Indexing and retrieval VRML models

Olivier Croquette, Jean-Philippe Vandeborre, Mohamed Daoudi, Christophe Chaillou

Indexing and retrieval VRML models

Olivier Croquette; Jean-Philippe Vandeborre and Mohamed Daoudi

ENIC Telecom Lille I / INT (MIIRE team)
Cité Scientifique - Rue G.Marconi
59658 Villeneuve d'Ascq cedex - France

ABSTRACT

In this paper we present a three-dimensional model retrieval system. A three-dimensional model is described by two invariant descriptors : a shape index and a histogram of distances between meshes. This work focuses on extracting invariant descriptors that well represent a three-dimensional model, and on combining theses descriptors in order to get a better retrieval performance. An experimental evaluation demonstrates the good performance of the approach.

1. INTRODUCTION

The use of three-dimensional image and model databases throughout the Internet is growing both in number and in size. Firstly, the development of the modeling tools, as well as the 3D scanners facilitates the creation of three-dimensional models. Secondly, 3D graphics hardware have become fast and cheap enough to allow three-dimensional data to be processed and displayed quickly on desktop computers. Finally, the World Wide Web is enabling access to three-dimensional models constructed by people all over the world, providing a mechanism for wide-spread distribution of high quality.

Techniques dealing with traditional information systems have been adequate for many applications involving alphanumeric records. They can be ordered, indexed and searched for matching patterns in a straightforward manner. However, in many three-dimensional databases applications, the information content of three-dimensional models is not explicit, and is not easily suitable for direct indexing, classification and retrieval.

Different kinds of approaches exist to represent three-dimensional object models. For example, the well-known 3D format for the Web, VRML[1], uses a hierarchy tree structure : an object is described as a tree where each node is a geometric information, an appearance information, or even a light position or a sound call-back, and so on. Many 3D tools are also able to use a simple data format : a set of flat polygons – often triangles – which can be described by listing their three-dimensional vertices and edges. The main interest of such format is that every other format can be exported to this simple one. This make it a very widely used format, which is why we are interested in indexing such three-dimensional models.

In this paper, we describe and analyze two methods for computing three-dimensional shape signatures and dissimilarity measures. The first shape index, introduced by Koenderink[2], is defined as a function of the two principal curvatures. This shape index is local but sensitive to noise. The second index, a distance index introduced by Osada[3], is based on a histogram of the distances between random points on the surface. This index is globally robust to the noise but does not take the local deformation into account.

Then, we propose to combine these three-dimensional shape signatures in different ways. The results obtained on the fifty three-dimensional models database show the effectiveness of our approach, and also the problems which are yet to be solved. In order to evaluate the performance of the proposed technique on each model of the database, we have used the recall/precision criteria. We have also used a classification matrix to see the global performance of our combining methods versus single shape and distance indexes used individually.

Further author information :
email : {vandeborre,daoudi}@enic.fr - tel : +33 3 20 33 55 77 - fax : +33 3 20 33 55 99
Send correspondence to J.Ph.Vandeborre - email : vandeborre@enic.fr - tel : +33 3 20 33 55 96

2. INVARIANT DESCRIPTORS

Indexing data is a way to find a suitable form for the information, for a given application. This form is called an index. In a search engine, the role of the index is to represent the original data in a very short way. Intuitively, this means that the index should be invariant to some geometric transformations of the object (translation, rotation, scaling), and should have a certain robustness to the noise.

Two indexes are used by our approach. The first one, shape index[2], is a local index and is invariant to geometric transformations, but it is not robust to noise. The second one, distance index[3], is a global index and is invariant to geometric transformations and is also robust to noise.

2.1. Local Descriptors

The proposed three-dimensional shape descriptor described in detail aims at providing an intrinsic shape description of three-dimensional mesh models. It exploits some local attributes of the three-dimensional surface. The surface and curvature notions are well developed in Ref. 4.

The shape index, introduced by Koenderink[2], is defined as a function of the two principal curvatures of the surface. The main advantage of this index is that it gives the possibility to describe the form of the object at a given point. The drawback is that it loses the information about the amplitude of the surface shape.

Let p be a point on the three-dimensional surface. Let us denote by k_p^1 and k_p^2 the principal curvatures associated with the point p. The shape index at point p, denoted by I_p, is defined as :

$$I_p = \frac{2}{\pi} \arctan \frac{k_p^1 + k_p^2}{k_p^1 - k_p^2} \qquad \text{with } k_p^1 \geq k_p^2. \tag{1}$$

The shape index value belongs to the interval $[0, 1]$ and is not defined for planar surfaces. The shape spectrum of the three-dimensional mesh is the histogram of the shape indexes calculated over the entire mesh.

The estimation of the principal curvatures is the key step of the shape spectrum extraction. Indeed, the spectrum extraction performances strongly depend on the accuracy of estimates. Computing these curvatures can be achieved in different ways, each with advantages and drawbacks; Ref. 5 proposes five practical methods to compute them. We choose to compute the curvature at each vertex of the mesh by fitting a quadric to the neighborhood of this vertex using the least-square method.

The parametric surface approximation is achieved by fitting a quadric surface through the cloud of the m points $\{(x_i, y_i, z_i)\}_{i=1}^m$ made at the centroids of the considered face and its 1-adjacent faces.

Let $S = (x, y, z) = (x, y, f(x, y))$ be a point of the three-dimensional surface defined by the analytical function f. Here, f is expressed as a second order polynomial :

$$f(x, y) = a_0 x^2 + a_1 y^2 + a_2 xy + a_3 x + a_4 y + a_5$$

where the a_is are real coefficients.

By denoting $a = (a_0, a_1, a_2, a_3, a_4, a_5)^t$ and $b(x, y) = (x^2, y^2, xy, x, y, 1)^t$, the previous equation can be expressed by using the standard matrix notations :

$$f(x, y) = a^t \cdot b(x, y)$$

The parameter vector $a = (a_0, a_1, a_2, a_3, a_4, a_5)^t$ is determined by applying a linear regression procedure. Given the data points denoted by $\{(x_i, y_i, z_i)\}_{i=1}^m$, the parameter vector corresponding to the optimal fit (in the mean square error sense) is given by the following equation :

$$a = (\sum_{i=1}^{N} b(x_i, y_i) b^t(x_i, y_i))^{-1} \cdot (\sum_{i=1}^{N} z_i b(x_i, y_i))$$

We can then calculate the two fundamental forms I and II :

$$I = \begin{bmatrix} <\vec{S}_u, \vec{S}_u> & <\vec{S}_u, \vec{S}_v> \\ <\vec{S}_u, \vec{S}_v> & <\vec{S}_v, \vec{S}_v> \end{bmatrix}$$

$$II = \begin{bmatrix} <\vec{S}_{uu}, \vec{N}> & <\vec{S}_{uv}, \vec{N}> \\ <\vec{S}_{uv}, \vec{N}> & <\vec{S}_{vv}, \vec{N}> \end{bmatrix}$$

$$\vec{S}_{uu} = \frac{\partial \vec{S}}{\partial u^2} \quad \vec{S}_{vv} = \frac{\partial \vec{x}}{\partial v^2}$$
$$\vec{S}_{uv} = \frac{\partial \vec{S}}{\partial u \cdot v}$$

The principal curvatures k^1 and k^2 are the eigenvalues of the Weingarter endomorphism W :

$$W = I^{-1} \cdot II$$

The shape index can now be computed by (1). We now have $NbFaces$ values, which can be represented as a histogram. This histogram (1024 intervals) is our first index for the object. Figure 2 shows an example of a such spectrum for the object quadru2 represented in figure 1.

2.2. Global Descriptors

Global descriptors are a mean to handle the object in its globality. This means that rather to be attached to the details of the object, we give more importance to its general aspect. The moments[6] are there the traditional mathematical tool. The moments present a main drawback : they rely on the object being described by a function, defined in the three-dimensional space, which could for instance associate 1 or 0 to each 3D point, depending on whether it is inside or outside the object. Very few objects are defined or convertible to such a function, which makes the moments method often impossible to use. To avoid this problem, we can also grab the Z-Buffer of an object's view, which consist of 2D data we could use to generate the moments. But we want to stay in the three-dimensional space, so we cannot use this method.

A new method has been recently proposed by Osada et al.[3], based on shape distributions. The main idea is to focus on the statistical distributions of a shape function measuring geometrical properties of the three-dimensional model. They are represented as histograms, just like for the local approach. The range of possible functions is very wide : from functions based on distances to functions based on angles, surfaces or volumes. The choice is limited by the properties the descriptor has to give. According to Osada et al.[3], the distance between random points gives good results compared to other simple methods.

We take two random faces of the objects, then we take two random points on those two faces. Finally, we compute the Euclidean distance between those two points. The method is iterated N times, N being big enough to give a good approximation of the distribution. We then build the histogram (1024 intervals) of all the values.

One could notice that the Euclidean distance, and thus this index, are not invariant by scaling for the moment. This is obviously because the Euclidean distance is not either. So, we have to normalize the spectrum before using it. One way to do it is to normalize the mean value of the distribution. We first compute this value :

$$MV(f) = \int_0^\infty x \cdot f(x) \cdot dx$$

The new histogram is then defined by :

$$f_{norm}(x) = f(\frac{x}{MV(f)})$$

The f_{norm} is the new distribution, invariant to scaling.

Figure 3 shows an example of a such distribution for the object quadru2 represented in figure 1.

Figure 1: Object quadru2 (5804 faces)

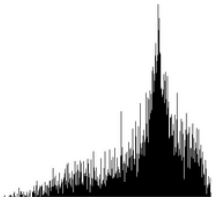

Figure 2: Shape histogram for the object quadru2

Figure 3. Distance distribution for the object quadru2

3. COMPARING DESCRIPTORS

Each object is now partially described by two histograms, which are used to compute a similarity between the objects, as shown on figure 4. The first thing to define is how the corresponding histograms of two objects will be compared, in order to find a distance between them.

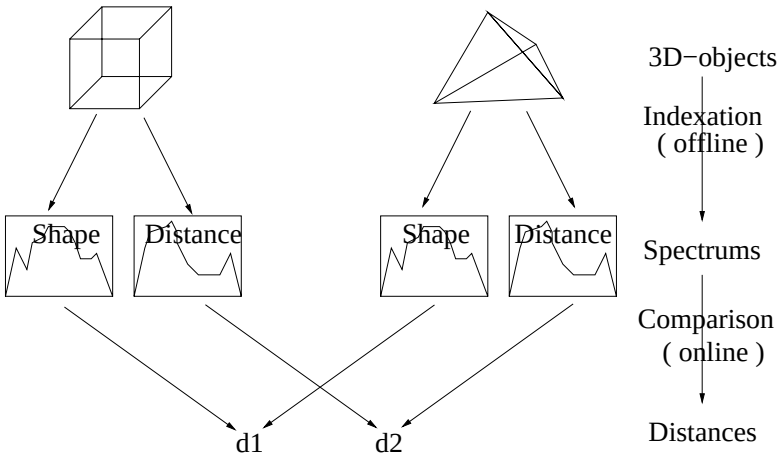

Figure 4: Working scheme

There are several ways to compare distributions : the Minkowski Ln norms, Kolmogorov-Smirnov distance, Match distances, and many others. We choose to use the L1 norm, as in Ref. 3, because of its simplicity and its good results :

$$d_{l1}(f1, f2) = \int_{-\infty}^{\infty} |f1(x) - f2(x)| \cdot dx$$

The histograms are being interpolated before anything else, in order to get rid of some problems, like quantization, noise etc. The interpolation is being made with 64 linear segments by the least-square method.

Then, a simple integration of the interpolations is made, giving a real number as a result. Notice that although the same method is applied to the two types of histogram, the numbers given are not comparable.

4. COMBINING RESULTS

We now have to join those two values, described in the above section, in one final mark. From now, please notice that the main goal of our approach is not to give an absolute distance between two objects, but rather to compare an object to a lot of others in a database, hunting the nearest object to the request. Our results can then be relative to our database.

We first compute the rank of each object, sorting them by decreasing "local distance", and next, by decreasing "global distance". We now have two integers, between 1 and $NbObjects$, named $Rank_s$ for the shape index, and $Rank_d$ for the distance index. Those values are merged in a single one, with a formula. The choice of this formula is very important, because the final results will greatly depend on it. Intuitively, there are different strategies :

- support the objects having good results with both approaches. This is what we call the "AND" method. It could be implemented with the following formula :

$$F = \left(1 - \frac{Rank_s - 1}{NbObjects}\right) \cdot \left(1 - \frac{Rank_d - 1}{NbObjects}\right)$$

- support the objects having good results with one approach, the other one having less importance. This is what we could the "OR" method. It could be implemented with the following formula :

$$F = 1 - \left(\frac{(Rank_s - 1) \cdot (Rank_d - 1)}{NbObjects \times NbObjects}\right)$$

- support the objects having good results with one approach, without caring about the other one. This is what we call the "MIN" method. It could be implemented with the following formula :

$$F = \frac{Min(Rank_s, Rank_d)}{NbObjects}$$

- use the mean of the two results :

$$F = \frac{(Rank_s + Rank_d)}{2 \cdot NbObjects}$$

This way, we now have one final real number, between 0 and 1, which represents the confidence one could have in the result.

5. RESULTS

5.1. Three-dimensional Models Database

The methods described above have been implemented and tested on a three-dimensional database containing fifteen models, classified by ourselves in seven classes. The models are simple meshes of approximatively 500 to 25000 faces, without any hierarchy structure.

Some of the objects are represented on figures 5 to 10. They have been extracted from a bigger database, in order to make some classes and to have a few objects per class. We choose some very different objects, so that the classes could be easily determined without ambiguity :

Class id	Content	Object range	Examples
A	8 airplanes	(1-8)	biplane, jets, 747...
M	5 misc	(9-13)	banana, statue, duck-toy, heart, "eight"
P	7 chess pieces	(14-20)	bishop, queen...
H	8 humans	(21-28)	men, women, dressed or not, alien
F	6 fishes	(29-34)	dolphins, shark...
Q	8 quadrupedes	(35-42)	triceratops, horse, cow...
C	8 cars	(43-50)	porsche, old car, cabriolet...

The "misc" class, being considered just as noise in our database, is not a real class. It is therefore meaningless to search for a "misc" object.

We have selected the objects which have a good mesh, objects with too many singularities have been eliminated. But some objects, generally the human-made objects, still have a mesh regularity problem : large flat surfaces are represented with a few number of big faces.

5.2. Evaluating the Method

One of the biggest problems relating to research and retrieval engines is their evaluation. The first difficulty is to define the word "similar", talking about three-dimensional objects. Does that mean that the objects are visually similar, or that they have the same usage ? Does a sailing ship look like an overcraft ? As far as the user is concerned, it depends. If the sailing ship is the only object at the user disposal, and that the user is looking for an overcraft, the answer would surely be "yes". But if the user is looking for all the fish boats, the answer would rather be "no". Actually, an overcraft does not visually look like a sailing ship, but the user thinks so because he/she knows that they both are boats.

As our database is classified, we assume, for each request, that the object from the same class as the request are relevant, and that the other are not. We can then use the recall-precision curves[7] to evaluate the algorithm. They give a mean to estimate the performance of the engine for some example requests extracted from the database. We can also use a classification matrix (even if the goal of our approach is not to classify the objects of a database), to have a global idea of the performance of our combining methods in comparison of the single indexes : shape and distance alone.

5.3. Experimental Results

The search engine finds good results very easily for most objects of the database. But a few objects do not give good results. This section shows some examples with the corresponding recall-precision curves. The first two examples are good working examples, and the third and fourth ones clearly show the problems in the way models are made.

5.3.1. A typical human object

A typical human object is represented on figure 6. We can see on the recall-precision curve (figure 11) associated with its research that the shape index gives excellent results. On the other hand, the distance index is much worse. The mixing methods are showing good results, although they are not perfect. We also note that the curves are crossing, showing that no method is always better than another.

5.3.2. A fish

The fish represented on figure 8 is interesting, because it shows the opposite case as the human. This time, the shape index gives bad results, and the distance index is the only trustworthy. We see that the mixing methods give good results, and even more, some are giving better results than the best single method (distance index). See figure 12 for the recall-precision curve.

This example shows the interest of the mixing methods.

Figure 5: Object chess1 (496 faces)

Figure 6: Object human6 (14946 faces)

Figure 7: Object human7 (14776 faces)

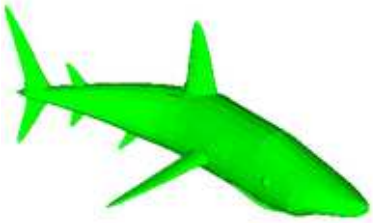

Figure 8: Object fish6 (23020 faces)

Figure 9: Object quadru6 (22258 faces)

Figure 10: Object car2 (6556 faces)

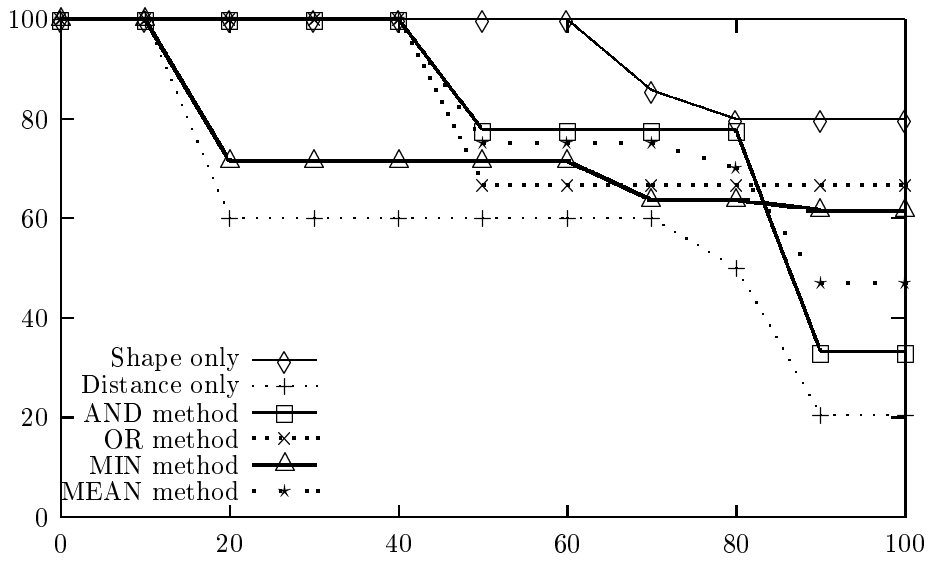

Figure 11: Recall-precision curve for human6

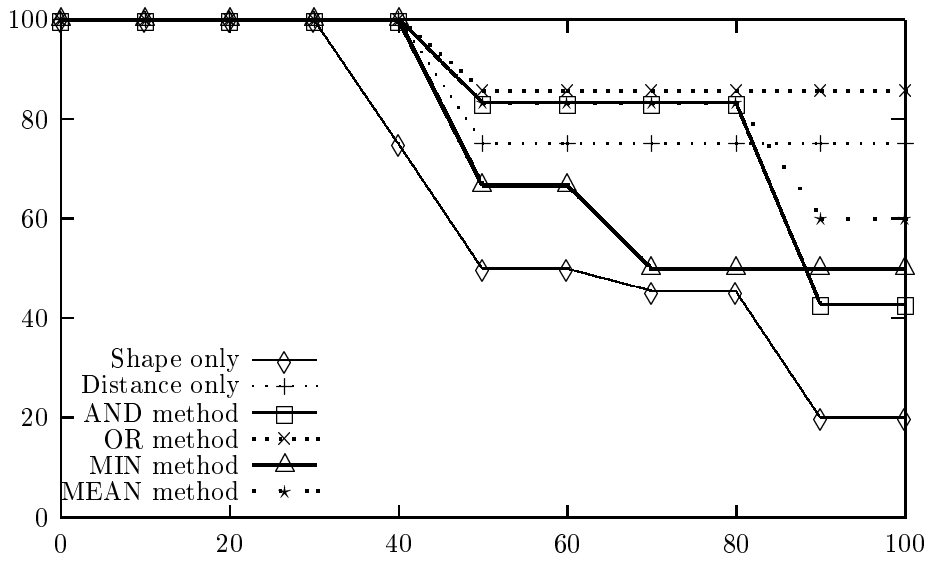

Figure 12: Recall-precision curve for fish6

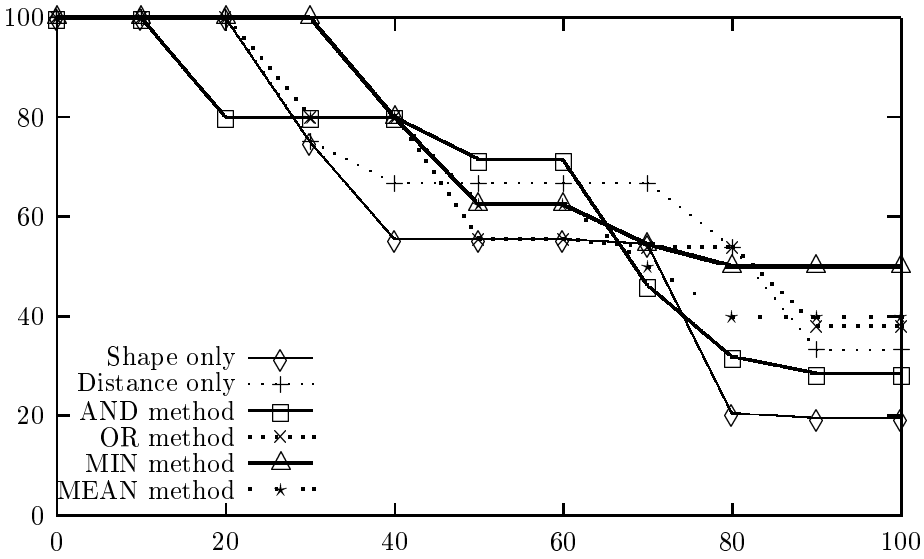

Figure 13: Recall-precision curve for human7

5.3.3. Another human object that does not work

The object represented on figure 7 is not really a human, but rather human shaped. As shown on figure 13, the two single indexes (shape and distance) are showing individually bad results, and so do the mixing methods. The problem in this case is that the morphology of the alien is quite different from the one of the humans, locally or globally. This example shows that we are missing another index, which could represent the fact that an object has two arms and two legs. Having this kind of structure (typically a hierarchy tree structure) from a mesh is not in the scope of our work : it is a difficult problem on its own.

5.3.4. A car object with a non regular mesh

Figure 10 shows the car the engine has been requested for. The recall-precision curve, as can be seen on figure 14, shows that the shape index has great difficulties to recognize the cars. This is typically the case of the human-made object, where the mesh is not regular because planar surfaces are represented by a few number of big triangles. Our algorithm cannot really compute a good shape index of such objects with a non regular mesh.

5.4. Classification

Although our aim is not to create a classifying engine, a good mean to evaluate the performance of the search engine is to see how it can classify the whole database. This section shows and analyzes some diagrams in this goal.

5.4.1. Classification matrix

The figure 17 shows a classification matrix. Each column represents an object of the database (column 1 corresponds to object 1 etc.) which has been used as a request for the search engine. Each little square shows how the object of the given row was ranked. If the object 23 was ranked last when the object 50 was requested, then the square at the intersection of column 50 and line 23 is white colored. The darker the square is, the best the rank is. This also means that the diagonal is completely black because each object is always the best result of its own request.

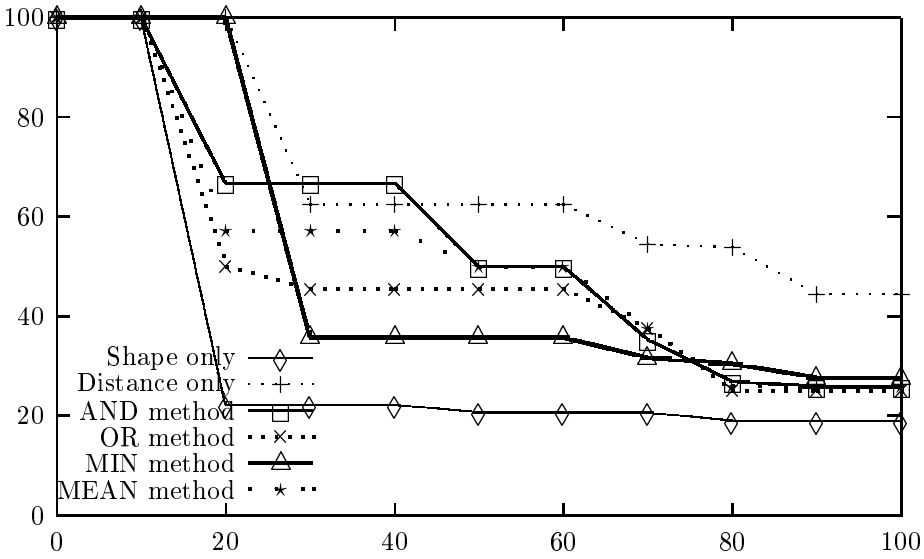

Figure 14: Recall-precision curve for car2

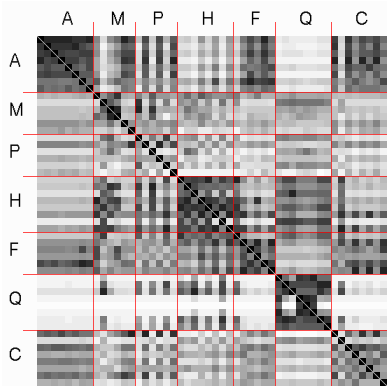

Figure 15: Classification matrix for the shape index

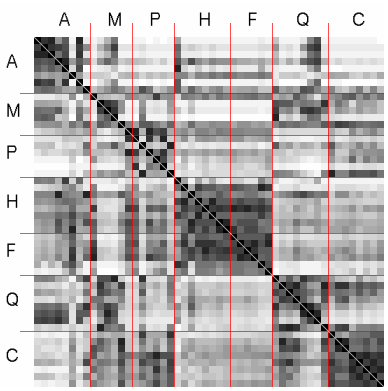

Figure 16. Classification matrix for the distance index

5.4.2. Single methods

The classification matrices for the single methods are shown in figures 15 and 16. Several comments can be made on these diagrams. First, both single methods have difficulties in retrieving some classes. The shape index mixes up airplanes and fishes. The reason for this is a common point between the classes : their objects are both locally cylindrical shaped. The distance index does not have this problem, thanks to the wings of the airplanes.

Similarly, the distance index has some advantages over the shape index. It has for instance no difficulty in classifying the cars, whereas the shape index mixes them up with airplanes.

This shows that a single index is not sufficient to provide a good search engine.

5.4.3. Mixing methods

First, we have noticed that the different mixing methods are quite equivalent in terms of results. We will therefore only analyze one method : the OR method, which seems slightly (but not always) better. Figure 17 shows the classification matrix for this method.

We notice that the method can correct some weaknesses of the single methods. For example, the shape index matrix shows some difficulties with the cars, and the distance matrix shows some difficulties with the airplanes. Those problems do not exist anymore with the OR method.

On the other hand, the shape index is quite able to classify humans and fishes, whereas the distance index has some problems with these classes. So, the OR method cannot totally classify these two classes correctly.

Nevertheless, the mixing method gives better results than single methods. It is easy to see on figure 17 that the diagonal of the matrix is dark colored, and the rest of the matrix is quite bright.

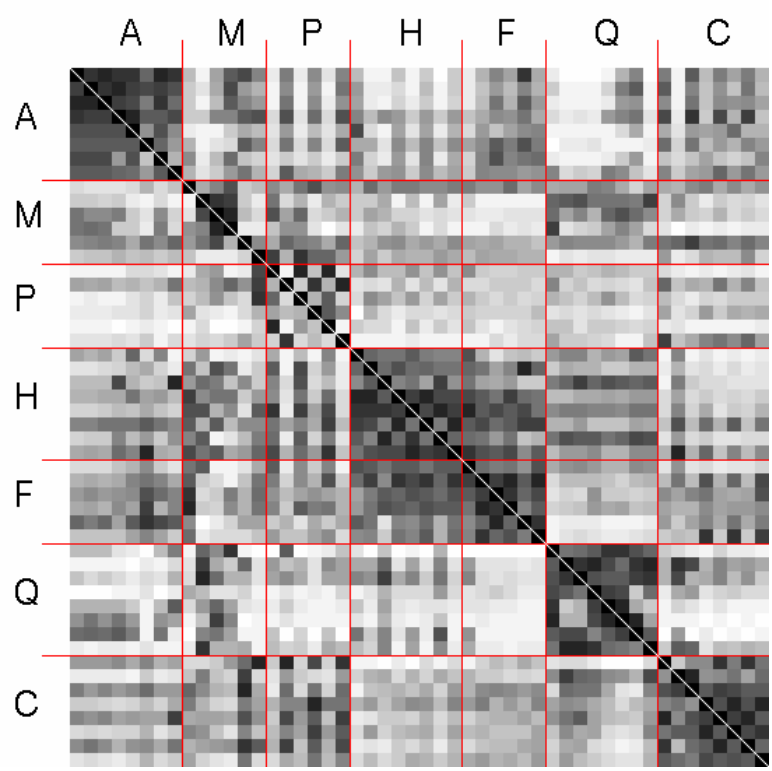

Figure 17: Classification matrix for the OR method

5.5. Computational requirements

In order to make a useful search engine, the search process has to be efficient. Our test machine was :

- CPU Pentium III, 666Mhz
- RAM 128Mo
- Linux system (kernel 2.2)

On this machine, the search engine needs 2ms by object in the database to give the scores. For our database of 50 objects, that means that it is only 1/10s of computing.

Please also note that the time needed to create the index is note included; it is typically 10-20s.

6. CONCLUSION

In this paper, we propose some algorithms which combine two three-dimensional invariant descriptors for three-dimensional mesh models. One is based on shape index and the second is based on the distribution of distances. We compare the results obtained by different strategies of combination. The results show that theses combinations are more performed than using one descriptor.

The results show the limitation of the approach when the mesh is not regular. Three-dimensional models come from different sources, so it is not always possible to control the mesh detail level. By the way, we are working to improve the approach with a preprocessing step which uses multi-resolution.

REFERENCES

1. J. Hartman and J. Wernecke, *The VRML 2.0 Handbook - Building Moving Worlds on the Web*, Addisson Wesley Developers Press, 1997.
2. J. J. Koenderink and A. J. van Doorn, "Surface shape and curvature scales," *Image and Vision Computing* **10**, pp. 557–565, October 1992.
3. R. Osada, T. Funkhouser, B. Chazells, and D. Dobkin, "Matching 3d models with shape distributions," in *Shape Modeling International*, May 2001.
4. M. P. do Carmo, *Differential Geometry of Curves and Surfaces*, Prentice-Hall, Inc., 1976.
5. E. M. Stockly and S. Y. Wu, "Surface parametrization and curvature measurment of arbitrary 3d objects : Five practical methods," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **14**(8), pp. 833–840, 1992.
6. F. Sadjadi and E. Hall, "Three-dimensional moment invariants," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2**(2), pp. 127–136, 1980.
7. I. H. Witten, A. Moffat, and T. C.Bell, *Managin gigabytes*, Mogran Kaufmann Publishers, 1999.