WPFM: THE WORKSPACE PREDICTION AND FAST MATCHING SYSTEM

J. C. Aylett, R. B. Fisher and A. P. Fothergill

Department of Artificial Intelligence
Working Paper No. 207

October 6, 1987

WPFM: The Workspace Prediction and Fast Matching System

Jonathan Aylett, Robert Fisher, Patricia Fothergill.
Department of Artificial Intelligence
University of Edinburgh
Forrest Hill
Edinburgh EH1 2QL

Abstract

This paper describes the Workspace Prediction and Fast Matching System the - *WPFM System.* The purpose of this system is to exploit the special constraints on the identity and placement of objects in a typical industrial robot workcell to perform rapid analysis of stereo vision data derived from such a scene. The *a priori* knowledge of the positions and identities of some objects can allow some stereo data to be quickly matched to model features, and subtracted out of the scene data, leaving unknown data to be dealt with by a more comprehensive (and more computationally expensive) vision matcher [9]. A workspace prediction graph (WP Graph) for a given scene can be created using the Robmod [7 & 8] solid modelling system (the WP System). The 3D model features in the WP Graph can then be matched against 3D stereo data derived from the equivalent real scene, using the Fast a priori Matching System (FAPM System). Matching is currently carried out only with edge features. The stereo data used in the matching process is derived from the real scene by a low level stereo vision capability (the GDB system [14]). Curved objects can be handled by the WPFM system, and these are represented in the WP Graph by polyhedral approximations, which are then matched to stereo data. The system was constructed as a subcomponent of a more comprehensive vision system, and is essentially a type of vision verification system.

Keywords : vision, solid modelling, verification, robot.

Acknowledgements

Funding for the WPFM project (originally called the 2 1/2 D Sketch Prediction Project) was under Alvey grant GR/D/1740.3. Vision test data was provided by T P Pridmore and S B Pollard of AIVRU Sheffield, who also provided help and advice with the GDB System.

1. Introduction

An important area for the application of machine vision is automatic assembly by robots. In general, the micro-world in which the industrial assembly robot may be working, although complex is also highly constrained. There is usually much information available *a priori* about objects and their disposition within a robot workcell. In order to maximise the efficiency of a general purpose vision system, we argue that it is often possible to use this *a priori* information to avoid extensive visual processing for constant scene features, or for objects in the scene for which there are accurate estimates of current position. This is the motivation for the development of the *WPFM System.* The different components and the operation of the system are shown schematically below (Figure(1)).

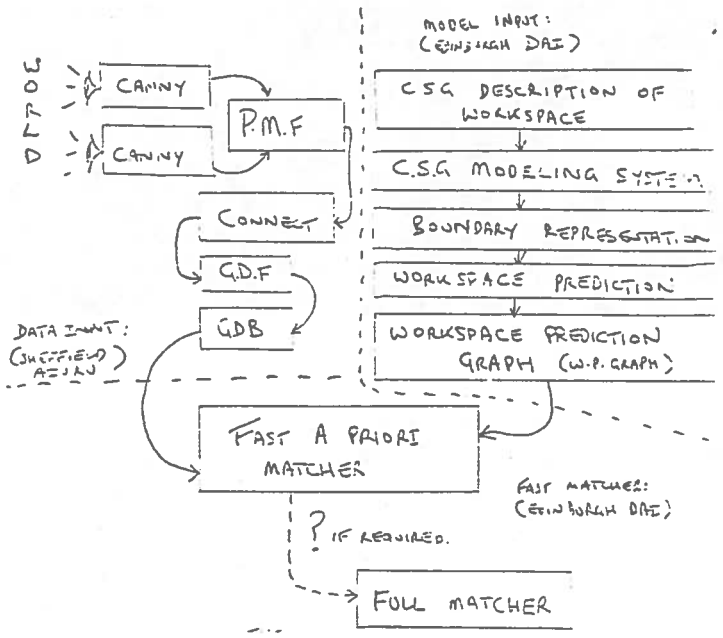


Figure 1 - The WPFM System

There are two main data paths into the system. Solid model data is created by a CSG modelling system and written out to a visibility prediction file, called the *Workspace Prediction Graph,* or *WP Graph.* This data structure is then input into the *FAPM System;* currently only the edge and vertex data is used. A pair of stereo camera images of a scene containing the predicted objects is analysed by the *GDB System* [14], and the 2 1/2 D data extracted is output to a *GDB file* which is then input to the *FAPM System;* only linear edge segments and circular arcs are used. The *FAPM system* then matches the *GDB data* to the *WP Graph* and outputs information about the matching process to the user as screen output, and in a new *GDB file* containing any unrecognised image data.

2. The Industrial Micro-World

A typical industrial robot operates in a robot workstation which is a tightly constrained micro-world (see Figure (2)), and within this small world, the positions of the majority of objects can be accurately determined. For a typical scene, this quantitative geometric information could be derived from a CAD based model of the robot workstation and from the known location of a camera system used to view the scene, in conjunction with the camera calibration parameters. The purpose of the WPFM system is to exploit this *quantitative a priori information* about the identity and disposition of objects, to enable us to carry out *fast a priori matching*

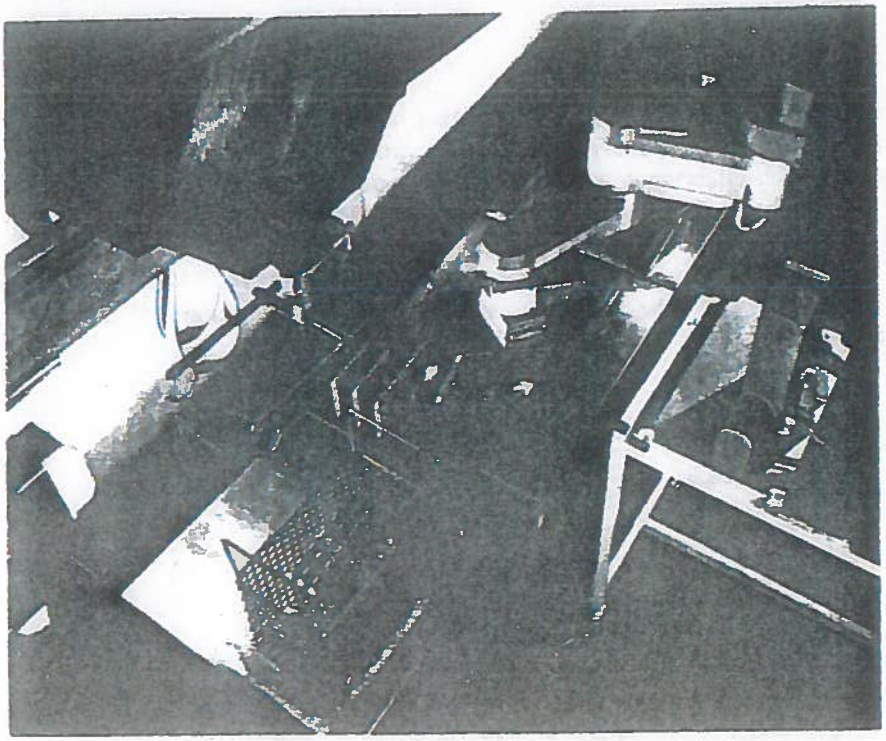


Figure 2 - A Robot Workstation

(FAPM), in order to do a rapid "*scene subtraction*". A schematic illustration of the type of scenario which we are attempting to tackle is shown in Figure(3), where the only difference between the two modeled scenes is that in one case the robot gripper has "dropped it's block". (These two pictures were generated by the Robmod solid modeling system).

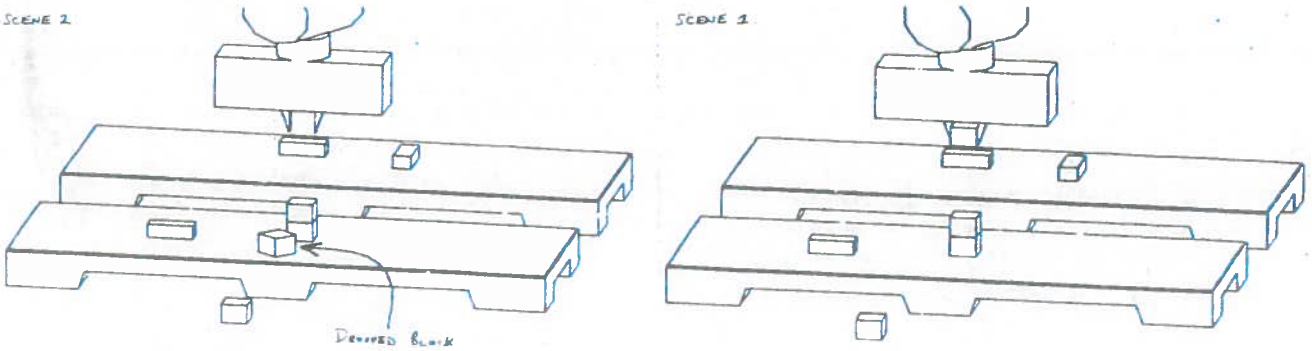


Figure 3 - A Typical Scenario

Some of the important features of an industrial robot workstation are :

1. There will be many objects in the scene.
2. Some objects will be static.
3. Some objects may move, but in a predetermined way, under control of the robot system
4. At any given moment the robot system should have good estimates available for object locations, except for objects that have been dropped, knocked over, or introduced into the workstation by some

other agency, such as a conveyer belt.

5. No assumptions can be made about lighting conditions, as this is difficult to control and may vary over time.

The scene subtraction process used in the *WPFM System* is a "*symbolic subtraction*" of edge features, rather than an "*image subtraction*" of pixels. This is partly because image subtraction would cause image artifacts and other errors which would produce misleading results, and partly because the encoding of the scene data into a *2 & 1/2 D data* description is a richer description than a pixel grey level image and it also condenses the large quantity of image pixel data into a succinct symbolic description in terms of *visible feature descriptions*. This symbolic representation can then be used in conjunction with an equivalent symbolic model of the scene to deduce the source of any data not matched. The model based approach also provides an effective mechanism for creating the scene predictions required for the subtraction process.

3. The Workspace Prediction Graph - (WP Graph)

Solid modelling systems are good vehicles for modeling man-made industrial type objects [4 & 17], like the robot in Figure(4). They have been used for modelling robot workstations, to predict collisions [7], plan robot movements [6] and develop robot programs off-line [13].

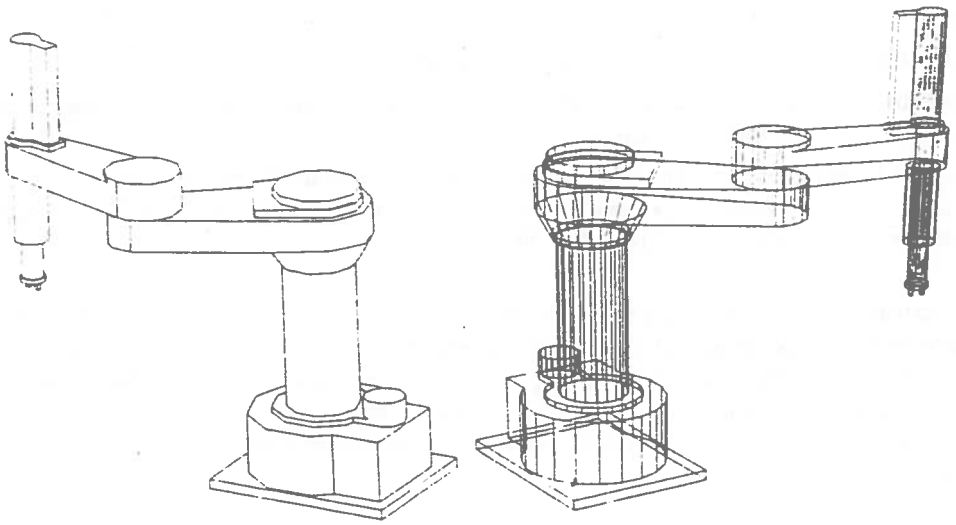


Figure 4 - Wireframe & Hidden Line Drawings of Robot Model

The Robmod CSG modeling system [7 & 8] is used at Edinburgh in conjunction with the RAPT system [13] to develop off-line robot programs. Unlike boundary based modelling systems, CSG systems build objects by combining a set of primitive solid shapes together, using boolean set operations. By including information about the way an object is constructed from these primitive solid shapes into a boundary representation, we can

construct a data structure containing the geometrical and topological information about an object, as well as a relational aspect. This relational aspect comprises a description of the *visually salient parts* that go to make up an object, and extends the concept of a boundary representation to include information specifically useful to vision processing [2]. This extended boundary description is a development from the classical winged edge type structures used by Weiler [19] and others [16 & 20], and a variation of this developed by IBM for use in machine vision applications [10]. Our variant has a hierarchical structure which starts by composing distinct objects into *separate assemblies*, then *CSG primatives* (convex or concave), then into *surfaces*, and then *edge boundaries* of the surfaces. The geometry is associated with edge and surface descriptions, which in the Robmod version are all polyhedral based, (see annotated skeleton in Appendix 1).

To produce a *Workspace Prediction Graph* we have further enhanced the Robmod system, so that a CSG model can be analysed from a particular viewpoint and visibility information added to the boundary representation described above. This is essentially a four stage process :

1. Produce a boundary representation for a modelled scene.
2. Perform hidden line removal of the modelled scene from the required viewpoint, storing the edge fragments produced, and connecting these into full edge segments.
3. Compare the non-occluded and partially occluded model edges to the boundary data, thereby tagging edges in the boundary representation with visibility status. Add any virtual vertices produced by partially occluded edges.
4. Output a boundary file with the extra visibility information.

The *WP Graph* produced is identical to the boundary representation described above, with the addition of the following extra information :

Total number of visible, part visible and non-visible edges.

Total number of visible, virtual and non-visible vertices.

Viewing parameters - viewing angles, viewed point and frame size.

Edges sorted by visibility type into lists - visible, part-visible and non-visible. Some edges may be tagged as extremal, for 'curved' primatives.

Vertices sorted by visibility type into lists - visible and non-visible. Where any model edges have been occluded, a virtual vertex is produced at the T junction. These are organised into a separate list, (and do not occur in the original boundary file).

The entire structure is organised as a graph, in the same way as the boundary file, with linking pointers between associated features (see annotated skeleton in Appendix 2). Note that the edges of the facets used by Robmod to model curved surfaces by polyhedral approximation, are treated as non-visible edges, as these are features which are not seen in real objects. The only exception to this is the *extremal edge* - which corresponds to a visible facet edge bounding the first non-visible facet. Thus, an extremal edge corresponds to a facet edge in the original boundary file.

Although the FAPM system only requires the use of the edge features in this graph, a more complete description is produced, as within the limitations of the planar approximation used by Robmod, an attempt has been made to produce a representation of real objects which would be as complete as possible [2]. Thus the WP Graph could be adapted to be used with a more general purpose vision system, or extra model features could be used in an enhanced FAPM system. The extra topological and geometric data in the WP graph could be then be used for disambiguating as far as possible, any uncertainties in matching data to model features.

As it is computationally expensive to produce a *WP Graph*, (about 1 to 10 minutes on a SUN - 2, depending on the complexity of the modelled scene) the *WP Graph* would be produced off-line for different *expected* scenes, as the positions of constant background and foreground scene parts, and the positions of a robot at different points in task program, could be determined in advance.

The Robmod modeller was used to generate the *WP Graph* as it was originally developed to model movable, jointed, and distinct objects, and has several important facilities to support this [4 & 5], such as the use of separate assemblies for distinct objects, which can also be linked by rotation axes, variables which can be dynamically set, and the facility to run programs using the Robmod programming language (a simple language similar to Basic).

4. The Fast a priori Matching System - (FAPM)

As described earlier, the matching system uses edges and their associated vertices to match model to data. However edges can have some disadvantages for use in matching, if many edges are fairly close to each other, or if objects are highly textured at a coarse scale, or objects are continuously smooth, and only have extremal or occlusion boundaries. Edge type features can also be produced by shadows, reflections and surface markings, although the use of stereo data can help to reduce these problems as real depth discontinuities can be detected in 2 1/2 D data.

However, many industrial type objects can be described adequately in terms of edges. This is fortunate, as passive vision systems such as the *GDB System* used here, can only extract (mainly linear) edges from images,

which therefor requires an edge based matching process. Also, an edge based model representation can be produced faster than a surface based description, which has to be produced by ray casting the CSG model, as for example in the IBM CSG modeller Winsom [15]. Visibility data can also be readily derived for edge features in such a boundary representation.

Although we use *a priori* information about an expected scene, there will not be a perfect correspondence between the features extracted from the scene by the *GDB system*, and the features produced by the *WP system*. The main problems encountered in matching data to model features arise from this imperfection, which can be caused by various factors, and can result in extra, missing or different features occurring in the *WP Graph* and the *GDB data*.

Imperfections in the WP Graph

1. Simplification of the shapes of the objects when modelled in order to be able to represent them in the modelling system. (eg, Robmod produces polyhedral approximations of curved objects such as cylinders).

2. The *WP graph* will include any part of a known object in the scene which is theoretically visible from the camera, (i.e. any feature that lies on an unobstructed ray from the camera). However, some of these features may be too small to recovered by the vision system.

3. Incorrect locations of a some model features, caused by objects with locations different to those predicted.

4. Missing model features, because the system will not attempt to predict features arising from objects with unknown locations.

Imperfections in GDB data

1. Features may not be appear in the data because lighting conditions can make them invisible (eg lost in the shadows).

2. Extra features can also be produced by the *GDB system*, caused by reflectance changes, shadows (shadow edges), specularities or texture.

3. Features may be occluded in the image that were predicted to be visible, so these features will exist in *WP graph* but not in the data. A single feature may be fragmented by an unpredicted partial occlusion, and so have an incomplete mapping to the corresponding model feature.

4. Data can be fragmented by imperfections in an image, such as noise, and may be incomplete, so several data edge segments may correspond to a single model edge.

5. Tangential occluding boundaries on curved surfaces are difficult to resolve accurately by a stereo process and may be placed inaccurately, particularly in depth.

6. Linear edge features parallel to the plane of the stereo camera system are difficult to resolve stereoscopically and may also be placed inaccurately.

7. Curved edge features may be inaccurately segmented into several linear data segments.

Other differences between the *GDB data* and the *WP graph* could be caused by errors in the imaging process and by inaccuracies in the robot system's model of the world. In our experiments, the tests with real data have highlighted the problems caused by inadequate camera calibration, and illustrated some of the problems with real data discussed above.

If we consider the reasons for extra and missing features in the data, as compared to the model, we shall see that they give rise to two types of discrepancy. One type is globally distributed local inconsistency, and the other type is locally distributed local inconsistency.

We call the first "*global incorrectness*" and the second "*local incorrectness*".

Global incorrectness is caused by lighting, texture, camera shift and random noise, and can occur anywhere in the image. Any discrepancies will be evenly distributed in the data, and require the use of tolerances by the matching process.

Local incorrectness is caused by an unexpected object obscuring features, or by an expected object not being visible. Any discrepancies will be localised in the data, and it is these areas of local discrepancy which should form the foci of attention for the full vision system.

Clearly, assumptions are required in order to evaluate any unmatched model or data features, and to establish how "good" a match is acceptable. To aid this process, the *FAPM System* produces a summary of how "good" a match has occurred at the end of a matching run.

5. FAPM - Implementation and Matching Algorithms

The *FAPM System* matches data input from a *GDB File* and model input from a *WP File*. Matching proceeds by pairing data to model edges using geometrical algorithms. The objective is to find all data segments that match the model, and to determine how well the data matches the model features. Before matching is attempted however, the model needs to be calibrated to the data. This requires scaling the model to the same scale as the data, as the GDB System outputs lengths in pixel units not in the actual metric lengths. Also, the model coordinate system may have to be transformed by a displacement along the Z axis so as to correspond to the coordinate frame of the stereo camera system. These transformation and scaling parameters are key *a priori* inputs to the matcher.

Tolerance values are also required for the matching process. The choice of values for these parameters depends on the quality of the data, as inferior data requires wider tolerances for matching. The tolerance parameters required are :

1. The *maximum divergence angle* between linear model and linear data segments. This sets a tolerance angle within which a linear model and a linear data segment are considered to be parallel.

2. The *maximum tessellation divergence angle* between model tessellation and linear data segments. This sets a tolerance angle within which a section of model curve (approximated by a linear tessellation) and a linear data segment are considered to be parallel.

3. The *maximum perpendicular divergence angle* between the axis of a circular arc in the data and a "curved" model segment. This parameter defines the tolerance angle within which the normal to the plane of the circular arc and a "curved" model edge segment are considered to be perpendicular, and

therefor the angle within which the model edge is considered to lie in the plane of the arc.

4. The minimum proportion of the data segment to model segment *overlap margin*. This defines the maximum length of data segment allowed to fall outside a model edge. This overlap length is calculated as a proportion of the model edge length.

5. The maximum *separation distance* between a linear model edge and a linear data segment. This sets the upper bound on the separation distance at the point of closest approach between a linear model edge and linear data segment.

Once the calibration and tolerance parameters have been set, model edges can be matched against data segments. We summarise this matching process below :

1. For each data segment, determine if it lies within the model circumsphere (see below), if not, mark as unmatched.

2. For each model edge, scan all data segments and attempt a match. Mark paired data segments and model segments as matched, and for each model edge, record a total accumulated length of data segments matched.

Stage 1 of the process acts as a coarse filter, and thereby improves performance. The circumsphere surrounds the modelled object, and is generated by Robmod as part of the WP Graph. For speed, the test used is a quick point test on each data segment. Data segments have both endpoint coordinates tested to determine if one or both lie within the circumsphere. This is a conservative test as it does not remove all features located too far from the model, partly as the circumsphere does not "fit tightly" (it is not a convex hull), and partly due to the use of approximate calculations which minimise the amount of computation required. However this test can remove a considerable number of data segments quickly from the matching process and becomes more significant as the number of distinct objects increases.

Stage 2 is the main matching process. Each model edge is matched against all the remaining data segments. For linear model edges, there is allowed only a unique pairing of data segments to model edges, such that if one data segment is matched to a linear model edge, that data segment cannot be matched to another model feature. The data segments are paired to the first matching model edge, and a "*best fit*" test for any competing model edges is not applied, as this situation should only occur with very loose matching tolerances or with a data/model mismatch. Hence for m model edges and d data segments, this matching process is of the order of $(m*d)/2$ complexity, although this is reduced by the use of the circumsphere test or increased as the number of "curved" model edges increases (as these do not have a unique pairing relationship to data segments). The computational complexity of the matching process is kept within these bounds, as it is not necessary to estimate a reference frame transformation to register model to data [12 & 18], or to select which model to match to data by a process of model invocation [9], as this information is determined *a priori*.

5.1. Matching Linear Features

The matching process for each linear model edge is subdivided into four tests (the tests are slightly different for "curved" model edges or curved data). Data is rejected at the first test in the sequence that fails. Linear edge segments in the model are represented by (v_m,m_1,m_2, M_l) and in the data by (v_d,d_1,d_2, D_l), where the four components are the unit direction vector ($v_m = m_2 - m_1$), the two endpoints and the length. The tests are as follows :

1. Quick point test. Tests if either point on the data segment has an approximate separation distance from a point on the the model edge greater than the length of the model edge. This acts as a computationally inexpensive coarse filter, considerably speeding up matching. The test will succeed if all the following conditions are true :

$$|m_{1i} - d_{1i}| < M_l$$

$$|m_{1i} - d_{2i}| < M_l$$

$$|m_{2i} - d_{1i}| < M_l$$

$$|m_{2i} - d_{2i}| < M_l$$

where $i \in (1,2,3)$, the vector components.

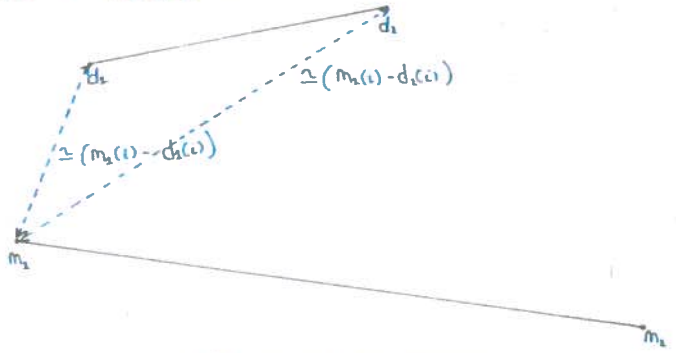


Figure 5 - Quick Point Test

2. *Angular deviation test.* Tests if the angle of separation of the direction vectors of the data segments and model edges is within the specified tolerance angle. This is a relatively fast test, and quickly removes a large number of false data to model pairings.

For a maximum deviation angle D_a the test will succeed iff

$$|v_m.v_d| > \cos(D_a)$$

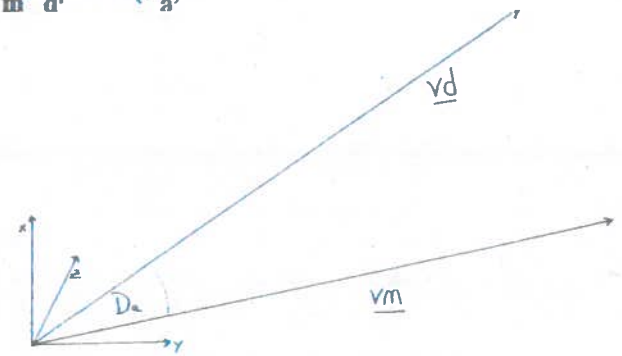


Figure 6 - Angular Deviation Test

3. *Sub-segment test.* Tests if the projection of a data segment onto a model edge falls within that model edge. A tolerance parameter requires the data to overlap the model edge by a fractional proportion of the model edge length, and for a linear data segment both endpoints must fall within this interval. The overlap parameters are given by :

$$Ov_1 = (d_1 - m_2).v_m$$

and $$Ov_2 = (d_2 - m_2).v_m$$

And the test will succeed iff

$$-F_o M_l <= Ov_1 <= M_l + F_o M_l$$

and $-F_o M_1 <= Ov_2 <= M_1 + F_o M_1$

Where F_o is the overlap proportion tolerance parameter.

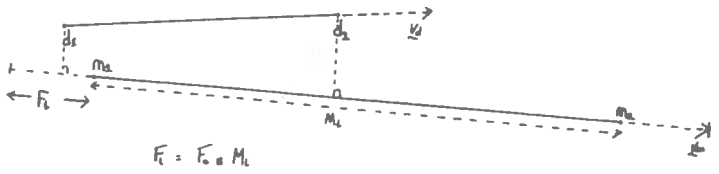


Figure 7 - Sub-segment Test

4. Minimum distance test. Tests to determine if the closest point between the the data segment and the model edge is within the specified separation distance limit. This test will succeed iff

$$|(Ov_1 \; v_m) + m_2 - d_1 | < S_d$$

and $|(Ov_2 \; v_m) + m_2 - d_2 | < S_d$

where S_d is the minimum separation tolerance parameter, and Ov_1 and Ov_2 are as defined in test(3) above.

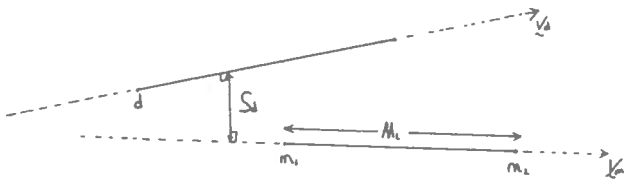


Figure 8 - Minimum Distance Test

The algorithms used in tests (2-4) above were derived from algorithms originally developed by Watson [18] for use in a wire frame based vision matching system, and are similar to those used in edge based combinatorial matching systems [11 & 12].

A linear data segment that passes all of the above tests will be paired to the linear model edge and the uniqueness criterion will be enforced by not rematching *data segments* to other model edges.

5.2. Matching Curved Features - Data and Model Tessellations

There are two further cases of matching, both of which involve the matching of a "*curved*" model edge to data. As curved model edges are represented in the WP Graph by a tessellation of the curve into a number of smaller linear edges, it is these *tessellation edges* that are matched to the data. The two cases differ in that curved features may be segmented by the GDB as *linear* or *circular arc* data segments. Hence in the first case,

the GDB tessellates curved data into a number of linear data segments, whilst in the second case circular features are segmented into one or more circular arcs. To match model tessellations to data tessellations, we use a similar (but not identical) method as used above for the linear case. This matching process is :

1. *Quick point test*, similar to test(1) above, except that only one pair of conditions is required to be true. Hence the test will succeed if :

$$|m_{1i} - d_{1i}| < M_l$$

and $|m_{1i} - d_{2i}| < M_l$

or $|m_{2i} - d_{1i}| < M_l$

and $|m_{2i} - d_{2i}| < M_l$

2. *Angle test* as test(2) above, although the maximum deviation angle is normally set to a larger value than used for the linear case, to allow for possible extra divergence between model and data segments which can occur if the tessellation of the data and the tessellation of the model are out of step. Ideally this has a maximum value when tessellations are misaligned by 50%, and so the maximum extra divergence angle is **pi/N**, where N is the number of tessellations in the polyhedral approximation of the original CSG primative (ie a cylinder). This model tessellation "scale" was set to 10 in all cases.

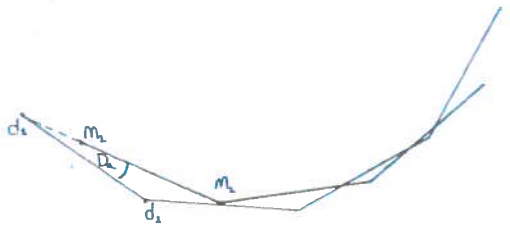


Figure 9 - Angular Deviation for Tessellated Edges

3. *Sub-segment test* as in test(3) above, except to allow for the possible "stepping" problem between tessellations, we use initially a zero overlap and require only one endpoint to overlap. If both overlap, the process is as before, otherwise we then run a subsidiary test to determine how much of the *data segment* overlaps the model edge. This subsidiary test is passed if a sufficient proportion of the data segment overlaps the model edge, (an overlap of 30% was used).

In some cases, the data segment can be larger than the model edge. This is usually caused by by the GDB segmenting a curve into one rather than several linear segments which will then correspond to several tessellation edges in the model. For this case, the algorithms are as as described above, except that the relationship between a model edge and a data segment is reversed as several model edge segments will be paired to one data segment.

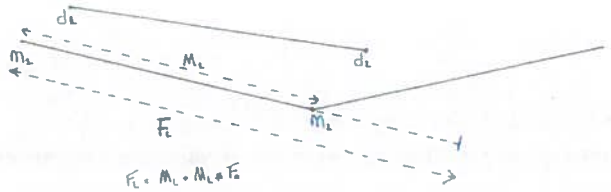


Figure 10 - Sub-segment Test with Tessellated Edges

4. *Minimum distance test* as in test(4) above, unless the overlap is only partial, in which case one minimum distance is tested corresponding to the one valid overlap.

Due to the "stepping" problem described above, the uniqueness criteria cannot be applied in this matching process. Because of this, and the poorer data produced by the GDB for curved features, some tolerance values and some tests have to be relaxed, as above. Hence to minimise the possibility of incorrect pairings between model and data occurring, the linear model edges are matched to the data before the tessellated model edges. The lower accuracy of this matching process is partially offset by the stronger pairing relationship implied by a match between a curved model edge and a curved data segment.

5.3. Matching Curved Features - Data Arcs

A circular arc in the GDB is represented by (R_1, c_c, d_1, d_2) where R_1 is the radius of the arc, c_c the centre point and d_1 & d_2 the two endpoints of the arc. To match a model tessellation edge to a circular arc, the test sequence is as follows :

1. *Quick point test* as in test(1) above, except that the point distance is the radius and the data point used is the centre of the circular arc.

2. *Angular deviation test.* The normal to the plane in which the data arc lies is compared to the direction vector of the model edge. It matches if these are perpendicular to each other to within the maximum deviation angle.

The unit direction vectors for the two radii are given by :

$$r_1 = (d_1 - c_c) / R_1$$

$$r_2 = (d_2 - c_c) / R_1$$

and the unit normal vector n_d to the plane of the arc is given by :

$$n_d = r_1 \times r_2 / \|r_1 \times r_2\|$$

For a maximum deviation angle D_a the test will succeed iff

$$|v_m . n_d| < \sin(D_a)$$

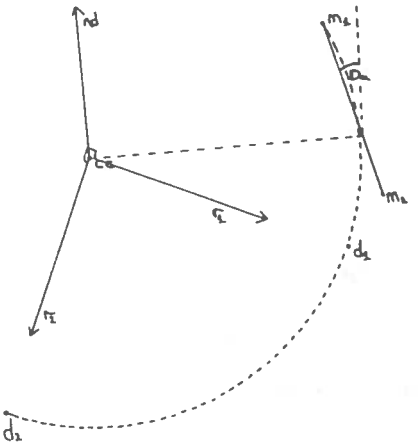


Figure 11 - Angular Deviation Test for Circular Arcs

3. *Minimum distance test.* Determines if the end points of the model edge fall within the radius distance from the centre of the data arc segment, within the separation distance limit. For a separation distance limit S_d the test will succeed iff

$$|(||(c_c - m_1)|| - R_l)| < S_d$$

and $$|(||(c_c - m_2)|| - R_l)| < S_d$$

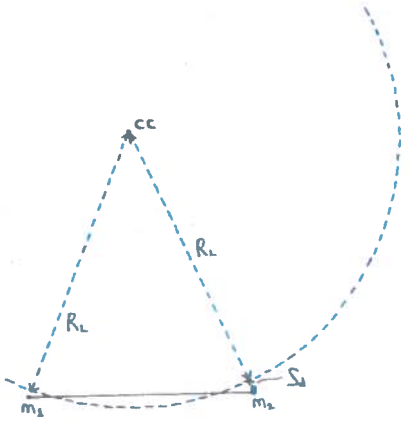


Figure 12 - Minimum Distance Test

4. *Circular arc overlap test.* This determines if the model edge overlaps the data arc, by intersecting the two angle ranges of the two model edge endpoints with the angle range of the data arc. The test passes if the required proportion of the angle range of the model tessellation edge is found to overlap the data arc, (the required overlap was set to 30%).

In some cases, the data arc can be smaller than the model tessellation edge, usually due to data fragmentation. For this case, the test is the same, except that 30% of the *data arc* is required to overlap the model edge for the test to succeed.

The angle range A_a of the data arc is given by :

$$A_a = \arctan ((r_2 . (n_d \times r_1)) / (r_2 . r_1))$$

this gives an angle value for A_a in the range 0 -> 2pi, after allowing for the signs of the numerator and denominator and adjusting any negative result to the appropriate positive angle.

To determine the angle ranges $[A_{m1}, A_{m2}]$ subtended by the model segment endpoints (m_1, m_2), we calculate the projection onto the plane of the data arc of the radial direction vectors to these endpoints, as follows :

$$p_{m1} = (m_1 - c_c) - n_d ((m_1 - c_c) . n_d)$$

$$p_{m2} = (m_2 - c_c) - n_d ((m_2 - c_c) . n_d)$$

and the angle ranges $[A_{m1}, A_{m2}]$ are then given by :

$$A_{m1} = \arctan ((p_{m1} . (n_d \times r_1)) / (p_{m1} . r_1))$$

$$A_{m2} = \arctan ((p_{m2} . (n_d \times r_1)) / (p_{m2} . r_1))$$

with appropriate adjustments to the resultant angle values as for angle range A_a above.

The overlap between the data and model angle ranges is then given by the intersection of the two

ranges :

$$O_v = | [0, A_a] \cap [A_{m1}, A_{m2}] |$$

and the test will then succeed iff

$$|O_v / (A_{m1} - A_{m2})| >= 0.3$$

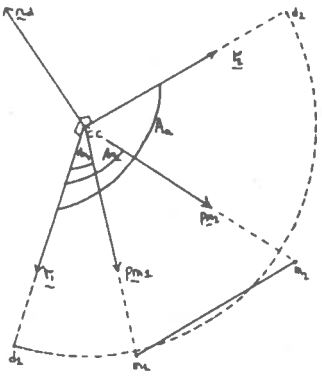


Figure 13 - Circular Arc Overlap Test

Although the algorithms used for matching circular data arcs differ from the previous case of matching tessellation to tessellation, the uniqueness constraint still does not apply in the matching process, as again one data segment may be required to be matched to several model tessellation edges. However, the problem of a lower resolution in the matching process is reduced as the "stepping" problem encountered with tessellation to tessellation matching does not arise.

6. Results of Matching

At the end of the matching process an updated GDB file is output, with indicators of which data segments have been matched, as well as graphics displays indicating which model edges and data segments have been matched. A matching "*goodness*" summary is also produced to indicate how close a match was obtained between model and data, and is summarised as the following :

Model edges predicted visible.

Model edges predicted visible and matched.

Total model edge length predicted visible.

Total model edge length matched.

Total data segment length matched.

Total data segment length matched as a fraction
of total matched model edge length (ML fraction).

Total data segment length matched as a fraction
of total model edge length predicted visible (VL fraction).

The values in the matching summary can then be used to determine whether a match is acceptable, and which data segments can be safely ignored in further vision processing [9].

7. Experimental Results

The workspace prediction and fast matching system has been implemented in C and is running on UNIX SUN-2 & SUN-3 computers at Edinburgh. We have run the system on several test cases, the results of which are indicated below. The module has currently only been run on its own, and not yet in conjunction with the overall vision system, which is the eventual objective. The test data consists of two sets of stereo images provided by AIVRU Sheffield and synthetic data generated by the IBM CSG Modeller Winsom. The Winsom modeller can generate a pair of stereo pixel images, given a CSG "scene" description, equivalent to real stereo images [15]. The resolution of all the test images was 65K pixels (256 by 256 square).

The system was run first on the Winsom data, as it was easy to calibrate this synthetic data to the model, and we were able to determine the maximum performance of the GDB System for an idealised test case. Also, the GDB System was capable of producing a number of circular data arcs, which enabled us to test this aspect of the matching system.

The real scenes contained a single known test object in isolation and in a cluttered scene. In all cases, we attempted to match only the test object at a predicted position. This test object is a moderately complex engineering type object generally known as the "widget". The dimensions of the rectangular base of this test object were (20, 50, 70)mm.

We now summarise the results of the matching process for each of these three cases.

7.1. Synthetic Winsom Data for a Widget

The GDB processing sequence is illustrated in Figure(14), starting with one of the stereo image pairs produced by the Winsom modeller (14a), the results of the edge detection process (14b), the output of the of 3D data fragments by the PMF stereo algorithm (14c), and the the GDB data derived from this (14d).

The matching sequence between model and data is illustrated in Figure(15), with the *WP Graph* to be matched (15a), the *GDB data* to be matched to this (15b), the unmatched model edges (15c), the unmatched data segments (15d), the matched model edges (15e) and the matched data segments (15f). Lastly, the data and model are shown overlaid (15g).

The matching parameters values used were :

Maximum divergence angle :	0.1 Radians.
Maximum tessellation divergence angle :	0.25 Radians.
Perpendicular divergence angle (circular arcs):	0.15 Radians.
Minimum Overlap Proportion :	90%.
Maximum separation distance limit :	8mm

The matching summary produced was :

MODEL EDGES PREDICTED VISIBLE	(60)
MATCHED MODEL EDGES	(38)
PROPORTION OF PREDICTED MODEL EDGES MATCHED	(63%)
TOTAL LENGTH OF MODEL EDGES PREDICTED VISIBLE	(891mm)

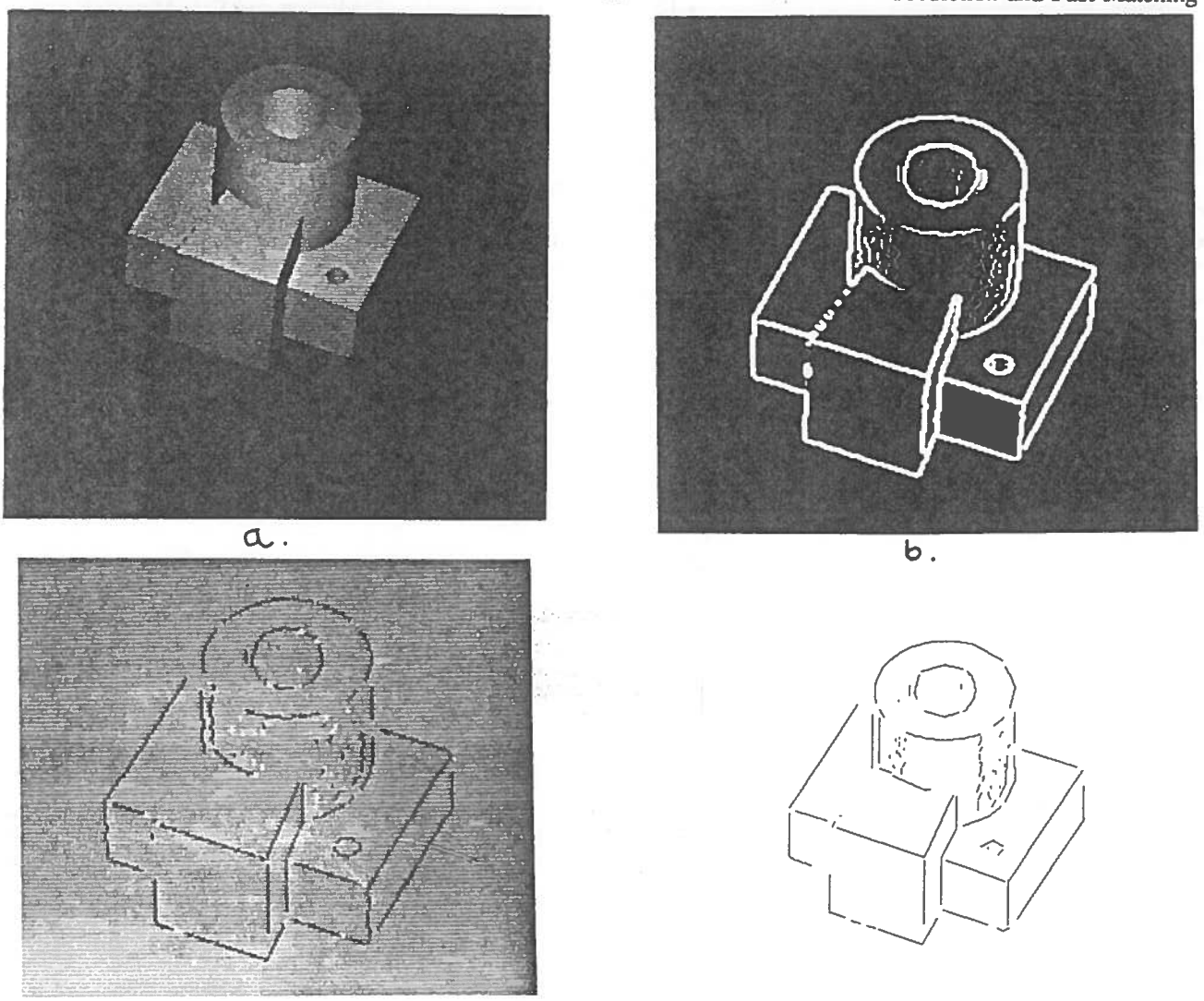


Figure 14 - Synthetic Data - Stereo Process

TOTAL LENGTH OF MATCHED MODEL EDGES	(762mm)
TOTAL LENGTH OF MATCHED DATA SEGMENTS	(620mm)
AVERAGE MODEL EDGE LENGTH MATCHED TO DATA	(81%)
PROPORTION OF PREDICTED EDGE LENGTH MATCHED TO DATA	(70%)

The matching process ran in 3 seconds on a SUN-2 and 1 second on a SUN-3.

7.2. Synthetic Data - Results Analysis

The results of this matching process were reasonably good. Of the model edges predicted to be visible 63% were matched to data, and 81% of the total length of these matched edges was accounted for by the total matched data length. The total matched data length also accounted for 70% of the total length of the model edges predicted to be visible.

This result is an indication of the upper limit of the potential resolution of the FAPM system, when using data produced by the GDB system. Some of the inaccuracies in the GDB Data can be seen when the model is over-laid on the data (Figure(15g)).

A perfect match to this synthetic data could not be achieved as not all edge features were extracted by the stereo system, and there were also errors in the localisation of some the features.

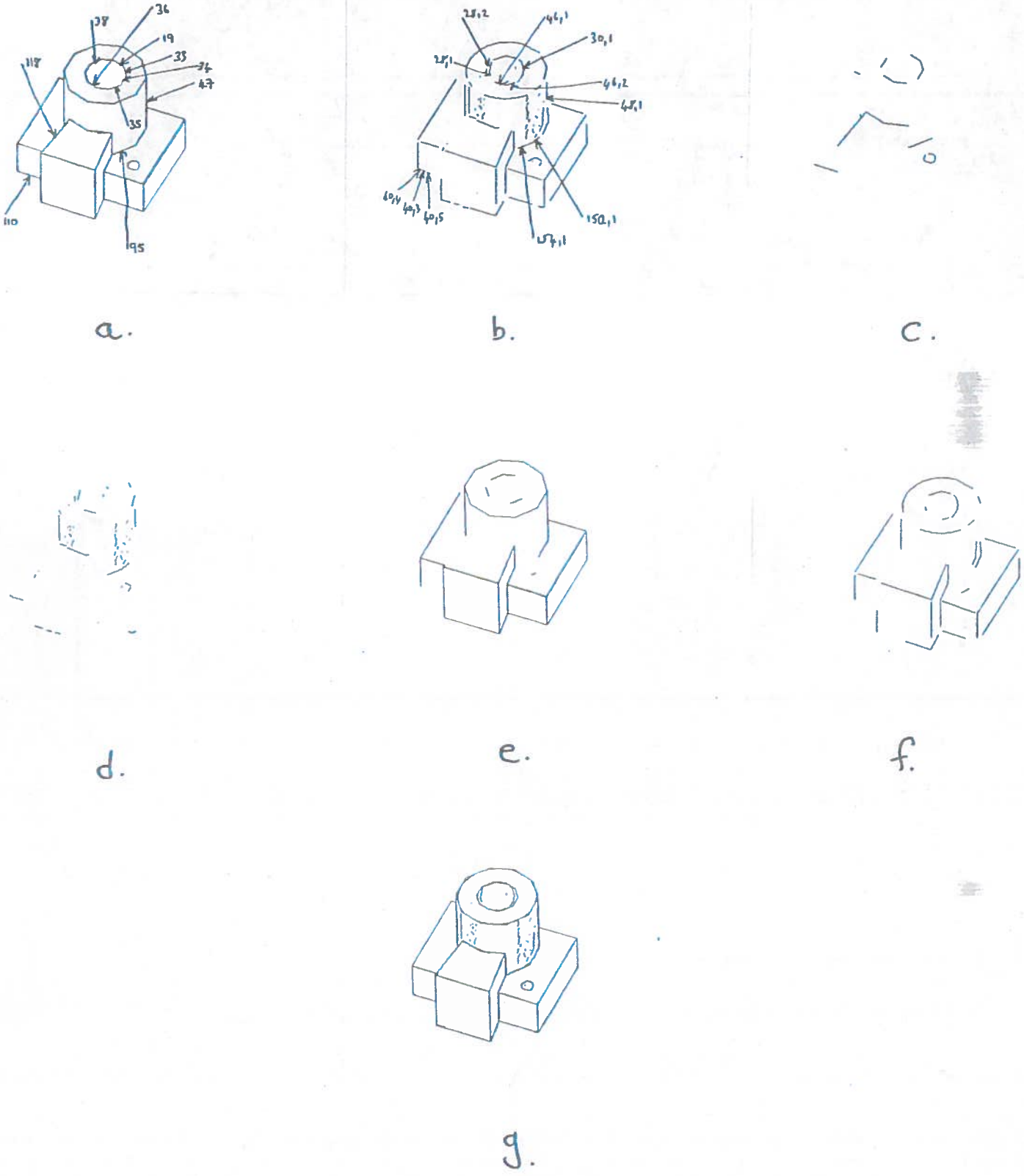


Figure 15 - Synthetic Data - Matching Process

September 22, 1987

The results of the tests for some of the failed matches are tabulated (see Table(1)), with model edges shown as *Edge(nnn)*, data segments as *(nnn,n)* and test result values given in radians, percentages and millimetres _

respectively. All the pairings tabulated passed the *quick point test*, so the results for this test are not shown. For each correct data/model pairing the result of each matching test is listed, up to the test which failed :

Table 1. Synthetic Data - Mismatch Analysis

Model Edge	Data Segment	Dev. Angle	Overlap	Separation
(Linear)				
Edge118	No data			
Edge110	40,4	0.21		
	40,5	0.72		
(Tessellation)				
Edge95	154,1	0.41		
	150,1	0.35		
Edge35	46,1	0.6		
	46,2	0.57		
Edge36	28,1	0.6		
	46,1	0.45		
Edge38	28,2	0.35		
Edge19	30,1(arc)	0.42		
Edge33	30,1(arc)	0.33		
(Extremal)				
Edge47	48,1	0.11		

There was virtually no data produced by the GDB to correspond to Edge118, so this match failed. The other model edges failed to be matched due to the inferior data produced by the GDB for the corresponding features. For Edge110, a number of small fragmented data segments were produced, which varied widely in angular direction and so failed the angular deviation test.

Most of the other main matching failures were associated with the difficulty in extracting accurate data for the curved features of the widget. Data segments 154,1 and 150,1 failed to match Edge95 as the curved concave surface discontinuity was inaccurately tessellated and placed by the GDB System and failed the angular deviation test. There were similar problems encountered in matching some of the model edges surrounding the larger cylindrical hole in the widget, and Edge35, Edge36 and Edge38 all failed the angular deviation test. This was partly due to the misalignment of tessellations and partly due to inaccurate placement of the data segments. Matching Edge19 and Edge33 to the data arc segment 30,1 failed as the model edges were not parallel to the plane of the data arc, and so these edges failed the arc coplanar test.

The problems with the larger hole were partly caused by shadows inside the hole "pulling" down the location of the data features, causing an incorrect placement and misalignment of these features. The curvature of the circular data arc was distorted in this way, although the distortion was less marked at the ends of this long segment, where a match (Edge34) did succeed. Although the tessellations between data and model can lead to the "stepping" problem causing matching failures, this is not the main cause of failure as this is mainly is due to the

inaccurate resolution of these curved features by the stereo system. This is borne out by the large angular deviations that were observed.

In the case of Edge47 failure was attributed to the difficulty in placing accurately an extremal boundary using passive stereo.

Most of the small cylindrical hole in the widget failed to be matched, as it is smaller than the effective resolution of the system, causing the GDB to produce an inaccurate and coarse tessellation of the data.

Finally, the synthetic data gave rise to a large number of small "texture" segments on the surface of the cylindrical part of the widget. These segments did not correspond to model edges and were not matched.

7.3. Real data for a Widget

The FAPM system was tested with two sets of real data and in both cases only the widget was matched, as model details were available only for this object. Although, the matcher requires the a priori location of objects, it was not available for this data and had to be estimated, so it is therefore not exact. In practice, this information would be derived from the workstation layout.

In the first scene, there was an isolated widget whereas in the second, there was a widget with other objects in a cluttered scene.

The tolerances used to match the synthetic data had to be relaxed slightly for the real data to enable a reasonable match to succeed. This relaxation of matching tolerance was kept to a minimum to avoid any degradation of matching resolution, which would have otherwise produced false matches and a greater uncertainty in the position of the data.

The tolerance values used for the real data were :

Maximum divergence angle :	0.25 Radians.
Maximum tessellation divergence angle :	0.30 Radians.
Perpendicular divergence angle (circular arcs):	0.20 Radians.
Minimum Overlap Proportion :	90%.
Maximum separation distance limit :	18mm

7.4. Isolated Widget - Matching Summary

The GDB processing sequence is illustrated in Figure(16), starting with one of the stereo image pairs of the scene (16a), the results of the edge detection process (16b), the output of the of 3D data fragments by the PMF stereo algorithm (16c), and the the GDB data derived from this (16d).

The matching sequence between model and data is illustrated in Figure(17), with the *WP Graph* to be matched (17a), the *GDB data* to be matched to this (17b), the unmatched model edges (17c), the unmatched data segments (17d), the matched model edges (17e) and the matched data segments (17f). Lastly, the data and model are shown overlaid (17g), and the data segments located outside the model circumsphere (17h).

The matching summary produced was :

MODEL EDGES PREDICTED VISIBLE	(55)
MATCHED MODEL EDGES	(21)
PROPORTION OF PREDICTED MODEL EDGES MATCHED	(38%)
TOTAL LENGTH OF MODEL EDGES PREDICTED VISIBLE	(794mm)
TOTAL LENGTH OF MATCHED MODEL EDGES	(422mm)
TOTAL LENGTH OF MATCHED DATA SEGMENTS	(297mm)
AVERAGE MODEL EDGE LENGTH MATCHED TO DATA	(70%)
PROPORTION OF PREDICTED EDGE LENGTH MATCHED TO DATA	(37%)

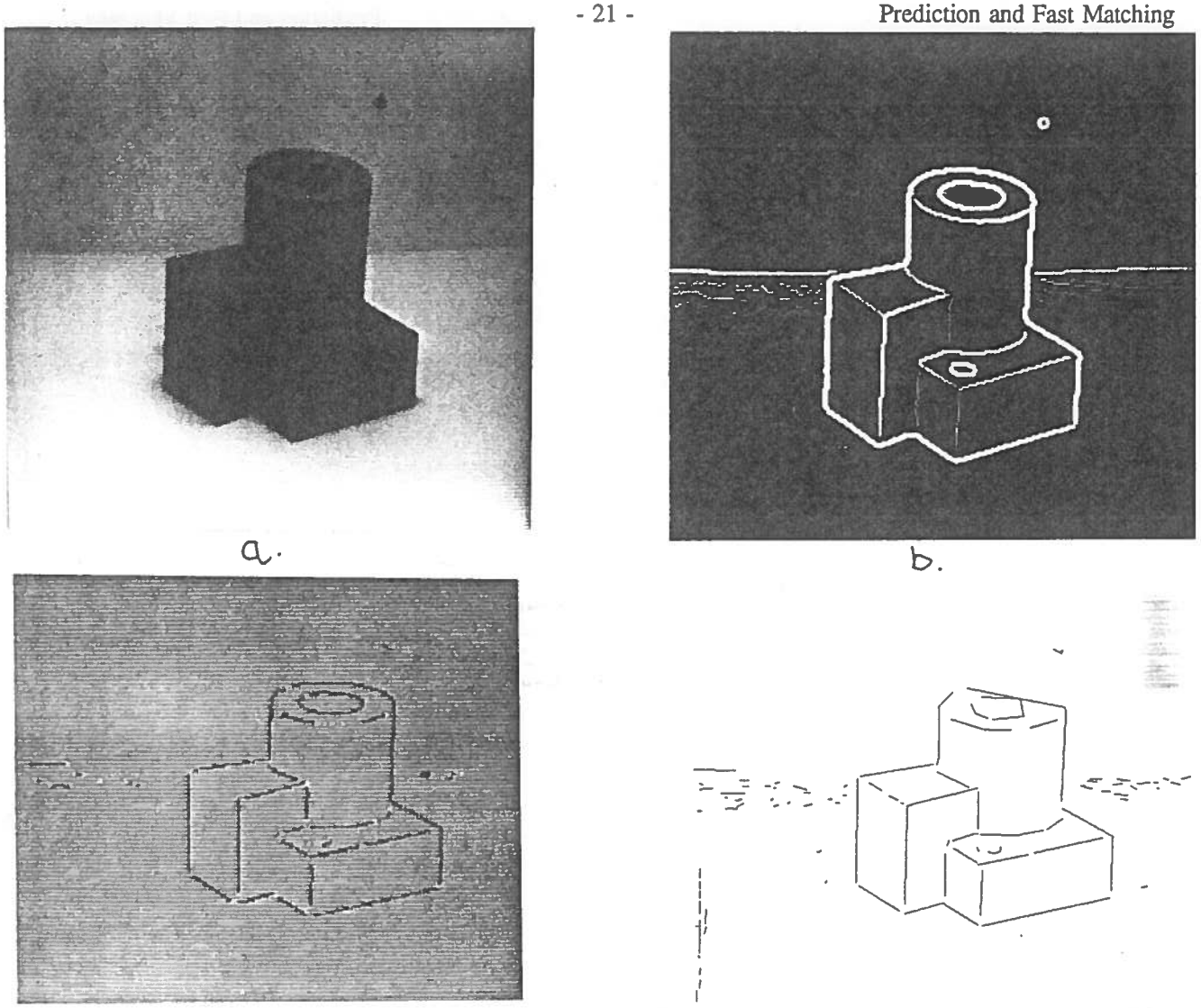


Figure 16 - Isolated Widget - Stereo Process

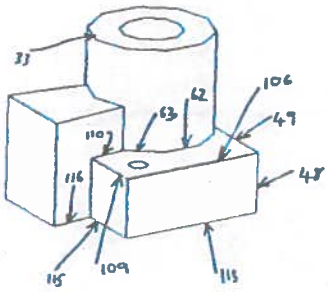
The matching process ran in 2 seconds on a SUN-2 and 0.5 seconds on a SUN-3.

7.5. Isolated Widget - Results Analysis

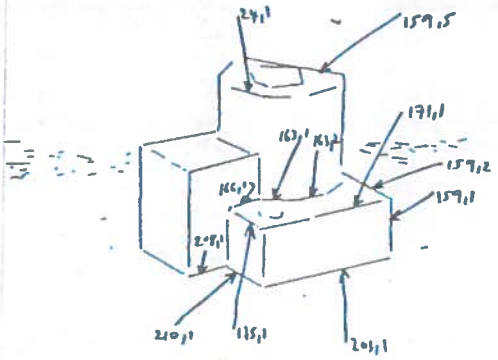
The results of this matching process were not as good as for the synthetic data. Of the model edges predicted to be visible 38% were matched to data, and 70% of the total length of these matched edges was accounted for by the total matched data length. The total matched data length also accounted for 37% of the total length of the model edges predicted to be visible.

Features not matched included some curved features which were too small to be resolved by the GDB, whilst others were segmented into tessellations which were inaccurate. A section of the smaller cylindrical hole was matched however, where the GDB succeeded in producing a circular arc for part of it. Several linear segments also failed to be matched, partly due to inaccurate data and partly due to the data/model calibration which was estimated and not exact.

The results of the tests for some of the failed matches are tabulated and are annotated as for the synthetic data above (see Table(2)). For some of the data/model pairings, matching tolerances were relaxed for some of the tests so as to obtain the results of tests subsequent to a test that failed. This enabled a more detailed analysis of the matching process.



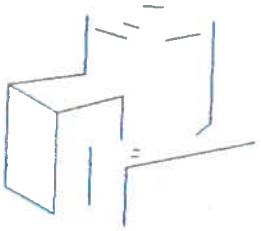
a.



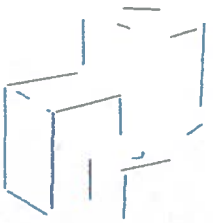
b.

c.

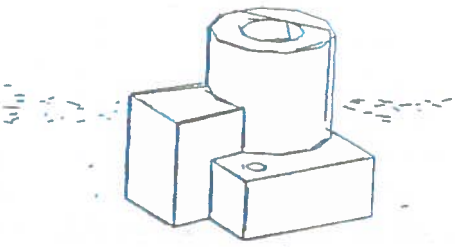
d.



e.



f.



g.

h.

Figure 17 - Isolated Widget - Matching Process

Table 2. Isolated Widget - Mismatch Analysis

Model Edge	Data Segment	Dev. Angle	Overlap	Separation
(Linear)				
Edge48	159,1	0.26(failed)	95%	15.7mm
Edge49	159,2	0.48		
Edge106	171,1	0.02	100%	21.1mm
Edge109	175,1	0.7		
Edge110	166,1	0.35(failed)	100%	6.3mm
Edge113	203,1	0.13	100%	31.5mm
Edge115	210,1	0.0	83%	
Edge116	208,1	0.34(failed)	100%	6.5mm
(Tessellation)				
Edge33	24,1	0.4		
Edge62	163,1	0.7		
Edge63	163,2	0.74		

Several linear features failed to be matched, data segments 203,1 and 171,1 are both misplaced in depth and fail the separation test. This is due to the difficulty in placing accurately in depth linear features which lie parallel to the epipolar direction within the stereo system, (the "washing line" effect).

Matching fails with Edge109 and Edge49 because the direction of the data segments (175,1 and 159,2) are misaligned. For segment 159,2, this is due to the T junction "pulling" up the direction of the segment where it is occluded by the cylindrical projection of the widget. For segment 175,1, the segment is misaligned near to the convex corner of the base. A similar problem causes data segment 210,1 to fail the overlap test.

Several of the curved features failed to be matched. In some cases the GDB produced no data corresponding to model edges. Data segment 159,5 was produced as an undefined space curve which the FAPM system could only interpret incorrectly as a linear segment, which could not be paired with any model edge. A number of other inaccurate tessellations were produced by the GDB, some of which failed to be matched. Examples were Edge33, Edge62 and Edge63, where the deviation angles of the data segments from the corresponding model edges were substantial.

7.6. Widget in a Cluttered Scene - Matching Summary

The GDB processing sequence is illustrated in Figure(18), starting with one of the stereo image pairs of the scene (18a), the results of the edge detection process (18b), the output of the of 3D data fragments by the PMF stereo algorithm (18c), and the the GDB data derived from this (18d).

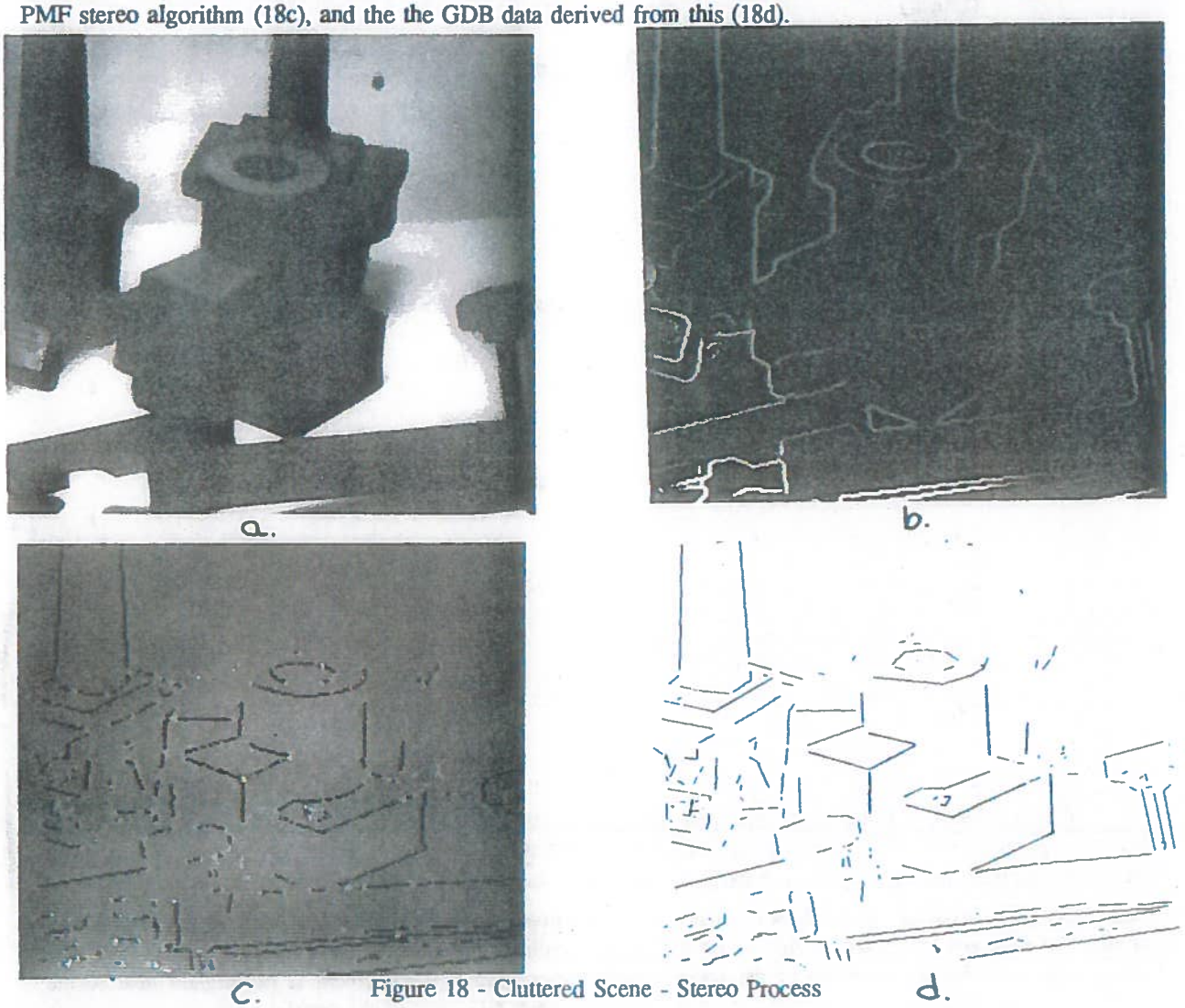


Figure 18 - Cluttered Scene - Stereo Process

The matching sequence between model and data is illustrated in Figure(19), with the *WP Graph* to be matched (19a), the *GDB data* to be matched to this (19b), the unmatched model edges (19c), the unmatched data segments (19d), the matched model edges (19e) and the matched data segments (19f). Lastly, the data and model are shown overlaid (19g), and the data segments located outside the model circumsphere (19h).

The matching summary produced was :

MODEL EDGES PREDICTED VISIBLE	(59)
MATCHED MODEL EDGES	(19)
PROPORTION OF PREDICTED MODEL EDGES MATCHED	(32%)
TOTAL LENGTH OF MODEL EDGES PREDICTED VISIBLE	(830mm)
TOTAL LENGTH OF MATCHED MODEL EDGES	(464mm)
TOTAL LENGTH OF MATCHED DATA SEGMENTS	(320mm)
AVERAGE MODEL EDGE LENGTH MATCHED TO DATA	(69%)
PROPORTION OF PREDICTED EDGE LENGTH MATCHED TO DATA	(39%)

The matching process ran in 3 seconds on a SUN-2 and 0.75 seconds on a SUN-3.

7.7. Widget in a Cluttered Scene - Results Analysis

The circumsphere test was useful (Figure(19h)) for this scene, eliminating much of the data from the matching process and enabling a faster run time.

Of the model edges predicted to be visible 32% were matched to data, and 69% of the total length of these matched edges was accounted for by the total matched data length. The total matched data length also accounted for 39% of the total length of the model edges predicted to be visible.

No data was produced by the GDB for a large part of the top of the cylindrical projection of the widget as the stereo system failed to resolve these features (although edge detection did extract them). The same problem occurred for the left front corner of the base and for the vertical edge on the right front corner. There was therefore no data for these features which could be matched to the equivalent model edges. As for the previous scenes, several curved features were not matched.

Unlike earlier scenes some parts of the widget were occluded by other objects. This explains the lower figure for the proportion of model edges predicted visible and matched (32%). In practice a model of the entire scene should be constructed which would reduce the number of model edges and the total edge length predicted visible for the widget. This would then increase the figures for the proportions matched.

However, the overall matching performance (69%, 39%) was somewhat better than in the previous case, and nearly all linear data segments were matched successfully. This was due to a more accurate position estimate and to the different presentation angle of the widget, which avoided the problem of horizontal edges being misplaced in depth (see Table(3)).

Apart from problems with data loss, only one linear feature failed to match completely. Several small linear data segments were produced by the GDB corresponding to Edge39, and although the longer segment (139,1) was matched the shorter segments failed due to large misalignments in direction. This problem is similar to that encountered with the synthetic Winsom data (Edge110).

One extremal feature failed match (to Edge82), due to a misalignment in direction. This was probably due to the partial occlusion of the this feature causing the data segment (68,1) to be "pulled" forwards at the T junction.

The remaining matching failures were all curved features, bounding the larger and smaller cylindrical holes and the large cylindrical projection. The data analysis above confirms that the reasons for the failures were similar to the those for the earlier scenes.

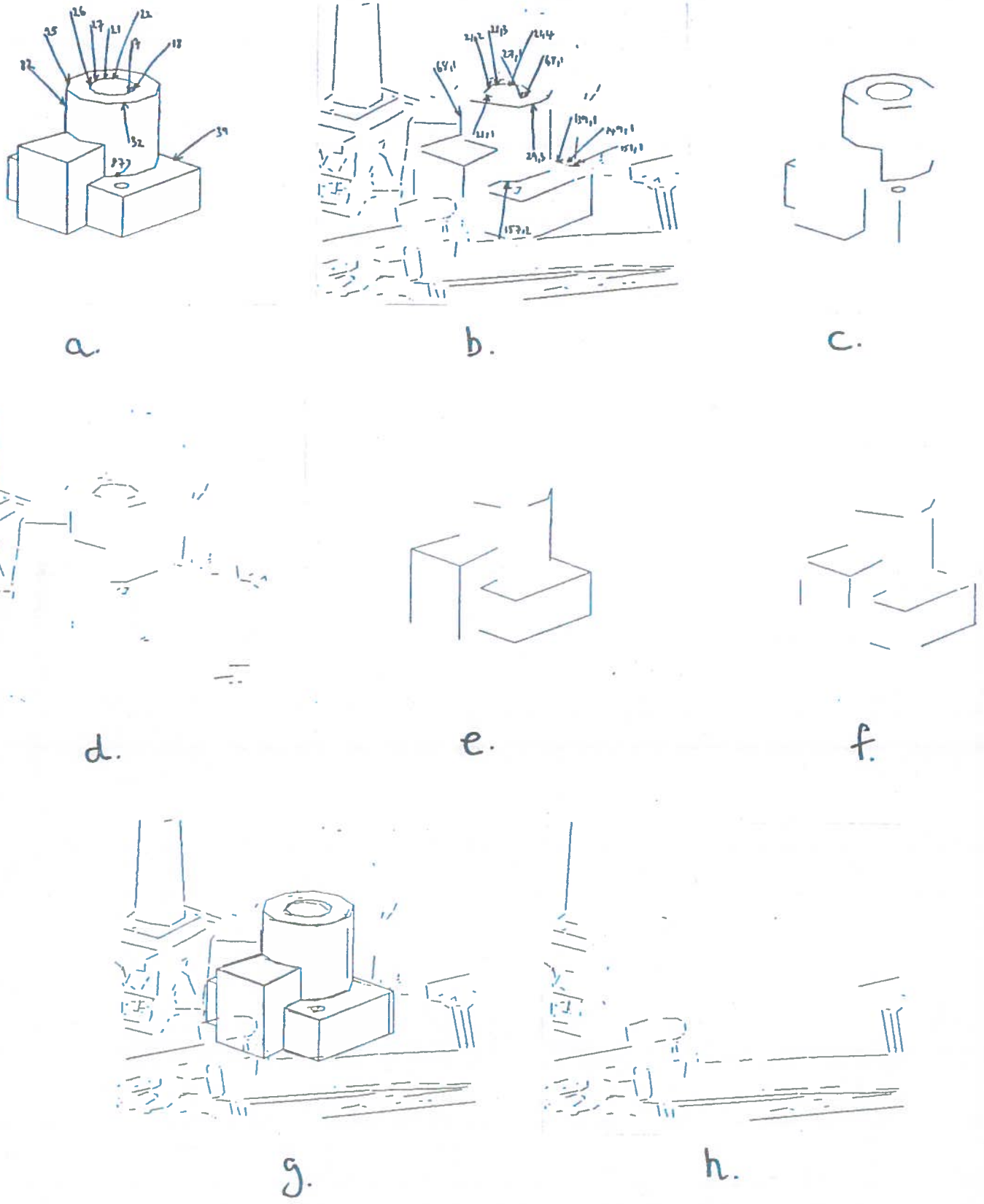


Figure 19 - Cluttered Scene - Matching Process

Table 3. Cluttered Scene - Mismatch Analysis

Model Edge	Data Segment	Dev. Angle	Overlap	Separation
(Linear)				
Edge39	139,1 149,1 151,1	0.14 0.81(failed) 1.30(failed)	100%	3mm(matched)
(Extremal)				
Edge82	68,1	0.30		
(Tessellation)				
Edge17	48,1	0.52		
Edge18	27,1	0.48		
Edge21	21,3 21,4	0.6 0.49		
Edge22	21,4	1.2		
Edge25	21,1	1.4		
Edge26	21,3 21,2	0.7 0.25	0%(failed)	
Edge27	21,3	0.25	19%(failed)	
Edge32	29,3	0.42		
Edge87	157,2	0.68		

8. Conclusion

The purpose of the *WPFM system* we have described is to identify areas of local discrepancy between the *GDB data* and *WP graph* as quickly as possible so that attention may be centred on them by a full vision system [9]. The *WP system* provides the model data which is used by the *FAPM system* to match and subtract expected data from a scene. The stereo data used by FAPM is provided by a passive stereo system, the *GDB System.* At the end of a FAPM run, an updated *GDB File* is produced, containing information about which features in a scene have been matched, and a matching "goodness" summary which indicates how tight a match was produced between model and data. Thus unmatched data could indicate a local area of discrepancy, which combined with information on any expected model features not matched, could indicate to the full vision system that objects have been occluded, or are missing. So this could then be a cue to the full vision system to expect to "see" the missing object somewhere else, assuming that other information indicates that the object has not been removed or obscured. This extra data unmatched in the GDB could be an indication of new objects in a scene, or of old objects moved to unexpected locations. However, some unmatched data will be due to noise and other distortions in the data. The matching goodness summary, combined with the use of the full topological and geometric information in the *WP Graph,* could be used by the full vision system to disambiguate noise and distortion from

real differences.

The system has been implemented and run on both synthetic and real test data, which it can process rapidly to produce a reasonable result. Matching performance with real data will probably be improved as the resolution of the GDB system is increased by the use of improved calibration techniques and better camera systems.

Future work could explore how well the system works in a greater variety of real situations. This could include an investigation of methods for tuning the "best" matching parameter values, and the establishment of better criteria to determine matching goodness. A second level matching could also be implemented, to match more data at a lower matching tolerance to model surfaces and edges, to eliminate noise and other features from the GDB, once a "good" match has been established. Investigation could also be made into the automatic recalibration of model to data by estimating appropriate reference frame transformations [12 & 18]. The possibility of the *FAPM system* using more of the topological and relational information in the *WP Graph* in order to improve the matching capability could also be explored.

References

[1] AIVRU (1986). GDB Release (1.0) User Documentation. AIVRU report 014, University of Sheffield.

[2] Aylett. J.C. (1986). A Boundary Representation for Machine Vision, DAI Working Paper, University Of Edinburgh. (In preparation).

[3] Aylett. J.C. (1986). Fast a Priori Matcher - User Guide, DAI Software Report, University Of Edinburgh.

[4] Baer, A. et al. (1979). Geometric Modelling: A Survey. Computer Aided Design, Volume 11 Number 5.

[5] Blake, A. and Mayhew, J.E.W. (1985). Alvey 2-1/2D sketch project: Proposed structure for a development system. AIVRU report, University of Sheffield.

[6] Boyse, J. W. (1982). GMSolid: Interactive Modeling for Design and Analysis of Solids. IEEE Computer Graphics and Automation.

[7] Cameron, S.A. (1984). Modelling Solids in Motion, PhD Thesis, University Of Edinburgh.

[8] Cameron, S.A., Aylett, J.C. (1986). Robmod User Guide, DAI Software Paper, University Of Edinburgh.

[9] Fisher, R.B. (1986). From Surfaces to Objects, PhD Thesis, University of Edinburgh.

[10] Gray, M. (1985). Proposal for Representing 3-D Objects. IBM UK Scientific Centre.

[11] Grimson, W. E. L. and Lozano-Perez T. Model-Based Recognition and Localisation from Sparse Range or Tactile Data. The International Journal of Robotics Research, Vol. 3, No. 3, Fall 1984.

[12] Pollard, S. B. et al. (1986). Matching Geometrical Descriptions in Three Space. AIVRU Working Paper 22, University of Sheffield.

[13] Popplestone, R.J., Ambler, A.P. and Bellos, I.M.. An Interpreter for a Language Describing Assemblies. Artificial Intelligence, 14, 79, 1980.

[14] Pridmore, T.P., Bowen, J.B., Mayhew, J.E.W. (1986). Geometrical description of the connect graph (version 2). The geometrical base description : a specification. AIVRU report 012, University of Sheffield.

[15] Quarendon, P. (1984). WINSOM User's Guide (UKSC 123). IBM UK Scientific Centre.

[16] Requicha A. A. G. (1980). Representations for Rigid Solids: Theory, Methods and Systems. Computing Surveys, Volume 12, No 4.

[17] Requicha A. A. G. and Voelcker, H. B. (1982). Solid Modeling: A Historical Summary and Contemporary Assessment. Computing Surveys, Volume 12, No 4.

[18] Watson, R . (1985). An Edge-Based 3D Geometrical Model Matching System. MSc Dissertation, Department of Artificial Intelligence, University of Edinburgh.

[19] Weiler, K. (1985). Edge Based Data Structures for Solid Modeling in Curved in Curved Surface Environments. IEEE Computer Graphics and Automation.

[20] Wilson, P. R. et al (1985). Interfaces for Data Transfer Between Solid Modeling Systems. IEEE Computer Graphics and Automation.

Appendix 1. Boundary File Description

Here is the robmod program used to create the output files shown in appendices 1 & 2.

```
1 set Frame "left"
10 cbody a (cyl 8 2 / cyl 8 1) roty 10 rotx -30
20 cbody b cub 4 4 4
30 cbody scene shapeof((a @ b to 2 3 10) rotx -10 roty -40)
35 wire scene
50 wbfile scene "CYLCUB.BF"
60 vpred scene "CYLCUB.P"
```

Picture produced by 'wire scene' command above:

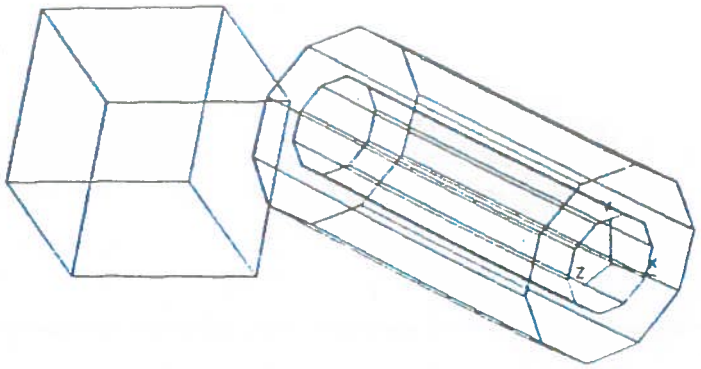


Figure 20

Boundary file annotated skeleton shown below:

ROBMOD - Scene Boundary Description(Version 2)

original name of output file.
(SCENE(CUBCYL.BF)

Identifies which reference frame was used when creating this boundary file:
(REFERENCE_FRAME(Left))

Circumsphere dimensions:
(BOUND_SPHERE

(RADIUS(10.491058350) CENTRE(-2.523521423 3.856894732 6.262558460)))

(FEATURES_TOTAL(SURFACES(28) BOUNDARY_LOOPS(30) EDGES(72) VERTICES(48)))

(SCENE_TOPOLOGY All the topology..
Original CSG tree used to make this model:
(CSG_TREE

(Csgnode1 ROTATE_Y(Csgnode2) ANGLE(-40.000000))
(Csgnode2 ROTATE_X(Csgnode3) ANGLE(-10.000000))
(Csgnode3 UNION(Csgnode4 Csgnode6))
(Csgnode4 TRANSLATE(Csgnode5) VECTOR(x 2.000000 y 3.000000 z 10.000000))
(Csgnode5 CSGPRIM(Cube1) DIMENSIONS(x 4.000000 y 4.000000 z 4.000000)
CENTRE(x 2.000000 y 2.000000 z 2.000000))

(Csgnode6 ROTATE_X(Csgnode7) ANGLE(-30.000000))
(Csgnode7 ROTATE_Y(Csgnode8) ANGLE(10.000000))
(Csgnode8 DIFFERENCE(Csgnode9 Csgnode10))
(Csgnode9 CSGPRIM(Cylinder1) AXISLENGTH(8.000000) RADIUS(2.000000)
AXISVECTOR(x 0.000000 y 0.000000 z 8.000000))

(Csgnode10 CSGPRIM(Cylinder2) AXISLENGTH(8.000000) RADIUS(1.000000)
AXISVECTOR(x 0.000000 y 0.000000 z 8.000000))

END_CSG_TREE)

List of separate assemblies:
(ASSEMBLY_LIST

(Assembly1
(CSG_PRIM_LIST

CSG primative components making up this assembly..

(Cube1(CONVEX) Note, these can be convex or concave..
(SURFACE_LIST

Remaining surfaces making up this CSG component..
(Surface1(PLANE) BOUNDARY_LOOPS(1) (EXTERIOR_BOUNDARY(Loop1)))
Each surface has an associated boundary loop..

(Surface2(PLANE) BOUNDARY_LOOPS(1) (EXTERIOR_BOUNDARY(Loop2)))
(Surface3(PLANE) BOUNDARY_LOOPS(1) (EXTERIOR_BOUNDARY(Loop3)))
(Surface4(PLANE) BOUNDARY_LOOPS(1) (EXTERIOR_BOUNDARY(Loop4)))
(Surface5(PLANE) BOUNDARY_LOOPS(1) (EXTERIOR_BOUNDARY(Loop5)))
(Surface6(PLANE) BOUNDARY_LOOPS(1) (EXTERIOR_BOUNDARY(Loop6)))
END_SURFACE_LIST))

END_CSG_PRIM_LIST)
End_Assembly1)

(Assembly2
(CSG_PRIM_LIST

(Cylinder1(CONVEX)
(SURFACE_LIST
(Surface7(PLANE) BOUNDARY_LOOPS(2) (EXTERIOR_BOUNDARY(Loop7)
Interior boundary loops are interior to the surface..
(INTERIOR_BOUNDARY(Loop8))))

etc..

END_SURFACE_LIST))

(Cylinder2(CONCAVE) concave primitive, in this case a hole.
(SURFACE_LIST
Facet surfaces are psuedo surfaces generated by the polyhedral
approximation of curved surfaces..
(Surface9(FACET) BOUNDARY_LOOPS(1) (EXTERIOR_BOUNDARY(Loop10)))

etc..

END_SURFACE_LIST))

END_CSG_PRIM_LIST)
End_Assembly2)

List of boundary loops..
(LOOP_LIST

The ordering of each edge in the loop is shown here,
and the type of loop and the ordering (sign) of vertices along the edge.

A polygon loop contains only true linear edges.
(Loop1(POLYGON) LOOP_EDGES(-> +Edge11 -> -Edge8 -> -Edge5 -> +Edge1))
(Loop2(POLYGON) LOOP_EDGES(-> +Edge4 -> +Edge2 -> -Edge3 -> -Edge1))
(Loop3(POLYGON) LOOP_EDGES(-> -Edge12 -> +Edge9 -> +Edge7 -> -Edge2))
(Loop4(POLYGON) LOOP_EDGES(-> +Edge12 -> +Edge3 -> -Edge11 -> -Edge10))
(Loop5(POLYGON) LOOP_EDGES(-> -Edge7 -> -Edge4 -> +Edge5 -> +Edge6))
(Loop6(POLYGON) LOOP_EDGES(-> +Edge10 -> +Edge8 -> -Edge6 -> -Edge9))

A curved loop is a polyline approximation to a space curve..
(Loop7(CURVED) LOOP_EDGES(-> +Edge52 -> +Edge13 -> +Edge15 -> -Edge16
-> -Edge26 -> -Edge27 -> -Edge28 -> -Edge49
-> +Edge50 -> +Edge51))
(Loop8(CURVED) LOOP_EDGES(-> -Edge25 -> -Edge24 -> -Edge23 -> +Edge22
-> +Edge21 -> +Edge20 -> +Edge18 -> +Edge17
-> -Edge14 -> -Edge19))

A facet loop, is the loop of edges around a facet surface..
(Loop9(FACET) LOOP_EDGES(-> +Edge59 -> -Edge13 -> -Edge44 -> +Edge47))

There may also be IRREGULAR loops, which are a mixture of CURVED
and LINEAR edges.

etc..

END_LOOP_LIST)

List of edges, indicating type, vertices and the two loops the
edge is in. Also included is the length and unit vector of the edge.

(EDGE_LIST

(Edge1(LINEAR) VERTS(Vert1 Vert2) LOOPS(Loop1 Loop2)
LENGTH(4.000000) UNIT_VECTOR(x -0.633022 y 0.173648 z 0.754407))

(Edge2(LINEAR) VERTS(Vert3 Vert4) LOOPS(Loop2 Loop3)
LENGTH(4.000000) UNIT_VECTOR(x -0.633022 y 0.173648 z 0.754407))

etc..

A curved edge is an edge that is part of a polyline curve
boundary loop..
(Edge13(CURVED) VERTS(Vert9 Vert10) LOOPS(Loop7 Loop9)
LENGTH(1.299671) UNIT_VECTOR(x -0.827954 y -0.554135 z 0.086177))

etc..

A facet edge is an edge that is part of a facet boundary loop..
(Edge29(FACET) VERTS(Vert12 Vert26) LOOPS(Loop10 Loop13)
LENGTH(8.000000) UNIT_VECTOR(x -0.351901 y 0.633022 z 0.689528))

etc..

END_EDGE_LIST)

END_SCENE_TOPOLOGY)

(SCENE_GEOMETRY(OBJECT_CENTRED) All the scene geometry..

(SURFACE_EQN_LIST
List of surface equations..
(Surface1 EQN(ax 0.111618884 by 0.984807789 cz -0.133022219 d -7.000000477))
(Surface2 EQN(ax 0.766044497 by 0.000000000 cz 0.642787576 d -6.000000000))
(Surface3 EQN(ax 0.111618884 by 0.984807789 cz -0.133022219 d -3.000000715))

etc..

END_SURFACE_EQN_LIST)

(VERTEX_LIST)

List of vertices, an edge at the vertex, and coordinates..
(Vert1 EDGE(Edge3) COORDS(x -3.484710932 y 9.324726105 z 13.487257004))
(Vert2 EDGE(Edge4) COORDS(x -0.952621460 y 8.630134583 z 10.469631195))
(Vert3 EDGE(Edge2) COORDS(x -3.931186199 y 5.385496140 z 14.019347191))
(Vert4 EDGE(Edge2) COORDS(x -1.399098158 y 4.690903187 z 11.001719475))

etc..

END_VERTEX_LIST)

END_SCENE_GEOMETRY)

END_SCENE)

Appendix 2. Workspace Prediction Graph Description

The visibility prediction (WP Graph) file is identical to the boundary file format in appendix(1), with the following differences:

ROBMOD - Scene Boundary Visibility Description(Version 1)

(SCENE(CUBCYL.P) as appendix(1)

(REFERENCE_FRAME(Left)) as appendix(1)

as appendix(1)
(BOUND_SPHERE(RADIUS(10.491058350) CENTRE(-2.523521423 3.856894732 6.262558460)))

(FEATURES_TOTAL(SURFACES(28) BOUNDARY_LOOPS(30) EDGES(72) VERTICES(48)))

Visibility information:
(FEATURES_VISIBLE(EDGES(34) PART_EDGES(3) VERTICES(33) VIRT_VERTICES(3)))

Viewing parameters:

Note all coordinates are with reference to the original robmod object centered coordinate frame. Any viewing transformations are added to these.

(VIEW_PARAMETERS

(VIEW_ANGLES(AZIMUTH(30.000000) INCLINATION(30.000000) ORIENTATION(0.000000)))
(VIEWED_POINT(x 0.000000 y 0.000000 z 0.000000))
(FRAME_SIZE(x 15.305500 y 15.305500))

END_VIEW_PARAMETERS)

(SCENE_TOPOLOGY

(CSG_TREE

As appendix(1).

END_CSG_TREE)

(ASSEMBLY_LIST

(Assembly1
(CSG_PRIM_LIST

etc..
As appendix(1)

END_LOOP_LIST)

Edge lists have visibility information attached to them:

(EDGE_LIST(VISIBLE)
Fully visible edges (excludes facet edges, which are never seen,
except as extremal boundaries).

(Edge1(LINEAR) VERTS(Vert1 Vert2) LOOPS(Loop1 Loop2)
LENGTH(4.000000) UNIT_VECTOR(x -0.633022 y 0.173648 z 0.754407))

etc..

(Edge60(EXTREMAL) VERTS(Vert23 Vert43) LOOPS(Loop23 Loop22)
LENGTH(8.000000) UNIT_VECTOR(x -0.351901 y 0.633022 z 0.689528))

etc..

END_EDGE_LIST)

(EDGE_LIST(PART_VISIBLE)
Partially visible edges, including virtual vertices:

(Edge49(CURVED) VERTS(Vert25 Virt_Vert2) LOOPS(Loop27 Loop7)
LENGTH(1.299687) UNIT_VECTOR(x -0.413176 y -0.766045 z 0.492404))

(Edge52(CURVED) VERTS(Vert9 Virt_Vert3) LOOPS(Loop7 Loop30)
LENGTH(1.299685) UNIT_VECTOR(x -0.926482 y -0.130565 z -0.352965))

(Edge66(EXTREMAL) VERTS(Vert48 Virt_Vert1) LOOPS(Loop30 Loop29)
LENGTH(7.999999) UNIT_VECTOR(x -0.351901 y 0.633022 z 0.689528))

END_EDGE_LIST)

(EDGE_LIST(NON_VISIBLE)

(Edge6(LINEAR) VERTS(Vert6 Vert5) LOOPS(Loop5 Loop6)
LENGTH(4.000000) UNIT_VECTOR(x -0.111619 y -0.984808 z 0.133022))

etc..

END_EDGE_LIST)

END_SCENE_TOPOLOGY)

(SCENE_GEOMETRY(OBJECT_CENTRED)

(SURFACE_EQN_LIST

As in appendix(1)

END_SURFACE_EQN_LIST)

(VERTEX_LIST(VISIBLE)

Visible vertices:

(Vert1 EDGE(Edge3) COORDS(x -3.484710932 y 9.324726105 z 13.487257004))
(Vert2 EDGE(Edge4) COORDS(x -0.952621460 y 8.630134583 z 10.469631195))
(Vert3 EDGE(Edge2) COORDS(x -3.931186199 y 5.385496140 z 14.019347191))

etc..

END_VERTEX_LIST)

(VERTEX_LIST(VIRTUAL)
Virtual vertices..

(Virt_Vert1 EDGE(Edge66) COORDS(x -3.675403118 y 3.437629700 z 6.534708500))
(Virt_Vert2 EDGE(Edge49) COORDS(x -4.627293587 y 5.042198658 z 4.611604691))
(Virt_Vert3 EDGE(Edge52) COORDS(x -3.660015821 y 3.456634521 z 6.560884476))

END_VERTEX_LIST)

(VERTEX_LIST(NON_VISIBLE)

Non visible vertices..
(Vert6 EDGE(Edge6) COORDS(x -4.463274479 y 4.690903187 z 8.430568695))

etc..

END_VERTEX_LIST)

END_SCENE_GEOMETRY)

END_SCENE)