Robotica (1998) volume 16, pp. 651–658. Printed in the United Kingdom © 1998 Cambridge University Press

Indexing and alignment of 3-D objects using geometric quasi-invariants

P. Vasseur, C. Pegard, E. Mouaddib and L. Delahoche

Perception in Robotics Department, GRACSY (Groupe de Recherche sur l'Analyse et la Commande des Systèmes), University of Picardie, 7, rue du Moulin Neuf, 80000 Amiens (France) e-mail: Pascal.Vasseur@sc.u-picardie.fr

(Received in Final Form: in Final Form: February 9, 1998)

SUMMARY

In this paper, we are introducing a system which is able to recognize polyhedral objects in an indoor environment. Our system is intended to be implemented on autonomous mobile platforms in order to enable the localization or research of a precise item. The algorithm is based on the use of geometric quasi-invariants associated to every object. These geometric quasi-invariants correspond to the ratio of the lengths as well as the angle formed by the pair of segments which are in relationship and which are constituting the object. We present some experimental results gained on one of our platforms in our laboratory.

KEYWORDS: 3-D objects; Indexing; Alignment; Geometric quasi-invariants; Indoor environment; Mobile robots.

1. INTRODUCTION

The problem of the recognition of objects based on a model can be described as follows: Thanks to a whole of models grouped together in a library, the aim is to know if one or more of these objects are on the image of a scene at a precise time and to indicate their position and their orientation.

1.1. Special conditions of the problem

These conditions will be relative to the kind of images which have to be treated and to the type of objects to be recognized. In our case, we only work with intensity images and not with range images.[1] Our application of recognition of objects will be implemented on mobile robots moving in an indoor environment in order to complete multi-agent missions of cooperation with each other.

We can consider that the greatest majority of recognized objects have a polyhedral form, and will thus limit our work to objects which have such characteristics. If we refer to the general view of objects recognition described in reference 2, the system can be divided into five successive stages which are detection of the primitives, grouping of the primitives in a significant whole, indexing, matching and, finally, verifying.

Detection and grouping of the primitives are low-level treatments[3] which refer to technics like the Hough transform[4] and perceptual organization,[5] in order to extract the desired primitives from the image and to associate them in significant groups so as to interpret them as objects.

Considering a collection of 3-D object models, the indexing must achieve an off-line pretreatment in order to generate a data structure to decide quickly if a given image contains one or more objects represented by 3-D models. Thus we obtain an indexing table which counts the different available models.

According to reference 6, two major indexing categories can be developed:

— The first one consists in the realization of indexing relative to the point of view[7, 8]
— The second category concerns indexing per invariants.[9] Hence functions will be identified and considered as extrinsic or intrinsic, and can be local, semi-local or global. Besides their particularity is to be invariant following some transformations which can be Euclidean, affine, perspective or projective.

The matching aims is to find the best pair between the primitives of the image and those of the models selected during the indexing. The goal is to research the suitable maximum measure which will define the best matching.

Numerous methods are often used to aim at this result. Among these methods, there is that based on interpretation trees.[10] In reference 11, a comparative study of 10 methods based on interpretation trees is proposed and stresses the difficulty to reduce the complexity of these algorithms.

The second approach corresponds to the matching of graphs.[12–14] In this case, two graphs respectively represent the model of the object and the object which is detected in the image. In this representation the nodes correspond to the primitives, such as segments or points, and the arcs correspond to the relation between these primitives. The principle of this approach consists in finding a subgraph isomorphism between both graphs. So, the two graphs which have the greatest similarity are matched.

The greatest defect of these first two methods is due to the combinatory explosion they are causing when the model and the image have a great number of primitives or when the number of models is important.

One of the possibilities to reduce this combinatory explosion is to use relaxation methods.[15] However, relaxation strategies do not necessarily converge and can allow ambiguous final solutions. Other methods based on transformation space search are based on the Hough transform philosophy which comes down to collecting votes.[16] The last sort of matching algorithm we mention, is named an alignment method.[17] This method consists in detecting a

https://doi.org/10.1017/S0263574798000575 Published online by Cambridge University Press

matching of a base from a limited number of primitives. Then, these primitives are used as a base to estimate the point of view transform to be applied to the model so that the remaining primitives are associated to these of the image.

In order to enhance the robustness of this method concerning the remaining primitives to be matched, Alter and Grimson[18] propose an uncertainty on the supposed position of the primitives. With this method, the error is propagated and sharpened in order to finally obtain a reliable position and direction.

Through these different methods, it is important to note that three parameters are mainly to be managed. First, it is necessary to avoid every method which involves a combinatory explosion when the number of primitives grows up. Then, it is important to have the possibility to manage the point of view notion without using a huge storage place for all the possible positions of an object in space, in order to spare memory place.

Finally, to obtain a noise robust method which can treat the cases of partial appearance of an object, it is necessary to develop a method managing mistakes, such as occlusions.

The last operation of the global object recognition process is meant to decide if one or many instances of one of the models can be created. This stage logically follows the matching stage and determines a measure of general quality on the matchings that may have been realized. If this measure is enough, so we can know where the object is located in the scene as well as its direction. The second part of this paper consists in a global introduction to our method.

2. SYSTEM OVERVIEW

The system we propose is divided into two main parts, respectively off-line and on-line. The first one has to deal with the organization of the database and integrates every model known from different points of view. The second one consists in matching an object detected in the scene with one of the models from the base. Figure 1 presents a global view of our system.

Our method essentially relies on the use of geometric quasi-invariants. We have chosen to index our classes of objects with these parameters as they display numerous advantages allowing a quick use and relative reliability.

The first advantage is the calculation, which is obvious, as the use of a ratio of lengths and an angle are enough. Then these quasi-invariants have the particularity to vary little whatever the point of view. So, a limited number of indexing classes can be formed, which limits the storage place needed for the memory of models. Finally, these values enable one to manage possible occlusions and objects with the same form but with different measures thanks to a vote method we will detail below. Generally speaking, indexing methods are based on the point of view involving comparison to the object.[19, 20] In our case, we are using the calculation of the quasi-invariant values, but we do not use it to determine the classes. We are only using these values to determine our indexing table with a classification method allowing a reduction of the number of classes, as it is explained further down, to give directly tolerance intervals for every class.

In this paper, we show that with our method we can manage the indexing of numerous objects and remain flexible, have a quick access and reliable results without developing too complex calculations. In order to explain the different stages of the algorithm, we are taking as an example a particular object which is a pyramid with a square base (Figure 2).

3. OBJECTS INDEXING

3.1. Introduction

The indexing stage in the algorithm of objects recognition is one of the main stages because it conditions the speed and

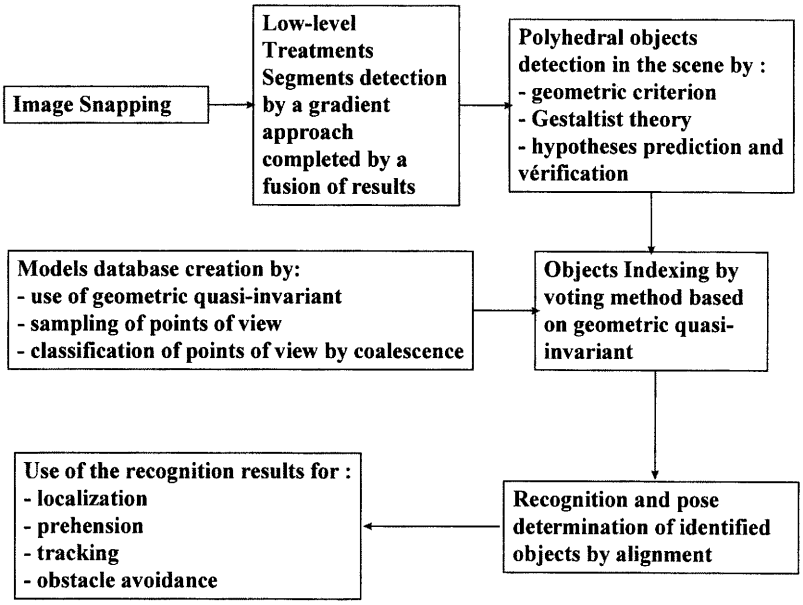


Fig. 1. Algorithm Description

https://doi.org/10.1017/S0263574798000575 Published online by Cambridge University Press

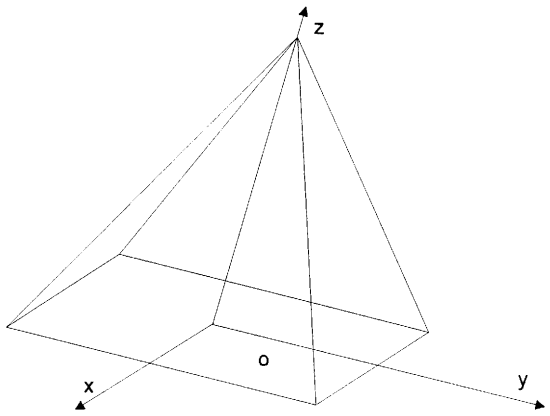


Fig. 2. Pyramid in its frame

the reliability of this recognition. It consists of giving a key to every object or to different point of view of objects to locate the object by calculating this key with information given by the scene. As we have seen before, this indexing can depend on the point of view or invariant parameters. The methods based on the point of view refer to the notion of aspect graph.[19, 20] This graph represents the partitioning of the view sphere around the object (Figure 3) and it represents the different points of view.

Every border limits sectors and represents the visual events symbolizing the aspect changes of the object.

In this precise calculation of the aspect graph, the different stages consist in looking for these visual events, then determining the different sectors on the sphere, and then calculating for every one of them, a prototype representation of the object.

3.2. Calculation of the quasi-invariants

In our method, we use the calculation philosophy according to the view sphere but we do not look for visual events.

We are using the sphere to form a whole learning of the different points of view. So for every point on the sphere, we are projecting the three-dimension object on the plane tangent to the sphere at this point and we eliminate the hidden lines. Once the object is projected, we calculate the value of the quasi-invariant for every couple of joined segments. Finally, once the whole of the sphere has been read through, we apply on this whole learning a classification which allows to link the points of view which supply identical or similar values.

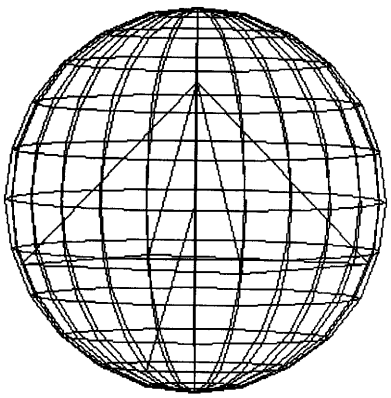


Fig. 3. Pyramid in its view sphere

In the proposed algorithm, we calculate the projection of the object from different points of view to determine the geometrical quasi-invariants forming it. Hence we had to choose between the perspective projection and the projection perpendicular to the graph. We suppose the objects to be sufficiently far from the camera, hence we are using the perpendicular projection.

Moreover, the latter displays several advantages which motivated our choice. First, the mathematical model which is associated with it, is simpler than that of the perspective. Secondly, it is useless to know data such as the foci and the center of the image. Thanks to its simplicity, the weak perspective comes out with fewer solutions to treat during the alignment. Moreover, no ratio of scale is necessary.

In order to generate a uniform distribution of the finite viewing directions, the unit sphere is partitioned into equilateral triangles. Ballard[21] suggests a way to generate tassellated spheres with different numbers of facets by subdividing the edges of the triangle of the icosahedron. In general, the facet number N is given by $N = 20 \times n^2$, where n is the division of the edge of triangle.

It is assumed that the object is located at the center of the tassellated sphere (Figure 3) and is viewed from the center of one of the facets. The viewing direction is defined by the unit vector from the center of the facet to the center of the sphere. The viewing direction can also be defined by two parameters of φ, θ. Figure 4 shows the tassellated spheres with 1280 facets and the definition of view angles φ, θ. For a given center point $P = x.i + y.j + z.k$, φ, θ can be obtained as follows:

$$\varphi = \arctan2(y, x) \qquad \text{where } 0 \le \varphi \le 2\pi$$
$$\theta = \arcsin(z) \qquad \text{where } -\pi/2 \le \theta \le \pi/2$$

When we obtain the point on the sphere, it is easier to calculate the tangent plane on this point and the projection of every vertex obtained with the following equations

$$\begin{cases} x_p = x_s + x_{view}.t \\ y_p = y_s + y_{view}.t \\ z_p = z_s + z_{view}.t \end{cases}$$

$$t = \frac{\sin\varphi.\cos\theta.(x_{view} - x_s) + \cos\varphi.\cos\theta.(y_{view} - y_s) + \sin\theta.(z_{view} - z_s)}{x_{view}.\sin\varphi.\cos\theta + y_{view}.\cos\varphi.\cos\theta + z_{view}.\sin\theta}$$

The triplets (x_s, y_s, z_s) and (x_p, y_p, z_p), respectively, represent the coordinates of the vertex centered on the object of the center of gravity of the object and the coordinates of this vertex projected on the plane tangent to the coordinate points $(x_{view}, y_{view}, z_{view})$. Hence when all the vertices have been projected, we have to determine which ones are hidden by the object itself.

3.3 Achievement of the aspect classes

The rest of the algorithm consists in grouping the whole points of view in some classes which present similar quasi-invariant values. Figures 5 and 6 show the values of the ratio of lengths and the corner formed by a pair of segments which belongs to our pyramid. These values have been calculated with values of φ and θ taken between 0 and 360 degrees with a sampling step of 5 degrees. So we can note that pics of value appear regularly to form different groups

https://doi.org/10.1017/S0263574798000575 Published online by Cambridge University Press

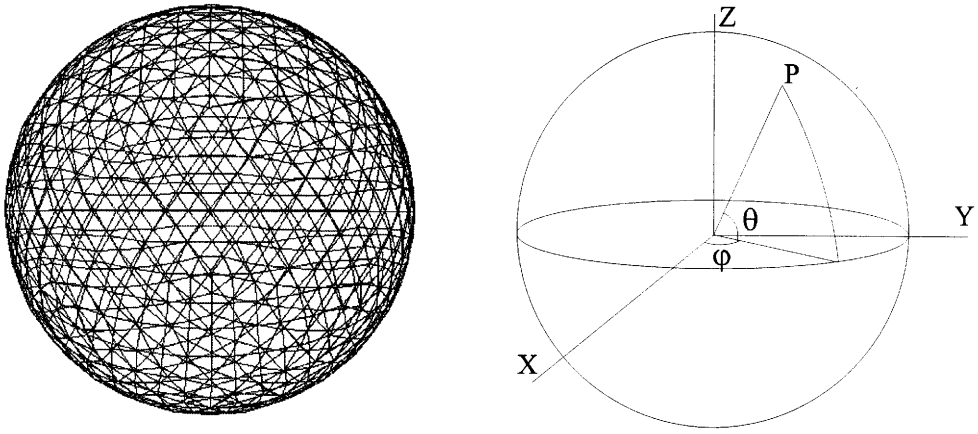


Fig. 4. Tassellated sphere with 1280 facets and definition of view angles

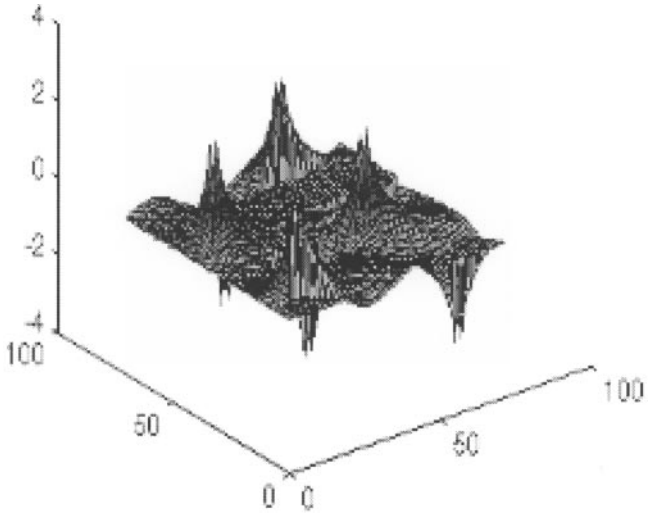


Fig. 5. Lengths ratio variation according to φ and θ for a single triplet of points.

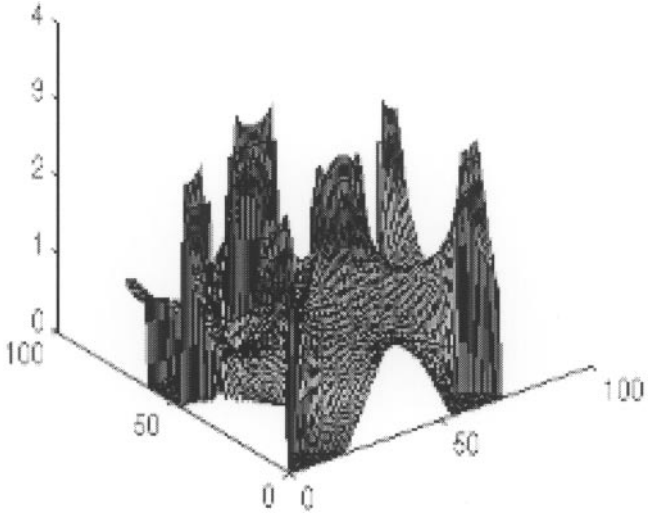


Fig. 6. Angle variation according to φ and θ for a single triplet of points.

which permit a classification of the points of view.

Hence every point of view is represented by a vector whose dimension is equal to twice the number of pairs of segments constituting the object. Thus every object can be defined by a different size of vectors. To achieve this taxonomy of the points of view, we use the tools given by the data analysis and, more precisely, the coalescence methods.[22] Among those, we have chosen a method which is nonsupervised, exclusive and by repartition. So every point of view of the object belongs to one unique class, and we use the information of proximity to obtain distinctive classes. To achieve such classification, we must define three essential points: an indication of proximity, the number of partitions and the quality criterion. The sign of proximity we have chosen is the Euclidean distance which will permit us to form our classes. The number of partitions is generally unknown. Still, if this one is not estimated, the number of possibilities becomes ultra-combinatorial. The estimation is important as the method of coalescence gives only local optimums and the number of classes is very big at the beginning.

In our case, we estimate that the number of classes is approximately equal to the number of visual events estimated from the geometry of the object. Finally, the last point to determine corresponds to the criterion of quality. We then consider the training set made of the following vectors:

$$X_i = (x_{1i} \quad x_{2i} \quad x_{3i} \ldots X_{di})$$

with d equal to twice the number of segments. So it is possible to calculate the general center of gravity and the matrix of intra-group and inter-groups dispersion given by:

$$\hat{m} = \frac{1}{n} \sum_{k=1}^{M} n_k . \hat{m}_k$$

$$\hat{\Sigma}_W = \frac{1}{n} \sum_{i=1}^{M} \sum_{j=1}^{n_j} (x_{ij} - \hat{m}_i).(x_{ij} - \hat{m}_i)^T$$

$$\hat{\Sigma}_B = \frac{1}{n} \sum_{i=1}^{M} \sum_{j=1}^{n_j} (\hat{m}_i - \hat{m}).(\hat{m}_i - \hat{m})^T$$

with M equal to the estimated number of classes and the criterion to majorate is:

https://doi.org/10.1017/S0263574798000575 Published online by Cambridge University Press

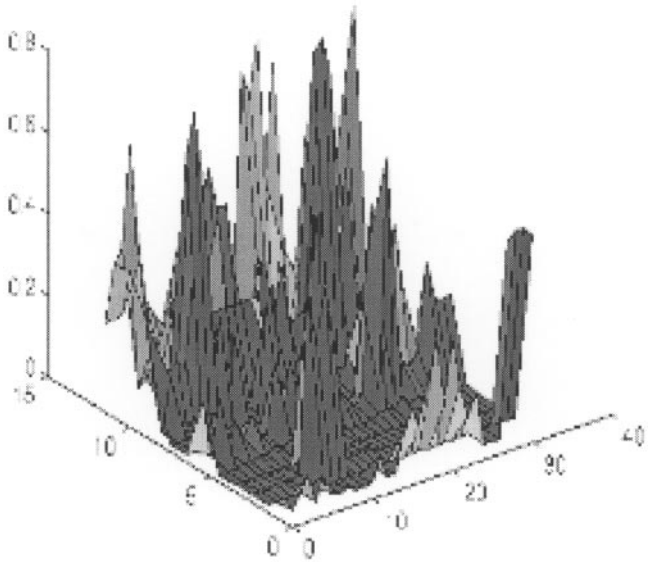


Fig. 7. Standard deviation for the vectors of each class formed for pyramid indexing

$$tr(\hat{\Sigma}_W^{-1}.\hat{\Sigma}_B)$$

When the algorithm has been executed for one object, after only a few iterations we obtain a classification giving first the vectors representing every class, but we also obtain the standard deviation associated to every value of the vectors. Thus in the case of the pyramid, we finally obtain 12 classes for vectors whose dimension is 32, and Figure 7 shows the value of the standard deviation for every vector of each class. So we can use intervals and no longer strict values for our indexing table. Incidentally, it is easy to note that the maximum standard deviation does not exceed the value of 0.8. This is very encouraging from the point of view of the accuracy of the achieved classification. Hence we can achieve our indexing with a minimum of information on the different objects and with a complexity which is acceptable considering that the algorithm was executed off-line. From this stage, it is possible to achieve the recognition of the object.

3.4. Creation of the indexing table

Once the aspect classes have been elaborated for different objects, it is imperative to range them in a computational structure which is simple, quickly available and easy to modify by the insertion or extraction of models. Besides, it is necessary that this structure immediately gives the mesh structure associated with every class. Hence we have created a table like that proposed by the Hough transform. The coordinates of this table correspond to the both geometrical quasi-invariant sampled regularly (Figure 8). Contrary to Hough, this table does not use any votes but we find in every box the point of view of different models of objects which are associated with the quasi-invariant values. Thus we see that this indexing table is very quick in access thanks to its two indexing keys which will permit us to use votes methods to identify the objects.

4. OBJECT RECOGNITION

The phase of on-line recognition of our algorithm is made of two parts which are first the identification of the object and then the determination of its position and its direction. The first stage uses the indexing table designed by the method described above to select a model. The second one uses the model chosen in this table to achieve matching per alignment.

Rap / Angle	0	0.1	0.2	.....	3.0	3.1	.....	4.7	.....
0	M1	M1	M1						
1									
2					M2	M2			
.....					M2	M2			
96									
97									
.....									
179	M1	M1M3	M3					M3	
180	M1	M1M3	M3					M3	

Fig. 8. Indexing Table

4.1. Model identification

In this first part, the aim is to achieve the identification of the objects detected above in the scene. This detection is not described here, but it is based on an algorithm which works with perceptual organization and a prediction and verification of hypotheses.[23]

Hence we have two input ranges which are the list of the segments constituting every object and the indexing table associated with the models. Our work is to determine to which model every detected object belongs; so we have chosen a votes method. The aim of this method is to concentrate the votes on the indexing classes and then on the identified models. The algorithm works as follows: For every detected object, the geometrical quasi-invariant associated with every pair of segments with a common endpoint are calculated. Then, the values are written in the indexing tables to find out to which defined models the values belong. So we use for every model a counter which is increased at every matching in the indexing table. Once all votes have been achieved, we simply have to see which candidates have the most votes. However, having the greatest number of votes is not sufficient because we want a threshold of probability to be passed to give to our method a supplementary criterion of reliability. This dynamic level is, in fact, a ratio between the number of votes accorded to a model and the dimension vector of the class associated with this model. Thus four cases of identification can be verified of which three were noted in reference 16.

The first case concerns the correct identification when the model corresponds to the object. The second case gives more than one model because the number of votes are the same. The third one is the case of a bad identification caused by the noise of the scene or by a bad detection in the previous stage. Finally, there is no identification because of scattered votes in the indexing table.

In that case, the last stage of the algorithm is not achieved and we try again recognition. In the other cases, the next

https://doi.org/10.1017/S0263574798000575 Published online by Cambridge University Press

stage will permit us to maintain, to refute or to decide between the obtained identification.

4.2. Determination of the object pose

This stage has two different aims: The first one, mentioned above, is to check the identification which is made, and the second consists of calculating the position and the direction of the identified objects, that is to say their setting.

To achieve these, we have decided to use a method based on alignment.[18] The alignment recognition consists of transforming a model with known transformations so as to obtain a form similar to that of the object on the image. This type of algorithm consists of four successive stages. First, we suppose a matching of three points from the model with three points in the image. Then, from this matching, we can calculate the 3-D setting of the model and estimate the position of the remaining points of the object in the image. We finally check the hypothesis by looking for the foretold points in the image.

Still, this kind of algorithm displays an important defect, that of the risk of the combinatory explosion and the difficulty of process management.

Thanks to the stage before the indexing, the problem of combinatory explosion is totally avoided as we can estimate matching between the segments of the models and of the object in the image from the calculated quasi-invariant. Hence we are looking for a couple of segments from the identified object which offers the best similarity with a couple of segments from the model, thanks to the quasi-invariants. Once the best matching has been made, we will strictly use the traditional stages of alignment to achieve the setting calculation.

All the transforms chosen are three rotations in space as well as the scale ratio, to manage the depth of the object in the scene.

Thus, when the proposed identification is correct, the setting is given quasi-immediately. In the case of a wrong identification or multiple identifications, impossible verifications are quickly detected and invalidations then permit us to avoid wrong hypotheses.

It is worth noting that, unlike Alter, we use a bounded error around the predicted locations of the endpoints in order to verify their presence in the image. This error is arbitrarily fixed to five pixels. Thus we summarize here the direct alignment and 3D pose obtained by Alter in order to prove the simplicity of the computation and the effectiveness provided by this method when a first matching between three points in the image and three points of the model is done.

Consider three model points $\vec{p}_0$, $\vec{p}_1$, $\vec{p}_2$ given in any centered coordinates, and three corresponding image points $\vec{i}_0$, $\vec{i}_1$, $\vec{i}_2$ given in image-centered coordinates. Then it is possible to compute the distances (R_{01}, R_{02}, R_{12}) between the model points and the corresponding distances (d_{01}, d_{02}, d_{12}) between the image points. Then the final aim is to give the image-centered coordinates a fourth model point. First we have:

$$a = (R_{01}+R_{02}+R_{12})(-R_{01}+R_{02}+R_{12})(R_{01}-R_{02}+R_{12})(R_{01}+R_{02}-R_{12})$$

$$b = d_{01}^2(-R_{01}^2+R_{02}^2+R_{12}^2)+d_{02}^2(R_{01}^2-R_{02}^2+R_{12}^2)+d_{12}^2(R_{01}^2+R_{02}^2-R_{12}^2)$$

$$c = (d_{01}+d_{02}+d_{12})(-d_{01}+d_{02}+d_{12})(d_{01}-d_{02}+d_{12})(d_{01}+d_{02}-d_{12})$$

Then, if $a \neq 0$

$$s = \sqrt{\frac{b+\sqrt{b^2-ac}}{a}}$$

$$\sigma = \begin{cases} 1 \text{ if } d_{01}^2+d_{02}^2-d_{12}^2 \leq s^2(R_{01}^2+R_{02}^2-R_{12}^2) \\ -1 \text{ otherwise} \end{cases}$$

$$(h_1, h_2) = \pm\left(\sqrt{(sR_{01})^2-d_{01}^2},\ \sigma\sqrt{(sR_{02})^2-d_{02}^2}\right)$$

$$(H_1, H_2) = \frac{1}{S}(h_1, h_2)$$

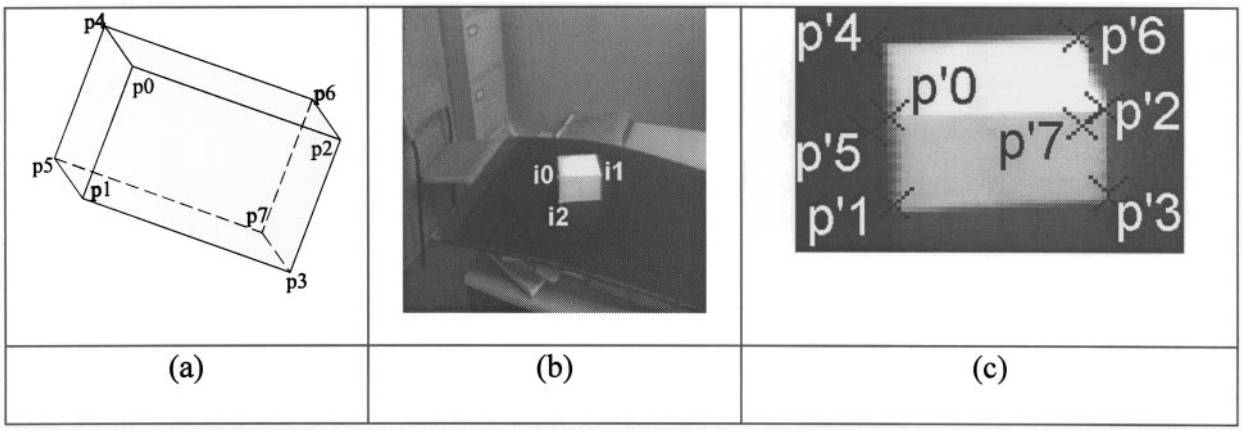


Fig. 9. Alignment results for a square: (a) CAD Model, (b) Original Scene, (c) Projected Model Points

https://doi.org/10.1017/S0263574798000575 Published online by Cambridge University Press

The next step consists in computing the "extended affine coordinates" of the fourth model point:

$$\vec{p}_3 = \alpha(\vec{p}_1 - \vec{p}_0) + \beta(\vec{p}_2 - \vec{p}_0) + \gamma(\vec{p}_1 - \vec{p}_0) \times (\vec{p}_2 - \vec{p}_0) + \vec{p}_0$$

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \text{inv} \begin{bmatrix} p_{1_1} - p_{0_1} & p_{2_1} - p_{0_1} & (p_{1_2} - p_{0_2})(p_{2_3} - p_{0_3}) \\ p_{1_2} - p_{0_2} & p_{2_2} - p_{0_2} & (p_{1_3} - p_{0_3})(p_{2_1} - p_{0_1}) \\ p_{1_3} - p_{0_3} & p_{2_3} - p_{0_3} & (p_{1_1} - p_{0_1})(p_{2_2} - p_{0_2}) \\ & & -(P_{1_3} - p_{0_3})(P_{2_2} - p_{0_2}) \\ & & -(P_{1_1} - p_{0_1})(P_{2_3} - p_{0_3}) \\ & & -(P_{1_2} - p_{0_2})(P_{2_1} - p_{0_1}) \end{bmatrix} \times \begin{bmatrix} P_{3_1} - p_{0_1} \\ P_{3_2} - p_{0_2} \\ P_{3_3} - p_{0_3} \end{bmatrix}$$

with $\vec{p}_0 = (p_{0_1}, p_{0_2}, p_{0_3})$.

Given the image points $\vec{i}_0 = (x_0, y_0)$, $\vec{i}_1 = (x_1, y_1)$ and $\vec{i}_2 = (x_2, y_2)$, and the following relations $x_{01} = x_1 - x_0$, $y_{01} = y_1 - y_0$, $x_{02} = x_2 - x_0$ and $y_{02} = y_2 - y_0$, the image location of the transformed and projected $\vec{p}_3$ is:

$$(\alpha x_{01} + \beta x_{02} + \gamma(\gamma_{01} H_2 - y_{02} H_1) + x_0,\ \alpha y_{01} + \beta y_{02} + \gamma(-x_{01} H_2 + x_{02} H_1) + y_0)$$

Once the image location of each model point has been determined, the final stage consists of verifying how many real endpoints are present in the image at the predicted positions in accordance with the fixed bounded error. Figure 9 presents the results obtained with such a method. The CAD model of a square (Figure 9(a)) has to be aligned with the object detected in the scene (Figure 9(b)). Finally, in (Figure 9(c)) the projected and transformed model points are represented by the notation p'n. We may note that six projected model points can be matched with image endpoints. Then it is possible to conclude that the square has been recognized and its pose determined.

Table I. Synthetic data results

Identification				Alignment
Good	Equality	False	Rate	Rate
65%	20%	15%	85%	95%

5. EXPERIMENTAL RESULTS

In order to test our algorithm, we have created two sets of tests. The first set is made of virtual objects which we direct arbitrarily and of which we calculate the quasi-invariant attributes. Then we identify the models with our indexing table built with our coalescence method. Thus we obtain the results mentioned in table I. Similarly, we make a calculation of the object pose with the algorithm proposed with the results of table I. These tests as a whole have been made with a library composed of 105 models and 50 objects. The last series of tests were performed on real images (Figure 10). Among these images, some of them display one object, divided objects or no object. Figure 10(b) displays the results obtained by our algorithm of objects detection. In this case, only two objects are identified, and Figure 10(c) shows the object which has been recognized. The other object detected has not been identified because of the overwhelming noise in its structure.

According to these results, we estimate that our process is relatively satisfactory. However, it may be important to improve its performance in order to avoid false recognitions which involve additional computational time requirements.

6. CONCLUSION

In this paper, we are proposing an algorithm of polyhedral object recognition. It permits us to manage an important number of objects without needing excessive memory. We propose an indexing method which allows us to manage the aspects of every object in function of the geometrical quasi-invariants associated to every couple of joined segments. Hence with this indexing we can quickly identify objects which are present in the scene and facilitate the calculation of the setting of every object in alignment. Our method proved efficient concerning the speed of the detection of the objects in the scene, as well as the reliability and the robustness of the results.

As for further work, our aim is to improve our system and to develop a panoramic sensor consisting of a linear camera

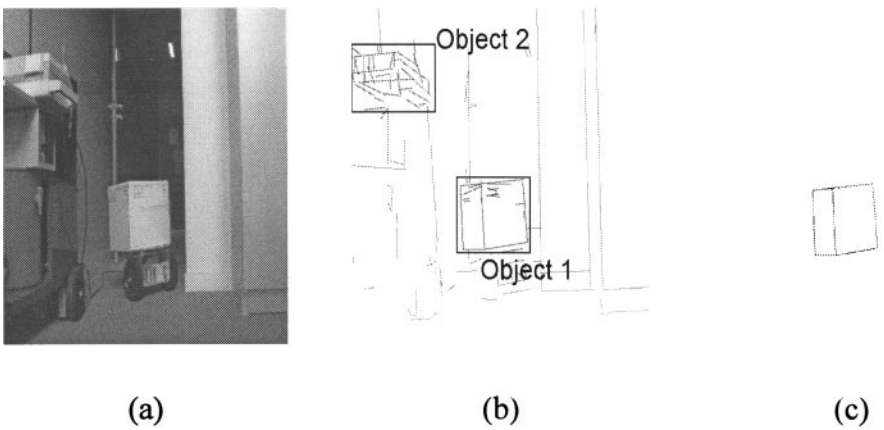


Fig. 10. (a) Indoor scene, (b) Objects detected, (c) Object recognized

https://doi.org/10.1017/S0263574798000575 Published online by Cambridge University Press

associated with a rotating platform. Indeed, the next stage is to develop algorithms which allow us to perform tasks, such as localization, prehension or tracking.

References

1. P.J. Flynn and A.K. Jain, "BONSAI: 3-D Object Recognition Using Constrained Search," *IEEE Trans. Pattern Anal. Machine Intell.* **13**, No. 10, 1066–1075 (Oct., 1991).
2. A.R. Pope, "Model-Based Object Recognition: A Survey of Recent Research," *Technical Report* 94-04 (Jan., 1994).
3. J.B. Burns, A.R. Hanson and E.M. Riseman, "Extracting Straight Lines," *IEEE Trans. Pattern Analysis and Machine Intelligence*, **8**(4), 425–455 (1986).
4. R.O. Duda and P.E. Hart, "Use of the Hough transform to detect lines and curves in pictures," *Commun. ACM* **15**, 11–15 (1972).
5. S. Sarkar and K.L. Boyer, "Perceptual organization in computer vision: A review and a proposal for a classifactory structure," *IEEE Trans. Pattern Anal. Machine Intell.* **PAMI-23**, No. 2, 382–399 (1993).
6. T.M. Breuel, "The 3D Indexing Problem," *IDIAP, Memo #93-08*, (June, 1993).
7. R. Horaud and H. Sossa, "Polyhedral Object Recognition by Indexing," *Pattern Recognition* **28**, No. 12, 1855–1870 (1985).
8. J.S. Beis and D.G. Lowe, "Learning Indexing Functions for 3-D Model-Based Object Recognition," *Proc. IEEE Computer Vision and Pattern Recognition* (1994) pp. 275–280.
9. Y. Lamdan, J. Schwartz and H. Wolfson, "Affine invariant model-based object recognition," *IEEE Trans. on Robotics and Automation*, **6**(5), 578–589 (1990).
10. J. Ben-Arie and A.Z. Meiri, "3D Objects Recognition by Optimal Matching Search of Multinary Relations Graphs," *Comput. Vision, Graphics, And Image Process.* **37**, 345–361 (1987).
11. R.B. Fisher, "Performance Comparison of Ten Variations on the Interpretation-Tree Matching Algorithm," *Proc. Euro. Conf. Comput. Vision* (1994).
12. L.G. Shapiro, R.M. Haralick, "Structural Descriptions and Inexact Matching," *IEEE Trans. Pattern Anal. Machine Intell.* **3**, No. 5, 504–519 (Sept., 1981).
13. R. Bergevin and M.D. Levine, "Generic Object Recognition: Building and Matching Coarse Descriptions from Line Drawings," *IEEE Trans. Pattern Anal. Machine Intell.* **15**, No. 1, 19–36 (Jan., 1993).
14. E.K. Wong, "Model Matching in Robot Vision by Subgraph Isomorphism," *Pattern Recognition* **25**, No. 3, 287–303 (1992).
15. A.J. Bray, "Object Recognition Using Local Geometric Constraints: A Robust Alternative to Tree Search," *Proc. Euro. Conf. Comput. Vision* (1990) pp. 499–515.
16. B. Lamiroy and P. Gros, "Rapid Object Indexing and Recognition Using Enhanced Geometric Hashing," *Proc. Euro. Conf. Comput. Vision* (1996) pp. 59–70.
17. S. Ullman and R. Basri, "Recognition by Linear Combinations of Models," *IEEE Trans. Pattern Anal. Machine Intell.* **13**, No. 10, 992–1006 (Oct., 1991).
18. T.D. Alter and W.E.L. Grimson, "Fast and Robust 3D Recognition by Alignment," *Proc. Int. Conf. Comput. Vision* (1993) pp. 113–120.
19. Z. Gigus, J. Canny and R. Seidel, "Efficiently Computing and Representing Aspect Graphs of Polyhedral Objects," *IEEE Trans. Pattern Anal. Machine Intell.* **13**, No. 6, 542–551 (June, 1995).
20. J.H. Stewman and K.W. Bowyer, "Direct Construction of the Perspective Projection Aspect Graph of Convex Polyhedra," *Comput. Vision, Graphics, And Image Process.* **51**, 20–37 (1990).
21. D.H. Ballard and C.M. Brown, *Computer Vision* (Prentice Hall, Englewood Cliffs, New Jersey, 1992).
22. B. Dubuisson, *Diagnostic et Reconnaissance des Formes* (Editions Hermes, Paris, 1990).
23. P. Vasseur, C. Pégard, E.M. Mouaddib and L. Delahoche, "Object Detection Using Perceptual Organization and Prediction/Verification of Hypotheses," *23rd Annual International Conference of the IEEE Industrial Electronics Society (IECON'97)*, New Orleans, Louisiana, (November 9–14, 1997).

https://doi.org/10.1017/S0263574798000575 Published online by Cambridge University Press