ADVANCES IN 3D MODEL IDENTIFICATION FROM STEREO DATA

John Knapman
IBM UK Scientific Centre, Winchester

Abstract

Volumetric (CAD) models of objects are converted to a wire frame representation which is compiled into a data base and represented in vectors and matrices that characterise both *local geometrical relationships* and the structure of the models. This characterisation is independent of position and orientation and supports variable size. Using a stereo vision system, instances of these objects are then identified from pairs of images containing single objects or more than one object.

Introduction

The term "data base" is taken to imply a repository of a number, potentially a large number, of models that are to be selected and matched to the 3D data obtained from a scene. Sometimes, selecting models in this way is called "invocation" and it appears in principle to be more difficult than matching a single, given model to the data.

Results have been obtained distinguishing among twelve objects in a data base using synthetic images employing the vision system developed at the AIVRU, University of Sheffield (Mayhew et al, 1986). This delivers 3D geometrical descriptions of the edges in a scene from a stereo pair of images. These are classified as straight, circular, planar or otherwise.

Some images have been of isolated objects, with results reported elsewhere (Knapman, 1987). Other images have contained more than one object with limited occlusion.

The work of Grimson and Lozano-Pérez (1984) emphasises the need to find powerful constraints that are nevertheless cheap to implement in order to reduce the magnitude of the combinatorial problem inherent in matching even one model to the data in a scene. One cannot afford the expense of generating and testing many hypotheses for the transformation between model co-ordinates and real world co-ordinates. Such a transformation in 3D involves three degrees of rotational freedom and three of translation. In addition, it may involve a scale factor. Recently, Grimson (1987) considers the further possibility of a stretching transformation. As Grimson (1986, p662) remarks, the use of constraints to reduce the number of hypotheses to be generated and tested is the key to efficiency.

The present work addresses ways in which objects can be recognised efficiently from a data base of models rather than just a single model. Consequently, there is even more need to discover powerful constraints that can be applied cheaply to avoid generating and testing too many hypotheses. The outcome of the model identification method described in this paper is a *short list* of model hypotheses that can then then be tested by established means, e.g. by the matcher of Pollard et al (1987). Often, however, the method leads to a short list containing only one hypothesis (modulo a symmetry or two).

Nature of the Sensing Device

Whereas Grimson and Lozano-Pérez aim to be independent of the sensing modality (sonar, tactile, visual) we are particularly interested in using the data from the binocular stereoscopic vision system (Mayhew et al, 1986). This delivers 3D vector descriptions of straight lines and (less reliably) circular arcs in a scene. So far, it has not been possible to deliver surface descriptions. The constraints used by Grimson and Lozano-Pérez assume the availability of surface normals and are not therefore ideally suited. They also rely solely on sparse, unconnected points, whereas the stereo system delivers descriptions of connected edge segments. Although the system often breaks edges and junctions, it very seldom connects them erroneously, adopting a conservative principle of least commitment. Consequently, there is no point in throwing away these edge descriptions.

Therefore, we describe pair-wise relationships between edges in the data rather than between points, whereas Grimson and Lozano-Pérez, followed by Murray (1986), consider pair-wise relationships between points and their surface normals in a surrounding neighbourhood.

The advantages of using pair-wise relationships are retained, namely that they provide a description that is independent of an object's position and orientation and can also be made independent of size. At the same time, an edge of some length can provide more geometrical information, and hence more discriminating power, than a neighbourhood of a point.

Stereo System Design

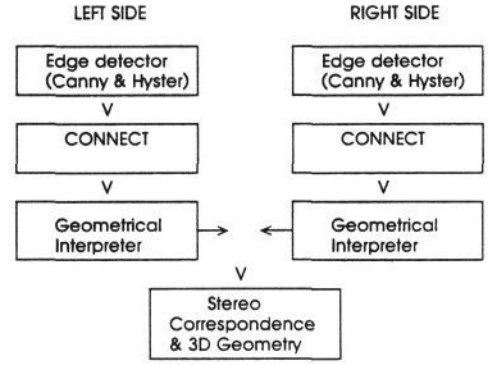

Figure 1. Alternative design of stereo vision system

Some of the experiments have been conducted using the Sheffield system and others with a somewhat different design, as illustrated in Figure 1. The primary

AVC 1987 doi:10.5244/C.1.10

motivation for trying this alternative design was to improve the quality of descriptions of circular arcs for use in the later matching process. Instead of fitting point by point as PMF (Pollard et al, 1985) does the method is to fit straight lines and ellipses to the monocular data and then solve the stereo correspondence problem between them, using the same constraints (notably disparity gradient) that PMF uses.

The results so far have been inconclusive. In some instances this method has yielded circular arcs with normals accurate to within 1° or 2° but in other cases has failed to classify them at all. Sometimes, the Sheffield system yields errors in normals as high as 28°, with commensurate errors in estimates of radius. Such errors render the arcs almost useless for recognition. However, better results have recently been reported with TINA (Frisby and Mayhew, 1987) and it is hoped that those improvements will be repeatable at this site.

Building Models

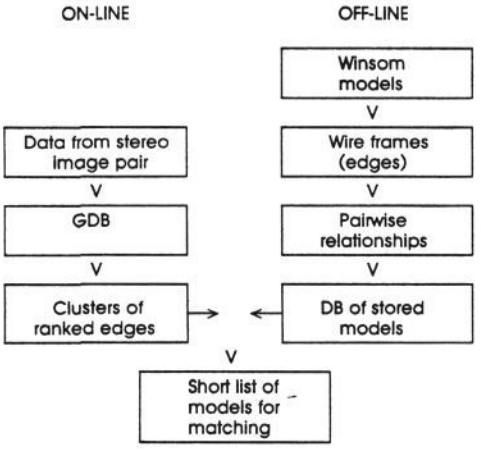

Figure 2. Summary of data flow

The IBM Winchester Solid Modeller - Winsom (Quarendon, 1984) - accepts volumetric descriptions of objects in terms of primitive solids. An extension of Winsom known as FASTDRAW (Halbert and Todd, 1987) produces a list of edge segments which are then classified (Herbert, 1987) as straight or circular. FASTDRAW was designed for a fast display of a model in outline for interactive use in defining models but it lends itself well to this different use.

The wire frames thus produced are then analysed to yield the pair-wise relationships between edges. These are stored in the data base in the form of *identification matrices* and *offset vectors* (see below). All this is done off-line as a kind of compilation.

On-line, the list of edges given by the vision system is examined. Earlier tests were conducted on scenes containing isolated objects. There, the edges were ranked by saliency, which in practice meant longest first. Later tests on scenes containing more than one object employed a simple clustering technique in which close edges (separation in 3-space less than an arbitrary threshold) were put together and ranked closest first.

Identifying Models

The recognition method can be thought of as providing an engine that implements a mapping from a set of features in the scene to a set of models in the data base.

D: $\{f \mid f \text{ in scene}\} \rightarrow \{m \mid m \text{ in DB}\}$

The set $M = D(F)$ is the short list of models that could be represented by the set F of features.

Underlying the mapping D are several primitive functions on features and pairs of features that produce vectors over the features and pairs in the data base. These are described below in the section "Primitives". They rely on the presence of matrices of data derived from the models and stored conveniently.

Perhaps the most important of these primitives is APPLY-RELATIONSHIP. It makes use of the structure of all the models in the data base to eliminate interpretations between pairs that share a common feature if they are inconsistent. This structure is encoded in offset vectors.

Whereas the recognition problem is usually posed as a search through n^k possible nodes for k features from the scene and n features in the data base, many of those possible nodes represent structurally inconsistent hypotheses. Here the problem is formulated as a step by step evaluation of the subsets of F. Each evaluation performs work in proportion to the number of models in the data base, so that the work increases linearly with the number of models. The actual operations involved are very simple, are local within models and hence are well suited to implementation on parallel hardware.

Clustering

In general a scene will comprise several objects and some sort of background. There exists the problem of segmenting the scene into objects. Since the vision system segments it into edges we are faced with a clustering problem in deciding which edges comprise a particular object. This is the "wire frame completion" problem. A complete solution would use visual cues such as connectivity, evidence for occlusion, type of junctions and uniformity of surfaces, as well as consistency with models in the data base.

Here we are interested mainly in the latter. The vision system delivers geometrical but not topological information about edges. A crude clustering scheme has been implemented, however, in which pairs of edges are sorted closest first and clusters are composed of those features nearer than an arbitrary threshold. This worked well on the example of two pegs (Figure 10).

Once a set F of features has been obtained, we wish to know whether it, or some subset of it, matches a small number of models in the data base. In other words, we wish to find $G \in P(F)$ such that $D(G)$ is non-empty but small, where $P(F)$ is the power set of F, the set of subsets of F which includes F itself.

The efficiency with which $P(F)$ is searched now determines the efficiency of the object recognition process. The heuristics employed assume that a set F of features is likely to consist predominantly of edges from one object. The rules are:

1. Grow a subset G starting from the closest two edges in F until $D(G)$ has one member (a unique identification) or $D(G)$ is empty (set G does not represent a known object).
2. If $D(G)$ is empty and G is of size j, consider each subset of G that is of size $j - 1$. Try to grow these without the discarded feature.

The most obvious thing that can go wrong is accidental identification of an incorrect model from a set of edges that actually lie on different objects. No matter how good the constraints built into the model identification mapping D, accidental identification will always remain a logical possibility because of the possibility of genuine coincidence in the scene. In such cases it will be necessary to rely on model verification.

In general terms, one can envisage a more sophisticated (intelligent?) process that could, for instance, reason about the possibilities of occlusion. The present system simply makes no assumptions about occlusion, allowing for the possibility that any edge may be partially occluded. Another improvement would be to re-process the scene through the stereo vision system to remove ambiguities once the first object is identified. It might also be possible to use contextual knowledge to further constrain the mapping D, although the very existence of the data base constitutes a strong form of knowledge already.

Primitives

The data base contains an ordered set of models, each containing features, each of which participates in relationships. Each relationship has a *source* and a *target*. Consider all the relationships $R_1, R_2, ..., R_q$ in the data base grouped by source feature within model. These can be thought of as defining a *vector base* over which *property vectors* and *score vectors* are defined. For some property (e.g. the angle between two lines) a property vector $\vec{p} = (p_1, p_2, ..., p_q)$ would have $p_m = 45°$ if relationship R_m in the data base has this value for this property.

Given a pair of features f_1, f_2 in the scene, we would like to know to which relationships in the data base it could correspond. If the angle between f_1 and f_2 is in the range (θ, ϕ) then the *Boolean vector* $b = (b_1, b_2, ..., b_q)$ is such that

$b_m = 1$ if $\theta \leq p_m \leq \phi$
$= 0$ otherwise

A Boolean vector is a special case of a score vector. The values in a score vector can also be numerical ranges (size factors). Union and intersection can be performed on them.

We define functions $E_{property}$ from pairs of features in the scene onto score vectors for every property that a relationship can have. The score vector of a pair is obtained as the union of these individual vectors.

$E(f_1, f_2) = E_A(f_1, f_2) \cup E_B(f_1, f_2) \cup ...$

Note that, although the functions $E_{property}$ are defined in terms of property vectors, they are implemented more efficiently using *identification matrices* (see below).

Structural Primitives

The two structural primitives are CONTRACT-BASE and APPLY-RELATIONSHIP. CONTRACT-BASE (abbreviated CB) changes the base of a score vector from relationships to features, or from features to models. The score s_m of relationship R_m is transferred to its source feature. Since a feature will have several relationships, a union is performed. Similarly, a union is performed contracting to a base of models.

APPLY-RELATIONSHIP (abbreviated AR) sets the score s_m of relationship R_m to be the same as that of its target feature.

The implementation of these two primitives relies on *offset vectors* that are prepared when the data base is loaded. They indicate the appropriate feature for each relationship R_m in the data base. Further efficiencies could be gained by exploiting the regularities in these offsets using bit masks and shift operations.

These two primitives allow the propagation of constraints from one relationship to another in a manner that is consistent with the structure of the models. Conceptually, this is done for all models at once. Suppose that a pair of scene features f_1, f_2 have a score vector $\vec{s} = E(f_2, f_1)$ (source f_2, target f_1). Then f_2 has a score vector $\vec{s}'_2 = CB(\vec{s})$. Now introduce a third scene feature f_3 giving score vectors $E(f_3, f_1)$ and $E(f_3, f_2)$. None of these three vectors takes account of the structural constraints implied by the other two. However, we can produce a score vector $\vec{s}'_3$ for f_3 that consolidates all the constraints as follows.

$$\vec{s}'_3 = CB(E(f_3, f_2) \cap AR(\vec{s}'_2)) \cap CB(E(f_3, f_1))$$

This score vector is now available to propagate to feature f_4. The general expression is

$$\vec{s}'_j = CB(E(f_j, f_{j-1}) \cap AR(\vec{s}'_{j-1})) \cap \bigcap_{i=1}^{j-2} CB(E(f_j, f_i))$$

Geometrical Constraints

These are much as described elsewhere (Knapman, 1987), relying on the angle between two lines and the two distances

CL orthogonal distance at closest separation of extended lines

CE average distance from each centroid perpendicular to the other lines

The length of a line is now regarded as a property of its relationships. This enables us to use the ratios of the lengths to CL and CE as properties of the relationship that are independent of size. The use of ratios minimises the need for size factor vectors, which are now only required for ensuring consistency between relationships that share a common feature.

The ratios utilised are

$CL/(CL + CE)$, $L1/CL$, $L2/CL$, $L1/CE$, $L2/CE$

where $L1$, $L2$ are the line lengths. (Similar arrangements have been implemented for circular arcs and relationships between arcs and lines.)

Tolerance Vectors

Before using data produced by the vision system from a scene, it is advisable to allow certain tolerances. Crudely, one may apply tolerances of $\pm 5°$ to all angles and ± 5 pixels to all distances computed as properties of pair-wise relationships, with additional allowance for lines that are nearly parallel. This is wasteful because it fails to take account of the high accuracy with which displacements in the image plane can be measured compared with displacements in depth. Consequently, discriminating power is thrown away. On the other hand, a scheme of tolerances must not be too sophisticated because efficiency must be maintained.

The tolerance in each vector is here described by a cuboid around it, defined by a pair of vectors representing opposite corners in a viewer centred co-ordinate frame. The vector representing the end point of a straight line, for example, has a cuboid with x- and y-sides of length 0.2 pixels and a z length corresponding to 5.2 pixels if the line length p in the image plane is 50 pixels. These express tolerances of ± 0.1, ± 0.1 and ± 2.6 respectively. (They are proportional to $1/\sqrt{p}$, reflecting the way that the accuracy of measurement depends on the number of points in both the left and right images.)

Such a vector pair is termed a *tolerance vector*. Operations, including dot and cross products, are defined on tolerance vectors by finding the maxima and minima of the individual components of these products. When using the dot product, a pair of numbers results. Care must be taken over the sign ambiguity when finding a range of angles by way of the inverse cosines of such a pair. This is done by checking for a sign change in the cross product.

Identification Matrices

In order to minimise arithmetic operations, a scheme of bit maps is introduced. It depends for its success on the use of size independent values of properties of pair-wise relationships wherever possible.

On a pair of straight lines, for instance, use of the tolerance vectors leads to a range of values of *CL* and *CE*. Hence ranges are found for the ratios and the angle between the lines. Lengths are regarded only as a lower bound on the true length because of possible occlusion or broken lines.

Once a range of values has been found for a property of a relationship between two edge features, *identification matrices* are used to produce a Boolean vector as illustrated below.

	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	...	R_n
90°	1	0	0	0	0	1	1	0	...	0
89°	1	1	0	0	0	1	1	0	...	0
.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.
1°	1	1	0	1	1	1	1	1	...	1
0°	1	1	1	1	1	1	1	1	...	1

In the identification matrix there is one Boolean vector (row) for each angular value between 0° and 90°. Every relationship (R_1, ..., R_n) in every model in the data base is represented by a column. Relationship R_2 has an angle of 89° so it has a 1 in the vector for 89° and in all those beneath it. In row θ°, the 1s indicate those relationships with angle greater than or equal to θ.

Our models are rigid and so the complement of this Boolean matrix is used to indicate those relationships with angle less than θ. More generally, two identification matrices could be used to support models with ranges of allowable values. This is a possible approach to describing hinged or articulated objects.

At present, when an angular range (θ, ϕ) is found from the scene by way of the vision system and the tolerances, we take the intersection of two vectors, one the complement of that found in the matrix, to obtain a Boolean vector showing those relationships R_i in the data base with angles ψ such that $\theta \leq \psi \leq \phi$. These are the relationships that could correspond to the relationship found in the scene.

A more sophisticated arrangement of identification matrices is used for the ratios that may range over thousands of significant values. The method uses overlapping ranges so that a potential set of 65536 vectors is reduced to an optimum set of 64. 17 operations on bit strings are then needed to obtain a Boolean vector representing a range of data values.

Results

The seven test objects illustrated in Figures 3 to 9 were distinguished against a data base containing models of all of them after considering the number of edge features from the scene indicated in the table. This test used the sizes of the objects. Each scene was of an isolated object.

Object	Number of scene features used
Widget	3
Plug	3
Wedge	2
Cube	3
Ice cream	2
Frame	6
Chair	2

In another test, the scene in Figure 10 consisting of two "pegs" was successfully divided into two clusters, both of which were identified correctly against a data base of 12 models. When the size was used, 3 features from each cluster were sufficient for a unique identification. When size was not used, 6 features from one cluster were needed and 7 from the other for unique identification.

Acknowledgements

I have had valuable discussions relevant to this topic with John Mayhew, John Frisby, Stephen Pollard, Eric Grimson, Daniel Huttenlocher, Bernard Buxton, Andrew Blake, Bob Fisher, Mark Orr, John Woodwark, Rod Cuff, Steve Rake, Alex Lebek and Tom Troscianko. I am grateful to Simon Herbert, Rod Cuff and John Holland for assistance and co-operation.

References

1. **Frisby, J P and Mayhew, J E W** 'TINA: A Stereo Computer Vision System' A I Vision Research Unit, University of Sheffield, UK (January 1987)

2. **Grimson, W E L and Lozano-Pérez T** 'Model based recognition from sparse range or tactile data' *International Journal of Robotics Research* 3(3) pp 3-35 (1984)

3. **Grimson, W E L** 'The Combinatorics of Local Constraints in Model-Based Recognition and Localization from Sparse Data' *JACM* 33(4) pp 658-86 (1986)

4. **Grimson, W E L** 'Recognition of Object Families using Parameterized Models' *Proc 1st International Conference on Computer Vision* pp 93-101, London, UK (June 1987)

5. **Halbert, A and Todd, S** 'FASTDRAW' IBM U K Scientific Centre, St Clement St, Winchester, UK (forthcoming 1987)

6. **Herbert, S** 'Description of FASTDRAW post processor prototype' unpublished memo, IBM U K Scientific Centre, St Clement St, Winchester, UK (May 1987)

7. **Knapman, J M** '3D Model Identification from Stereo Data' *Proc 1st International Conference on Computer Vision* pp 547-51, London, UK (June 1987)

8. **Mayhew, J E W et al** 'GDB Release 1.0 User Documentation' A I Vision Research Unit ref no 014, University of Sheffield, UK (1986)

9. **Murray, D W** 'Model-based recognition using 3d shape alone' GEC Research Ltd., Wembley, UK (1986)

10. **Pollard, J, Mayhew, J E W and Frisby, J P** 'PMF: A stereo correspondence algorithm using a disparity gradient limit' *Perception*, 14, pp 449-70 (1985)

11. **Pollard, S B, Porrill, J, Mayhew, J E W and Frisby, J P** 'Matching geometrical descriptions in three space' A I Vision Research Unit ref no 022, University of Sheffield, UK (1986)

12. **Quarendon, P** 'Winsom User's Guide' IBM U K Scientific Centre report no 123, St Clement St, Winchester, UK (August 1984)

Figures

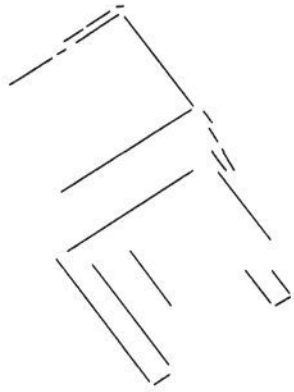

Figure 3. The chair

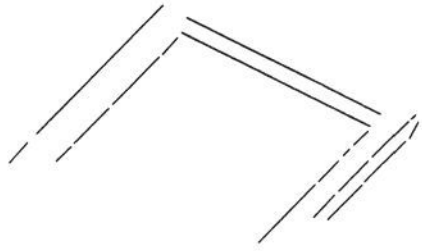

Figure 4. The frame

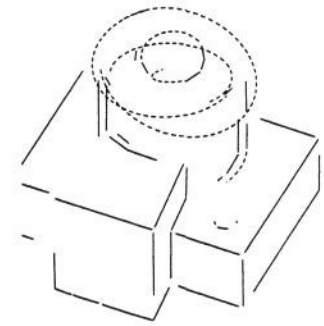

Figure 5. The widget

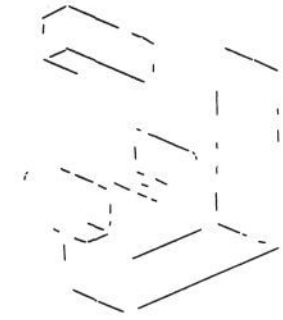

Figure 6. The plug

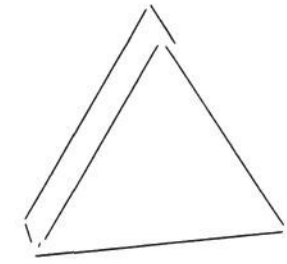

Figure 7. The wedge

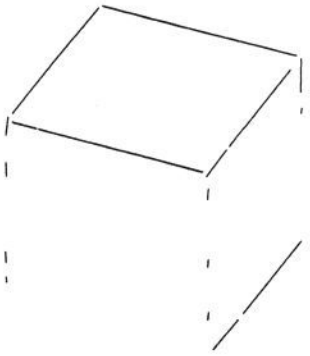

Figure 8. The cube

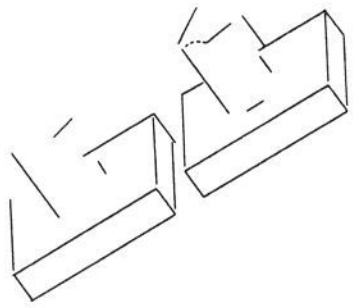

Figure 10. Two pegs

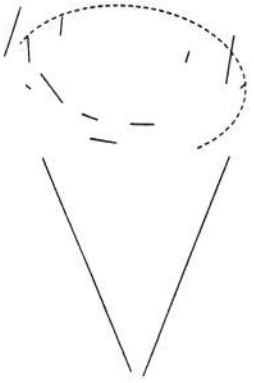

Figure 9. The ice cream cone