Available online at www.sciencedirect.com
ScienceDirect
Applied Mathematics and Computation 196 (2008) 318–332
APPLIED MATHEMATICS AND COMPUTATION
www.elsevier.com/locate/amc

3D object indexing and recognition

Saliha Aouat *, Nacéra Laiche, Feryel Souami, Slimane Larabi

LRIA, Computer Department, University of Science and Technology, Houari Boumediene, Algeria

Abstract

In this paper, we address the problem of 3D object recognition from a single 2D image using models database. We propose a method based on geometric quasi-invariant features of the 2D images. We index the 2D images in a model base using a modified quad-tree technique that enhance the research process. The final vote that matches the 2D object image to the 3D object of the database is solved by a vector approximation file which overcomes the difficulties of high dimensionality by following not the data partitioning approach of conventional index methods, but rather as filter based approach.

Keywords: Model base; Geometric quasi-invariant; Content based indexing and retrieval

1. Introduction

An easy way to recognize an object in an image is to find the more reassembling object in database models. This problem can be considered as an indexing problem which consists in calculating an index key from a set of image features and comparing these keys to find similar images. The keys comparison gives a global similarity between images and the rate of matching images. Several image features can compose image index. We are interested in geometric ones.

It exists a large number of contributions to the indexing problem [1–4] with geometric features. Segments are particularly interesting features because of their robustness to noise and their connectedness constraint that reduces the possibility of false matches, since it is based on a topological reality in the image. They also have the properties to vary slightly with a small change in the viewpoint, and to be invariant under similarity transform of the image [13–22].

Since these features are used to match objects, we needed to use geometric invariant features. We considered therefore as 2D image features the intersecting segments and transformed them to couples of quasi-invariants features (ρ, θ) [11]. The set of (ρ, θ) couples find in the image are to be the object indexes.

* Corresponding author.
E-mail addresses: saouat@usthb.dz (S. Aouat), nlaiche@usthb.dz (N. Laiche), fsouami@usthb.dz (F. Souami), slarabi@usthb.dz (S. Larabi).

0096-3003/$ - see front matter
doi:10.1016/j.amc.2007.05.062

Our aim is to develop a complete recognition system based on matching geometric quasi-invariant indexes [8]. This induces that recognition and indexing is restricted to 2D–2D matching and recognition. Instead of directly interpreting 3D object information, we stored several 2D features (quasi-invariant) of a 3D object, and perform the object retrieval in the 2D indexes representation space [7]. Once the object has been identified, it is easy to backtrack the 3D information. To enhance the search process, the quad-tree algorithm [5] is adopted to code these indexes instead of the hashing table.

When a request image is to be compared to the images of the image base, this is done by comparing the request object indexes to those in the geometric model base. The method we propose begins by extracting the request image geometric features, and build the object indexes (quasi-invariants). We then search in the geometric model base the nearest indexes. A final vote step will identify the best matching object from the geometric model base (Fig. 1). This last step is performed by a vector approximation file [6] which overcomes the difficulties of high dimensionality by following not the data partitioning approaches of conventional index methods, but rather act as a filter based approach.

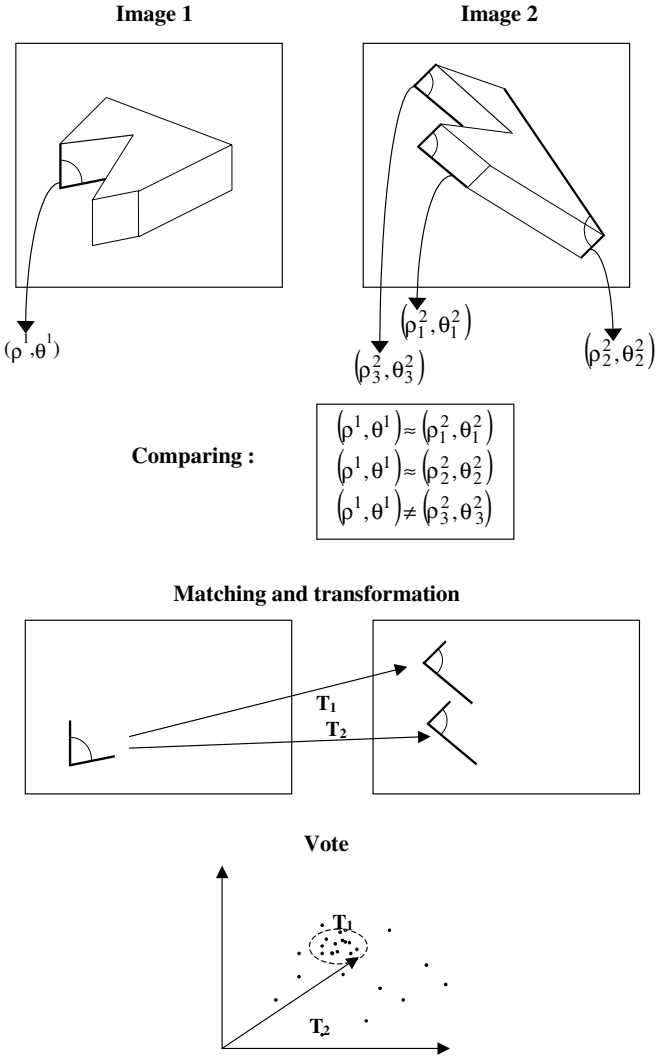


Fig. 1. Matching process.

2. Indexes analysis

2.1. Quasi-invariants and similarity

Intersecting segments are extracted for each image of the image base by a derivative operator. Intersecting segments are the geometric configuration used to calculate the quasi-invariants parameters for which a set of similarities are defined.

2.1.1. Geometric quasi-invariants

The quasi-invariants (ρ, θ) are defined as: the angle θ between the intersecting segments, and the segments length ratio ρ [9]:

$$\rho = \frac{\overrightarrow{a_0a_1}}{\overrightarrow{a_0a_2}} \quad \theta = \arccos \frac{\overrightarrow{a_0a_1} \cdot \overrightarrow{a_0a_2}}{\|\overrightarrow{a_0a_1}\| \cdot \|\overrightarrow{a_0a_2}\|}.$$

The (ρ, θ) couples find in each image are the image indices (Fig. 2). They vary slightly with a small change in the viewpoint, and are invariant under similarity transform of the image [12].

2.1.2. Similarity

The similarity $(k, \alpha, \overrightarrow{T})$ is composed by a homothety k, rotation α and translation $\overrightarrow{T}$. When considering two geometric configurations (Fig. 3), the similarity is expressed [10]:

$$k = \frac{1}{2}\left(\frac{l'_1}{l_1} + \frac{l'_2}{l_2}\right),$$
$$\alpha = |\theta'_0 - \theta_0| + \frac{1}{2}(\theta' - \theta),$$
$$T = \begin{pmatrix} x_b - k(x_a \cos\alpha - y_a \sin\alpha) \\ y_b - k(x_a \sin\alpha + y_a \cos\alpha) \end{pmatrix}.$$

2.2. Geometric indices composition

The indices used to build the geometric model base are composed by a set of geometric configurations parameters: quasi-invariant (ρ, θ), intersecting segments length and the image identifier (Fig. 4).

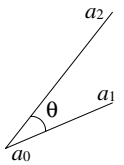


Fig. 2. Quasi-invariants (ρ, θ).

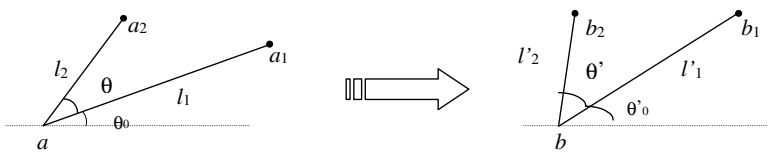


Fig. 3. Geometric configurations similarity.

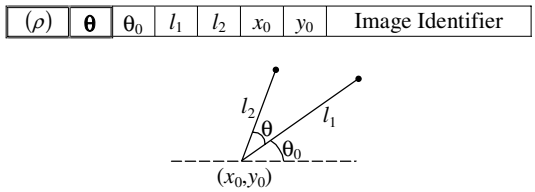


Fig. 4. Indices composition.

The indices are built in a way that two adjacent segments can not have the same indices. They express the similarity between two segments configurations.

2.3. Geometric indexes composition

In order to determine the indexing structure of such indices, we first analyzed the distribution of major indices component (θ, ρ).

2.3.1. (θ, ρ) distribution

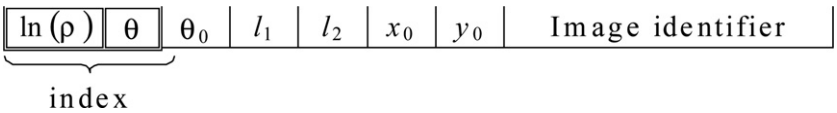
Fig. 5 shows the θ distribution. The θ value equal to $\pi/2$ is a value frequently encountered in the geometric indices.

Fig. 6 shows a non-uniform distribution of the indices number in regard to the component ρ.

To perform a better indexing, the indices components have to follow a uniform distribution [4]. Therefore, a logarithmic function is considered $\ln(\rho)$. Fig. 7 shows that values beyond the bound $[-4, +4]$ are not to be considered because the indices are not relevant.

2.3.2. Indexes

The geometric indices are composed as follows, and the geometric model base indexes are $\ln(\rho)$ and θ.



For each image in the image base, sets if geometric indices are calculated to build the geometric model base indexes.

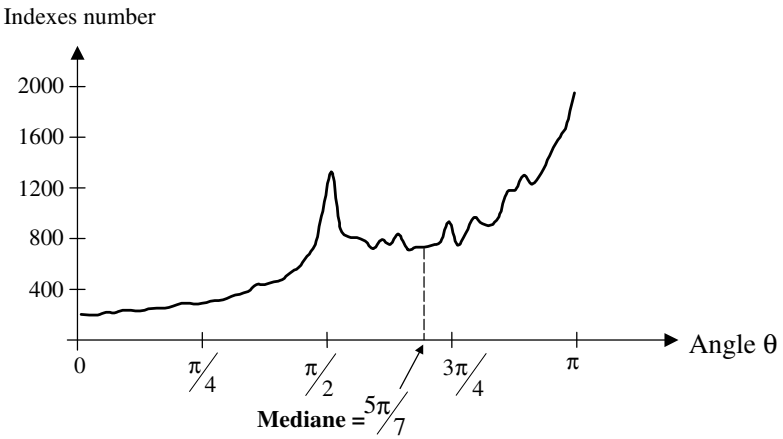


Fig. 5. Indices distribution versus θ values.

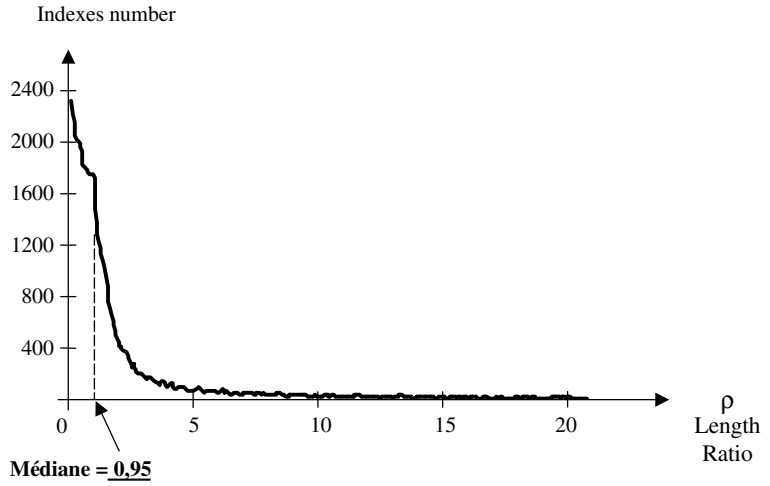


Fig. 6. Indices distribution versus ρ values.

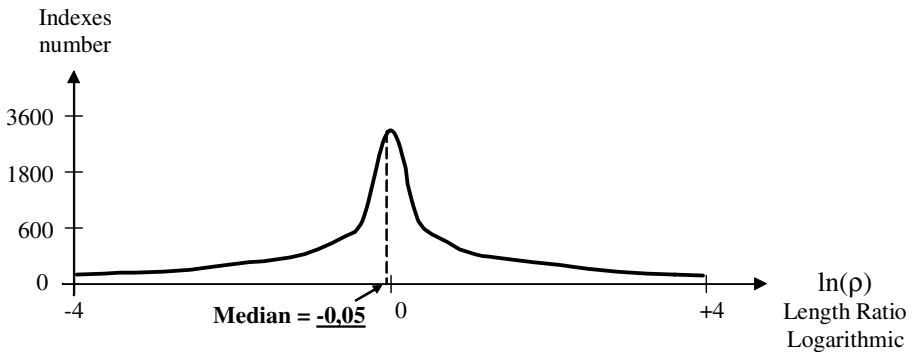


Fig. 7. Indices distribution versus ln(ρ) values.

3. Geometric models database

Building the geometric model base is an indexing process which is an off line process. On the other hand, the retrieval process is an on line process which good performances is much more of our concern. Therefore, the geometric model base is designed to fulfill the retrieval process needs: velocity and accuracy.

For efficient query processing in large data sets, it is necessary to build an index structure that reduces the size of the retrieved set needed to answer a query. The general approach is to prune the search space and eliminate irrelevant data objects without accessing the corresponding features subspace. To achieve that, the data set is organized such that only a partial representation (e.g. 2 out of d dimensions) of each object is examined. This had lead to several index structures proposed in the literature. Examples include Kdb-tree [29,30], hb-tree [28], R-tree [26], R*-tree [23], SS-tree [30], Tv-tree [27], X-tree [24], Pyramid Technique [5], Hybrid technique [25]. Various algorithms for similarity searching have been developed in conjunction with these indexing mechanisms.

In our case, the geometric features lead to a two-dimensional indexes space. Therefore, we have chosen an adapted quad-tree indexing structure.

3.1. Indexes quad-tree structure

The quad-tree indexing consists in splitting the index bidimensional space in equal squares. Each tree level corresponds to a split part. The tree nodes correspond to index space split point. The indexes space can be split in sub spaces of different sizes which is adapted to non-uniform data (Fig. 8).

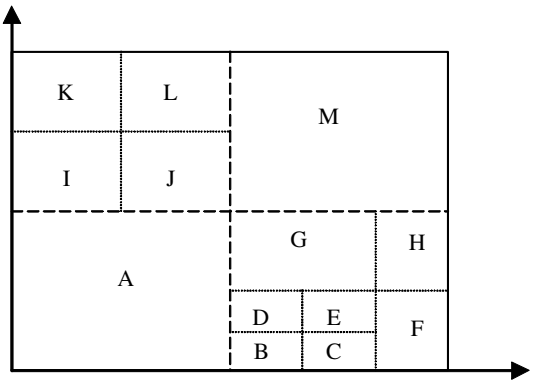


Fig. 8. Indexes structure.

To improve the retrieval process, we create four structures for indexes storage:

- A structure composed by the quad-tree leaves.
- A structure composed by the quad-tree nodes.
- A structure composed by the objects models.
- A structure composed by the different object aspects.

3.2. Building geometric model base

The geometric model base is meant to represent polyhedric objects. For each object, several images of different points of view are considered. For each image, we extract points and segments that provide the adjacent segment configurations needed to calculate the quasi-invariants and therefore, the geometric indices, for each of which the indexes are calculated and added to the geometric model base structure.

The indices are stored in two structures. The indexes are in the quad-tree structure (primary memory) and the indices features in a secondary memory. The quad-tree nodes contain only the indexes, which make the model base access for retrieval process faster.

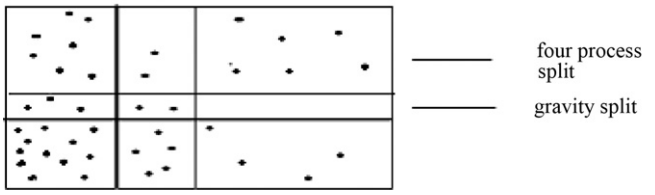


Fig. 9. Split processes.

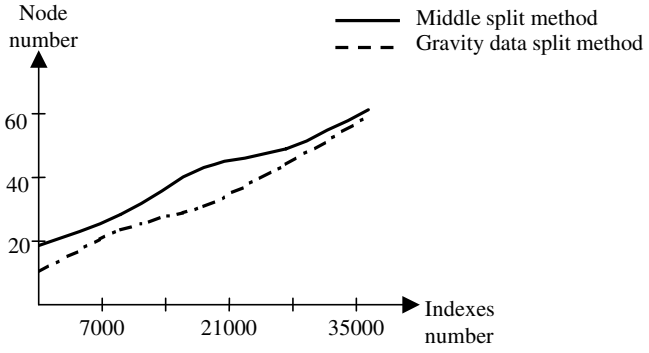


Fig. 10. Split process improvement.

We wanted to fill the quad-tree structure in a way that avoids empty leaves. It is well known that for each leave split, when adding data in the structure, the splitting process create four new leaves from the initial one, leading to bad data distribution in the leaves. This misdistribution will again lead to another split in the new leaves with dense data. To avoid such situation, the split process has to create new leaves with equivalent data distribution. Therefore, we propose, instead of the conventional split in four processes to a gravity center driven split process (Fig. 9).

Fig. 10 shows that the center driven split process, regardless to the indices number, improve the quad-tree composition and the ''partitions'' are balanced. This improves the retrieval process time.

4. Retrieval process

The retrieval process is a two step process. In the first one, the 3D object image is processed in the same way done for the images of the image base. The request image indices are then compared to the ones in the geometrical model base to find the 3D object they refer to. The second step is then a match step. The matched indices from the geometric model base lead to the images of 3D objects that can be represented in the request image.

To match the request image indices with those in the geometric model base, we calculate similarities of each request indices with its neighbors in the indices space.

4.1. Comparing indices – similarity estimation

The similarity is calculated for each request indices neighbors. To do so, a recursive search process in the tree structure is needed. The usual search zone is elliptical. When going down in the tree, we look for the square ''partition'' included in the ellipse (Fig. 11). In this ''partition'' lie the most similar indices to the request indices. To improve the request access time, we used, instead of the elliptic zone, a square zone.

At the end of this search process, we obtain a list of possible matches between request image indices and databases indices. The corresponding similarities forms clusters in the four-dimensional representation space. By analyzing these clusters, we find the best request match by a vote step process.

4.2. Vote and VA-file

The vote step is used to eliminate all not matching objects of the geometric model base. The vote is performed in the similarity representation space. Due to quasi-invariants properties, dense clusters in representation space represent the good matches. The best match is the one with the higher density cluster.

Finding the dense clusters is a search for neighboring data. Conventional approach to nearest-neighbor search uses some form of multidimensional access method (such as quad-tree). In these methods work the data space according is used for search queries.

Unfortunately, while these methods perform well for a low dimensionality, performance degrades as dimensionality increases. To overcome this well known ''dimensional curse'', we used to analyze the clusters density, a vector approximation file descriptor (VA-file) [6]. The VA-file is an array of compact geometric approxima-

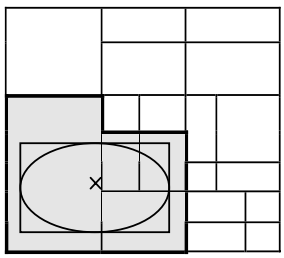


Fig. 11. Square and elliptic research zones.

tions to data points. Each approximation determines a lower and an upper bound on the distance between its data point and the query. These bounds are sufficient to filter most of the vectors from the search.

Therefore, the VA-file is composed of two sets of data: one for indexes and one for their approximations. The last one is kept in the primary memory to keep a quick access.

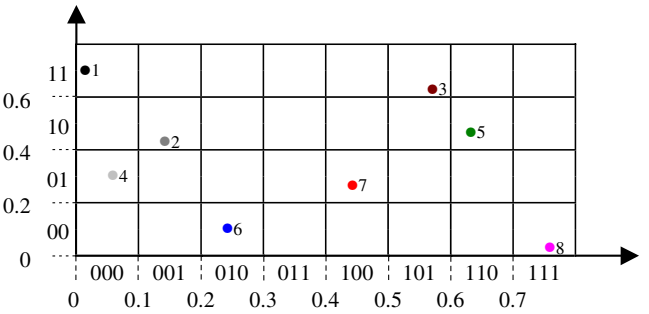
While creating the index (Section 2), the VA-file geometric approximation split in 2 each data space dimension d_i (this is coded on d_i bits). In the same way, the d space dimension are split in 2^b hypercubes ($b = \sum d_i$). The indexes are then sequentially approximate by the VA-file approach. The index approximation is determined by the bounds it lies between. The approximation corresponds to number of the interval it lies in. To avoid poor or empty partitions, intervals with less than one index are ignored (Fig. 12).

During the retrieval process, the request image indexes are approximate by the same process. The VA-file answering request seeks then for the closest hypercubes to the request. A distance is measured in between hypercubes, and those for which the distance is bigger then a predefined ε value are rejected (Fig. 13). This step is more like a filter that eliminates all non-suitable hypercubes. This improves the request process regarding to the sequential one used in tree research.

Our vote process is a three steps process:

- Initialization: for each request image, an empty hypercube counters list, labeled with hypercube numbers, is created.
- For each similarity, we determine the interval number it lies in ($i^{\text{ème}}$ dimension that contain its $j^{\text{ème}}$ component). This gives us the hypercube number that contains these similarity parameters.
- If the hypercube number already appears in the initial list, its counter is incremented. Other wise, the hypercube number is added to the list and it counter set to 1.

When all the similarities have been treated, the matched indexes refer the more resembling 3D objects in the geometric model base. The corresponding images (of the image base) are selected and sort out by the vote number.



		VA-File	Vectors
1	●	000 11	0.02 0.72
2	●	001 10	0.15 0.45
3	●	101 11	0.58 0.67
4	●	000 01	0.06 0.52
5	●	110 10	0.63 0.50
6	●	010 00	0.25 0.13
7	●	100 01	0.45 0.30
8	●	111 00	0.77 0.05

Fig. 12. Approximation VA-file.

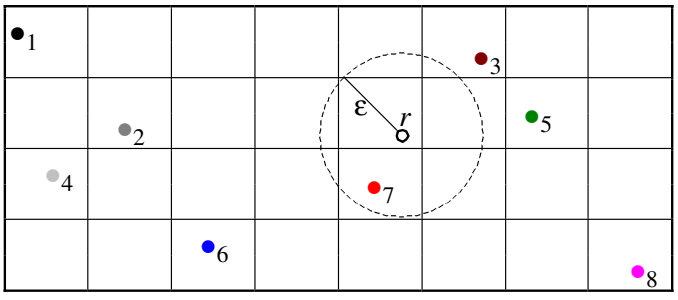


Fig. 13. Request process.

5. Evaluation

5.1. Image base

We considered 28 polyhydric objects and several images (856 images) of each object taken under different points of view (average object rotation is 20°). Identical views have been eliminated of the image base to avoid redundancy (Fig. 14). We then extract the geometric features: the intersecting segments.

5.2. Geometric model base

When indexing the geometric model base, several tests have been conducted. In particular, we were concerned by the leaves composition.

5.2.1. Leaf fullness

We conducted this test first on the hole base (49 764 indexes), and then on half the base (24 380 indexes), and finally on the third of the base (16 652 indexes). Leaves composed by more than 400 indexes present the best fullness rate (Fig. 15).

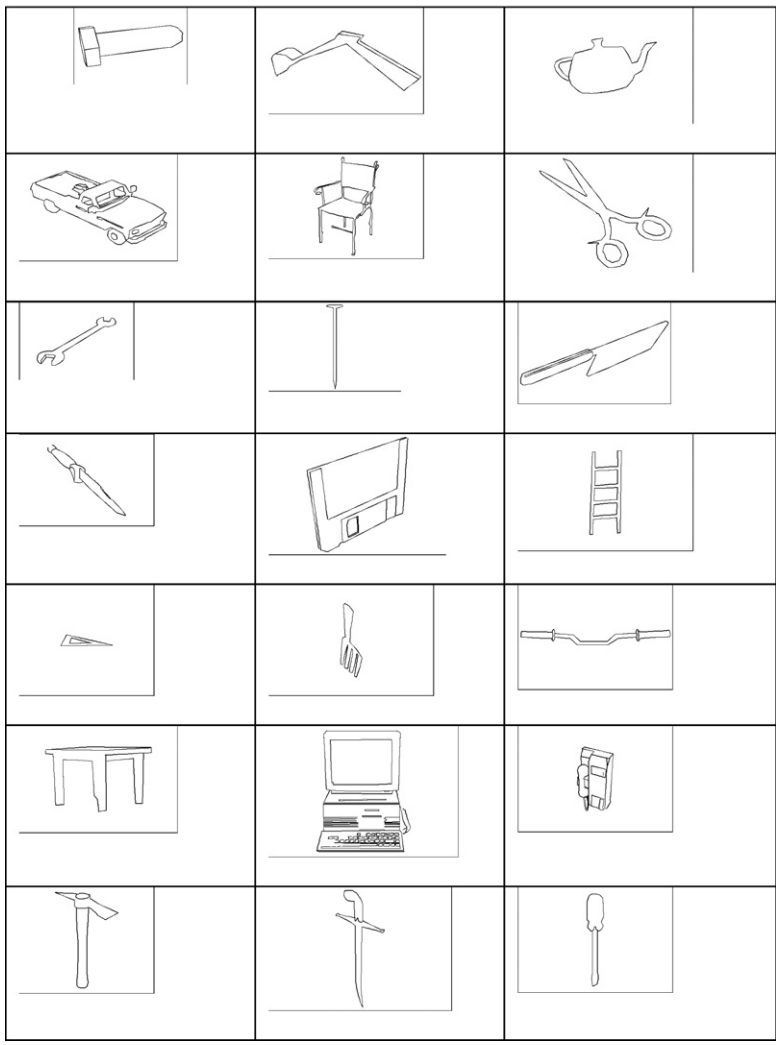


Fig. 14. Objects used in experimentation.

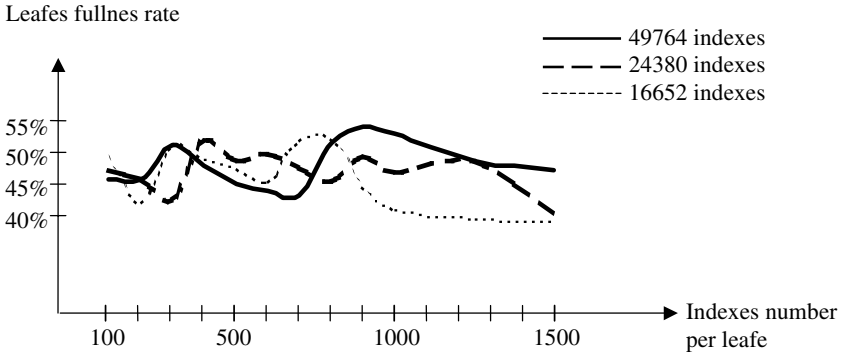


Fig. 15. Leaf fullness.

5.2.2. Leaf size versus request time

We tested the request response (Fig. 16) by considering a request composed of 100 images (long request index).

In our experiment, the number of indexes in a leaf (size of a leaf) which gives the worst time request is around 100. On the other hand, over 400 indexes, the request time is quite equal. Therefore, we used leaves of 400 indexes.

5.3. Indexes similarity analysis

We analyze the similarities so to determine the similarities values of interest regarding the request indexes (Fig. 17). The index is composed by the quasi-invariant: $\ln(\rho)$ and θ.

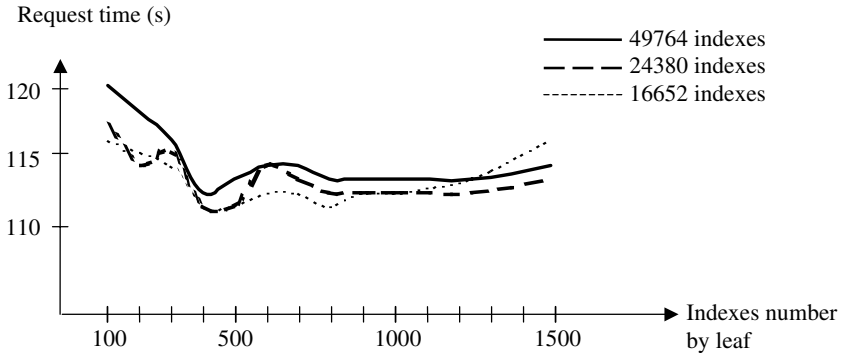


Fig. 16. Time request.

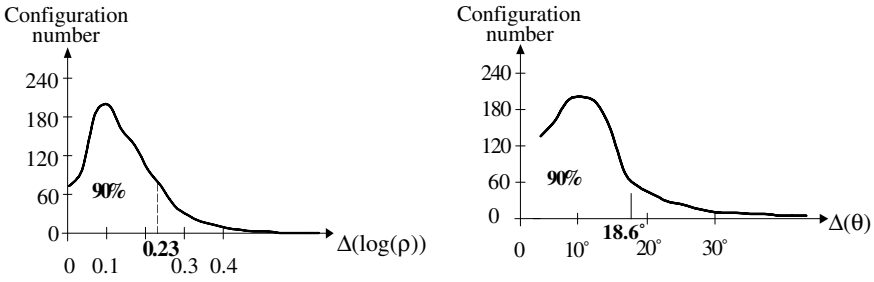


Fig. 17. Similarities of quasi-invariants.

The tests were performed on 24 images couple of the same object. For each image couple, we analyze identical geometric configurations and evaluate the difference in between the quasi-invariants we extracted from 90% of these couples show (Fig. 16) a quasi-invariant distance inferior to: $\ln(\rho) = 0.23$; $\theta = 18.61°$.

Half these values are the threshold we consider in accepting or not a quasi-invariant as similar to the request process one: $(\ln(\rho), \theta) \Leftarrow (0.14, 11.2)$.

5.4. VA-file parameters

In the VA-file process, the similarities representation space is split into 2^b hypercubes, and each dimension is divides in 2_{b_i} intervals, with the condition:

$$b = \sum_{i=1}^{4} b_i.$$

The VA-file parameters are the hypercubes bounds and the intervals number b^i. They are defined in regard to the similarities variations. In our case, the similarities are homothety k, rotation α, and translation over X- and Y-axis.

The homothety distribution (Fig. 18) resembles the length ratio ρ distribution (non-uniform distribution) (see Figs. 19–21). In the same way, we use a logarithmic function to ensure a uniform distribution (Fig. 22).

The interval numbers for each dimension is determined by the existing similarities. We use image couples, and analyze identical geometric configurations. We evaluate the difference in between each similarity. The average similarity distance is then evaluate and leads to these intervals numbers:

$$b_k = 6, \quad b_\alpha = 6, \quad b_x = 8, \quad b_y = 8.$$

Theoretically [3], the wrong similarities distribution can be approximate by a uniform distribution. In real cases, this is not verified. We had then to split the similarities representation space in a way we can get close to the uniform distribution (needed by the vote).

We build the interval of parameters variation in a two step process. First, each dimension is split in 2_{b_i} equal intervals. In second, each dimension is split in 2_{b_i} intervals with the same similarities number. This is performed by classifying, in a increasing order, each parameter of the similarities, and then by splitting each parameter variation interval in 2_{b_i} intervals, with the same numbers of similarities each.

5.5. Results

The request image has an average of 55 indexes. The response of the request from the geometric model base gives an average of 241 similarities for each request index.

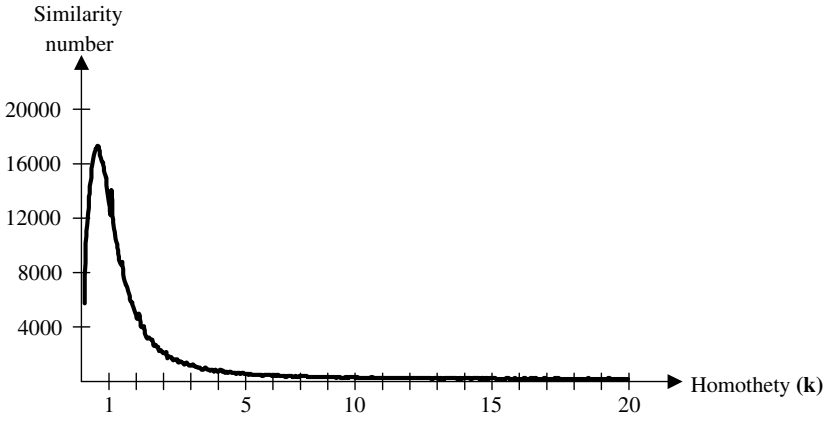


Fig. 18. Similarities distribution – homothety (k).

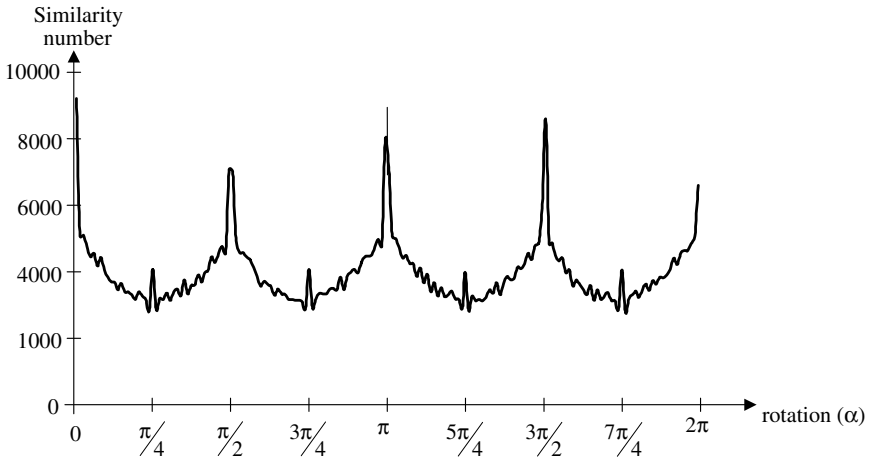


Fig. 19. Similarities distribution – rotation (α).

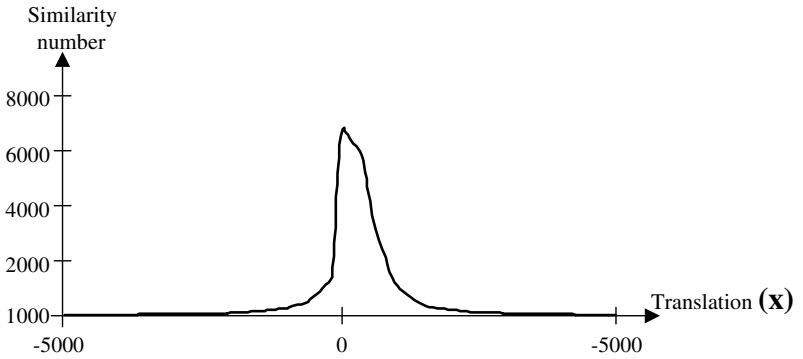


Fig. 20. Similarities distribution – translation X.

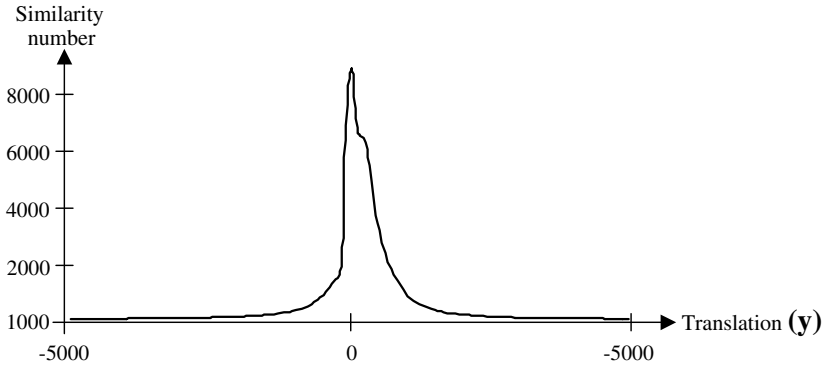


Fig. 21. Similarities distribution – translation Y.

We evaluate the retrieval rate with two types of images: images of object of the image base, and image of unknown object.

For each image, we used two space representation split methods: one that guaranty the uniform distribution (uniformity split), and one that creates hypercubes of same volume (equal interval split).

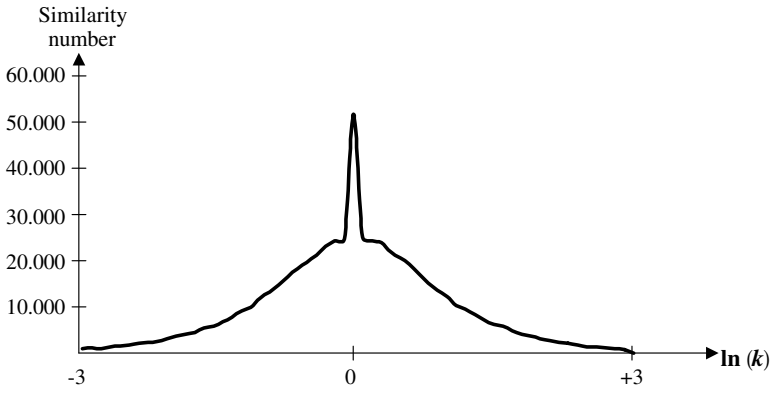


Fig. 22. Homothety distribution ln(k).

5.5.1. Matching rates for an image in the image base

For the two split approaches, the matching rates are equivalent. The rate is around 98%. The few fail cases are for ambiguous images of objects due the point of view they were looked through.

5.5.2. Matching rates for an unknown request image

In the uniformity split case, the matching rate is 63%. In the fail case, the request image is matched with the right object with 58% vote rate and with wrong objects with 22% vote rate.

In the equal interval split case, the matching rate is 69%. In the fail case, the request image is matched with the right object with 56% vote rate and with wrong objects with 37% vote rate.

The two split procedure give quite similar results. Furthermore, the uniformity split method did not improve the matching rate but made request time longer.

5.5.3. Request time

The representation space equal split shows good matching rate and request time. More than 40% of the request images have a request time lower then the average request time for the image issued from the image base. Most of the request time (94%) is for the vote process step (see Fig. 23).

In the case of representation space uniform split, the request time increases of 40%.

5.5.4. Test image

We show below a 3D object request image (Fig. 24) and the best matched images in the image database (Fig. 25). The object, a disk, is well identified even if two different objects are looked similar. This vote rate

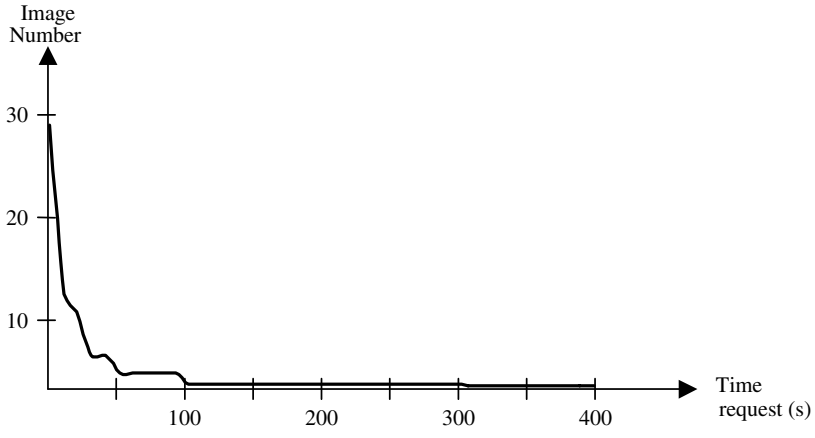


Fig. 23. Image distribution.

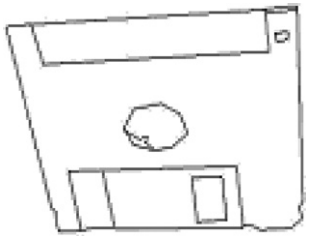


Fig. 24. Request image.

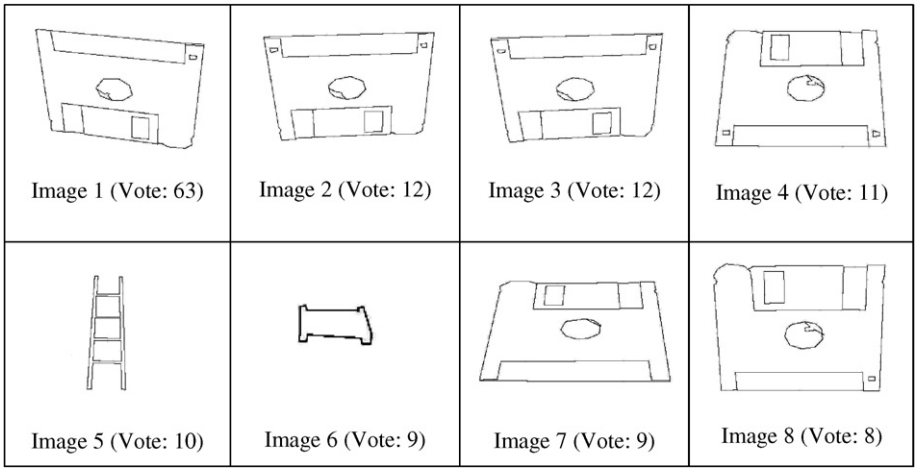


Fig. 25. Best matched images.

is due to a non-complete description of the object. The quasi-invariant features used to describe and identify objects from images needs do not provide a unique description. We propose in a future work a new geometric description that insures uniqueness of the description.

6. Conclusion

We have presented a method to index and efficiently retrieve an object in the presence of a geometric model base. We developed a method for coding the indexes and their similarities so to improve the request time. We tested our approach with images of polyhydric objects taken with different points of view. The improving in the time request and matching rate shows the efficiency of the method proposed.

The indexes used the geometric quasi-invariants features. These features can be extracted from several images regardless to the way they have been taken and without any information on the point's view (no calibration parameters are needed). Indexes can in the future be a combination of geometric and photometric features. This combination will be used during the vote process to reject indexes which geometric similarities that leads to false matching.

Acknowledgements

This work was conducted with the help of two engineers S.A. Garici and S. Akkouche. We thank them for conducting the tests.

References

[1] M. Daoudi, S. Matusiak, New multiscale planar shape invariant representation under a general affine transformations, in: ICPR'2000, Barcelona, September 3, 2000, pp. 794–797.

[2] Y. Lamdan, H.J. Wolfson, Geometric hashing: a general and efficient model based recognition scheme, in: ICCV'88 Second International Conference on Computer Vision, 1988.
[3] S. Matusiak, New multiscale planar shape invariant representation under a general affine transformations, 1999.
[4] B. Lamiroy, P. Gros, Rapid object indexing and recognition using enhanced geometric hashing, in: Fourth European Conference on Computer Vision, Cambridge, England, 1996.
[5] S. Berchtold, C. Bohm, H.P. Kriegel, The pyramid-technique: towards Breaking the Curse of dimensionality, in: Proceedings of the International Conference on Management of Data, ACM SIGMOD, Seattle, Washington, 1998.
[6] R. Weber, H.J. Schek, S. Blott, A quantitative analysis and performance study for similarity-search methods in high dimensional spaces, in: Proceedings of the 24th VLDB International Conference on Very Large Data Bases, New York, US, August 1998.
[7] L. Amsaleg, P. Gros, S.-A. Berrani, A robust technique to recognise objects in images, and the DB problems it raises, in: Proceedings of the Workshop on Multimedia Information Systems, Capri, Italie, November 2001.
[8] B. Lamiroy, P. Gros, S. Picard, Combining local recognition methods for better image recognition, in: British Machine Vision Conference, BMVC'2000, vol. 2, Bristol, UK, September 2000, pp. 735–744.
[9] P. Gros, New descriptors for image and video indexing, Dagstuhl Seminar on Content-Based Image and Video Retrieval, Dagstuhl, Germany, December 1999.
[10] P. Gros, Using quasi-invariant for automatic model building and object recognition: an overview, in: Proceedings of the NSF-ARPA Workshop on Object Representations in Computer Vision, New York, USA, December 1.
[11] P. Gros, L. Quan, 3D projective invariant from two images, in: Proceedings of the SPIE Conference on Geometric Methods in Computer Vision II, San Diego, California, USA, July 1993, pp. 75–86.
[12] K. Aström, Affine and projective normalization of planar curves and regions, in: Proceedings of the Third European Conference on Computer Vision, Stockholm, Sweden, May 1994.
[13] Joseph L. Mondey, Andrew Zisserman, Geometric Invariance in Computer Vision, MIT Press, 1992.
[14] E. Tuncel, H. Ferhatosmanoglu, K. Rose, VQ-index: an index structure for similarity searching in multimedia databases, ACM Multimedia, Juan Les Pins, France, December 2002.
[15] M.J. Fonseca, J.A. Jorge, Towards content based retrieval of technical drawings through high dimensional indexing, Computers and Graphics (2003).
[16] C. Li, E. Chang, H. Garcia-Molina, G. Wiederhold, Clustering approach for approximate similarity search in high dimensional spaces, IEEE Transactions on Knowledge and Data Engineering 14 (4) (2002) 792–808.
[17] N. Katayama, S. Satoh, Distinctiveness-sensitive nearest neighbor search for efficient similarity retrieval of multimedia information, ICDE, 2001, pp. 493–502.
[18] R. Bellman, Adaptive Control Processes: A guided Tour, Princeton University Press, 1961.
[19] T.O. Binford, T.S. Levitt, Quasi invariants: theory and exploitation, in: Proceedings of DARPA Image Understanding Workshop, 1993.
[20] H. Murase, S.K. Nayar, Visual learning and recognition of 3D objects from appearance, International Journal of Computer Vision (1995).
[21] A.R. Pope, D.G. Lowe, Learning object recognition models from images, in: ICCV'93, Fourth International Conference on Computer Vision, 1993.
[22] H. Sossa, Reconnaissance d'objets polyhédriques dans une base de modèles, Thèse de doctorat, Institut National Polytechnique de Grenoble, Décembre 1997.
[23] N. Beckmann, H. Kriegel, R. Schneider, B. Seeger, The R* tree: an efficient and robust access method for points and rectangles, in: Proceedings of the ACM Sigmod International Conference on Management of Data, May 23–25 1990.
[24] S. Berchtold, D. Keim, H.P. Kriegel, The X-tree: an index structure for high dimensional data, in: Proceedings of the International Conference on Very Large Data Bases, Bombay, India, 1996.
[25] K. Chakrabarti, S. Mehrotra, The hybrid tree: an index structure for high dimensional feature spaces, in: Proceedings of the International Conference on Data Engineering, Australia, 1999.
[26] A. Guttman, R-trees: a dynamic index structure for spatial searching, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1984.
[27] K. Lin, V. Jagadish, C. Faloutos, The TV-tree: an index structure for high dimensional data, VLDB Journal (1995).
[28] D.B. Lomet, B. Salzberg, The Hb-tree: a multi attribute indexing method with good guaranteed performance, ACM Transactions on Database Systems (1990), December.
[29] J.T. Robinson, The Kdb-tree: a search structure for large multi dimensional indexes, in: Proceedings of the ACM SIGMOD International Conference on Management of Data, 1981.
[30] D. White, R. Jain, Similarity indexing the SS-tree, in: Proceedings of the International Conference on Data Engineering, 1996.