

3D models recognition from a 2D sketch

Jean-Philippe Vandeborre¹, Mohamed Daoudi², Christophe Chaillou¹

¹ LIFL : Laboratoire d'Informatique
Fondamentale de Lille
Bâtiment M3, Cité Scientifique
59658 Villeneuve d'Ascq - France

² GRIF : Groupe de recherche
Image et Forme - ENIC
Rue G. Marconi, Cité Scientifique
59658 Villeneuve d'Ascq - France

e-mail : vandebor@lifl.fr, daoudi@enic.fr, chaillou@lifl.fr

Abstract

Three dimensional models become more and more important in computer images. The problem which consists to recognize a three dimensional model in a two dimensional image appears. We are interesting in the recognition of simple polyhedral objects. To do that we have developed two interfaces. The user sketch interface allows a user to express his request graphically with adapted sketching tools. Another interface allows us to build the object database : the administrator interface computes characteristic views from a three dimensional object. These interfaces avoid the problem of processing bitmap images and its problems of segmentation : shadows, deleted lines, low resolution, lack of details etc... In this approach, both the objects to be recognized and the user sketch are represented by weighted graphs and indexed by a polynomial characterization of characteristic views in a hash-table.

keys words : image indexing, three dimensional image database, graphs, user sketch.

1. INTRODUCTION

Actually, and more in the future, images occupy a preponderant place in many domains like medicine, art, education, design. Professionals and the great public can find these images in large databases. Thus the problem is to match object features with image request features. One way to resolve that problem is to have textual attributes for each image in the database (models) and to be able to describe in the same way the request image. But this type of description is approximate, subjective and is confronted to the linguistic barrier. We must then find features which describe the shape of objects without the use of any spoken language. That puts the fundamental problem of primitives extraction.

S.Matusiak, M.Daoudi and F.Ghorbel [1] propose to use invariant descriptors which are independent under a general affine transformation to index two dimensional image database. B.Lamiroy and P.Gros [2] use similitude invariants. L.Morin, P.Brand and R.Mohr [3] use projective invariants. F.J.Stein [4] proposes to use super-segments which approximate a curve in a polygonal manner. Many methods are adapted for two dimensional image database.

In this paper we propose to resolve the problem which consists to find three dimensional models corresponding to a two dimensional sketch made by a user. This paper is organized as follows. The second section presents the characterization of our models in terms of graphs. Section three describes the indexing principle based on polynomial characterization of the graphs. Section four

focuses on the obtained results and the applications based on the method described in the above sections. Finally, the fifth section draws some conclusions.

2. MODEL CHARACTERIZATION

We have some hypothesis on the models : they are simple polyhedral objects as the one shown in Figure 1. This means that an object has, at most, three edges meeting at a vertex.

Hence, we can build an exhaustive catalogue of the possible appearances of the edges (twenty) of a polyhedral object. Each geometric node appearance will have a code associated with it. These codes will be used to weight the graphs.

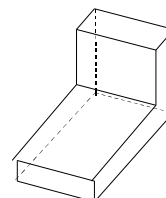


Figure 1. A simple polyhedral object in "L" shape

A three dimensional object is represented by a set of characteristic views and each characteristic view is described in term of a graph (P.J.Flynn and A.K.Jain [5]). A characteristic view is obtained by rotation of a virtual object around itself or by the movement of a virtual

camera around the virtual object. Of course, these two processes are equivalent. The number of characteristic views must be sufficient to describe the object completely (see Section 4). Each graph is weighted according to the geometry (the appearance) of his edges given in the exhaustive catalogue and then decomposed into subgraphs. An n vertices characteristic view graph is decomposed into n subgraphs. A subgraph is formed by a beginning node as well as the nodes that are at a distance less or equal to p edges away from this node (see Figure 2).

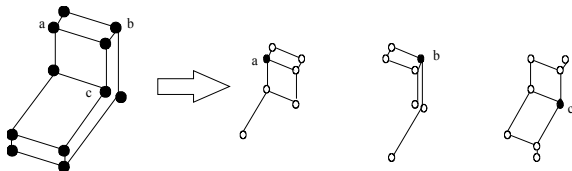


Figure 2. An example of a characteristic view of an object and a few subgraphs extracted from this view

With a small value of p , the obtained subgraphs are not discriminative and with large value, all the subgraphs tend to be the same as the characteristic view from which they are extracted. In this method, p is equal to three. Figure 2 shows a characteristic view of an "L" shaped object and three of its eleven subgraphs. At last, a polynomial characterization is computed for each subgraph. These polynomial characterizations will be used for the indexing process as described in the next section.

With this decomposition, we have a three layer database structure. Graphs (subgraphs) are extracted from characteristic views. Characteristic views are extracted from objects. An object may well have more than two characteristic views associated with it (see Figure 3). A subgraph does not necessarily belong to only one characteristic view. In the same way, a characteristic view can be extracted from two or more different objects. For example, every objects with a rectangular side have a common characteristic view which consists of a simple rectangle.

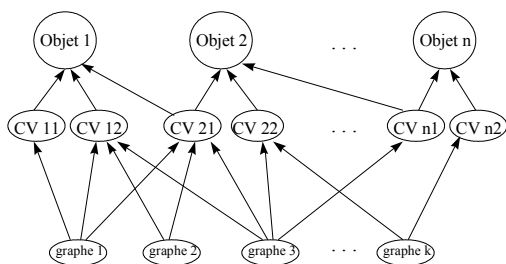


Figure 3. The database structure

Each unknown subgraph extracted from the user's sketch (which represents a drawn characteristic view) votes for a number of objects of the higher layer of the database structure. This process is repeated for each unknown subgraph belonging to the unknown sketch. The characteristic view that received the largest number of votes is the model that best matches the unknown view. According to the database structure there can be two or

more views that match with the unknown sketch. A good example is the case of a simple rectangular sketch made by the user. Every object with a rectangular side has a view that match with the sketch.

3. INDEXING

We are facing the problem of comparing a request object with many objects stored in a database. We cannot compare the unknown object with each object in the database. This process spends too much time. It is then necessary to introduce the notion of indexing function. This function allows to quickly access to the searched object without comparing the unknown object with each object in the database. It gives a quasi-constant access time to the database.

The indexing function computes a key (a code) for the unknown object and compare it with those stored in the database. Ideally two different elements to be indexed should have different keys. Practically this is not the case but a good code can limit the collision problem. When a code collision happens (ie. two different elements to be indexed have the same key) it can be solved by the key storage structure. The keys can be sorted or stored in a search tree or in a hash-table.

The graph matching problem is a very hard problem. As R.Horand and H.Sossa [6], we use the polynomial characterization, we reduce practically the graph matching problem to a polynomial equality problem.

A lot of books deal with graphs computation but it can be useful to remind that graphs can be represented by their adjacency matrices. The adjacency matrix of a graph G is a $n \times n$ square matrix where n is the number of nodes of the graph G . The adjacency matrix M of a graph can be defined as follows :

$$M_{i,j} = \begin{cases} weight(i, j) & \text{if there is an edge between nodes } i \text{ and } j \\ 0 & \text{if } i = j \text{ or if there is no edge between the two nodes} \end{cases}$$

where $weight(i, j)$ is the weight of the edge between node i and node j of the graph G according to code in the exhaustive catalogue of possible edge appearances.

The location of the coefficients in the adjacency matrix depends on the numbering of the nodes of the graph. If A_1 and A_2 are the adjacency matrices of two graphs G_1 and G_2 , we can easily see that G_1 is equivalent to G_2 if and only if there exists a permutation matrix P such that :

$$A_2 = P \cdot A_1 \cdot P^{-1}$$

To solve the problem of finding the permutation matrix, we use an algebraic characterization which is invariant by permutation of lines and columns between themselves : the polynomial characterization. It is computed as follows :

$$p(G) = \det(x \cdot I - M_G)$$

where M_G is the adjacency matrix of the graph G and I is the identity matrix.

Mathematically, the polynomial characterization equality is not sufficient for two graphs to be equivalent. But in practice, there is a few number of graphs with the same

polynomial characterization that are not equivalent. These cases are counterbalanced by the vote process described in Section 2.

In our method, we use a hash-table to store the polynomial characterization with links to the corresponding models. As we can see in Figure 4, characteristic views are not kept in the implanted structure because they are not useful in the recognition process.

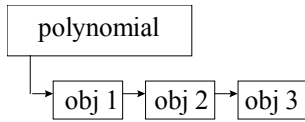


Figure 4. An element of the hash-table

The hash-code is computed with the coefficients of the polynomial characterization.

4. RESULTS AND APPLICATIONS

The theoretical part of the project (graphs and polynomial characterization) has been entirely developed and tested. Characteristic views were manually entered, point by point, to test the system. This is the kernel, the search engine of this recognition system but the interface part is very important in our approach.

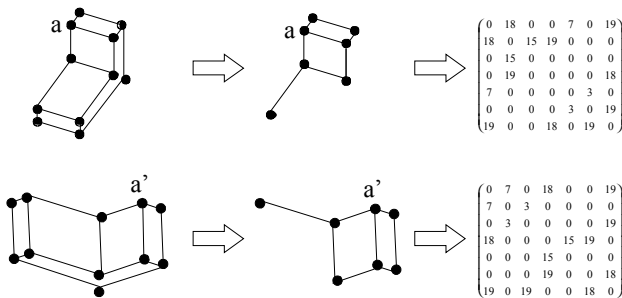


Figure 5. A view of an "L" shaped object

The Figure 5 shows a view of an "L" shaped object and another view in the same conditions but with a transformation by a rotation and a light scale. Except the transformation these are two same views of a same object. A subgraph is extracted from the two views from the same point location (*a* and *a'* on the Figure 5). These subgraphs give different adjacency matrices because of the different numbering of the graphs nodes. The polynomial characterizations given by the kernel of our method are equal for these two different matrices :

$$x^7 - 2014x^5 + 545476x^3 - 11364856x$$

Then they are recognized to be extracted from the same view of an object, which is true.

We are interesting in recognition of three dimensional models from a two dimensional sketch made by a user. That's why we have developed two types of interface : a user sketching interface (Figure 6) and a characteristic views construction interface (Figure 8).

With the sketching interface, the user can express his request in a graphical manner. He draws a view of the

object he wants to find in the data base. The interface offers tools to keep the sketch in the limits described in the hypothesis at the beginning of the second section of this paper :

- only straight lines can be drawn.
- ends of line are guided by a grid of cursors.
- no more than three lines meeting at an edge.
- no crossing lines which would form a four vertices edge.
- no isolated lines.
- closed shape.

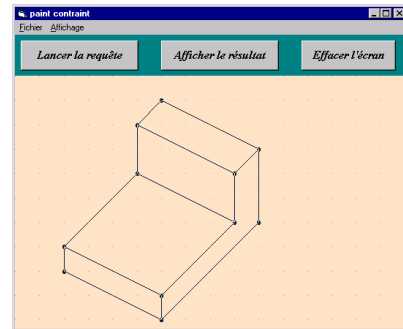


Figure 6. The sketching interface

The sketch can be saved in a file for future changes. Once the sketch is finished, a button must be pressed to start the recognition process. User's sketch is a request picture, then it is considered as a characteristic view. The final sketch is stored in a file to make the communication between the user interface and the search engine. This file consists of a list of points coordinates and links to others points. The kernel read this file, computes results and then write a file with them. Finally, the sketching application reads the results file and displays the results on the screen as a list of found objects and a representation of the three dimensional objects considered.

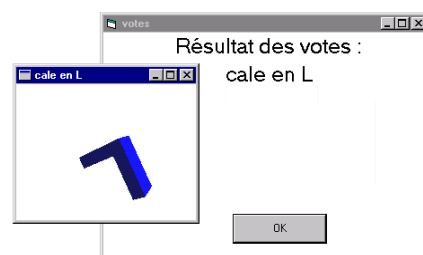


Figure 7. The result of a request

The second interface is a characteristic views construction interface. At the beginning of our search engine tests, we were forced to draw needed views on paper, estimate points positions and enter each set of points in a separate file (one file for a view) of a standard form. Thanks to the characteristic views construction interface all we have to do is building a three dimensional model and the interface displays it and makes it rotate around itself. When we think the obtained view is interesting, the interface write a standard characteristic view file to be opened by the search engine. All these files of characteristic views are used by the search engine to compute its hash-table.

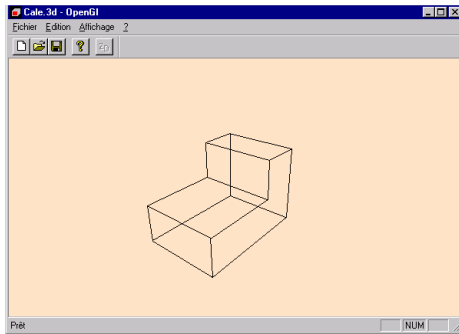


Figure 8. The characteristic views construction interface

5. CONCLUSIONS

Characteristic views are extracted from three dimensional objects and represented by graphs. This representation in characteristic views is independent of the spatial position of the three dimensional object. The objects can be rotated, translated or scaled, the graphs of a characteristic view will not change. The polynomial characterization of the graphs allows practically the reduction of graphs matching problem to a polynomial equality problem that is simpler (two polynomials are equals if they have the same degree and if all their coefficients are equals). We choose polynomial characterization for simplicity of implementation but there exist others characterizations based on polynomial like the second immanantal polynomial associated with an $n \times n$ Laplacian matrix of a graph described in [6].

Our method is application oriented. The user sketch interface allows the expression of a request in a graphical manner. There is no need to have a picture of an object. A bitmap picture needs to be processed and many errors of segmentation may occur. The user sketch interface gives tools to the user to express his request while keeping it in the limit of the model hypothesis described in Section 2. The characteristic views construction interface can be considered has a database administrator tool. This interface allows the creator or the administrator of the model database to add an object and create the characteristic views of this object without anything else than the three dimensional object.

6. REFERENCES

[1] S. Matusiak, M. Daoudi et F. Ghorbel, "Indexation d'une base de données d'images par une description de formes invariantes aux affinités", CORESA'97, 26-27 mars 1997.

[2] Bart Lamiroy and Patrick Gros, "Reconnaissance d'objets par indexation géométrique étendue", équipe MOVI, GRAVIR-IMAG & INRIA Rhône-Alpes, 1996.

[3] Luce Morin, Pascal Brand and Roger Mohr, "Indexing with Projective Invariants", SSPR'94

[4] Fridtjof Johannes Stein, "Structural Indexing for Object Recognition", a dissertation presented for the degree of Doctor of Philosophy (Computer Science), April 1992.

[5] Patrick J. Flynn and Anil K. Jain, "CAD-Based Computer Vision : From CAD Models to Relational Graphs" in IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, n°2, February 1991.

[6] Radu Horaud and Humberto Sossa, "Polyhedral Object Recognition by Indexing" in Pattern Recognition, Vol. 28, n°12, pp. 1855-1870, 1995.