Structural Learning from Iconic Representations

Herman M. Gomes* and Robert B. Fisher
Division of Informatics, Edinburgh University
5 Forrest Hill, Edinburgh EH8 9SH, Scotland UK,
{hermang,rbf}@dai.ed.ac.uk

Abstract

This paper addresses the important problem of how to learn geometric relationships from sets of iconic (2-D) models obtained from a sequence of images. It assumes a vision system that operates by foveating at interesting regions in a scene, extracting a number of raw primal sketch-like image descriptions, and matching new regions to previously seen ones. A solution to the structure learning problem is presented in terms of a graph-based representation and algorithm. Vertices represent instances of an image neighbourhood found in the scenes. An edge represents a relationship between two neighbourhoods. Intra and inter model relationships are inferred by means of the cliques found in the graph, which leads to rigid geometric models inferred from the image evidence.

1 Introduction

Within the context of visual learning, our ultimate goal is to design a vision system that is capable of automatically learning objects or parts of objects and their relationships from generic scenes. But this is not an easy task in a completely autonomous context, in which there is no one to define the appropriate training sets with its objects already segmented, normalised and separated into classes. In order to deliver objects in such way, a system has to somehow deduce the object's shape, and its position, scale and orientation in the scene.

Generally, object recognition research falls into three main categories: (1) geometric, symbolic, or structure based recognition; (2) property, vector or feature based recognition; and (3) iconic (image) based recognition. Most of the research found in the first category is related to 3-D object recognition systems and usually involves either volumetric relationships [1] or surface relationships [2]. Relational or graph matching [13, 8] are common techniques used in this area to do the matching between two relational or graph-based descriptions. The second category presents a wider range of techniques varying from the use of specific feature vectors, multiple filtering to global descriptors for shape, texture and colour, or a combination of techniques [10, 12]. This kind of approach is popular amongst applications involving image database indexing [9]. Finally, the third category is characterised by the direct use of images. In this case, the most popular technique to recognise objects is template matching. But, when

*Supported by CNPq, Brazil. On leave from DSC/COPIN/UFPB - Federal University of Paraíba, Campina Grande PB, Brazil, hmg@dsc.ufpb.br

using the traditional sensor architecture, the number of pixels involved can be too high to allow for a more elaborate computation. An alternative is to use a log-polar representation [11] which requires less pixels to represent an image once it is space variant.

Within this context, an iconic vision system based on primal sketch features extracted from a log-polar representation was developed [6, 3]. In this system, iconic models are represented by geometric relations, which are in turn used during recognition to strengthen the evidence that a particular object model has been found based on other nearby matches. But the training sets to build the models and the model relationships themselves had to be defined manually. The work presented in this paper fits in the category of iconic based recognition using geometric relationships and expands the work described in [6, 3, 5]. It addresses the important question of whether or not is possible to to learn rigid geometric models from 2-D image evidence (iconic models) acquired from a sequence of scenes. An affirmative answer to this question is given.

We assume that a model consists of 2-D representations learnt from unsegmented and cluttered scenes by means of an iconic vision system which is inspired by some of the mechanisms found in the mammalian visual system: (a) *Foveated vision*: the input light is processed through a set of overlapping receptive fields (resembling the human retina) which produces an image smaller in size but retaining high resolution in the middle; (b) *Visual attention*: fixating the retina at interesting regions of a scene prevents having to process the entire scene at once, and, provided that an appropriate attention mechanism is defined, the fixation points can be seen as places where object features (or components) are most likely to be found; (c) *Primal sketch*: it is hypothesised that primal sketch features, like *edges*, *bars*, *blobs* and *ends* [7] are used by humans as more compact and intelligible representations for image data and also as cues for an attention mechanism. The following sections explain how to combine the above mechanisms and present an algorithm to solve the structure learning problem.

2 Learning object feature models

This section explains how the process of learning of object features or primitive models is implemented. The algorithm for learning relationships between these object features, described in the following section, relies upon certain aspects of the above solution.

We assume a vision system architecture which is based on an existing system described in [6], see Fig. 1. The figure shows only four main modules that are directly related to this paper. Module (a) is responsible for converting input pixels into a retina-like (log-polar) image representation, which in turn is used by module (b) to generate primal sketch features at a number of orientations and contrasts through a neural network [5]. During scene examination an attention mechanism (c) continuously updates a map which weights points of interest in the scenes based upon the primal sketch features and colour information. The foveation area is smaller than the scenes so that just a smaller section of the input image is analysed at a given time. Finally, module (d), described in this section, clusters primal sketch planes (representing primitive objects) into model classes. It also stores information about the scale, position, orientation and similarity between the clustered objects in order to allow for a subsequent examination of the possible relationships between primitive models to form larger structured models. When these relationships are identified, they can in turn be used to improve the attention and matching processes.

The position of the retina, and any underlying local object feature, is obtained by simply

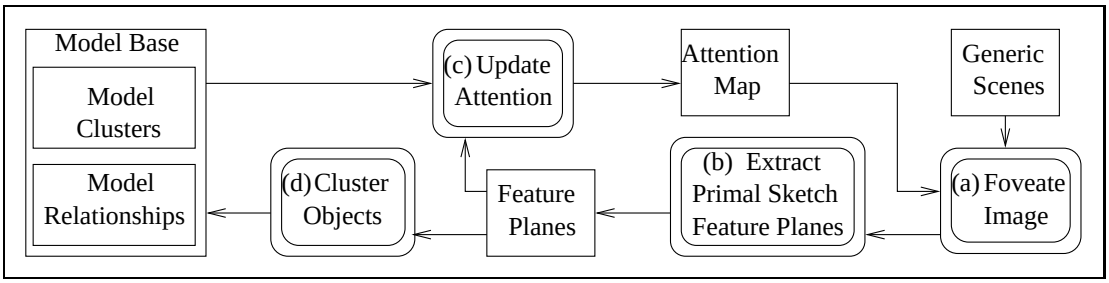

Figure 1: System's main modules.

looking at the interest map. The log-polar map implemented in our retinal representation helps the process of figuring out the relative scale and orientations of an object feature with regard to another previously stored cluster. This is possible by translating the log-polar map in both radial and angular directions to inexpensively transform the features into a number of new orientations and scales, which are then matched against already learnt occurrences of feature instances for all the existing feature models. The best match will possibly be with the model the feature is represented by. Then the feature together with its position, relative orientation and scale are stored. Algorithm 1 gives a straightforward solution to clustering. Without loss of generality this can be used as a prototype for designing more efficient algorithms.

Algorithm 1: Clustering object features
for each scene image S^i **do**
  for each foveation point $P^{i,j}$ on the scene **do**
    obtain the object feature $f^{i,j}$ at position $P^{i,j}$
    if the model base is empty, **then create** a new model and **store** $f^{i,j}$ on it.
    else
      generate a set of scaled and rotated versions $f'^{i,j}$ of $f^{i,j}$
      find the model F_t that gives the highest average similarity score $\bar{C}$
      between its internal object features $f^{k,l}$ and one of the $f'^{i,j}$ variations.
      if $\bar{C} >$ threshold **then**
        store $f^{i,j}$ in F_t.
        store the similarity scores $Sm(f^{i,j}, f^{k,l})$, the relative scales $rS(f^{i,j}, f^{k,l})$
        and the relative orientations $rO(f^{i,j}, f^{k,l})$ $\forall f^{k,l} \in F_t$.
      else create a new model and **store** $f^{i,j}$ on it.

3 Learning relationships via a representational graph

In this section we explain how geometric relationships between object feature models can be found. A general principle adopted here is that the recognition of consistent geometric relations allows the inference of larger structural object models. We assume that objects can only have 2-D rigid body transformation. The approach adopted to solve the structure learning problem was to build a graph-based representation. Vertices are defined as the Cartesian product of the sets of object feature instances of a same model class found in each of the images. Vertices are ranked according to a function of the similarities between their

component object instances. An edge in the graph represents a hypothesis for the relationship between the features described by the two connecting vertices and is valued according to the compatibility between these two vertices. The problem is reduced to finding cliques of maximum scores within this graph.

3.1 Vertices

Each object feature model F_t, created by the algorithm described in Section 2, contains a set of instances $f^{i,j}$ found in each scene image S^i. An initial problem when designing an algorithm to learn relationships from these sets was to deal with the occurrence of multiple unrelated model instances of a same class in the scenes. In other words, how to account for the combinations of instances that appear at consistent positions and orientations and separate them from those who don't? We solve this problem by taking the complete combinatorial set of instances. Each N-tuple from this set will be a vertex in our graph. The set of all vertices V_t of a given feature type is the Cartesian product between the sets of instances f_t^i of type t found in scene i (Eq. (1)).

$$V_t = f_t^1 \times f_t^2 \times \ldots \times f_t^N \tag{1}$$

where N is the total number of scene images analysed.

Equation (1) can be expanded as $V_t = \{v_1, v_2, \ldots, v_M\}$ where M is the total number of vertices. Each vertex v_r can be expanded as $v_r = (f_t^{1,j}, \ldots, f_t^{N,l})$, where $j \in \{1, \ldots, \#(f_t^1)\}$ and $l \in \{1, \ldots, \#(f_t^N)\}$. As we will not need the positions j, l of the instance within an image, for the sake of simplicity, we are going to remove these indexes throughout this section, unless otherwise stated. Also, we are going to drop the type t for the feature model class, as from now on they will be distinguished by the letter representing the instance itself.

In order to cope with the possibility of model instances being missing in some images, as the attention mechanism might fail searching at some locations, occlusion might have happened and so on, we introduce a * (wild-card) model instance, which is added to the sets of instances f^i found in scene i before computing the combinatorial sets that define the vertices.

Vertex ranking. If the similarity scores between pairs of vertex elements $Sm(f^i, f^j)$ are thought as the probability of those elements belonging to a same feature class, then a natural way of defining the rank of a vertex is by multiplying all the similarity scores (Eq. (2)). As a result, for a vertex to be strong all of its elements have to be very similar to each other.

$$Rank(v_r) = \prod_{i<j}^{N} Sm(f^i, f^j) \tag{2}$$

where N is the number of feature instances in the vertex (N is also the number of images analysed), Sm is the similarity function between two vertices f^i, f^j (this function is obtained via the Algorithm 1, and produces values within the range $[0, 1]$). We assume that the similarity between the wild-card and any other feature instance is one: $Sm(*, *) = Sm(*, f^j) = Sm(f^i, *) = 1 \ \forall i, j$.

Vertex pruning. One side effect of the addition of the wild-card instances is that there will be now a number of vertices with many *'s when compared to the number of real object features, which can cause relationships being learned between loose features, or vertices that do not represent any plausible real objects. To reduce the number of this kind of vertices, we allow only K *'s per node during the node creation process, where $K << N$ (N is the total number of images). Limiting K also reduces the combinatorial explosion of vertices. For simplicity, we have chosen $K = 1$ for the case study developed in this paper.

3.2 Edges

An edge $e = (a, b)$ connects two compatible vertices a and b in the graph. The vertices $a = (a^1, \ldots, a^N)$ and $b = (b^1, \ldots, b^N)$ are compatible if for each pair of feature instances in different images (a^i, a^j) and found in the first vertex, which are related by a given scale and orientation $R = (rS(a^i, a^j),\ rO(a^i, a^j))$, the corresponding pair (b^i, b^j) in the second vertex has its components related through a similar relative scale and orientation. Moreover, each pair of feature instance coordinates (P_{a^i}, P_{b^i}) and (P_{a^j}, P_{b^j}) taken from the same vertex positions will roughly define a unique vector angle A and length D ($Q = (A, D)$) when taking into account the feature's relative scales and orientations. We illustrate the above concepts in Fig. 2.

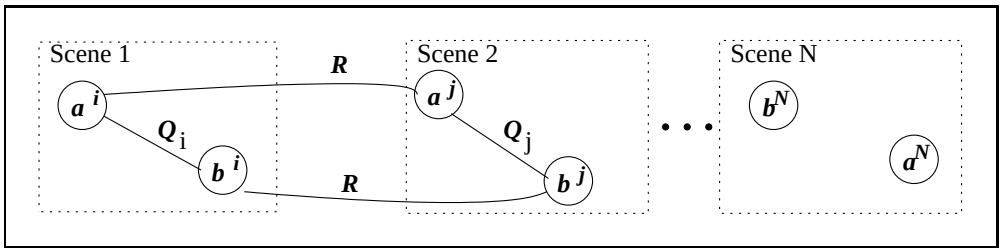

Figure 2: Relations between vertex components: $a = (a^i, \ldots, a^j, \ldots, a^N)$, $b = (b^i, \ldots, b^j, \ldots, b^N)$ are vertices connected through an edge $e(a, b)$. R has the relative scale and orientation between two vertex components across images. Q has the angle and norm of a vector linking two components from distinct vertices in a single image.

Edge ranking. From the previous paragraph is possible to conclude that the rank of an edge is defined as a function of four main quantities: (a) the relative scales and (b) the relative orientations within pairs of features of the connecting vertices; (c) the angles and (d) the norms of the vectors defined by a pair of corresponding instance coordinates taken from the two connecting vertices. One of the simplest, yet powerful, ways of comparing these quantities is by using normalised absolute differences. Equation (3) shows how to compare the relative scales of corresponding features (a^i, a^j), (b^i, b^j) found in two vertices a and b.

$$\Delta S^{i,j} = 1 - \frac{abs(rS(b^i, b^j) - rS(a^i, a^j))}{rS(b^i, b^j) + rS(a^i, a^j)} \tag{3}$$

Equation (4) shows the same for relative orientations, with the difference that a normalisation function $\hat{O}$ (Eq. (5)) is now required to take into account the fact that orientations are measured in a closed circle.

$$\Delta O^{i,j} = 1 - \frac{\hat{O}(abs(rO(b^i, b^j) - rO(a^i, a^j)))}{180} \quad (4)$$

$$\hat{O}(x) = \begin{cases} 360 - x, & \text{if } x > 180 \\ x, & \text{otherwise} \end{cases} \quad (5)$$

The angle $A(a^j, b^j)$ of the vector defined by a pair of corresponding instance coordinates (from a same scene), taken from the two connecting vertices, is expected to be the same angle found in any other pair of instance coordinates (at another scene), apart from the rotation that each of the feature pairs might have suffered from one scene to another. Here we have to decide which feature pair gives the best estimate for the angle on the second scene, so we compute two differences (Eq. (6)) and take the minimum between these differences (Eq. (7)). Although normally both values are the same, there is the possibility of imprecise calculations at earlier stages, due to noise for example. Note that the normalisation function $\hat{O}$ has to be used again as orientations are compared.

$$\begin{aligned} dA_a^{i,j} &= abs(A(a^j, b^j) - (rO(a^i, a^j) + A(a^i, b^i))) \\ dA_b^{i,j} &= abs(A(a^j, b^j) - (rO(b^i, b^j) + A(a^i, b^i))) \quad (6) \end{aligned}$$

$$\Delta A^{i,j} = 1 - \frac{MIN(\hat{O}(dA_a^{i,j}), \hat{O}(dA_b^{i,j}))}{180} \quad (7)$$

Similarly to the angle comparison, the length of the vector connecting two features in an image should be preserved in any other image, apart from the change in scale that each of the feature pairs might have suffered from one scene to another. Again, the feature pair that gives the best estimate for the scale on the second scene (Eq. (8)) has to be chosen (Eq. (9)). Note that this time, the normalisation factor (called here $q^{i,j}$) depends on the minimum value that is chosen (Eq. (10)).

$$\begin{aligned} dD_a^{i,j} &= abs(D(a^j, b^j) - (rS(a^i, a^j) \times D(a^i, b^i))) \\ dD_b^{i,j} &= abs(D(a^j, b^j) - (rS(b^i, b^j) \times D(a^i, b^i))) \quad (8) \end{aligned}$$

$$\Delta D^{i,j} = 1 - \frac{MIN(dD_a^{i,j}, dD_b^{i,j})}{q^{i,j}} \quad (9)$$

$$q^{i,j} = \begin{cases} D(a^j, b^j) + (rS(a^i, a^j) \times D(a^i, b^i)), & \text{if } dD_a^{i,j} < dD_b^{i,j} \\ D(a^j, b^j) + (rS(b^i, b^j) \times D(a^i, b^i)), & \text{otherwise} \end{cases} \quad (10)$$

Finally, we define the rank of an edge e as the average of all four quantities explained above (Eq. (11)). Edges connecting vertices that have at least one wild-card are not taken into account by this function.

$$Rank(e) = \sum_{\substack{\forall i,j \in 1,\ldots,N \\ a^i, b^i, a^j, b^j \neq *}} \frac{\Delta S^{i,j} + \Delta O^{i,j} + \Delta A^{i,j} + \Delta D^{i,j}}{2 \times N \times (N-1)} \quad (11)$$

Edge pruning. The number of edges that can potentially be created from a set of vertices is quadratic in the size of the vertices. Two mechanisms are used to prune the edge space. The first one acts during the edge creation process by eliminating edges that link pairs of vertices containing at least one common instance at the same feature within the vertex list, as they cannot correspond to any real feature relationships. The second, thresholding, is used only after all edges have been created and evaluated.

3.3 Cliques

A standard algorithm is used to find cliques. The algorithm takes as input a graph $G = (V, E)$ and returns in the superset $CLIQUES$ all the cliques found:

Algorithm 2: Finding cliques $G = (V, E)$

$i := 1$; $C := CLIQUES := \emptyset$; $L(k) = \emptyset \ \forall \ k = 1, \ldots, size(V)$
while $i > 0$
  if $\exists \ v_a \in V$ with $v_a \notin L(k)$, for all $k \leq i$ **then**
    $L(i) := L(i) \cup \{v_a\}$
    if $\exists$ an edge $e_a = (v_a, w_a) \in E$ for all $w_a \in C$ **then**
      $C := C \cup \{v_a\}$
      let $CLIQUES := CLIQUES \cup \{C\}$
      $i := i + 1$
    else **remove** the $i - 1^{th}$ vertex from C
      $L(i) := \emptyset$
      $i := i - 1$

Clique ranking. The final stage is to rank the cliques. The rank of a clique is the product of the averages of all its internal vertex and edge ranks.

$$Rank(CLIQUE) = \bar{V} \times \bar{E} \times \frac{\#CLIQUE}{\#CLIQUE_{max}} \tag{12}$$

where $\bar{V}$ and $\bar{E}$ are the average values of all vertices and edges in the clique, respectively. $\#CLIQUE$ is the clique size and $\#CLIQUE_{max}$ is the size of the maximal clique(s).

4 Case Study

In order to help focusing on the structure learning process, and to keep away from other aspects of our system (like attention, lighting invariance, dealing with clutter and so on) which are not the main issue of this paper, we tried to make this case study as simple as possible. Three scene images were created from two top view pictures of a telephone handset and its base unit, taken against a black background. The two pictures were placed inside a large black image under varying scales and orientations. In scene S^1 the handset and base were place parallel to each other. In scene S^2, the handset was translated, rotated by 90^o and scaled down by a factor of 70% with respect to its first occurrence. Finally, in scene S^3 the base unit was scaled down by a factor of 60% of its original size and the handset was rotated by 300^o with respect to its first occurrence, see Fig. 3. A set of interest points have been manually selected and passed to the system. These points consisted of: three pairs of central microphone/speaker positions within the telephone handsets; three consistent 'led' positions in the base units and three dark spots within the base units. A set of two distractor

points (not belonging to any distinguishable feature) have also been selected in two of the scenes. From this, we want our system to learn that the handset and base units are each one a structured model, but, as the handset and base do not obey a rigid body transformation, they should not form a structured model.

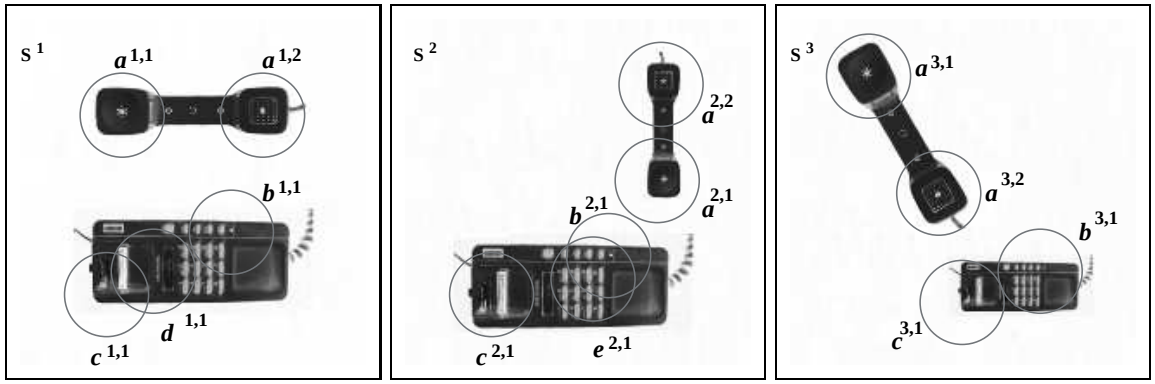

Figure 3: Scenes used in the case study. In order to facilitate visualisation, the original image intensities were inverted. The circles represent the retinal areas centred at the interest points. The feature types a, b, c, d, e obtained by Algorithm 1 are also shown.

Algorithm 1 (Section 2) was applied to the set of interest points. The results are summarised on Table 1. As one might expect, five different feature types were automatically identified: type a for describing speaker and microphone areas within a telephone handset; b for describing the features centred on the 'led' of the base unit; c for the dark spot features; and d, e for the distractor features. The next step was to use the object feature models and relations to build the graph according to what is described in Section 3. By using a threshold of 0.8, five cliques were obtained, which are listed below:

$$(a^{1,1}, a^{2,1}, a^{3,1}) \overset{0.92}{\longleftrightarrow} (a^{1,2}, a^{2,2}, a^{3,2}) \qquad (a^{1,1}, a^{2,1}, a^{3,2}) \overset{0.91}{\longleftrightarrow} (a^{1,2}, a^{2,2}, a^{3,1})$$
$$(a^{1,1}, a^{2,2}, a^{3,1}) \overset{0.90}{\longleftrightarrow} (a^{1,2}, a^{2,1}, a^{3,2}) \qquad (a^{1,1}, a^{2,2}, a^{3,2}) \overset{0.89}{\longleftrightarrow} (a^{1,2}, a^{2,1}, a^{3,1})$$
$$(b^{1,1}, b^{2,1}, b^{3,1}) \overset{0.94}{\longleftrightarrow} (c^{1,2}, c^{2,1}, c^{3,1})$$

The four cliques involving features of type a indicate that the telephone handset features define a rigid geometric model governed by the relationships between the clique vertex components. The reason why there are four cliques describing the same geometric relation is because the handset features were classified as the being of the same type, so they can be interchanged within a vertex without breaking the geometric constraint. The remaining clique corresponds to a structural model for the base unit.

5 Conclusions

In this paper we provide an answer to the question of whether or not is possible to to learn rigid geometric models from 2-D image evidence (iconic object models) acquired from a sequence of scenes. We found that structured models can indeed be learned in such a context by using a graph-based representation and algorithm. In a case study we have shown how our approach works in practice. More complex case studies are currently under development and will be

	$a^{1,1}{}_{(236,587)}$	$a^{1,2}{}_{(562,589)}$	$a^{2,1}{}_{(598,431)}$	$a^{2,2}{}_{(598,659)}$	$a^{3,1}{}_{(177,676)}$	$a^{3,2}{}_{(342,395)}$
$a^{1,1}$	1,0,1	1,180,0.96	0.70,90,0.96	0.70,270,0.95	1,300,0.97	1,120,0.96
$a^{1,2}$		1,0,1	0.70,270,0.96	0.70,90,0.98	1,120,0.97	1,300,0.99
$a^{2,1}$			1,0,1	1,180,0.96	1.44,210,0.96	1.44,30,0.97
$a^{2,2}$				1,0,1	1.44,30,0.96	1.44,210,0.98
$a^{3,1}$					1,0,1	1,180,0.97
$a^{3,2}$						1,0,1

	$b^{1,1}{}_{(488,313)}$	$b^{2,1}{}_{(476,256)}$	$b^{3,1}{}_{(587,225)}$
$b^{1,1}$	1,0,1	1,0,0.99	0.58,0,0.99
$b^{2,1}$		1,0,1	0.58,0,0.99
$b^{3,1}$			1,0,1

	$c^{1,1}{}_{(188,188)}$	$c^{2,1}{}_{(176,131)}$	$c^{3,1}{}_{(408,152)}$
$c^{1,1}$	1,0,1	1, 0,0.99	0.58,0,0.96
$c^{2,1}$		1,0,1	0.58,0,0.97
$c^{3,1}$			1,0,1

Table 1: Results of the Algorithm 1. The three smaller sub-tables present the relationships $(rS, rO, Sm))$ between feature instances of type a, b, c, respectively, in different images as well as the positions (x, y) where these features were found. The lower diagonals of the sub-tables are not shown because they are symmetric. The feature types d and e have only one instance and therefore were not included in the table, their coordinates are as follows: $P_{d^{1,1}} = (314, 226)$ and $P_{e^{2,1}} = (441, 215)$.

available in [4]. An important difference between the way we learn models and the existing traditional approaches is that our system is designed to search the visual field for objects in an attentive way, like humans and some other animals do. In this way, the relative position of clustered features can be recorded and, with the help of the features' relative scale and orientation, possible relationships amongst features can be worked out.

Obviously, there are some issues related to the algorithms described in this paper that require further research. For instance, there are other ways of defining the rank of a vertex, as for example the average of the similarity scores between all the pairs of vertex elements. A study on how the functions used to rank vertices, edges and cliques influence the learning results is currently under investigation. The vertex creation process is not yet the optimal solution to the problem as it suffer from a scalability problem: the size of the resulting combinatorial set grows exponentially with the number of images. However it is still a reasonable solution for a few tens of images. One way to reduce the number of combinations would be to pre-group multiple instances of same model class as if it were a new type of object. Finding a more computationally attractive vertex definition is left as future work.

References

[1] I. Biederman. Human image understanding: recent research and a theory. In A. Rosenfeld, editor, *Hum. and Mach. Vision II*, pp 13–57. Acad. Press, 1986.

[2] R. B. Fisher. *From Surfaces to Objects.* John Wiley and Sons, 1989.

[3] R. B. Fisher and A. MacKirdy. Integrating iconic and structured matching. In *Proc. of Europ. Conf. on Comp. Vision*, vol. II, pp 687–698, Freiburg, June 1998.

[4] H. M. Gomes. Model learning in iconic vision. PhD Thesis, Division of Informatics, Edinburgh University, to be submitted, August 2000.

[5] H. M. Gomes, R. B. Fisher, and J. Hallam. A retina-like image representation of primal sketch features extracted using a neural network approach. In *Proc. of Noblesse Workshop on Non-Linear Model Based Image Analysis*, pp 251–256, Glasgow, July 1998.

[6] T. D. Grove and R. B. Fisher. Attention in iconic object matching. In *Proc. of Brit. Machine Vision Conf.*, vol. 1, pp 293–302, Edinburgh, 1996.

[7] D. Marr. *Vision*. W. H. Freeman and Co., 1982.

[8] A. R. Pearce, T. Caelli, and W. Bischof. Rulegraphs for graph matching in pattern recognition. *Patt. Recog.*, 27(9):1231–1247, 1994.

[9] A. Pentland, R. W. Picard, and S. Sclaroff. Photobook - content-based manipulation of image databases. *Int. Journal of Comp. Vision*, 18(3):233–254, 1996.

[10] R. P. N. Rao and D. H. Ballard. An active vision architecture based on iconic representations. *Artif. Intell.*, 78:461–505, 1995.

[11] G. Sandini and M. Tristarelli. Vision and space-variant sensing. In H. Wechsler, editor, *Neural Netw. for Percep.*, vol. 1, chap. II.11, pp 398–425. Acad. Press, 1992.

[12] B. Schiele and J. L. Crowley. Probabilistic object recognition using multidimensional receptive field histogram. In *Proc. Int. Conf. on Patt. Recog.*, vol. B, pp 50–54, Vienna, August 1996.

[13] G. Vosselman. *Relational Matching*. Number 628 in Lecture Notes in Computer Science. Springer-Verlag, Berlin Heidelberg, 1992.