparisons but has since been used elsewhere.

The first issue with edit distances is defining the family of possible edit operations.
Sebastian et al. define four categories of shock graph edit operations. The first three
correspond to topological edits based on the six types of shock transitions enumerated
in [8]. The fourth edit operation is a relabeling, or deforming, of the attributes of shock
graphs with identical topology.

Once the edit operations have been enumerated and explicated it is necessary for costs
to be associated with them. Properly defining the costs associated with edit operations
is critical to having meaningful edit distances. Sebastian et al. begin this process of
cost definition by noting that each topological deformation occurs at the boundary of
a sequence of non-topological deformations. Hence, to get a consistent set of costs for
all edit operations they begin by defining the cost of a non-topological deformation and
define the topological edit costs as the limiting value of the non-topological deformation
along the appropriate path.

A non-topological deformation corresponds to changing the times of formation of the
shocks in a node \(u\). Associated with each node \(u\) are two boundary segments of the
original shape. The cost of the deformation is then defined in terms of aligning and
deforming the associated boundary segments into the desired form. Specifically, as a base case, when matching single infinitesimal segments let $ds$ be the length of segment $s$, $d\hat{s}$ be the length of segment $\hat{s}$ and $d\theta$ and $d\hat{\theta}$ be the curvature of segments $s$ and $\hat{s}$ respectively. Then the cost of matching these two segments is defined to be $\mu(s, \hat{s}) = |d\hat{s} - ds| + R \left| d\hat{\theta} - d\theta \right|$ for some constant $R$. In the more general case, let $S$ and $\hat{S}$ be two clusters of shocks and let $\alpha$ be an alignment curve used to mediate the match as described in [18]. Then the cost of the deformation from $S$ to $\hat{S}$ is defined as

$$d(S, \hat{S}) = \min_{\alpha} \mu(S, \hat{S}; \alpha)$$

where $\mu(S, \hat{S}; \alpha)$ is roughly the integral over $S$ and $\hat{S}$ of $\mu(s, \hat{s})$ using the alignment curve $\alpha$.

With this now well defined the definition of the topological edit operations are immediately defined in terms of the limit of $d(S, \hat{S})$ and Sebastian et al. omit a detailed discussion of these costs.

### 3.3 The Edit Distance Algorithm

With any edit distance based algorithm the most important issue is calculating (or at least accurately estimating) the edit distance efficiently. In this work Sebastian et al. use a notion of “simplification” in order to limit the number of possible single edits available at any point. Specifically, to match two shock graphs corresponding to shapes $A$ and $B$, at each step both trees are simplified by an edit operation until both trees reach a common, simplified shape $C$. The edit path from $A$ to $B$ is then simply the combination of these two paths. The algorithm used runs in polynomial time and was developed in detail in [10, 11].
3.4 Experiments and Results

In order to demonstrate their methods matching performance, Sebastian et al. constructed a database of 9 categories such as fish, rabbit, airplane, etc. For each category the database consisted of 11 shape variations such as occlusion, articulation and other visual transformations. Then each of the 99 shapes in the database were matched against the other 98 and the results were sorted by edit distance. Of these matches the three best matches were always from the correct category. The next three were 99% of the time the correct category and the last four were 97%, 96%, 95% and 87% of the time in the correct category.

Another database was also used which consisted of 18 categories and 12 shapes per category. The same testing method was run. As percentages, the first eleven matches were the correct category (100, 100, 100, 100, 100, 100, 100, 99, 99, 97, 98, 95). In other words, for every object the top six matches were always the correct category.
4 Graduated Assignment: A Graduated Assignment Algorithm for Graph Matching

In this work Gold and Rangarajan [9] formulate the problem of matching two weighted graphs as the minimization of a cost function over the space of match matrices. Unfortunately, this formulation is intractable for general cost functions. As a result Gold and Rangarajan construct an algorithm to efficiently approximate the discrete optimization problem using graduated assignment. The algorithm is then demonstrated on a few image based graphs and on randomly generated and permuted graphs.

4.1 Graph Matching Formulation

Assume we are matching two weighted graphs $G_1 = (V_1, E_1, W_1)$ and $G_2 = (V_2, E_2, W_2)$ where $E$ are the edges and $W$ maps $E$ to $\mathbb{R}$. Define the match-cost matrix $C$ in the following manner:

$$C_{i,j} = \begin{cases} 
0 & \text{if } (a, b) \not\in E_1 \text{ or } (i, j) \not\in E_2 \\
 c(W_1(a, b), W_2(i, j)) & \text{otherwise}
\end{cases}$$

for some match-cost function $c(\cdot, \cdot)$. Intuitively, this matrix defines the cost of matching edge $(a, b)$ in graph $G_1$ to edge $(i, j)$ in $G_2$. The match matrix between $G_1$ and $G_2$ is
then defined as the following:

\[ M_{ai} = \begin{cases} 
1 & \text{if node } a \text{ in } G_1 \text{ corresponds to node } i \text{ in } G_2 \\
0 & \text{otherwise}
\end{cases} \]

Finally then, the match error function is:

\[ E(M) = -\frac{1}{2} \sum_a \sum_i \sum_b \sum_j M_{ai} M_{bj} C_{aibj} \]

There are several properties of this match matrix which will become important. Specifically, \( M_{ai} \in \{0,1\} \) and all the sums of the rows and columns are less than or equal to one. In other words, this dictates that each node can be matched to at most one node.

The match-cost function chosen was:

\[ c(w_1, w_2) = 1 - 3|w_1 - w_2| \]

Looking ahead a bit, this was chosen so that the expected value would be zero if the weights were chosen from a uniform distribution over the unit interval.

### 4.2 Constraint Satisfaction

The constraints on the match matrix are a significant issue which must be addressed in order to find a good, approximate solution to the problem. As noted above, the rows and columns must sum to a number less than or equal to 1. The first step in handling these problems is removing the inequality. This is done by adding an extra row and column to the match matrix which contains the “slack” of the match. By doing this the rows and columns now must sum to exactly 1.

To actually handle these constraints algorithmically, Gold and Rangarajan first deal with the case of only one constraint. In this case assume we have a row or column of
numbers \{X_i\} where \( X_i \in \mathbb{R} \). For the match matrix then, we want \( \{m_i\} \) where \( \sum m_i = 1 \) and \( m_i \in \{0, 1\} \). To that end, the final goal is:

\[
m_j = \begin{cases} 
1 & \text{if } X_j \text{ is the maximum element of the set } \{X_i\} \\
0 & \text{otherwise}
\end{cases}
\]

In order to make this discrete problem tractable, it is reformulated as a continuous problem using continuation methods. Continuation methods is characterized by minimizing a series of functions which are indexed by some control parameter, say \( \beta \). As \( \beta \to \infty \), the continuous solution converges to the desired discrete solution. In this work, Gold and Rangarajan select the softmax function:

\[
m_j = \frac{\exp(\beta X_j)}{\sum_i \exp(\beta X_i)}
\]

To then extend this work to satisfy the two-way constraint, they cite Sinkhorn [25], wherein it is proved that any square matrix whose elements are all positive will converge to a doubly stochastic matrix simply by iteratively normalizing the rows and columns.

### 4.3 Error Minimization

In order to find an estimate to minimize the match-error function Gold and Rangarajan examine the first order Taylor expansion of the error functions. Specifically:

\[
E(M) = -\frac{1}{2} \sum_a \sum_i \sum_b \sum_j M_{ai} M_{bj} C_{aibi} \\
\approx -\frac{1}{2} \sum_a \sum_i \sum_b \sum_j M_{ai}^0 M_{bj}^0 C_{aibi} - \sum_a \sum_i Q_{ai}^0 (M_{ai} - M_{ai}^0)
\]
where $M^0$ is some initial guess for the match matrix and

$$Q_{ai} = -\frac{\partial E_{wg}}{\partial M_{ai}} = + \sum_b \sum_j M_{bj} C_{aibj}$$

Then, to minimize $E$ we simply need to maximize

$$\sum_a \sum_i = Q_{ai} M_{ai}$$

which is an assignment problem from combinatorial optimization. This problem has been solved using the method described in the previous section in [12] and that is how it is used here.

### 4.4 The Final Algorithm

The final algorithm starts with an initial value for $\beta$ and an initial guess $M$ for $M$. Gold and Rangarajan using $M_{ai} = 1 + \epsilon$ where $\epsilon$ is machine precision. Then, for a fixed number of monotonically increasing values for $\beta$ do the following:

1. For a fixed number of iterations or until $M$ converges do the following:
   
   a) Calculate $Q_{ai} = -\frac{\partial E_{wg}}{\partial M_{ai}}$ with the current $M_{ai}$.

   b) Calculate $M_{ai} = \exp(\beta Q_{ai})$.

   c) Iteratively renormalize the rows and columns of $M_{ai}$ using softmax as described above until either a fixed number of iterations or until $M$ converges.

Once the above has been done for the prescribed values of $\beta$ a heuristic cleanup is performed. In this work, this cleanup consisted of setting the maximum element of each column to 1 and the remaining elements to 0.
4.5 Extending the Algorithm to Attributed Relational Graphs

The preceding algorithm was designed and works for weighted graphs but can also be extended to support general attributed relational graphs. This is done by redefining the error function in a way similar to the following:

\[
E_{arg}(M) = -\frac{1}{2} \sum_a \sum_i \sum_b \sum_j M_{ai} M_{bj} C^{(2)}_{abij} + \alpha \sum_a \sum_i M_{ai} C^{(1)}_{ai}
\]

where \( C^{(2)}_{abij} \) is defined identically to \( C_{abij} \) above, \( C^{(1)}_{ai} = c(a,i) \) is the cost of matching the attributes related with nodes \( a \) and \( i \) and \( \alpha \) is a constant parameter which controls the impact of node attributes on the matching.

4.6 Experiments and Results

Three basic experiments were done using this method. The first was a repeat of an experiment done by Eshera and Fu [6] where Gold and Rangarajan reused the ARG data from [6] in their algorithm. The second experiment was running on a set of hand designed ARGs based on images. The third experiment was run using randomly generated graphs of different types which were tested under various noise conditions. As a control for the third experiment, simple probabilistic relaxation was run on randomly generated graphs as well.

The first experiment ran consisted of matching an ARG from an image of a wrench with an ARG from a cluttered image with a number of overlapping objects (including the wrench). With this experiment, their algorithm found the correct match between the two ARGs. The second experiment, similar in nature, consisted of matching two hand-crafted ARGs. The first based off of an image of a coffee mug and the second, the same coffee mug (at a different scale) with a coffee pot added to the scene. Again, their
algorithm found the correct assignment between the two ARGs.

The third experiment was considerably more involved and consisted of matching randomly generated graphs. Specifically, a 100 node graph was randomly generated of a specific type. Then the graph was randomly permuted in some way and the original graph was matched to the permuted graph. Three types of graphs were used: 1) simple zero-one connected graphs, 2) weighted graphs and 3) ARGs. The permutations applied included removing links and nodes and adding random noise to weights and attributes. Obviously, the combination of these types and permutations, when run over varying levels of noise, resulted in numerous experiments, we will summarize a selection of them.

The algorithm performed well in the presence of noisy weights. With 40% of the nodes removed and relatively sparse connectivity (10% and 15%) in weighted graphs the number of incorrect node matches was insignificant (essentially 0) when the weights on the graphs were unperturbed. The percentage of incorrect matches remained roughly under 10% for up to 5% noise and under 20% for up to around 8% noise.

When matching ARGs, the results were even better. With as many as 80% of the nodes deleted and up to 10% noise in the weights and attributes the algorithm still only mismatched less than 5% of the nodes. This was for nodes having both 3 and 5 binary attributes.

However, when links (as well as nodes) were randomly added, removed or mislabeled, the performance became significantly worse for both ARGs and weighted graphs, with this permutation having considerably more impact on the weighted graphs than on the ARGs.

Also, as mentioned above they ran standard probabilistic relaxation in similar situations to compare with the algorithm. As might be expected, in all the tests they ran, their algorithm considerably outperformed probabilistic relaxation.
5 Graph Eigenspaces: On the Representation and Matching of Qualitative Shape at Multiple Scales

In their work “On the Representation and Matching of Qualitative Shape at Multiple Scales” Shokoufandeh, Dickinson, Jonsson, Bretzner, and Lindeberg [21] use an image feature representation which is similar the one used in [22] which was reviewed in Chapter 2. Specifically, image features are defined in terms of blobs and ridges at multiple scales. These features are ordered by scale and orientation and placed in a directed, acyclic graph. The eigenvalues of the graphs adjacency matrix are used as a permutation invariant representation of graph topology. This representation is showed to be relatively stable under minor changes in graph structure. The rooted, undirected tree matching algorithm first explicated in [24] is then extended to match directed acyclic graphs in general. The resulting method is applied to gesture and face recognition.

5.1 Blobs and Ridges

The image feature representation used is based on convolution of the image signal $f$ by a Gaussian kernel $g(\cdot; t)$ with variance $t$. Specifically, blobs are detected as scale-space
local maxima over the normalized Laplacian operator,

\[ \nabla^2_{\text{norm}} L = t(L_{xx} + L_{yy}) \]

where

\[ L(\cdot; t) = g(\cdot; t) * f(\cdot) \]

Intuitively, blobs are roughly circular regions of contrast in the image.

Similarly, ridges are elongated regions of contrast which are detected via a multi-scale ridge detector

\[ R_{\text{norm}} L = t^{3/2}((L_{xx} - L_{yy})^2 + 4L^2_{xy}) \]

Associated with each region is a matrix \( \Sigma \) which is defined as follows:

\[
\Sigma = \int_{\eta \in \mathbb{R}} \begin{pmatrix} L_x^2 & L_x L_y \\ L_x L_y & L_y^2 \end{pmatrix} g(\eta; t_{int}) d\eta
\]

for some scale \( t_{int} \) which is proportionally related to the scale of the detected feature. This matrix provides important information about the detected feature. Specifically, 1) orientation and 2) anisotropy. If \( \lambda_1 \) and \( \lambda_2 \) are the eigenvalues of \( \Sigma \) and \( \lambda_1 > \lambda_2 \) then the orientation is the direction of the eigenvector corresponding to \( \lambda_1 \). The anisotropy is defined as \( 1 - \lambda_1/\lambda_2 \).

This process is likely to result in many overlapping and redundant features. To handle this a measure of feature similarity is used and two close features for which similarity surpasses a certain threshold are merged. Further, multiple ridges may be detected corresponding to a single, longer ridge. A process of ridge linking detects and links such ridges into single features.

In their work Shokoufandeh, Dickinson, Jonsson, Bretzner, and Lindeberg perform this detection process on four image signals corresponding to the R, G and B channels of the
image as well as an intensity image. The results are compositd such that reoccurring features are given greater significance.

5.2 Feature Graphs

Once the features have been detected they are used to construct a directed acyclic “blob” graph. Each feature is represented as a node in the graph. The graph construction process works as follows:

1. Select a feature at the coarsest scale to use as a root node.

2. Features that overlap the root at a finer scale are then added as children of the root.

3. Each child feature node is then used as a root node and repeat 2 and 3 until there are no more overlapping features.

4. If there are features still unassigned select a new root node at the coarsest scale of remaining features and return to step 2.

Once this is completed sibling edges are added between all nodes which share a common parent. While these edges do not end up in the structural description of the graph, thus ensuring that it is indeed a DAG, they are used to encode relational information between sibling nodes which is then encoded as a histogram in the nodes themselves.

Specifically, associated with each edge (both sibling and parental) is the following information:

- Two different measures of inter-feature distance is encoded, both scale independent,
- Relative orientation,
- Relative bearing,
• Scale ratio, and

• Edge type.

5.3 Topological Signature Vectors

In [24] the idea of using topological signature vectors (TSVs) was introduced as a representation of graph structure. Briefly we will define the TSV of a node $V$ in a DAG. Let $C_1, \cdots, C_{\delta(V)}$ be the children of $V$ where $\delta(V)$ is the branching factor at node $V$. Let $A(C_i)$ be the adjacency matrix associated with the subgraph rooted at node $C_i$. For each $C_i$ compute the magnitudes of the eigenvalues of $A(C_i)$ and sum the $\delta(C_i) - 1$ largest of them. Let this sum be $S_i$. Sort the $S_i$ such that $S_1 \leq S_2 \leq \cdots \leq S_{\delta(V)}$. The TSV $T(V)$ is then a $\Delta$ dimensional vector where $\Delta$ is the maximum branching factor of the graph with the first $\delta(V)$ entries $S_1, \cdots, S_{\delta(V)}$ and the remaining elements set to zero.

5.4 Geometric Information

As noted above when discussing the construction of the DAG, the sibling edges were there for informational purposes and would not be included in the final structural representation of the DAG. The information contained within them is associated with each node in the following way. Assume we’re talking about a DAG $G = (V, E)$. For every node $u \in V$ and relationship $p$ define the set $P(u, p) = \{p(u, u') | (u, u') \in E\}$. Then we define a translatable Hausdorff distance metric $d_{p,t}(u, u')$ between $P(u, p)$ and $P(u', p)$ with respect to relation $p$

$$d_{p,t}(u, u') = \max \left\{ \max_{a \in P(u, p)} \min_{b \in P(u', p)} |a + t - b| \right\}$$
for a translation factor $t$ which serves to normalize the features. We then use this to define

$$w_p(u, u') = \min_{t \in \mathbb{R}} d_{p,t}(u, u')$$

which is the geometric similarity between nodes $u$ and $u'$ with respect to feature $p$. This is then used to define the general geometric similarity between nodes $u$ and $u'$

$$W(u, u') = e^{-\sum_{p \in \mathcal{R}} w_p(u, u')}$$

where $\mathcal{R}$ is the set of all such relationships.

### 5.5 Graph Matching

The matching algorithm used by Shokoufandeh et al. works by computing the maximum cardinality, maximum weight bipartite matching between between the nodes of two graphs. The weighting is a function of topological similarity (the difference between the associated TSVs) and geometrical similarity as defined above.

From this matching, the maximum weight match is greedily selected and added to the solution set. Then the subgraphs rooted at the selected nodes are recursively matched in the same manner. As each node is matched, it is marked so as to avoid duplication which could occur if a node had multiple parents. This recursive process is done repeatedly on the remaining unmatched nodes until all the nodes of one graph have been matched.

### 5.6 Experiments and Results

No systematic experimentation is reported in the paper but the methods described are applied to two problem domains: gesture recognition and face recognition.

In the gesture recognition demonstration there were five different exemplars for each of seven different gesture classes. Within each class the exemplar whose sum distance to
all other exemplars in the class was minimum was chosen as a model for the class. The remaining exemplars were used as a query. The resulting matching process yielded 100% recognition of gesture class over the 28 trials (four queries for each of the seven classes). A further gesture recognition experiment was run which matched five queries with complex backgrounds from each of the seven classes against the models selected for the previous experiment. The recognition rate in this case was 57% (20/35). Of the 15 which were misclassified, 10 were second closest to the correct class. The authors speculated that more complex queries tended to favour more complex models whose additional fingers could be used to explain the additional complexity in the query. They note that they are currently exploring using a stronger weighting on the geometric similarity term to improve the matching in these cases.

In the last experiment reported on in the paper, five cluttered scenes were matched against a model of a face. In each case the portions of the scenes graph matched to the models graph were the face in the scene.
6 Conclusions

6.1 Analysis

One common limitation has shown itself among each of the papers reviewed here and others on the subject of object recognition in general and graph-based object recognition in particular: there are no common benchmark datasets or baseline methods by which different methods can be compared in a quantitative manner. The machine learning field has developed several of these over the years (using methods such as naive Bayes and K-nearest neighbors as baselines) but the vision community has yet to start using any baseline methods for comparison nor a standard dataset and performance measure.

The approach taken by Shokoufandeh et al. as described in Section 5.6 is common, where small compartmentalized tests are run to demonstrate specific features of a method (such as its performance in the presence of background noise or occlusion). While the randomized experiments run by Gold and Rangarajan which were described in Section 4.6 were certainly exhaustive in nature, their relevance to real-world graphs and their associated noise processes is certainly not clear.

There has been much good work done in object recognition recently, however the question which cannot be well answered yet is: how much better are we doing? We are almost certainly doing well, but how well? With many different methods now available for solving object recognition problems within various contexts which all work reasonably efficiently and seem to behave well it is time that the community begins to work on standardizing datasets and performance metrics and selecting methods which can serve
as baselines.

6.2 Future Work

This paper has covered several areas with respect to graph matching in object recognition but the research being done is still far more diverse than the four papers covered here and there is far more work to be done.

For one, each of the papers reviewed here (and most of the related papers that were referenced) are restricted one-to-one matching of graphs. This restriction is quite limiting in the face of noise, 3D transformations and other errors where a single node in one graph may correspond to several in the other. Unfortunately, the problem of many-to-many graph matching is much harder and few algorithms for the task exist which are both robust and efficient. Some exceptional work on an efficient method for many-to-many matching was recently done by Demirci et al. in [5] which seems very promising.

Another issue not covered here which remains open is that of indexing. Graph indexing was originally intended to be covered but a lack of space has forced its omission. However, the topological signature vector described in Section 5.3 has been used as an indexing key to reduce the space of potential matches such as in [24] and [13]. This has shown itself to be a powerful method for handling large databases of graphs but is certainly not optimal, as it only prunes based on topological similarity.
Bibliography


