Unsupervised Neural Network Classifier for Semantic Graphs

Denis Boulanger and Denis Poussart

Laboratoire de Vision et Systèmes Numériques
Département de génie électrique, Université Laval.
Québec, Canada, G1K 7P4
e-mail:boubou@gel.ulaval.ca, poussart@gel.ulaval.ca

Abstract

In this paper we describe a parallel neural network architecture for the recognition of semantic graphs. Graphs are a high level representation of knowledge used in computer vision and in many other fields. When the recognition process begins, geometric values attributed to nodes and arcs of the graphs initiate a parallel search through the neural network. This search results in the selection of a graph from the database. A parallel relaxation is then initiated in order to find a match between the selected graph and the shown one. The algorithm uses some properties of neural networks such as fast access to classes, immunity to noise, and auto-classification. The database is built from example graphs extracted from 3-D and 2 1/2D images.

1. Introduction

In computer vision, the recognition problem can be summarized in two major steps:

- searching into the database,
- comparing and validating of the selected models.

However before initiating any recognition process, a system must acquire an image of a scene, using sensors such as CCD cameras or laser range finders. From raw data, a recognition system should be able to extract pertinent information in order to describe the scene. Many segmentation techniques [5] have been studied using intensity or range images. One common point emerges: if a recognition system is to be efficient, the segmentation process should give a constant representation independently of the point of view of the observer and of the orientation of the object in the scene. A representation is said to be constant if the properties of the segmented parts of an object are equivalent independently of the view of the part.

In a recognition system using geometric descriptors, each segmented patch of the image is described using geometric attributes. For example, a plane may be described by its surface and its perimeter [6]. Taken individually, each part is not enough to describe a complex object. Three square planes may describe a corner of a cube as well as some faces of a complex polyhedral object for example. The description can be refined by introducing geometric constraints between patches. Such constraints can be derived from two adjacent patches, such as the angle between two planes, or from three or more. The choice of the geometric parameters is also critical for the recognition steps. The constraints that give a constant description of an object and the number of these parameters needed for an optimal description, are of fundamental importance.

Unary and binary constraints can be viewed as a graph representation where nodes are the lists of geometric descriptors for each patch, and arcs are the lists of geometric features between adjacent patches. The goal of recognitionthen is to find in a database the model graph which best matches the one extracted from the image. If the number of models in the database is rather small, then an exhaustive comparison of all models is easily performed. The problem arises when the number of models in the database increases. The search through the database has to be optimized in order to limit the number of comparisons between possible models and the input graph.

Once a limited number of possible models are found, the recognition system must verify if the selected graphs match the one given. The problem of graph matching consists of associating each node of the input graph with only one node of the model graph. Taken individually each input node can be associated with many nodes of the model. Using binary constraints, which are the arcs of the graph, one can limit the number of possible matches to each node. Methods such as tree search or relaxation labeling can be used to satisfy every constraint at the same time. This is known as the consistent labeling problem [3,4].

In this paper we describe a parallel neural network classifier for semantic graphs. The algorithm uses some properties of neural networks such as fast access to classes, immunity to noise, and auto-classification. The network is based on an ART system described by Grossberg and *al.* [1].

In the next section, the ART system is described. In the following sections, a general description of the network

and its operation are given, followed by a more detailed description. Finally, some experimental results demonstrate the operational modes of the network.

2. ART system

Carpenter and Grossberg have described a number of classification networks based on their adaptive resonance theory (ART). Their systems, like ART1 or ART2, implement a massively parallel algorithm for the non-supervised classification of binary and continuous patterns. Basically, an ART system is made of two layers of units connected by a series of distributed links. The bottom level is generally used both as an input and as a comparison level. In the top level, the network is made of classification units. Weighted links are established between both levels in both ways. The weights are modified during a learning phase and asymptotically approach the mean value of inputs. Each different input is assigned to a different class and memorized in the connections of the network.

The recognition process begins by correlating the input values with the bottom-up links (*c.f.* figure 1). This inner product assigns to each classification unit a value proportional to the distance between the input values and the memorized values associated with this class. The unit receiving the highest product is then enhanced and selected. The selected class generates back to the bottom level an image of the memorized model through the top-down connections. The relative difference between the input and the generated image is computed and compared to a threshold value. If the difference is too large between the two images, the selected classification units is reset by the network. If a reset occurs, the recognition cycle is re-activated and another classification unit is selected and validated. This cycle continues until a match is found or until a new class has been assigned to the input. When the input matches the selected class, the network is said to be in resonance.

Every time the network is in resonance, the bottom-up links and the top-down links are gradually modified in order to adapt to the variations between the input and the models. After being shown a series of noisy inputs of the same nature, the memorized model will represent an average model of the different inputs.

The ART approach has many interesting properties. First, models in the database are built automatically from examples and are averaged among the prototype inputs. The search in the database is fast, direct and conducted in parallel. Finally, its immunity to noise makes the network a good approach to the recognition process. The semantic graph network described here has been designed while keeping these principles and properties in mind.

3. Semantic graphs classifier

In the semantic graph classification network, a graph is memorized in two levels. Each list constituting a node and each arc list are associated with different classes. These

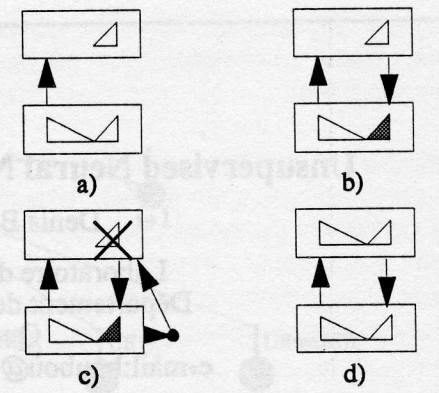

Figure 1: Search process of an ART system: a) The network first selects the class which receives the highest correlations value from the input and the bottom-up links. b) An image of the selected class is then fed back to the bottom level. c) If the relative distance between the input values and the image values is higher then a certain threshold, the network resets the selected classification unit d) The process continues until the selected class matches the input, or until a new class has been created.

classes are grouped and linked with a second level of classes which memorizes the graphs. As shown in figure 2, the network is made of two different levels of classification units and one input level. Between the input level and the association matrices, a series of distributed weighted links help memorize the nodes and the arc values in the network.

The association matrices give the possible memorized classes of nodes and arcs that can be associated with each input node and arc. Links between these matrices and the graph's classification units are created to memorize the node and arc classes in the selected graph. Finally, links between the two associating matrices are used during the relaxation phase to establish the relations between node and arc classes.

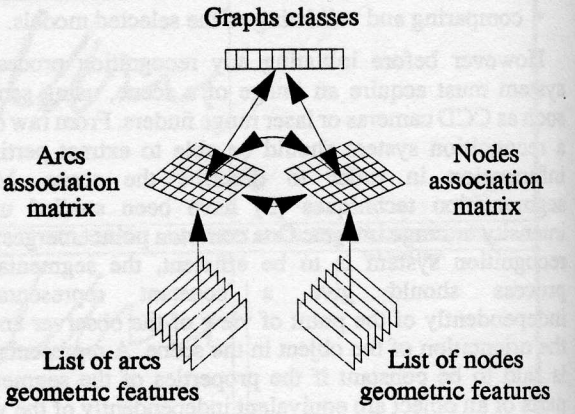

Figure 2: Structure of the semantic graphs classifier network: The network is made of two different levels of classification units and one input level. Links between levels (arrows) are used to memorize the graphs.

The search and recognition process, illustrated in figure

3, can be subdivided into four major steps. First, the lists of geometric values are compared individually with all the memorized values associated with each node and arc of the stored graphs. These values are memorized in the weighted links between the input level and the association matrices. If the difference between an input item list and memorized one is lower than a certain threshold, then the class representing the memorized list is associated with the input list. Each node and arc of the input graph is associated in this way with a series of possible classes. This builds an association matrix between each node and arc of the input graph, and the memorized classes.

From a previous learning phase, the network knows which node and which arc are associated with each graph. These associations are directly obtained from the links between the matrices and the graph classification units. In this way, each node and arc class found in the first step cast a vote on every linked class of graph. The class which has a number of nodes and arcs equivalent or closer to the input classes is then selected first.

In the next step, the classes of nodes and arcs which constitute the selected graph are fed back to the association matrices. These classes enable the matrices to eliminate the possible associations which are not compatible with the generated graph. However, there may still remain multiple associations for every node and arc. Such associations are consistent locally but may not be consistent globally.

Shapiro and Harlick [2,3,4] have described this situation as the consistent labelling problem: given a set of objects and relations to be labelled, a set of possible labels and a set of constraints, try to assign labels to every object without violating the constraints and the relations. In the case of graph matching we want to assign each node of the model graph to only one node of the input graph, while satisfying the relations of both graphs.

Many methods for finding a solution to the labelling problem have been described. One of them was introduced by Rosenfeld and *al.* [7], who have described a parallel continuous relaxation operation which resolves the constraint satisfaction problem. In this method, a weight or assumed value is first assigned from every object to each label. The algorithm transforms all weights at once into a new set conforming to the set of constraints. This transformation is repeated until the weights converge to stable values.

A similar situation occurs with the network. Here, a set of input nodes and a set of possible classes are to be associated with each other. The same has to be done with arcs of the input graph. The goal of the relaxation method is to find one model node for each input node and one model arc for each input arc, all of which satisfy the local constraints. In other words, the networks should find the graph homomorphisms. A new scheme, based on continuous relaxation, is used to solve this problem.

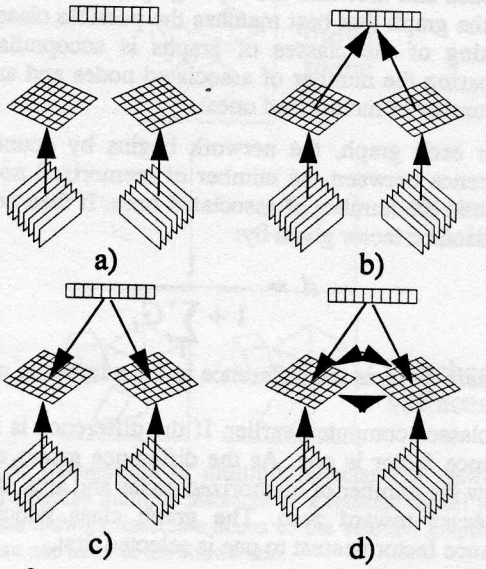

Figure 3: Semantic graph classifier search process: a) First, each list of features of nodes and arcs of the input graph is compared with the memorized classes. b) Then each possible class casts a vote for its associated graphs, memorized in the links. The graph having the least different number of nodes and arcs, is then selected. c) The nodes and arcs associated with this graph are fed back to the association matrices and helps eliminate the incorrect matches. d) A relaxation scheme finally generates a one to one association between the input graph and the selected graph.

4. Nodes and arcs class selection

Each node and arc of the graph represents a list of geometric attributes which describe the scene. These lists form vectors which are compared with the memorized values of each class of nodes and arcs. In a normal feed-foward neural network, a dot product is calculated between the input vector and the memorized vectors, giving for each class a distance measurement from the input vector. In our case, the vector items may not have the same range or may be non-numerical, so a dot product for this kind of vector is meaningless. Instead, the network computes the difference between every item of the input vector and the memorized items. A threshold value proportional to the memorized values gives the range where the input value is valid. A fuzzy threshold can also be used where the difference between values is associated with a certain degree of confidence. The minimum difference factor of each list gives an association factor between each input node and arc and the memorized ones. This accumulation of evidence produces two association matrices, one for the nodes and another for the arcs, where the possible matches of nodes and arcs of the input graph with the memorized ones, are mapped.

5. Graph class selection

Given the association matrices, which generates the possible association factors between the memorized classes

of nodes and arcs and the input graph, the network must find the graph that best matches the possible classes. The indexing of the classes of graphs is accopmlished by comparing the number of associated nodes and arcs with the number of memorized ones.

For each graph, the network begins by counting the difference between the number of memorized nodes and arcs and the number of associated ones. It then computes the distance factor given by:

$$d = \frac{1}{1 + \sum_i G_i}$$

where $\sum_i G_i$ is the difference in the number of node and arc classes computed earlier. If the difference is null, the distance factor is one. As the difference grows above or below the number of memorized nodes and arcs, the factor decreases toward zero. The graph class which has a distance factor nearest to one is selected first.

Once a graph has been selected, its nodes and arcs are fed back to the association matrices. These nodes and arcs are memorized in the top-down connections of the network. The network keeps only the classes which are common to the selected graph and the input graph. From the initial association, the selected graph narrows the possible association for each input node and arc. However each node or arc might still have more than one class that can be assigned to it. This happens when one or more node or arc in the memorized graph are equivalent to each other.

6. Labelling relaxation

In order to remove the ambiguity of certain associations, the network goes into a relaxation phase. The relaxation method insures that not more than one class is associated with each node and arc. If each node and arc is taken individually, all the associations made are valid. However globally, the network must find the correct combination of labeling of nodes that satisfies the possible arcs.

The method is based on a constraint satisfaction network [7]. At each relaxation cycle the degree of association of every node is update according to the following:

$$\frac{\partial x_i}{\partial t} = (1 - x_i)\left(\sum_j \lambda_j x_j + A\right) - x_i \sum_k x_k$$

where $\sum_j \lambda_j x_j$ is the sum of local constraints from the adjacent nodes of the input graph, A is a decay factor, and $\sum_k x_k$ the sum outputed by the units of the same row and column of the active unit satisfying the constraints of one association by node. After every cycle, only arc associations which are activated by two active nodes are kept. The others assocaitions are deleted.

When more than one solution exists for each node, the network falls into a local minimum. The network climbs out of this minimum by selecting the most probable association and restarts the relaxation process.

This relaxation cycle enhances units which satisfy the constraints directly or indirectly, while ensuring at most one association for each node and arc.

7. New graphs

After the relaxation network has stabilized, the network evaluates the homomorphism between the two graphs. If the selected graph is a sub-graph of the input, some nodes and arcs of the input graph do not have any possible match. However, if the input graph is a sub-graph of the selected one, every node and arc of the input will find a match but not all nodes of the selected graph will be matched. When the two graphs are not equivalent the network resets the selected graph class and continues with another class. When no graph has been found, a new class is assigned to the input graph.

When a new class is created the network first tries to find the maximum sub-graph that matches the input using relaxation. The network allocates new classes to all nodes and arc which have not found any matches.

8. Memorization of the graphs

When the network has found a graph that matches or when it has allocated a new class to the input graph, the weight of the links of the network are updated. The items of the list of each node and arc are memorized using a simple delta rule function. Using this function insures convergence, and is more immune to abrupt variations of the items values.

The nodes and arcs classes are memorized directly by making unit weight links from the activated classes to the selected graph class. Finally the constraint links of the input graph, between the two association matrices, are copied between their associated nodes and arcs classes.

9. Experimental results

The network was trained using synthetic polyhedral objects. Each planar patch was described using geometric features as well as geometric constraints between adjacent planes. The geometric features used to describe each object were [6]:

- Perimeter.
- Area.
- Minimum and maximum radius
- Angle between two planar patches.
- Type of intersection *i.e.* convex or concave.

The network was trained using twenty 3-D model graphs of polyhedral objects. The network classified each graph in a different class of graph using common classes for nodes and arcs. During the training phase, every time a new graph

was shown to the network, it selected the graph closest to the input and attempted to match the nodes, represented by faces of the objects (*c.f.* figure 4). If every node of the input graph was not matched, the network reset the selected class and retried with another one. The network continued its search until it found a match. If no match was possible, the network create a new class and the input graph was memorized.

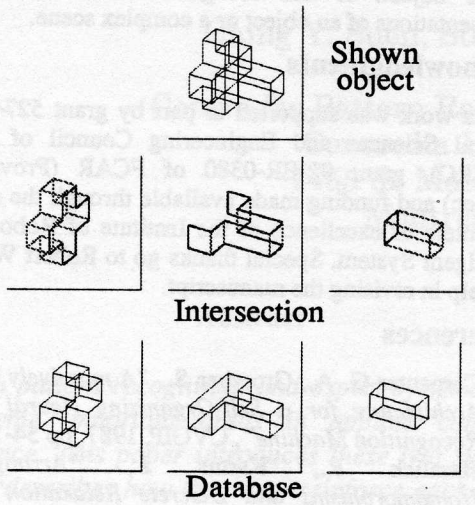

Figure 4: Search in the database: The graph of the object is shown to the network and matched against the memorized graphs beginning with the one closest to the input. If no match is found, a new class is created.

When a new class is created, the network finds the best sub-graph from all the memorized graphs, that matches the input. Classes of nodes and arcs of this sub-graph are memorized together with new ones assigned to nodes and arcs that did not find a match.

After the training phase, the network accesses the proper model the first time when a 3-D object graph is shown, even though the attributes' values are noisy. When presented with graphs of the projected objects, the network usually accesses the database correctly the first time.

By letting the network associate a node of the selected graph with more than one node of the input graph, the network is able to recognize multiple instances of selected models in a complex scene (*c.f.* figure 5).

One other interesting way of using the network is as a classifier of aspect graphs. Aspect graphs are built from projected views of an object. Using multiple views of the same object as models, an object can be recognized whatever its orientation with respect to the observer. When presented with a series of graphs obtained from multiple views of an object, the network classifies each graph in a different class using only a minimum number of classes for nodes and arcs of the input graphs. The network recognizes common surfaces from one view to another, and memorizes the next view using those surfaces (*c.f.* figure 6).

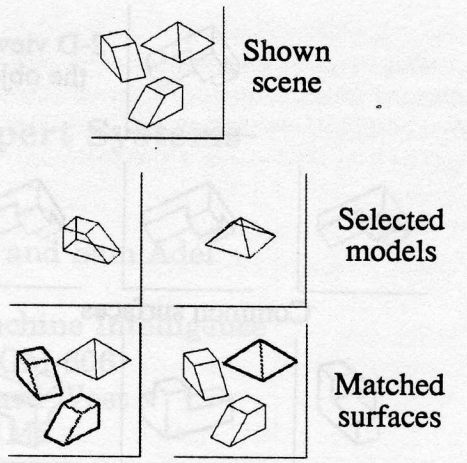

Figure 5: Recognition of multiple objects: The network is able to recognize multiple instances of a selected model just by letting the network associate nodes of the selected graphs with more than one node of the shown one.

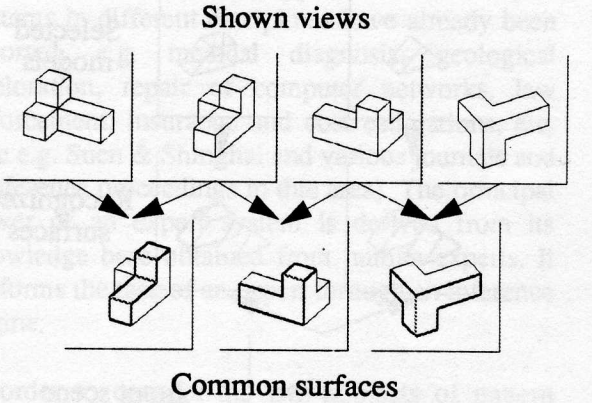

Figure 6: Classification of aspect graphs: The network classifies Each view of the object using common classes for nodes and arcs of their graphs.

After memorizing all possible views, the network always accesses more than one memorized graph when shown another view (*c.f.* figure 7). Therefore, the network is able to determine if the views are redundant or not just by counting the number of selected graphs.

Finally, the recognition of partially occluded surfaces is easily performed using a fuzzy association for nodes and arcs. In the first step of a network cycle, each value of node and arc lists of the input graph is compared to the memorized values using a fuzzy thresholds which gives an association factor ranging from 0 to 1. If a surface is partially hidden, its geometric attributes can be altered. Even then, a partial match can be accopmplished by giving a fuzzy associations factor to the input nodes and arcs. During the relaxation phase these fuzzy associations can then be enhanced and selected (*c.f.* figure 8).

However these associations are very sensitive to the type

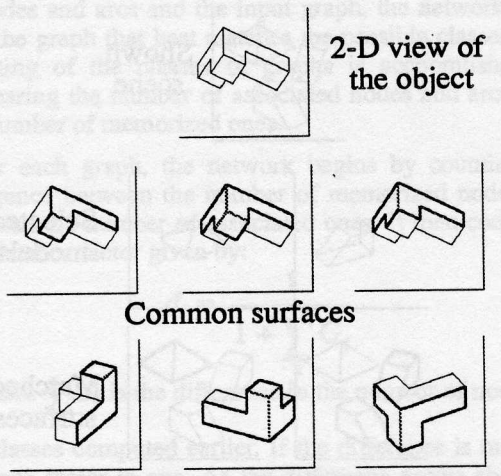

Figure 7: Recognition from aspect graphs: Every time a new view of the object is shown, the network accesses more than one class and matches the common surfaces to the selected memorized views.

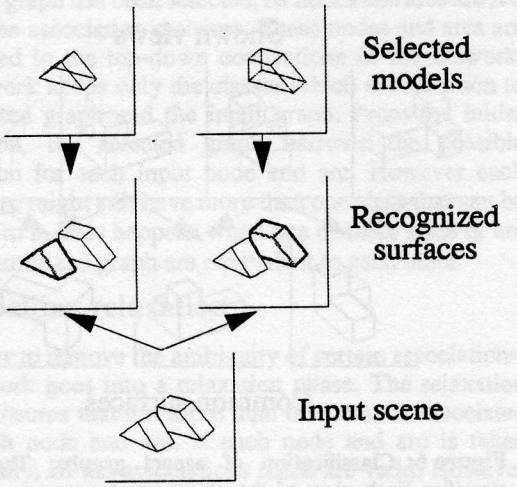

Figure 8: Recognition of partially occluded objects: Each node and arc of the input graph is matched against the memorized values using fuzzy thresholds. The network can then match these surfaces during the relaxation phase.

of geometric attributes used. For example, the perimeter of a visible surface can vary considerably if it is partially occluded. Using attributes less sensitive to occlusions would enhance the recognition process.

10. Conclusion

Many properties characterize the present semantic graph classifier. Its parallel structure makes the implementation easy and direct in a parallel architecture computer. It exploits the properties of neural networks such as fast access to the classes and immunity to noise. Because the network builds its database from examples, it adapts itself to the noise level of the prototypes shown. The network is not limited to classifying numerical values but can also classify strings, making it applicable not only to vision problems but to many other fields of artificial intelligence. Finally, the network optimizes the number of classes of nodes and arcs by memorizing only classes which are different from the memorized ones.

The network can be configured to recognize partially occluded objects in a complex scene. It can also be used to classify and recognize aspect graphs from multiple views of an object. It can recognize 2 1/2-D and 3-D representations of an object or a complex scene.

Acknowledgments

This work was supported in part by grant 5274 of the Natural Sciences and Engineering Council of Canada (NSERC), grant 92-ER-0380 of FCAR (Province of Québec) and funding made available through the network of centers of excellence of the Institute of Robotics and Intelligent System. Special thanks go to Robert Wolfe for his help in revising the manuscript.

References

[1] Carpenter G. A., Grossber S., *"A massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine"*, CVGIP, 1987, pp 54-115.

[2] Haralick R., Kartus J., *"Arrangements, Homomorphisms and Discrete Relaxation"*, IEEE Trans. on System, man and, cybernetics, SMC-8, No 8, August 1978, pp. 600-612.

[3] Haralick R., Shapiro L., *"The Consistent Labeling Prolem: Part I"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, PAMI-1, No 2, April 1979, pp. 173-184.

[4] Haralick R., Shapiro L., *"The Consistent Labeling Problem: Part II"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, PAMI-2, No 3, May 1980, pp. 193-203.

[5] Lowe D.G., *"Three Dimensional Object Recognition from Single Two-Dimensional Images"*, Artificial Intelligence, Vol-31, 1987, pp 355-395.

[6] Oshima M., Shirai Y., *"Object Recognition Using Three Dimensional Information"*, IEEE Trans. on Pattern Analysis and Machine Intelligence, PAMI-5, No 4, July 1983, pp. 353-361.

[7] Rosenfeld A., Hummel R., Zucker S., *"Scene Labeling by Relaxation Operations"*, IEEE Trans. on System, man and, cybernetics, SMC-June 1976.