Model Indexing: the Graph-Hashing Approach

Humberto Sossa, Radu Horaud

Model Indexing: The Graph-Hashing Approach

Humberto Sossa & Radu Horaud
LIFIA–IRIMAG
46, avenue Félix Viallet
38031 Grenoble France

1 Introduction and approach

The problem of object recognition in computer vision is the problem of interpreting image features in terms of object features and of describing the image in terms of objects and relationships between objects. In the past, attempts to solve this problem have contributed to the following computational paradigm:

1. Describe both the object and the image in terms of features and
2. Establish a one-to-one correspondence (matching) between a subset of image features and a subset of object features.

Various implementations of this paradigm take the form of either a search-graph [1] or a search-tree [2, 8, 7]. The matching being combinatorial in complexity, the rationale has been to use constraints (those provided by assuming rigid objects) such that the search space (the graph or the tree) is as reduced as possible. Hence, any search process (including exhaustive search) is constrained to visit a few hundreds of nodes rather than millions.

Nevertheless, if many objects, rather than a single object are present in a database of objects to be recognized then the search process just described exploses again because the complexity grows with the number of objects [6]. Therefore, step 2 of the above paradigm should be further decomposed into: *model indexing* and matching.

Model indexing can be briefly described as follows: Given a group of image features "rapidly" extract from the list of objects those objects containing this group of features.

Nevatia & Binford [14] were among the first to describe indexing as a part of an object recognition system. Ettinger [4] described a hierarchically organized object library well suited for indexing.

The idea of using hashing in conjunction with object recognition was introduced by Kalvin & al. [10]. Stein & Medioni [15] were able to recognize 3-D objects from 3-D data using *structural* hashing. A hash-based approach was also proposed by Breuel [3].

The method that we propose in this paper stems directly from the work of Ettinger, Kalvin & al., Breuel, and Stein & Medioni. However, unlike these methods which deal with specific object types and feature types, our method operates on an abstract representation of features, more precisely, groups of features. In practice, this abstract representation takes the form of a graph: Nodes in this graph represent features and edges in this graph represent feature relationships. Such a graph may well describe a whole object or any object sub-part. The representation is equally valid for describing 2-D data, 3-D data, and objects.

In this paper we deal with binary graphs only. That is, only one feature-type and one feature-relationship-type can be embedded in the representation. However, it is straightforward to overcome this limitation by using a multiple binary graph representation.

The remainder of this paper is organized as follows. In Section 2 we briefly present a graph-theoretic method for characterizing a binary graph by a set of numbers such that this characterization is unique. In Section 3 we use the results of Section 2 in order to build and index a database of binary graphs under the form of *graph hashing*. Section 4 describes a model-indexing system based on graph characterization and graph hashing. Section 5 presents some preliminary experimental results obtained with polyhedral objects and suggests some directions for future work in the light of the current limitations.

2 Characterization of a graph

In this section we describe one way to characterize a graph G with n nodes and m edges. One way to think of such a characterization is the characteristic polynomial $det(xI - A(G))$ wich allows the computation of the eigen values of a matrix. $A(G)$ is the adjacency matrix of the graph G. However $det(xI - A(G))$ may be the same for two distinct graphs and hence this characterization is not a good one. Therefore we seek a polynomial associated with a graph, say $p(G)$, such that:

$$\begin{cases} \text{if } G_1 \neq G_2 \text{ then } p(G_1) \neq p(G_2) \\ \text{if } p(G_1) \neq p(G_2) \text{ then } G_1 \neq G_2 \end{cases} \quad (1)$$

Two graphs are said to be equal if they have the same number of nodes and if they are isomorphic. Two polynomials are equal if they have the same degree and if their coefficients are equal. It is worth reminding that graph isomorphism is known to be a NP-complete problem. If a polynomial satisfying the above condition exists, it follows

0-8186-2855-3/92 $3.00 ©1992 IEEE

that the problem of comparing two graphs of the same size is equivalent to the problem of comparing the coefficients of their associated polynomials.

Notice however that graph characterization with a polynomial allows one to state whether two graphs are isomorphic or not but it doesn't provide the node-to-node isomorphic mapping between the graphs. Graph characterization is therefore exactly what one needs for model indexing, i.e., rapidly state whether some sensed data *equal* some object data. The search of an isomorphic node-to-node mapping is the task of matching.

The first example of such a polynomial described in the literature [12] is the *permanental* polynomial of the *laplacian matrix* associated with G: $per(xI - L(G))$. In practice the problem of computing the coefficients of this polynomial is computationally intractable [16].

The second example is the *second immanantal* polynomial of the laplacian matrix: $d_2(xI - L(G))$ or the d_2-polynomial [11]. Unlike the permanental polynomial, the coefficients of the second immanantal can be computed using determinants [11]. The drawback is that for graphs of size greater than 12 the unicity condition described by eq. (1) is not guaranted: there exist a few non-isomorphic graphs (with size > 12) with the same d_2-polynomial. However, in our case this is not a strong limitation as it will be discussed in the next section. In the remainder of this Section we give the formulae for computing the coefficients of the d_2-polynomial.

The laplacian matrix of a graph G, $L(G)$ is defined as follows:

$$l_{ij} = \begin{cases} d_i & \text{if } i = j \\ -1 & \text{if nodes } i \text{ and } j \text{ are joined} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

d_i being the number of edges meeting at node i and is called the degree of a node. The laplacian matrix is simply a richer graph description than the wellknown adjacency matrix.

The d_2-polynomial associated with the laplacian matrix of a graph is:

$$d_2(xI - L(G)) = c_0(G)x^n - ... + (-1)^n c_n(G) \quad (3)$$

The coefficients $c_o, ..., c_n$ of this polynomial can be computed using the following formulae [11]:

$$\begin{aligned} c_0(G) &= n - 1 \\ c_1(G) &= 2m(n-1) \\ c_k(G) &= \sum_{X \in Q_{k,n}} \left(\sum_{i=1}^{n} l_{ii}\ det\ (L(G)\{X\}(i)) - \right. \\ &\quad \left. det\ (L(G)\{X\})\right) \end{aligned} \quad (4)$$

$Q_{k,n}$ denotes the collection of C_n^k k-element subsets of the set $\{1, 2, ..., n\}$. For any $n \times n$ matrix M and for $X \in Q_{k,n}$ let $M[X]$ be the $k \times k$ principal submatrix of M corresponding to X. $M\{X\}$ is the $n \times n$ matrix:

$$M\{X\} = \begin{pmatrix} M[X] & 0_k \\ 0_k & I_{n-k} \end{pmatrix} \quad (5)$$

where I_{n-k} is the identity matrix of size $n - k$ and 0_k is the null matrix of size k. $M\{X\}(i)$ is the matrix obtained from $M\{X\}$ by removing the i-th row and the i-th column. For a graph with n nodes, the complexity of computing the coefficients of the d_2-polynomial is n^4 [11, 13]. We remind that the complexity needed for computing the determinant of a matrix is n^3.

To conclude, we described a way of characterizing a graph G with n nodes and m edges by a set of $n + 1$ integers: $c_0, c_1, ..., c_n$.

3 Graph hashing

We consider now the following problem. A large collection of graphs are stored in a database. We want to devise a method to rapidly answer the question of whether an unknown graph is present in the database or not. One way to perform this search is to use *hashing*. Hashing can be briefly described as follows. Each object of the database has a numerical key associated with it. Then a *hash function* maps this key onto the address of an array of a manageable size. The address thus computed for an object is also called the hash-code of the object.

Database construction consists of computing such a hash-code for each object to be stored in the database. Several objects may well have the same address (hash-code). Therefore a list of objects will be associated with each address. The database takes therefore the form of a hash-table, a list of objects being stored at each hash-table address.

Indexing consists of computing the address (hash-code) of an unknown object in order to determine whether this object is in the hash-table or not.

It is important to outline that the indexing process retrieves a short list of database objects and not a unique object. We remember that, in theory, a graph with n nodes is characterized by $n+1$ integers. One cannot compare two graphs with a different number of nodes. The first integer, c_0 directly encodes the number of nodes. This suggests that c_0 and the set $\{c_1, ...c_n\}$ are used sequentially. Moreover, we have noticed that among the latter set c_{n-1} is the most discriminant one. Therefore one may use c_0 and c_{n-1} for hash-coding any graph. Of course, this couple of integers does not uniquely characterize a graph but such a unique characterization is outside the scope of hashing.

Another issue of crucial practical importance is the range of values on n: Lower and upper bounds for n will be discussed below.

In practice, *Graph hashing* uses c_0 and c_{n-1} sequentially. A "vertical" table allows the selection of one of the hash-tables upon the value of c_0. Next, c_{n-1} allows the selection of an address in the hash-table.

4 A model-indexing system

This Section addresses three issues: The object representation scheme that is being used, the organization of the database of objects, and the indexing mechanism.

Object representation has been thoroughly studied in Computer Vision and a recent paper by Flynn & Jain [5] provides a good state of the art. In general there are two possible representation classes: Object centered and viewer centered representations. Within our approach we use a representation that is not tight to a specific frame. An object is mainly described by a list of characteristic views. In the representation that we use the definition of a characteristic view (CV) should be understood in a broad sense. It is a network of object features and feature relationships that are simultaneously visible from some viewpoint. Such a representation is by no means limited to the aspect graph representation of an object. The features in the network may well be either 2-D or 3-D, object-centered or viewer-centered. The important characteristic here is not as much the dimensionality of the features or the frame to which they relate, but instead, the *intrinsic* topology of the feature network. As we already mentioned such a feature network can be convieniently represented by a binary graph with the restriction that only one feature-type and one feature-relationship-type are described in such a graph.

In general, the data associated with a sensor rarely encodes a whole CV associated with some object view. The data are corrupted by noise, occlusions, and accidental alignments. Therefore it is not wise to directly encode the characteristic views in the hash-tables and to try to index into a list of characteristic views. Instead, we added one more layer to our object representation scheme: Each CV is decomposed in a number of, possibly overlapping, fundamental figures (FF) where each FF is simply a subgraph of the graph encoding the CV. In short, a CV (large graph) is composed of fundamental figures (small graphs). There are several reasons in support of such a choice:

- One can compare only graphs of the same size. The "raw" graphs extracted from the data rarely have the same size as the graphs associated with the objects' characteristic views.
- The graph characterization scheme based on the computation of the d_2-polynomial is optimal for graphs not larger than 12 (see Section 2).
- A small group of image features is easier to detect than a large one. Plus, because of the occlusion problem, it is impossible to find in an image a large group of features that belongs to the same object.
- The set of fundamental figures increases sub-linearly as one adds new objects and hence new characteristic views to the database. In other words, the set of fundamental figures is an "alphabet" allowing a large "dictionary" of characteristic views.

A compromise must be made between very small fundamental figures (which are too ambiguous) and large ones (wich are hard to detect in the data). At the limit, a single-feature fundamental figure may belong to thousands of objects and model indexing is not very useful in this case; the extraction from the data of a very large fundamental figure is computationally intractable but such a large figure is less ambiguous than a small one and it may uniquely describe a characteristic view.

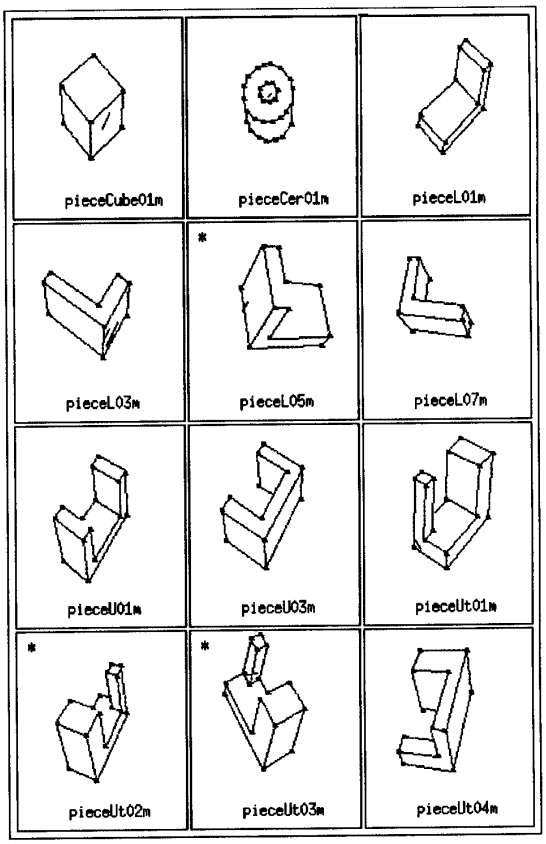

Figure 1: An example of a database of 12 characteristic views from 5 different objects. The database views matching the unknown object are marked by a "*".

In practice, the size of the fundamental figures, i.e., the number of nodes of the associated graph, varies between 5 and 9. Obviously, such a choice is consistent with the d_2-polynomial graph characterization, on one side but poses the problem of ambiguity, on another side: such a fundamental figure may belong to many different characteristic views. In order to overcome the problem of ambiguity we introduced a characterization of the intersection of two fundamental figures. Within a characteristic view, the decomposition into fundamental figures is such that the intersection of any two fundamental figures is not null: a fundamental figure intersection (FFI) is just another binary graph whose numerical characterization is provided by its d_2-polynomial.

To summarize, an object representation contains three layers. The top layer is a list of objects. At the middle layer, each object is described by a list of characteristic

views. At the bottom layer, a characteristic view is described by a list of fundamental figures and a list of fundamental figure intersections.

The organization of the database of objects follows the three-layer structure just described, but up-side down: a list of fundamental figures and a list of fundamental figure intersections, a list of characteristic views, and a list of objects. Only the FFs and the FFIs are organized in hash-tables following the graph-hashing model described in Section 3.

The indexing mechanism considers a set of fundamental figures and their intersections as extracted from an image and attempts to determine which object (or objects) contain this figures and hence which object (or objects) are likely to be present in the image.

5 Preliminary results and discussion

The model indexing system that is currently under development in our laboratory operates on 2-D characteristic views extracted from 3-D polyhedral objects. Both the database and the data to be interpreted are extracted from 2-D images.

The offline database construction process proceeds as follows. For each object to be stored in the database a number of images are taken from various "characteristic" viewpoints. Each such image is therefore a potential characteristic view associated with an object. The image is processed such that straight-lines and junctions are extracted. A junction is the intersection of two or more lines. The network of lines and junctions is mapped into a graph, where each node encodes a junction and each edge encodes a straight line [9]. Figure 1 shows a collection of 12 characteristic views. These characteristic views are those stored in our preliminary experimental database.

It is worthwhile to notice that these CVs are not perfect. Some lines are missing, some other lines are split into several pieces, etc. One important built-in feature of our model-indexing system is that it is robust with respect to these imperfections. The robustness is achieved through the use of fundamental figures and their intersections, both being local in nature, and through a voting mechanism as described below.

The decomposition of a characteristic view into fundamental figures is not an easy task. It is related to the task of decomposing a graph into sub-graphs under some constraints. So far we do not have a general-purpose decomposition technique. Instead we use a simple technique which considers, for each node in the graph, a sub-graph "around" this central node. For instance, one may consider all the nodes at a distance less than k around a central node. Finally, for each characteristic view, its associated fundamental figures and fundamental figure intersections are stored in the database.

The indexing process analyses an image and attempts to find which objects are present in the image. The current preliminary implementation allows the analysis of images containing only one object view at a time.

The same process as described in the previous paragraph extracts a graph from an image and decomposes this graph into sub-graphs (fundamental figures). The fundamental figures thus obtained are considered pairwise together with their intersection. Obviously, fundamental figures which do not intersect are not considered by the indexing mechanism. Each such triplet indexes one or several objects in the database following the procedure described in Section 4. Hence, each triplet votes for a number of CVs. The CV receiving the largest number of votes is the identified one.

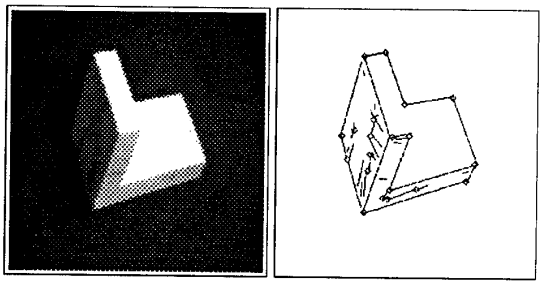

Figure 2: The image of an object to be recognized (left) and the extracteg binary graph (right).

Figure 2 shows the image of an object and the graph extracted from it. This image is not among the images used for constructing the database. The database views matching this unknown object are marked by a "*" on Figure 1.

We believe that the method described herein is attractive because of the mathematical and computational properties of the graph characterization scheme. The preliminary experimental results seem to confirm this attractiveness. Nevertheless, the indexing system that we implemented still suffers of two limitations:

- The decomposition of a graph into subgraphs is not optimal and it doesn't take into account geometric and structural constraints in order to select subgraphs such that each such subgraph describes a small number of characteristic views, and
- The indexing method cannot deal with images containing more than one object.

In fact these two limitations are tightly related. Consider a graph extracted from an image with several objects. The decomposition into sub-graphs must be such that each extracted sub-graph corresponds to one and only one object. This decomposition problem is in fact similar to the feature grouping problem and recent work in Computer Vision proposed a few practical solutions to the grouping problem that we intend to investigate.

References are available on request.