Directed Search In A 3D Objects Database Using SVM

Michael Elad, Ayellet Tal[1], Sigal Ar[1]
HP Laboratories Israel[2]
HPL-2000-20(R.1)
August 23rd, 2000*

E-mail: elad@ee.technion.ac.il, ayellet@ee.technion.ac.il, ar@ee.technion.ac.il

3D objects, search in databases, moments, support vector machine, quadratic programming

This paper introduces a content-based search algorithm for a database of 3D objects. The search is performed by giving an example object, and looking for similar ones in the database. The search system result is given as several nearest neighbor objects. The weighted Euclidean distance between a sequence of normalized moments is used to measure the similarity between objects. The moments are estimated based on uniformly distributed random generation of 3D points on the objects' surface.

An important feature of the search system is the proposed iterative refinement algorithm. Marking successful and failure decisions on previous results, this user's feedback is used to adapt the weights of the distance measure. The adaptation causes the successful objects to become nearer, and the failure decisions to become distant. Training the distance measure is done using the well-known SVM algorithm, which introduces robustness to the weight parameters. The above process may be repeated several times, accumulating 'Good' and 'Bad' examples and updating the distance measure to discriminate between the two with a maximal margin. This way, for each search and for each different user applying it, a different measure of similarity that reflects the user's desires and the searched object properties, is obtained.

Simulations done on a database of more than 500 objects show promising results. It is shown that with only 2-3 such refinement iterations, the search results are successfully directed towards better suited 3D objects.

[1] Ayellet Tal and Sigal Ar are with EE Department in the Technion-Israel Institute of Technology
[2] Technion City, Haifa 32000, Israel
* Internal Accession Date Only Approved for External Publication

1. Introduction

Recent years' progress in computer networking, digital storage, and computational power, turns various multi-media applications of manipulating text, sound, image, and graphics to practical. Among these applications, systems for searching specific sound files, images, or graphic objects on the World Wide Web (WWW) by their content become common [1-6]. Search by content will become far more desirable in future applications, when the number of such files available on a disk/network is expected to grow rapidly. Having many such files prevents the ability to manually assign textual attributes to each one.

As opposed to the conventional text-based search algorithms, the content-based search requires a deep understanding of the specific data representation. Indeed, content-based search algorithms share the need to define an effective feature space representing the data, and a similarity measure that is used upon these features. The effectiveness of the chosen features should be measured both in terms of their compactness (having as few as possible features for efficient search algorithms), and their ability to represent the essence of the data. The chosen similarity measure is required to faithfully describe similarities between objects, similar to the results of the human perception.

An additional issue, common in such search systems, is the refinement of the search results [1-6]. In many cases, the initial search may result with some inappropriate answers. It is therefore vital to supply the user with a refinement mechanism that directs the search to learn the user’s requirements and adapt to them. At this part of the system, computational learning algorithms may be used for such adaptation.

In this paper we introduce a new algorithm for searching a database of 3D objects. The database is assumed to contain many instances of 3D polyhedra, represented by a common three-dimensional description, such as polygonal meshes. The algorithm disregards textures and colors of the objects, and focuses on shape alone. The features, which are used for the actual search, are the moments of the 3D bodies. We introduce a method for robustly estimating and normalizing these moments' values, in a manner that removes spatial shift, scale or rotation dependence. The Weighted-Euclidean distance measures the distance between objects. For the initial search, all the weights have a unit value.

Giving the user the ability to give feedback to the system, ranking the search results, allows the search algorithm to refine its results. The search prompts the user with 10-20 possible answers, assuming that some of these are successful search results. By marking several of the results as matches and some as mismatches, the search system adapts to the user's preferences. Using the groups of 'Good' and 'Bad' examples, the adaptation is done by finding new weight values, which reflects the new information given to the system. The above process may be repeated again and again, collecting 'Good' and 'Bad' examples, until the user is satisfied with the search results.

Simulations of the proposed search algorithm, performed on a database of more than 500 objects, show promising results. It is shown that 2-3 refinement iterations suffice to direct the search towards better suited 3D objects.

This paper is organized as follows: In the next section we introduce the features and the similarity measure for the search process. Section 3 describes the iterative refinement algorithm, which uses SVM for training the weights of the distance measure. In section 4 we give several results and explanations as to how the system operates. We conclude this paper in section 5.

2. Features and Similarity Measure

This work concentrates on the of 3D objects' shape. Therefore, our choice of features is physical moments. These moments are computed for the surface of the objects, assuming that all objects are hollow. Denoting the object D surface by ∂D, its $[k_1, k_2, k_3]$ moment is given by

$$M[k_1, k_2, k_3] = \int_{\partial D} x^{k_1} y^{k_2} z^{k_3} dxdydz \,. \qquad (1)$$

One practical way to evaluate this integral is to compute it analytically for each facet and sum over all facets that comprise the overall object. An alternative approach, used in our work, yields an approximation of the moments. We draw a sequence of random 3D points (x,y,z) for each facet, where their position is uniformly distributed over its surface, and their number is proportional to the facet's area. Denoting the list of such points as $\{x_j, y_j, z_j\}_{j=1}^{N}$, the moment is approximated (up to some constant) by

$$\hat{M}[k_1, k_2, k_3] = \frac{1}{N}\sum_{j=1}^{N} x_j^{k_1} y_j^{k_2} z_j^{k_3} . \qquad (2)$$

Since we want the similarity measure to be invariant to spatial position, scale, and rotation, we need to normalize the moment values. The normalization starts by computing the first moments ($\hat{M}[1,0,0]$, $\hat{M}[0,1,0]$, and $\hat{M}[0,0,1]$), which represent the spatial object's center of mass, and removing them from the data-points by

$$\{x_j, y_j, z_j\}_{j=1}^{N} \Leftarrow \{x_j - \hat{M}[1,0,0], y_j - \hat{M}[0,1,0], z_j - \hat{M}[0,0,1]\}_{j=1}^{N} \qquad (3)$$

This way the all the different objects are centered on the point (0,0,0). At the second stage, using the updated triplets $\{x_j, y_j, z_j\}_{j=1}^{N}$, the second moments ($\hat{M}[2,0,0]$, $\hat{M}[0,2,0]$, $\hat{M}[0,0,2]$, $\hat{M}[1,1,0]$, $\hat{M}[1,0,1]$, and $\hat{M}[0,1,1]$) are computed. These values represent the body scale and orientation. Using these values as a 3 by 3 matrix S, an SVD (Singular Value Decomposition) [7-8] is performed, producing

$$\mathrm{U}\Delta\mathrm{U}^{\mathrm{T}} = \mathrm{SVD}\{\mathrm{S}\} = \mathrm{SVD}\left\{\begin{bmatrix} \hat{M}[2,0,0] & \hat{M}[1,1,0] & \hat{M}[1,0,1] \\ \hat{M}[1,1,0] & \hat{M}[0,2,0] & \hat{M}[0,1,1] \\ \hat{M}[1,0,1] & \hat{M}[0,1,1] & \hat{M}[0,0,2] \end{bmatrix}\right\}. \qquad (4)$$

The unitary matrix U stands for the rotation that this body went through. By multiplying the triplets $\{x_j, y_j, z_j\}_{j=1}^{N}$ with this matrix, we are actually rotating the body back to its canonic pose. The diagonal matrix Δ represents the scale in each axis, where the first axis holds the largest, and the third holds the smallest scales. By dividing all the triplets

$\{x_j, y_j, z_j\}_{j=1}^{N}$ by $\Delta(1,1)$, the body is re-scaled so that its largest scale becomes 1. Summarizing this stage, each triplet is replaced by

$$\begin{bmatrix} x_j \\ y_j \\ z_j \end{bmatrix} \Leftarrow \frac{1}{\Delta(1,1)} \mathrm{U} \cdot \begin{bmatrix} x_j \\ y_j \\ z_j \end{bmatrix} \tag{5}$$

As a third and last stage, we have to take care of flipped objects. Such situations do not conflict with any of the previous normalization stages, since a mirror image of an object has the same normalized moment values. This stage is done by counting the number of triplets on each side of the body (from the center), and flipping the body such that the larger count is on the positive side.

Figure 1-3 show several examples of objects and their normalization stages. As can be seen by the values on the axes, the first stage simply moves the object to the

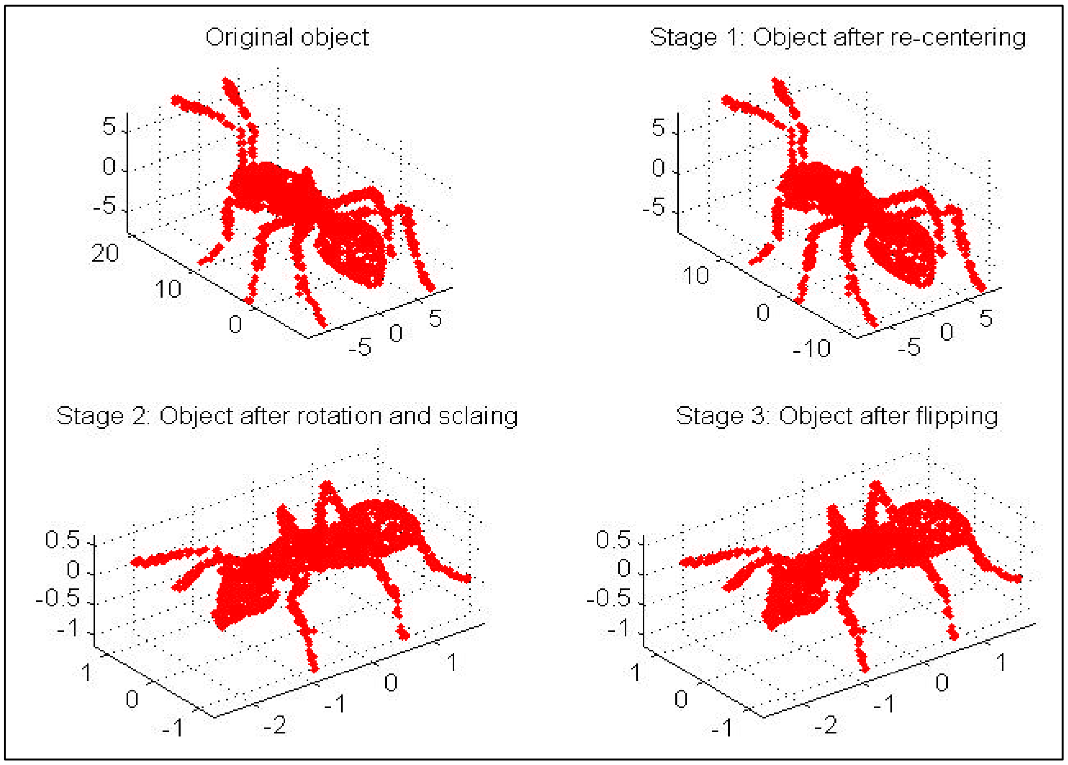

Figure 1: Ant - An example of the normalization stages

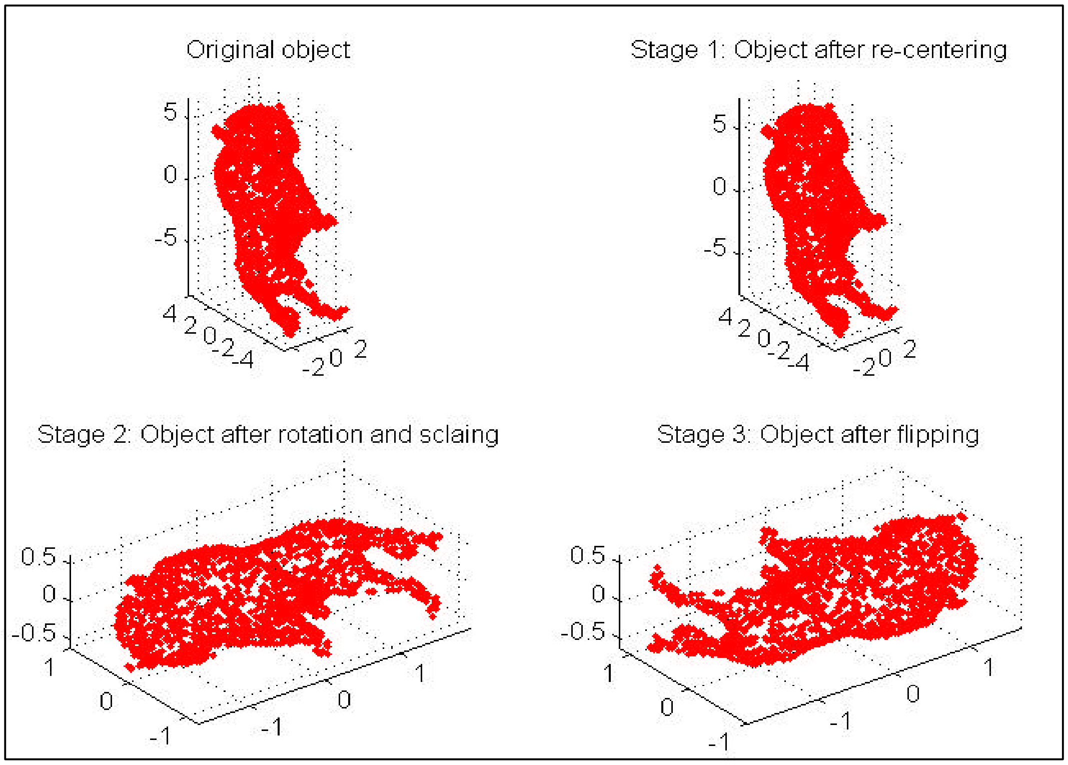

Figure 2: Buffalo - An example of the normalization stages

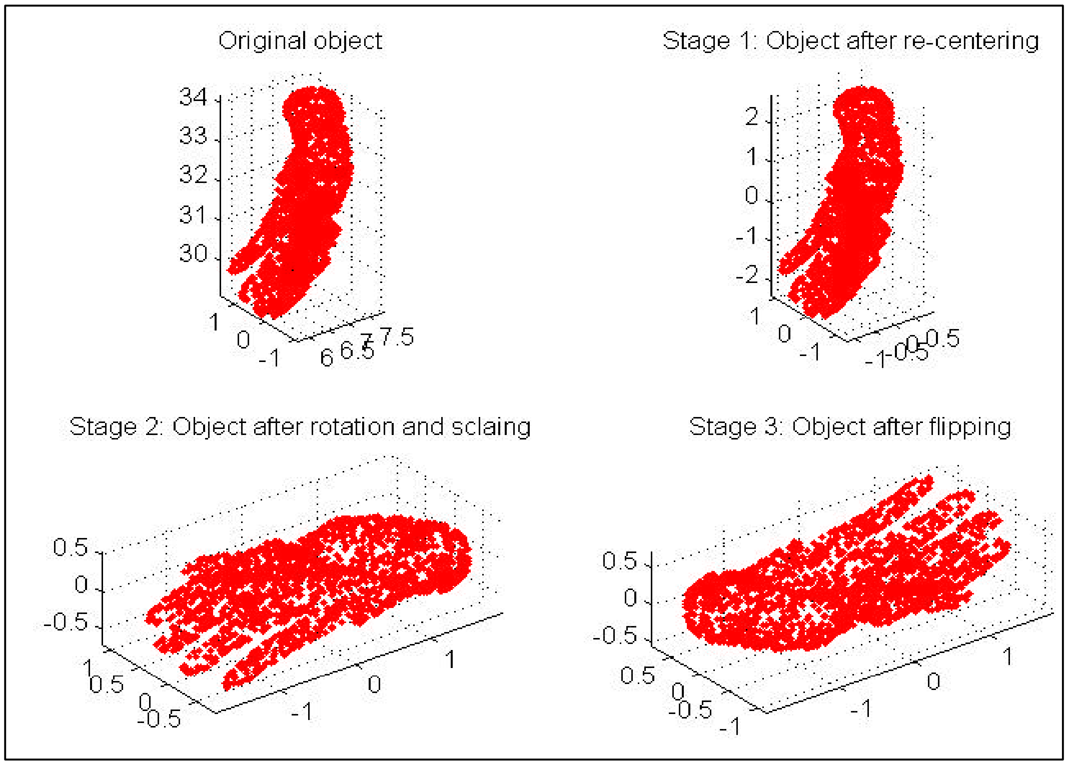

Figure 3: Hand - An example of the normalization stages

(0,0,0) point. The second stage performs a change of the pose of the object by rotating and scaling. The third stage in some cases (Figure 1) does not change a thing, and in others (Figures 2 and 3) it simply flips the object in all three axes simultaneously. In all three examples, 10,000 3D-points represent each of the objects, and only 2,000 of them are plotted in the graphs.

After applying these normalization stages, the moments are computed again. The feature vector contains a list of scalar values, which correspond to moments up to some pre-specified order. Of-course, there is no point in using the moment values $\hat{M}[1,0,0]$, $\hat{M}[0,1,0]$, $\hat{M}[0,0,1]$, or $\hat{M}[2,0,0]$ since the normalization fixes them to the values (0,0,0,1) respectively. An important benefit of the above normalization process is that the values after the normalization are close to zero, a property which helps in getting numerical stability.

As to the similarity measure, having two sets of moments, X and Y, of the objects D_X and D_Y, respectively, their Euclidean distance

$$d\{D_X, D_Y\} = \|X - Y\|^2 \ , \tag{6}$$

is used as the proximity measure between these two objects. In the next section we introduce parameterization of this distance measure, so that learning can be applied.

In order to conclude this section, we note that based on the features and the similarity measure we have defined, we can conceptually perform a search in the 3D-object database. The search starts by introducing an example object D_Z. The goal is to find objects in the database, which are similar to D_Z. The actual search is done by computing the features of the given example, Z, and computing its Euclidean distance to each and every feature vector of every object in the database. The L closest objects are shown as results to the user. A pre-processing stage of computing the features for all the objects in the database is imperative, for an efficient search.

3. Iterative Refinement

The similarity measure and the proposed features are a pretentious attempt to imitate the human perception. As such, it is quite reasonable to expect that searching with the above features and similarity measure may frequently fail. More specifically, if the system responds with L nearest-neighbor answers (typical value of L is 10-20), getting that large portion of these answers it unsatisfactory is quite understandable. By granting the user the ability to respond and rank the results, we are able to adapt to his/her reasoning, with the ability to refine the search results in subsequent stages.

Suppose that after the first search as described above, the user marks several of the results as successful, and some as failures. This way we have two lists, $\{X_k\}_{k=1}^{N_X}$ and $\{Y_k\}_{k=1}^{N_Y}$, of 'Good' and 'Bad' example results respectively. These lists are of feature vectors in $\Re^M$, i.e., there are M scalar feature values in each such vector. We would like to change the similarity measure in Equation (6) such that it exploits the above two lists. Instead of using a fixed form for the similarity measure, $d\{D_X, D_Y\}$, we propose the use of a parametric form $d\{D_X, D_Y, \theta\}$, where the θ are to be determined such that the 'Good' and 'Bad' example sets are reflected somehow. The value of the parameter vector θ is adapted as the search progress.

Since we chose the Euclidean distance measure, we 'parameterize' it by adding weights and a bias value

$$d\{D_X, D_Y\} = [X - Y]^T \mathbf{W} [X - Y] + b, \tag{7}$$

where **W** may be diagonal or full matrix, and *b* is a scalar bias value. In the following analysis our learning scheme assumes that **W** is a diagonal matrix.

The adaptation is obtained by using the sets $\{X_k\}_{k=1}^{N_X}$ and $\{Y_k\}_{k=1}^{N_Y}$ and requiring that the new distance between Z and each of the 'Good' examples should be small, whereas the distance between Z and the 'Bad' examples should be high. We formulate these requirements by defining the following set of constraints

$$\begin{aligned}&\forall k=1,2,\ldots,N_x, \quad d\left\{D_{X_k},D_Z\right\}=[X_k-Z]^T\mathbf{W}[X_k-Z]+b\le 1\\&\forall k=1,2,\ldots,N_y, \quad d\left\{D_{Y_k},D_Z\right\}=[Y_k-Z]^T\mathbf{W}[Y_k-Z]+b\ge 2\end{aligned}\tag{8}$$

These inequalities require that all the 'Good' examples get a distance to *Z* which is smaller than 1. The 'Bad' examples get a distance to *Z* which is higher than 2. This way we actually create a margin between the two sets. The above inequalities are linear with respect to the entries of **W**. Denoting the main diagonal of **W** as $W=\mathrm{diag}\{\mathbf{W}\}$, the above constraints may be re-written as

$$\begin{aligned}&\forall k=1,2,\ldots,N_x, \quad d\left\{D_{X_k},D_Z\right\}=[X_k-Z]^2W+b\le 1\\&\forall k=1,2,\ldots,N_y, \quad d\left\{D_{Y_k},D_Z\right\}=[Y_k-Z]^2W+b\ge 2\end{aligned}\tag{9}$$

where the notation V^2 means the square of each entry in the vector *V*.

Another important requirement is that the distance of *Z* to itself should be the smallest possible distance, which implies that the entries of *W* are all non-negative. Note that we do not require *b* to be non-negative, which may result with a non-metric similarity measure.

However, the above constraints alone are insufficient. In typical cases we have $N_x+N_y<<M$, which implies that there are more unknowns than inequalities. That means that there are numerous possible solutions {W, *b*} that satisfy the above requirements. Among all these possibilities, we would like to use a method that will be robust to the number of examples from each set, and to the other objects in the database. By minimizing the norm of the weights vector, *W*, we get such robustness. By forcing the distances to be lower than 1 and higher than 2, as in Equation (9), a margin is not guaranteed, since simply multiplying the weights causes an increase in the gap. The minimization of the weights' norm is directly related to the margin size, as already shown in SVM [9-10]. Thus, training is essentially the solution of the following Quadratic Programming (QP) problem:

$$\begin{aligned}
&\text{Minimize :} && \|W\|^2 \\
&\text{Subject To :} && \forall k = 1,2,\ldots,N_x, \quad d\{D_{X_k}, D_Z\} = [X_k - Z]^2 W + b \le 1 \\
& && \forall k = 1,2,\ldots,N_y, \quad d\{D_{Y_k}, D_Z\} = [Y_k - Z]^2 W + b \ge 2 \\
& && W \ge 0
\end{aligned} \tag{10}$$

This QP problem can be solved directly or through the dual problem [9-11]. If the number of constraints is much lower than the number of unknowns, i.e., $N_x + N_y << M$, the dual problem is easier to solve [9-11]. The use of the bias in the formulation is crucial, since it frees us from considering the boundary values, and therefore, choosing these values to be 1 and 2 does not loose generality. The proposed idea may be applied in several variations:

1. As described - collect the 'Good' and the 'Bad' sets.
2. Let the user define only the 'Good' examples, and refer to all the other results as 'Bad' ones.
3. Use the defined 'Good' and 'Bad' examples in a different manner, where for the 'Good' ones we require that the cross-distances between X_k, and X_j be smaller than 1.
4. Extend the previous idea by requiring in addition that the distance between every pair X_k, and Y_j should be larger than 2 as well.

In our application we chose to focus on the first simple choice, mainly because it is the simplest, and it works quite well.

To summarize this section, we have introduced a way to adapt the applied distance measure such that it becomes user-dependent and search dependent. Its training is done through the solution of a QP problem of moderate size, directly related to the SVM [9-11].

4. Results

In this section we show several search examples, and the evolution of their results as the iterative refinement is activated. In order to ease the visual interaction with the user, we created an icon image for each 3D object. These icons where created in advance, and are used for display purposes only. The icons where constructed by projection of the canonic 3D triplets $\{x_j, y_j, z_j\}$ (i.e., object after the normalization stage) to a 2D plane $\{x_j, y_j\}$ simply by removing the third coordinate, which represents the least energetic axis of the object.

In the first example, we used object number 9 - an Allosaurus Dinosaur, shown in Figure 4. Using a feature vector of 34 entries which contain up to the 4-th order moments, the first stage search results are given in Figure 5. There are 20 objects organized according to their distance from the Allosaurus Dinosaur. Note that for each result in this figure, its number in the database is given. We have chosen the objects [10, 11, 224] as 'Good' results, and [295, 538, 40, 410, 90, 318] as 'Bad' ones. The updated search results are shown in Figure 6. Assigning [48, 139, 49, 138] as new 'Good' examples, and [296, 75, 153, 340, 463, 480] as additional 'Bad' examples, the final results are shown in Figure 7. As can be seen, results improve both in terms of new objects that match the requirements, and their rank in the similarity measure. At the final stage, all the 7 dinosaurs are given as the first 7 closest objects. The 'Good' results in all examples and all results are marked by a Red smiling face.

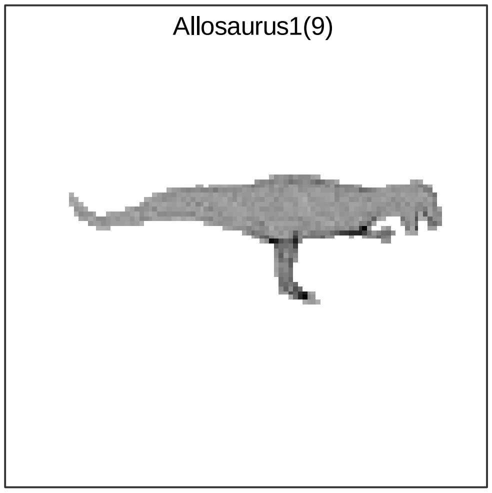

Figure 4: Search example #1 - Allosaurus (number 9 in the database)

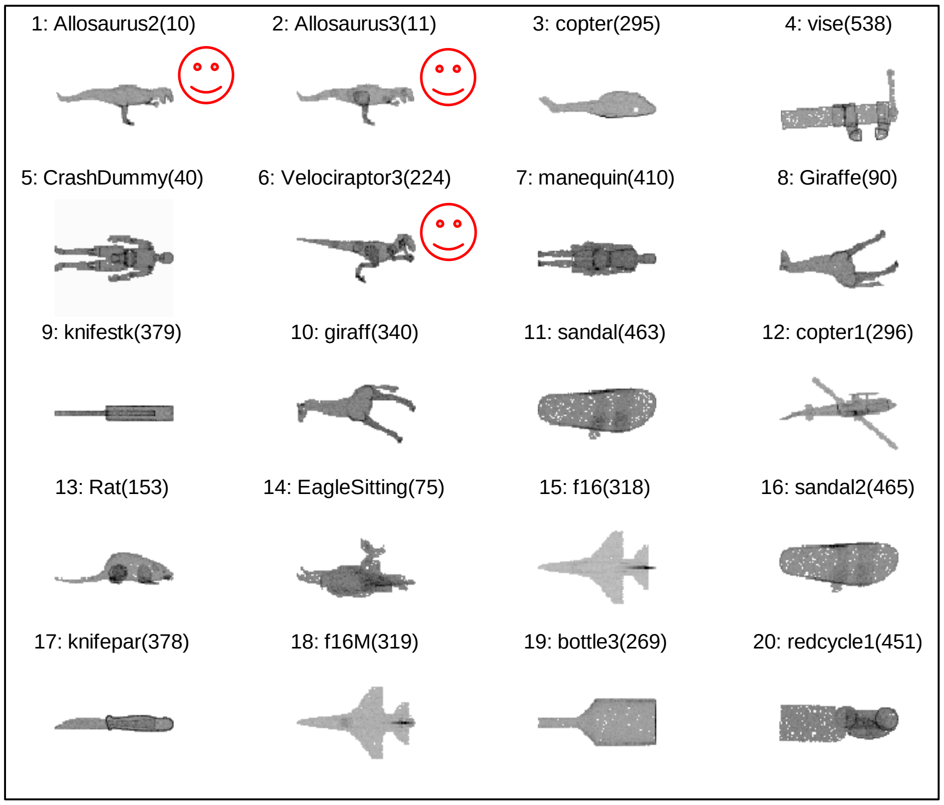

Figure 5: Search example #1 - The first search round results

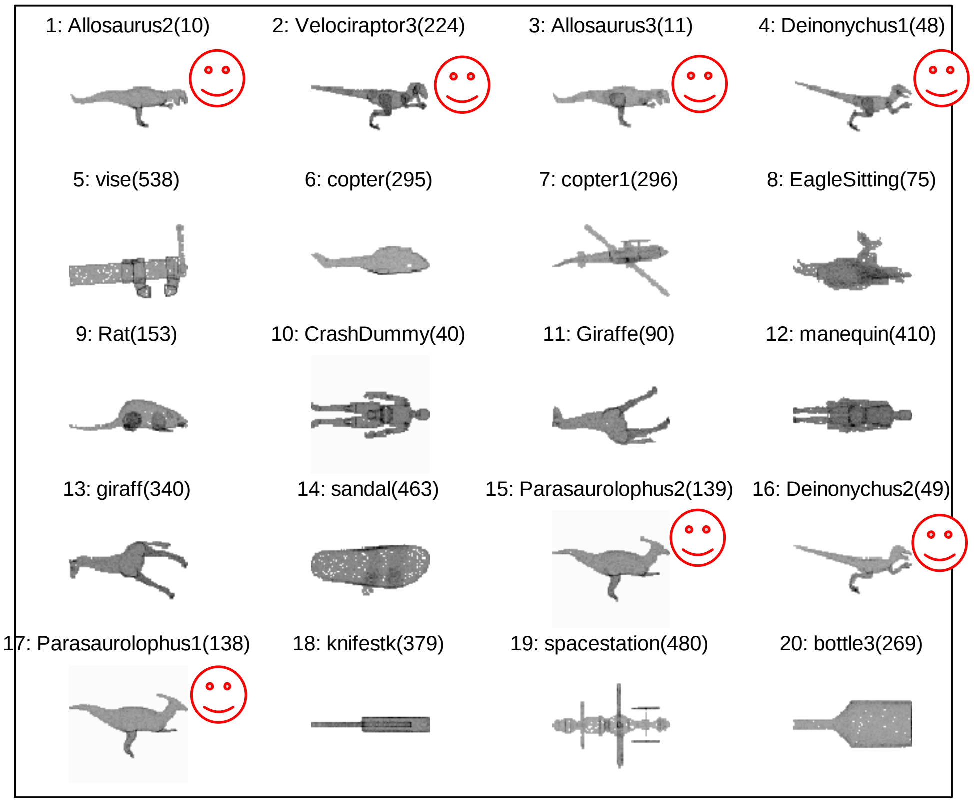

Figure 6: Search example #1 - The second search round results

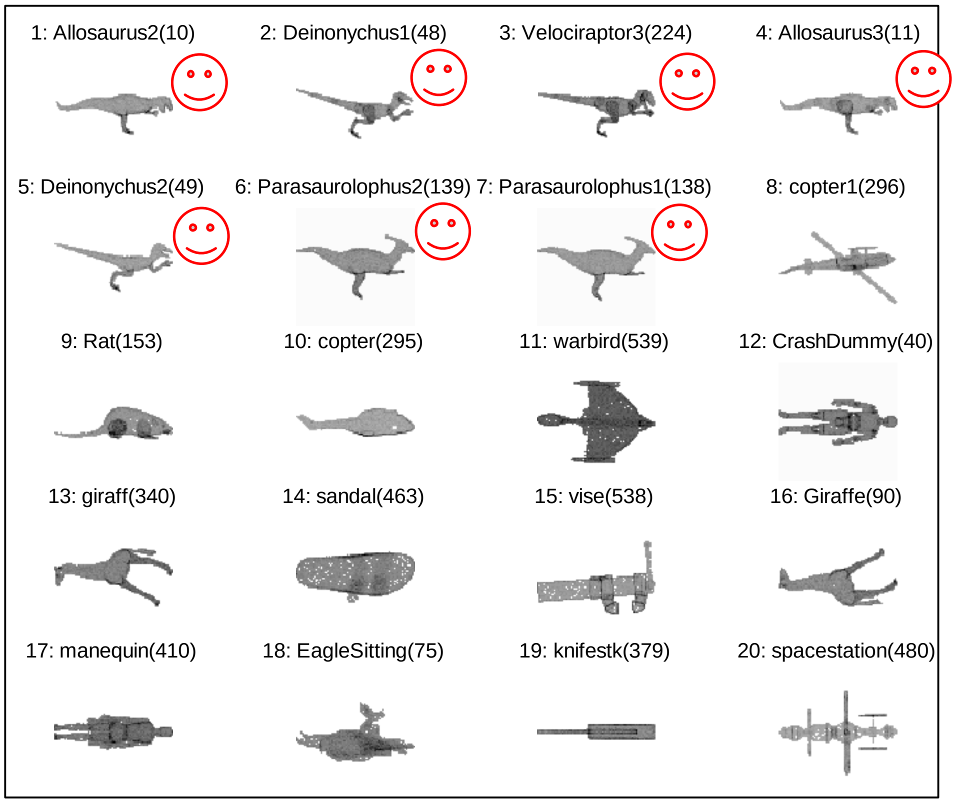

Figure 7: Search example #1 - The third search round results

Similarly, we performed a search for the object Shark (number 160) shown in Figure 8. The results are given in Figures 9, 10 and 11. Second stage is obtained by defining [161, 112] as 'Good' and [23, 177, 255] as 'Bad'. Third stage is obtained by choosing [162, 63, 226, 227] as 'Good' and [276, 318, 384, 491] as 'Bad'. Again, the improvement in the results is self-evident.

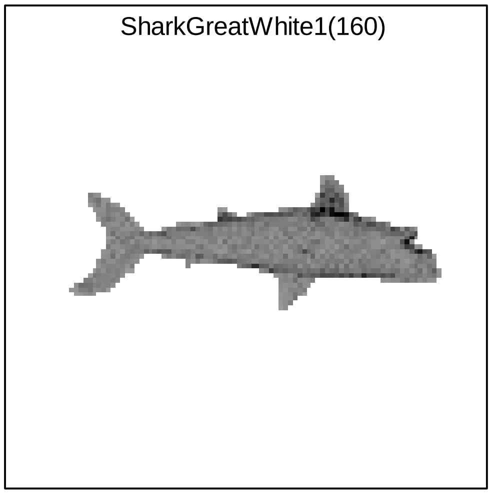

Figure 8: Search example #2 - Shark (number 160 in the database)

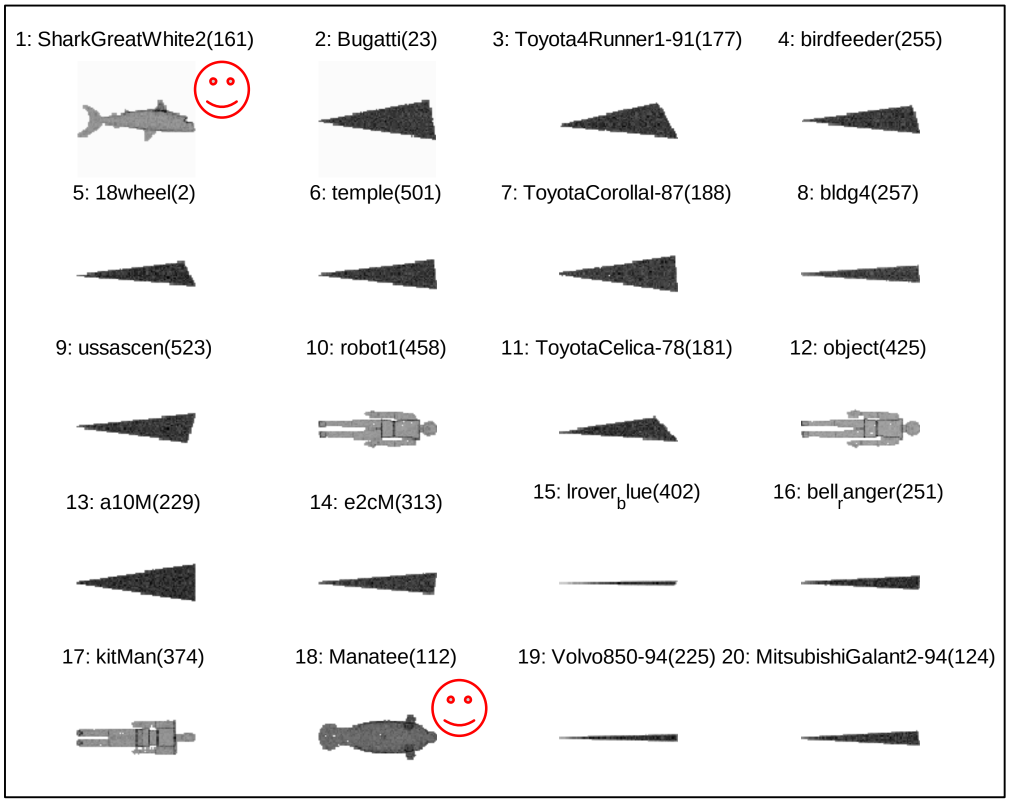

Figure 9: Search example #2 - The first search round results

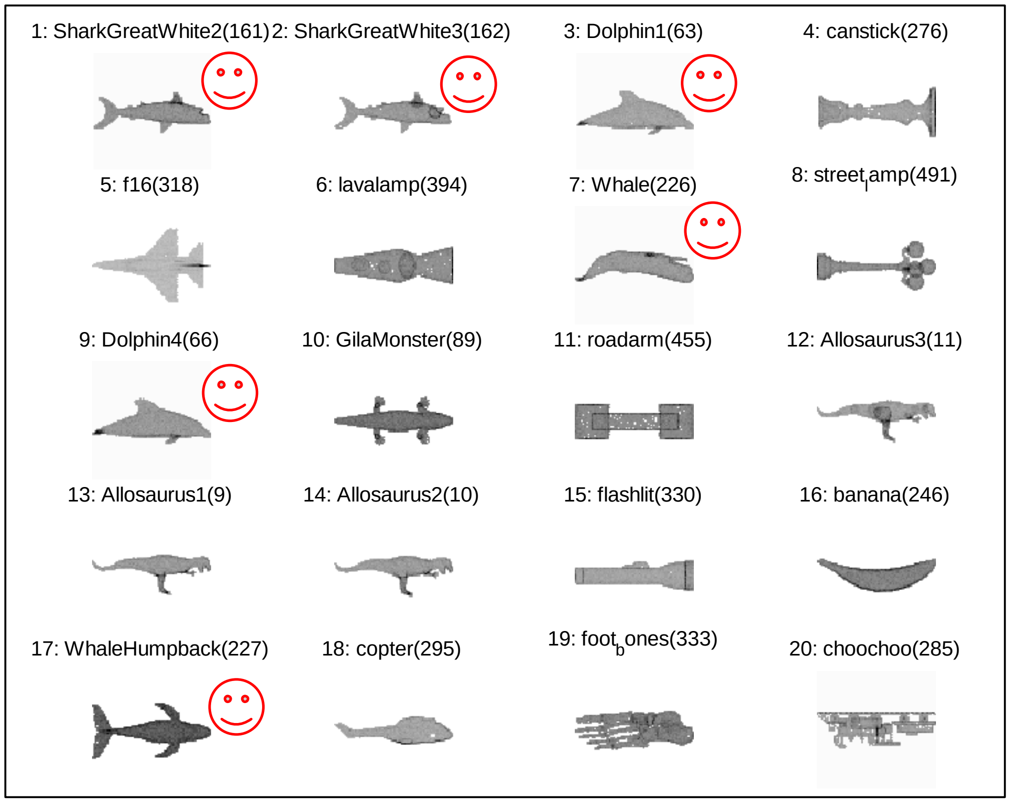

Figure 10: Search example #2 - The second search round results

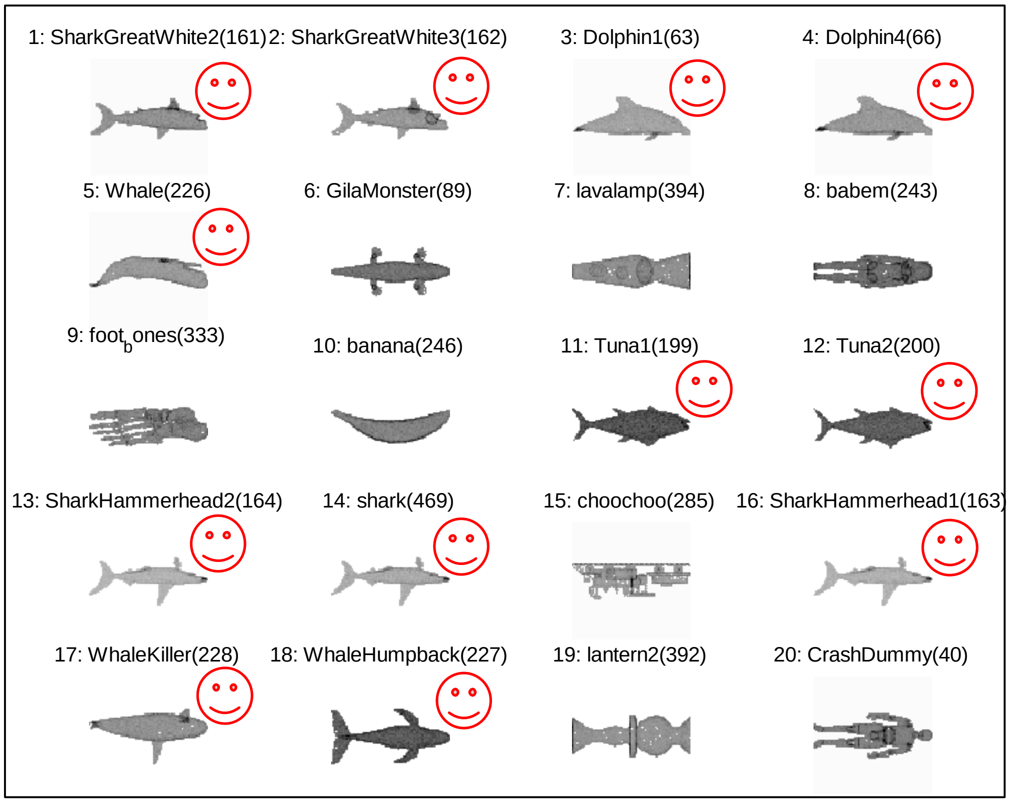

Figure 11: Search example #2 - The third search round results

As a last example, we present a search for the object Man (number 408). In this case only one refinement iteration is required. Figure 12 shows the example Man, and Figures 13 and 14 show the search results. By choosing [283, 243, 548, 459] as 'Good', and [199, 200, 181, 164, 180] as 'Bad', the results are improved markedly.

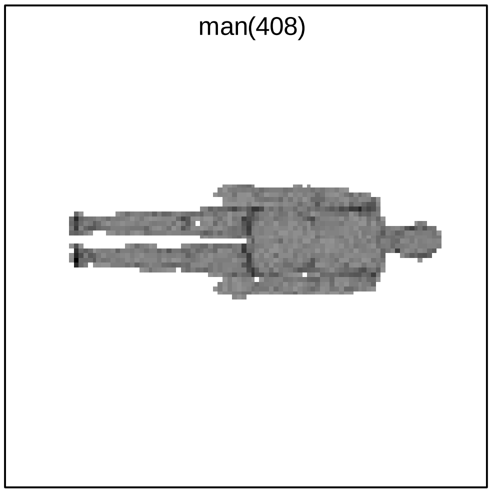

Figure 12: Search example #3 - Man (number 408 in the database)

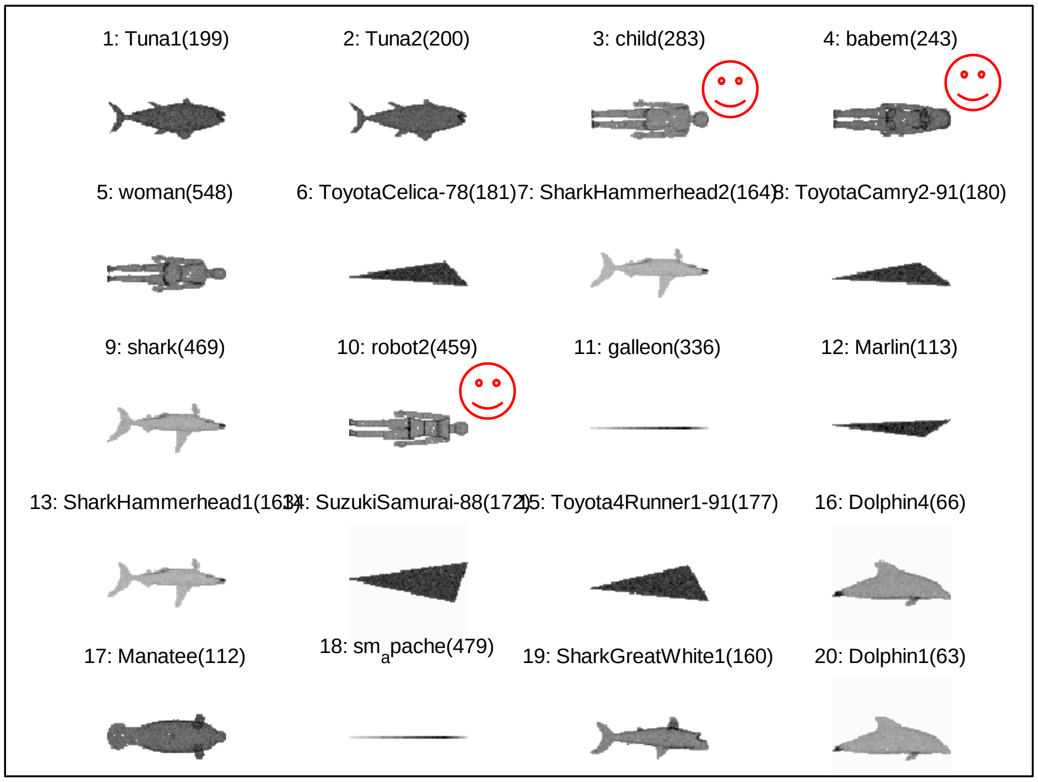

Figure 13: Search example #3 - The first search round results

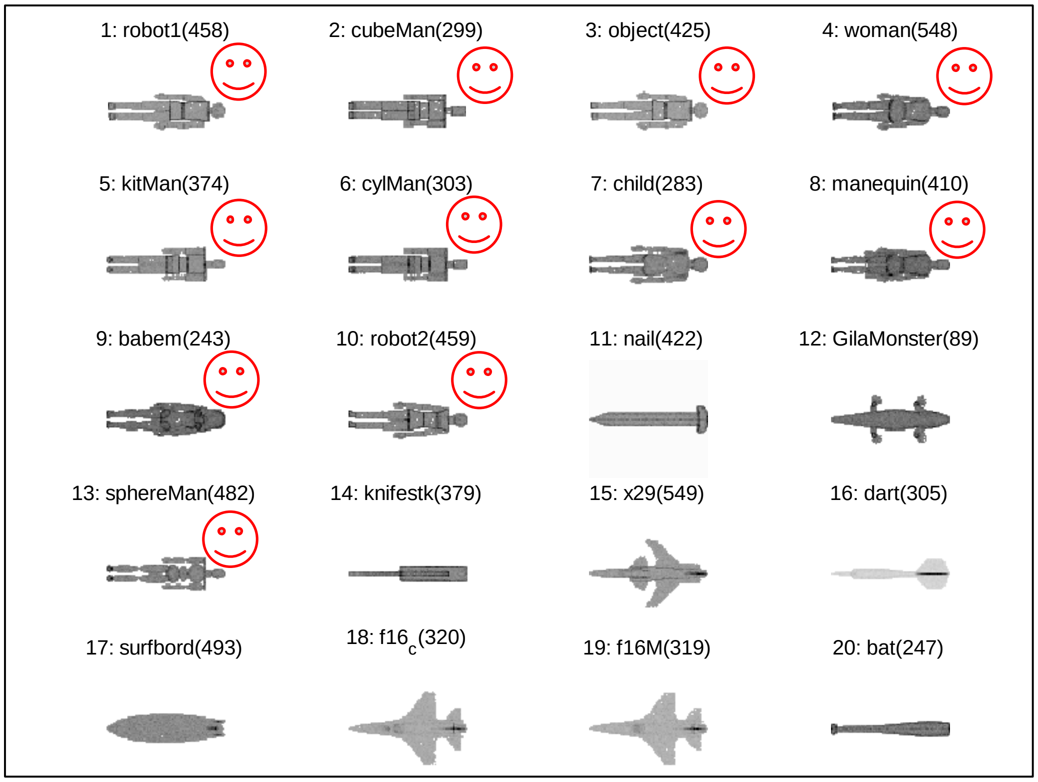

Figure 14: Search example #3 - The second search round results

5. Conclusions

In this paper we have introduced a novel application of content-based search in a database of 3D objects. Using normalized moments and a Weighted-Euclidean distance measure, the obtained nearest neighbor objects are used as first stage results. We have proposed a novel learning mechanism that enables adaptation of the applied metric, based on the user feedback. By marking some of the results as 'Good', and some as 'Bad', the weights are updated to reflect these user decisions. Training is done using the well-known SVM algorithm, which assures robustness. Simulations of several such search operations and their results through the refinement process are presented, to demonstrate the validity of the proposed learning scheme.

This paper concentrates on the dialog between the user and the application, and the way to adapt the search by this dialog. Further work is required in order to incorporate more features to the search paradigm, such as topological ones, colors and textures, etc. Also, section 3 lists several improvement possibilities for the training algorithm. Both these directions are currently under investigation.

References

[1] Y. Rubner, C. Tomasi, and L.J. Guibas, "A Metric for Distributions with applications to Image Databases", Proceedings of IEEE 6th International Conf. On Computer Vision, pp. 59-66, January 1998.

[2] J. Malik, J. D.A. Forsyth, M.M. Fleck, H. Greenspan, T. Leung, C. Carson, S. Belongie, and C. Bregler, "Finding Objects in Image DataBases by Grouping", Proceedings of IEEE 3rd International Conf. On Image Processing, pp. 761-764, September 1996.

[3] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz, "Efficient and Effective Querying by Image content", Journal of Intelligent Information systems: Integrating Artificial Intelligence with Database Technologies, Vol. 3, pp. 231-262, July 1994.

[4] C. Ta-Chun, A.L.P. Chen, and L. Chih-Chin, "Music DataBases: Indexing Techniques and Implementation", Proceedings of International Workshop on Multimedia Databases Management Systems, pp. 46-53, August 1996.

[5] E. Paquet and M. Rioux, "A Content Based Search Engine for VRML DataBases", Proceedings of IEEE Conf. On Computer Vision and Pattern Recognition, 1998.

[6] A. Del-Bimbo, M. Campanai, and P. Nesi, "3-D Visual Query Language for Image DataBases", Journal of Visual Languages and Computing, Vol. 3, No. 3, pp. 257-271, September 1992.

[7] Gilbert Strang, ***Linear Algebra and Its Applications, Saunders College Publishing***, 3rd edition, 1988.

[8] James W. Dammel, ***Applied Numerical Linear Algebra***, SIAM - Society of Industrial and Applied Mathematics, Philadelphia, 1997.

[9] Vladimir N. Vapnik, ***The Nature of Statistical Learning Theory***, Springer-Verlag New-York, 1995.

[10] C. Cortes and V. Vapnik, "Support Vector Networks", Machine Learning, Vol. 20, No. 3, pp. 273-297, September 1995.

[11] Dimitry P. Bertsekas, ***Non-Linear Programming***, Athena Scientific, Belmont Massachusetts, 1995.