POLYHEDRAL OBJECT RECOGNITION WITH SPARSE DATA IN SIMD PROCESSING MODE

by Derrick Holder and Hilary Buxton*

Queen Mary College, London.

The method of Grimson, Lozano Pérez and others, for the recognition of polyhedral objects with sparse data, has been developed and implemented on a distributed array processor, the AMT DAP 500, which operates in SIMD mode. Measurements involving the location vectors and the surface normals at m data points, considered in pairs, are compared with the corresponding maximum and minimum values associated with n×n pairs of object model faces, in a process that exploits n×n parallelism. The overall processing time is essentially proportional to m×(m-1)/2, to explore the interpretation tree to its full depth.

This paper discusses the nature of the comparisons between object models and data, together with the need to make these comparisons in a particular sequence, and results of test runs with a variety of object models and different geometric constraints are presented herein. Comparison is made with the corresponding sequential process, and with the more costly method of Flynn and Harris, in which n^m processing elements are required to achieve, at best, a processing time of the same order of magnitude.

INTRODUCTION

In many applications the key task for a robot's vision system, according to Murray and Cook[1], is "to supply the control unit with a quantitative and symbolic description of its surroundings: in other words, tell the robot what is where in the scene being viewed."

Grimson and Lozano-Pérez[2] describe the problem, with regard to polyhedron objects as follows:-

"To identify an object from among a set of known objects and to locate it relative to the sensor. The object sensed is assumed to be a single, possibly non-convex, polyhedral (for which we have an accurate geometric model). The object may have up to six degrees of freedom relative to the sensor (three translational and three rotational). The sensor, which could be tactile or range, is assumed to be capable of providing three-dimensional information about the position and local surface orientation of a small set of points on the object".

With geometric object model descriptions stored in a database, the robot is confronted with geometric (shape) data from low level vision sources such as binocular and photometric stereo, structure from motion, shape from contour, shading and texture, structured light and active range finder measurements.

We start with the premise that, on the basis of sensor readings, the positions of some points on the surface of the object in question can be determined to lie within some small volume in space relative to the sensor, and that the direction of surface normals at these points can be recovered to within a cone of uncertainty. Our goal is then to use this information to determine the location and orientation of an object that is consistent with the sensed data.

The general approach to the problem is first to generate feasible interpretations by means of simple, generally pairwise, geometric comparisons between data and object models, and then to test the feasible interpretations, in detail, for compatibility with the surface equations of a particular object model. An interpretation is valid if, and only if, it is possible to solve for a rotation and translation that would place each data point sufficiently accurately on a surface of a particular object. If no match is found, i.e. there are no consistent position and orientation to be found, a particular object is excluded from the investigation. The shortlisting of feasible interpretations is of the utmost importance because, given m data points to be matched against a polyhedral model with n faces, the number of possible interpretations is n to the power m, and it is clearly not a viable proposition to do the detailed model calculations for every one.

A number of sequential algorithms, with regard to polyhedral object recognition, have been implemented by different authors, but they are not generally fast enough to offer a practical solution to the problem. The aim of the present authors has been to implement the methods of Grimson and Lozano-Pérez, and others, with efficient parallel algorithms. The implementation is on a distributed array processor, the AMT DAP 500, which operates in SIMD (single instruction - multiple data) lockstep mode.

GENERAL CONSIDERATIONS.

The generation of feasible alternatives proceeds as follows:-

> For each data point, or pair of data points, we consider alternative assignments to the faces of a particular object model. The alternatives may be recorded in an interpretation tree, with each node representing a single assignment and the alternative paths representing the sequences of assignments

*The first author in on leave of absence from City of London Polytechnic.

embodied in alternative interpretations of the data set.

The pairwise geometric match between the data and a given object model is then investigated, and the interpretation tree is pruned accordingly. A geometric match is said to be achieved when the values of certain primitives, such as the distance between two points and the angle between two surface normals, associated with a given pair of data points are compatible with the geometric constraint, or ranges of values, associated with a particular pair of object model faces.

The trial assignment to object model faces of the pair of data points associated with a given geometric match is then checked against the assignments made at higher levels in the interpretation tree. The need to do this arises because a data point cannot sensibly be assigned to more than one surface, if we exclude measurements at the intersections between surfaces with ambiguity in surface normals.

The term "consistent match" is coined herein, to describe the situation where a pairwise geometric match is consistent with the assignments made at higher levels in the interpretation tree, and this provides a basis for further substantial pruning of the interpretation tree.

With a sequential processor, it is absolutely essential that the interpretation tree is pruned, as soon as it is established that the interpretation of a subset of the data is found to violate the model constraints. For example, with eight data points to be matched against a dodecahedron there are 429,981,696 alternative interpretations to be considered. To make matters worse, there are 56 possible relationships between pairs of data points that may be tested against the corresponding relationships between 144 pairs of object model faces, producing 1.64×10^{120} nodes down to level eight in the interpretation tree!

Even with inconsistent interpretations eliminated, i.e. those which associated any given data point with more than one object model face, but with no other pruning, there would, in this example, be over ten hours of processing to do at a processing rate of one tenth of a millisecond per comparison, say, for a single object.

With a parallel processing facility there is scope for a substantial reduction in overall processing time from data capture to object recognition, but the size of the unpruned interpretation tree remains a problem, in that it is likely to place a totally unacceptable demand on memory requirement.

However, we note that the sub-tree from nodes at a particular level in an unpruned interpretation tree are independent of the path from the root to those nodes, and will be reproduced many times over. The initial comparison between the data and a particular object model may then be represented by a network, and stored more efficiently in an array.

The network approach lends itself to parallel processing in which all of the alternative interpretations of the relationships between a particular pair of data points, i.e. the entries in a given row of the array, are tested in parallel. This is in stark contrast with the approach of Flynn and Harris[3],in which separate processors are allocated to each one of the alternatives in an internally consistent, but otherwise unpruned, interpretation tree. For 250 objects, each with 10 faces, and three data measurements they use 250,000 of the 256,000 processors available on the MIT Connection Machine, and their requirement for processors would increase by a factor of 10 with each additional data point!

Murray[4], Grimson and Lozano-Pérez and others have observed that the efficiency of interpretation tree pruning is improved when data are presorted so that the most effective pruning occurs near the root of the tree. They have found that sorting on the basis of pairwise separation avoids a combinatorial explosion, and substantially reduces the size of the interpretation tree.

In test runs with an object model representing an electrical plug with 27 faces, 14 of which were visible in a given view, and with 56 sorted data points at which measured normal directions were accurate to within 0.05 radians, Murray managed to arrive at a unique interpretation out of 10^{80} possibilities, after just 108 pairwise comparisons between object model and data. 1614 comparisons were required with unsorted data, and we note that even the hardware intensive approach of Flynn and Harris might well have involved 1540 steps before reaching a conclusion.

However, it may be argued that the separation of data points will not always be the most significant measure with regard to the effective pruning of the interpretation tree; for example, a strangely oriented face such as at the damaged corner of a chipped brick might well provide the most important key to the solution of the problem. Fortunately, there is no need to prejudge the issue, because the recognition method that is described herein lends itself to a much more flexible approach. The most effective constraints are likely to be reflected in a relatively small number of geometrical matches between an object model and the data, and it is a simple matter to sort the data into ascending order of geometrical matches, before launching into the recursive procedure that checks for consistent assignments of data points to object model faces.

Grimson and Lozano Pérez considered both visual and tactile data, but the emphasis in the present work is on the analysis and interpretation of data from photometric sources alone. Even with a multiple camera system, such data will not normally be available for every surface of the object that is being viewed, and we find that the absence of data with regard to hidden surfaces may cause problems in distinguishing between similar objects such as cuboids and parallelepipeds. We are then obliged to look for the most likely interpretation of a scene, perhaps with a fairly subjective ordering of the object model alternatives from which the choice is to be made.

Murray has concentrated on shape data alone, without absolute size. Working with optical flow data, he has observed that the precision of size information is often poor, unless an active range finder or some very high accuracy stereo ranging system is used, and that it is frequently necessary to recognize classes of objects of similar sizes. He maintains that shape data are highly effective in constraining the search space of matches to stored 3D object model. However, there is no need to prejudge the availability and quality of size information here, because it is possible to develop and implement the algorithms required for interpretation quite separately from the object model descriptions and the data involved in the matching process.

A SIMPLE EXAMPLE

Consider the two-dimensional object shown in Figure 1. We shall call it a flag, but the real world interpretation does not matter. We are only interested in the generation of feasible interpretations at this stage, and these will be considered in terms of abstract relationships between the sides of the object. Nor does it matter that we are looking at a two-dimensional object when we are really interested in the problem of object recognition in a three dimensional world. In two dimensions we consider edges, the sides of a polygon, in three dimensions surfaces, the faces of a polyhedron. Furthermore, it does not really matter at this stage whether we are dealing with visual or tactile data; the algorithms are essentially the same.

Suppose that the flag is described in terms of the directions of the outward normals at the edges, and that the directions of outward normals have been measured at four points, to within ±7½ degrees. The orientation of the flag is unknown, so we can not compare the measurements directly with a description of the flag to establish which sides should be associated with particular data points. We have to look at the differences between pairs of measurements, and consider whether they are consistent with the proposition that given pairs of sides are associated with particular pairs of data points.

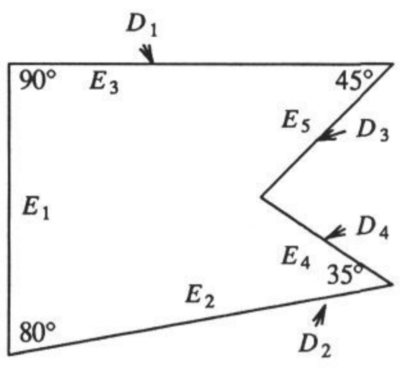

Figure 1 - The Flag with Four Data Points

The interpretation tree for this problem down to level 3, i.e. for the first three data relationships, is shown in Figure 2, and it can be seen that the three subtrees from the level 2 nodes at which a match has been achieved between model and data are exactly the same. The tree has been pruned wherever the relationships in the object model fail to match those in the data, but there has been no backtracking to ensure that points have been assigned consistently to particular object edges.

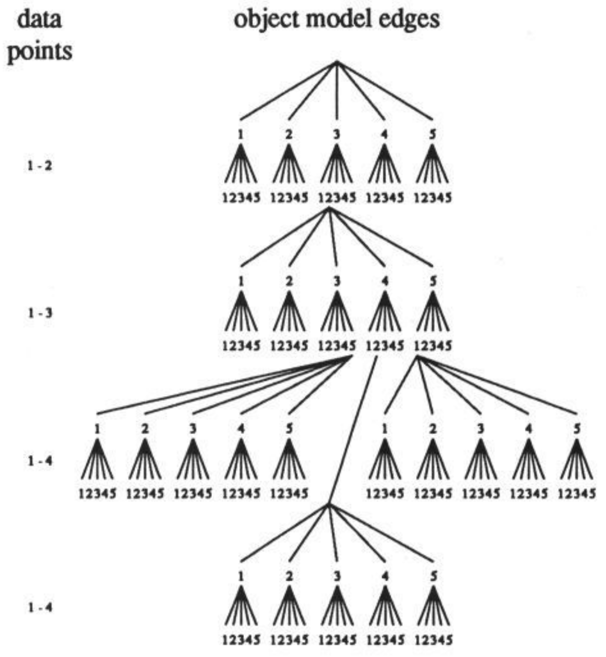

Figure 2 - The Interpretation Tree for the Flag

The same information is stored more compactly in an array, in Table 1, where we note that all paths downwards through true values must be explored.

		edge_i	1	2	3	4	5
		edge_j	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5	1 2 3 4 5
datum_i	datum_j						
1	2		.....	.....	.T...	.....	.....
1	3		.....	.....	....T	.T...	T....
1	4		.....	.....	...T.	.....	.T...
2	3		.....	....T	.....	..T..	.....
2	4		....T	...T.	.....	.....	..T..
3	4		T....	.....	.....	T....	...T.

Table 1 - The Matching Array for the Flag

It is immediately evident from Figure 2 and Table 3 that the data points D_1 and D_2 are associated with edges E_3 and E_2, respectively (the order may be reversed if we allow the flag to be turned over). Considered in isolation, D_1 and D_3 might be associated with E_3 and E_5, respectively, or with E_4 and E_2 or E_5 and E_1, but we have already assigned D_1 to E_3 and, to be consistent, we must now assign D_3 to E_5. Similarly, when we consider the relationship between D_1 and D_4, we quickly establish that D_4 must be assigned to E_4 and the interpretation is complete. We note that the remaining two relationships between the data points merely serve to echo the interpretation, without really providing any additional data.

There is no information for the location of the data points to be determined more precisely, or for the validity of the interpretation to be confirmed more rigorously, but the orientation of the flag may be established with reference to the directions of the normals at the separate data points.

Incidently, if the margin of error in the measurements was as much as ± 10 degrees, the interpretation would no longer be unique.

THE GEOMETRIC CONSTRAINTS IN THREE DIMENSIONS

There is an obvious ambiguity in the two-dimensional example described above, because all of the angles are reversed when the flag is turned over, and the problem is compounded in three dimensions. When the signs of the angles are included in two-dimensional object model descriptions, they may be inconsistent with the signs observed in the data, but spurious interpretations may be introduced when these signs are disregarded. Furthermore, when we consider the equivalent constraint in three dimensions, the direction of the unit vector associated with the angle between two surfaces is ambiguous. If this is disregarded, there may be such a proliferation of spurious interpretations as to render the recognition process useless, without the inclusion of additional constraints.

A distance constraint, i.e. a comparison of the distances between pairs of data points against the maximum and minimum values associated with pairs of object model faces, in addition to the angle constraint, does little to alleviate the problem. What is required is a set of constraints that completely exploits the information that is available, in a manner that is independent of the global coordinate frame of reference.

One way to get coordinate-frame-independent constraints, with regard to data points considered in pairs, according to Grimson[5], is to construct a local coordinate frame relative to the data points, and this may be done with the unit normals to the surfaces at the two points, and the unit vector in the direction of their cross product, used as axes. Given such a local basis, one set of coordinate-frame-independent measurements is provided by the components of the separation vector **d** along each of the three directions defined above, together with the unsigned angle between the two normals $\mathbf{n}_1$ and $\mathbf{n}_2$.

We thus obtain:-

$$\mathbf{n}_1 . \mathbf{n}_2$$
$$\mathbf{d} . \mathbf{n}_1$$
$$\mathbf{d} . \mathbf{n}_2$$

and

$$\mathbf{d} . \mathbf{n}_1 \times \mathbf{n}_2$$

as a basis for comparison between object model and data.

THE RECOGNITION ALGORITHM

The recognition processes described in the example are applied to three-dimensional objects, and formalised in a fairly obvious notation, as follows:-

```
scan(1)

procedure scan(i)

for $object\_face_1$ := 1 to n
  $clear_1$ := $face[datum_1[i]] = 0$
  if $clear_1$ then $face[datum_1[i]]$ := $object\_face_1$
  for $object\_face_2$ := 1 to n
    $clear_2$ := $face[datum_2[i]] = 0$
    if $clear_2$ then $face[datum_2[i]]$ := $object\_face_2$
    {consider geometric match between
    data and object model faces}
    $match[object\_face_1, object\_face_2, i]$ :=
      $model[object\_face_1, object\_face_2] = data[i]$
    {check for consistent assignment of
    data points to object model faces}
    $consistent\_match$ :=
      $match[object\_face_1, object\_face_2, i]$
      and ($object\_face_1 = face[datum_1[i]]$)
      and ($object\_face_2 = face[datum_2[i]]$)
    if $consistent\_match$ then
      if i<$mpairs$ then $scan(i+1)$
      if i=$mpairs$ then $report\_interpretation$
      {prepare to backtrack for
      alternative interpretations}
      if $clear_1$ then $face[datum_1[i]])$ := 0
      if $clear_2$ then $face[datum_2[i]])$ := 0
    endif
  endfor
endfor
```

By far the most important single step in our quest for parallelism, remembering that the geometric matching process is totally independent of the preceding partial interpretation, is to deal with that part of the algorithm once and for all, before entering the recursive procedure. The process translates conveniently into:-

```
   DO 10 i = 1,mpairs
     match(,,i) = ...
     icount(i) = SUM(match(,,i))
10 CONTINUE
```

in DAP FORTRAN, with all of the possible matches between the object model and a particular pair of data points computed in a single matrix assignment.

A simple tag sort procedure may then be used to sort the data pairs into ascending order of geometric match. Using the MINP function in DAP FORTRAN, which returns a logical matrix with true value(s) corresponding to the minimum value(s) of the argument, and ELN, which returns the index of the first true value of the matrix regarded as a long vector, a tag sort may be implemented as:-

```
   DO 20 i = i,mpairs
    k = ELN(MINP(icount))
    itag(i) = k
    icount(k) = 9999
 20 CONTINUE
```

although a process known as bitonic sort would be marginally more efficient.

Although at any stage the check for consistency is then dependant on the preceding partial interpretation, it can be performed as a parallel process within the recursive procedure. The subsequent assignments of data points to object model faces can then be made conditional on the outcome, with a construct of the form -

while any *consistent_match* do ...

replacing the nested do loops.

There is no WHILE statement in DAP FORTRAN, but there is an ANY function that returns a true value, provided that at least one element of a logical argument is true, and we may implement the construct with successive true values of consistent_match identified by ELN, switching values to false as soon as they have been processed.

Thus, with matrices object_face1 and object_face2 set up with every column of object_face1 containing the row number, and every row of object_face2 containing the column number, we obtain:-

```
    SUBROUTINE scan(iparam)
    ...
    i = itag(iparam)
    clear1 = (face(datum1(i)).EQ.0)
    clear2 = (face(datum1(i)).EQ.0)
    consistent_match = (match(,,i)
    .AND. (clear1.OR.(face(datum1(i)).EQ.object_face1))
    .AND. (clear2.OR.(face(datum2(i)).EQ.object_face2)))
    GO TO 20
 30  k = ELN(consistent_match)
     consistent_match(k) = .FALSE.
     face(datum1(i)) = object_face1(k)
     face(datum2(i)) = object_face2(k)
     IF (i.LT.mpairs) CALL scan(iparam+1)
     IF (i.EQ.mpairs) CALL report
     IF (clear1) face(datum1(i)) = 0
     IF (clear2) face(datum2(i)) = 0
 20 IF (ANY(consistent_match)) GO TO 30
```

The processing within the loop is of a sequential nature, but this seems inevitable if the demands on processing elements are to be kept within reasonable bounds. Nevertheless, highly effective pruning of alternative interpretations is thus achieved, because the consistent_match matrix is generally very sparse. The ANY and ELN functions are efficiently implemented in DAP FORTRAN, and the loop contains only a few simple assignments.

THE METHOD OF FLYNN AND HARRIS

Although the method of Flynn and Harris makes enormous demands on the hardware, it avoids the need for recursion and provides a yardstick against which the present method can be evaluated, at least for simple objects with just a few data points. Their method is summarised in the algorithm below.

```
n_total := $n^m$
for i:=1 to m
 same_f := $n^{i-1}$
 f := 0
 for j:=1 to n_total in steps of same_f
  f := f + 1
  for k:=1 to same_f
   temp [j+k-1] := f
  endfor
  face [,,i] := temp
 endfor
endfor

match := true
for i:=1 to m-1
 for j:=1 to m
  match := match
  and (model [face [,,i],face [,,j]] = data [i,j])
 endfor
endfor

while any match do
 report interpretation
endwhile
```

We note that m represents the number of data points, not the number of pairs of data points, and that *object_face*[,,i], i:=1 to m, is a matrix of up to 1024 possible interpretations with regard to a particular data point, on a 32×32 DAP (if *n_total* > 1024, e.g. n=8 and m>3 or n=5 and m>4, the face and match matrices must be partitioned).

Unfortunately, the use of matrices as mapping functions, for the purpose of indirect addressing, is not provided for on the DAP. It is necessary to form look up tables in serial mode, one for each constraint for each pair of data points, before embarking on the comparison, perhaps with several sets of data. The comparison then becomes

```
match := true
ijpair :=0
for i:=1 to m-1
 for j:=1 to m
  ijpair := ijpair + 1
  match := match and (model[,,ijpair] = data[i,j])
 endfor
endfor
```

and this is easily translated into DAP FORTRAN.

TEST RESULTS

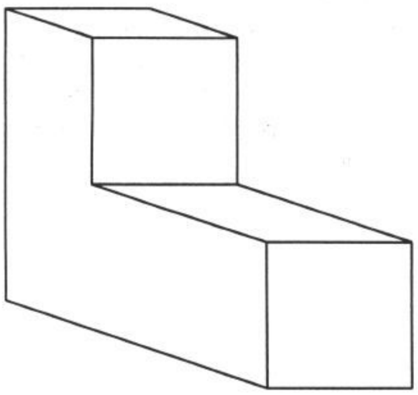

Figure 3 - The L Shape

Early test runs with the L shaped object model illustrated in Figure 3, with one data point on each visible face, produced some encouraging results. The program returned the original interpretation, together with its mirror image in the plane of symmetry, with a linear variation in run times, with respect to the cumulative number of consistent matches between pairs of data points and object model faces. Furthermore, there was a significant benefit when the data pairs were sorted into ascending order of geometric match, with run time being reduced from 14.8ms to 7.8ms. The run time with data pairs sorted into descending order of geometric match, as a 'worst case' test, was 305.4ms.

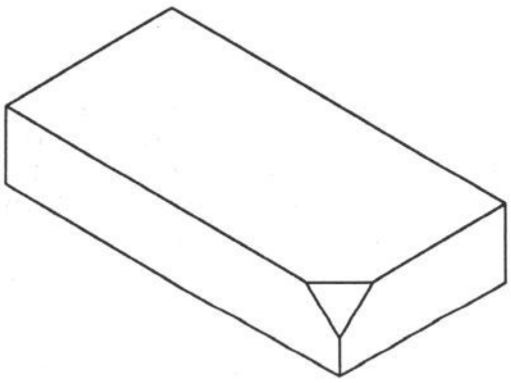

Figure 4 - The Chipped Brick

Further test runs have been made with data points at the centre of the faces of the chipped brick illustrated in Figure 4, with the first data point on the chip face. The first four data points relate to the visible faces of the brick, and the remainder relate to the hidden surfaces. The latter may be regarded as tactile data, but the distinction between visual and tactile data is of no consequence whatever at this stage in the recognition process.

The effect of the number of data points on run times with Grimson's constraints applied to unsorted data is illustrated in Figure 5.

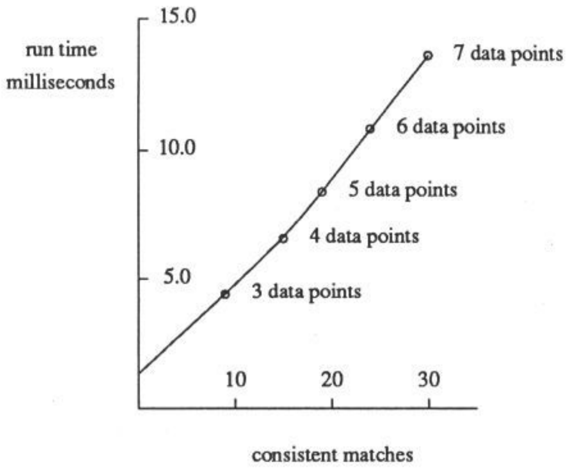

Figure 5 - The Relationship between Runtime and Consistent Matches for the Chipped Brick

There is just one interpretation with four or more data points, and the cumulative number of consistent matches increases from 15 with four data points to 30 with seven data points, with the corresponding run time increasing from 6.6ms to 13.6ms.

The run time is in fact increased from 8.4ms to 10.8ms with 5 data points sorted into ascending order of geometric matches, but these times are both regarded as highly satisfactory, especially by comparison with the 'worst case' run time of 67.0 ms.

The total number of geometric matches is increased from 75 to 146 when the constraints involving $\mathbf{d}.\mathbf{n}_1$ and $\mathbf{d}.\mathbf{n}_2$ are dropped. The time required to check these for consistency is increased from 4.8ms to 14.6ms, and the program returns 4 alternative interpretations. There is only one additional geometric match when the constraint involving $\mathbf{d}.\mathbf{n}_1 \times \mathbf{n}_2$ is omitted, but this feeds through to a cumulative total of 12 additional consistent matches to be explored further, in the partial interpretations.

Finally, a comparison is made with the performance of a sequential program, and of an implementation of the method of Flynn and Harris. The run times with 3 data points are 71.2ms and 650.3ms, respectively, including a set up time of 648.1ms in the latter case, by comparison with a total run time of 4.4ms by the present method.

There is scope for some reduction in the time required to form the look up tables that are required by Flynn and Harris. In any case, these tables may be installed in an initialisation process, in which case the time required to do so becomes relatively unimportant.

Of much greater significance is the way the memory requirement of Flynn and Harris escalates as the number of data points increases. With just 7 data points on the chipped brick, every look up table would have to be processed in 805 segments, on a 32×32 DAP processor. The time required to establish a geometric match would increase pro rata, from about 5ms to over 4 seconds, by comparison with a total run time of 13.6ms by the present method. Actual run times for the chipped brick are compared with projected run times by the method of Flynn and Harris for up to 5 data points, in Figure 6.

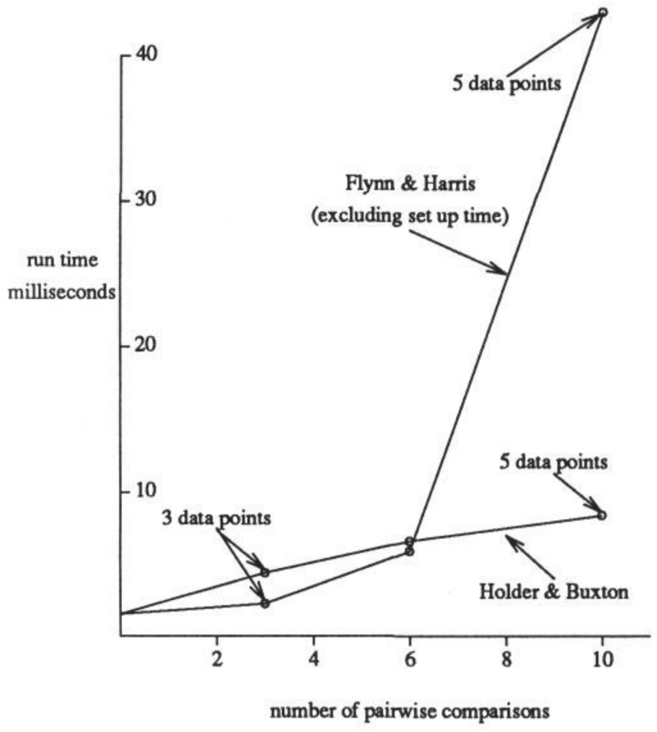

Figure 6 - Run Times for the Chipped Brick by the Present Method and the Method of Flynn and Harris.

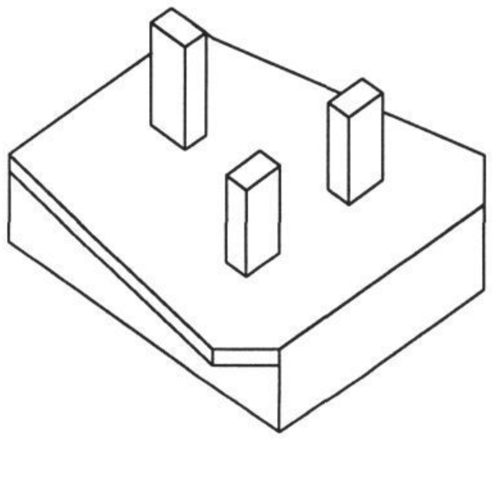

Figure 7 - The Electric Plug (View 1)

We have established that the present method works well with simple object models, but it remains to be seen whether the same applies with more complex object representations. A three-pin electric plug, similar to that used by Murray[1], has therefore been set up with a view to further performance tests.

The plug has 27 faces, and is viewed from three different positions, with data points in each case at the centre of every visible face. The first view, which is illustrated in Figure 7, has 14 visible faces, and involves 91 pairwise comparisons; the second view, looking towards the back of the plug has 12 visible faces, involving 66 comparisons; the third view looks straight at the ends of the pins, with just 4 visible faces, involving 6 comparisons.

An early run forcefully demonstrated the need to sort the data into ascending order of geometric match, before proceeding to check for consistent assignment to object model faces. Once this is done, the program quickly homes in on the correct interpretations with regard to Views 1 and 2, with run times of 65ms and 37ms, respectively, for complete pairwise matching of the two data sets against model data. Not too surprisingly, the program fails to distinguish between the ends of the two short pins in View 3, and returns four interpretations in 10ms. The effects of working with subsets of the data on the sequence of consistent matches with View 1, and the resultant run times for Views 1 and 2, are illustrated in Figures 8 and 9.

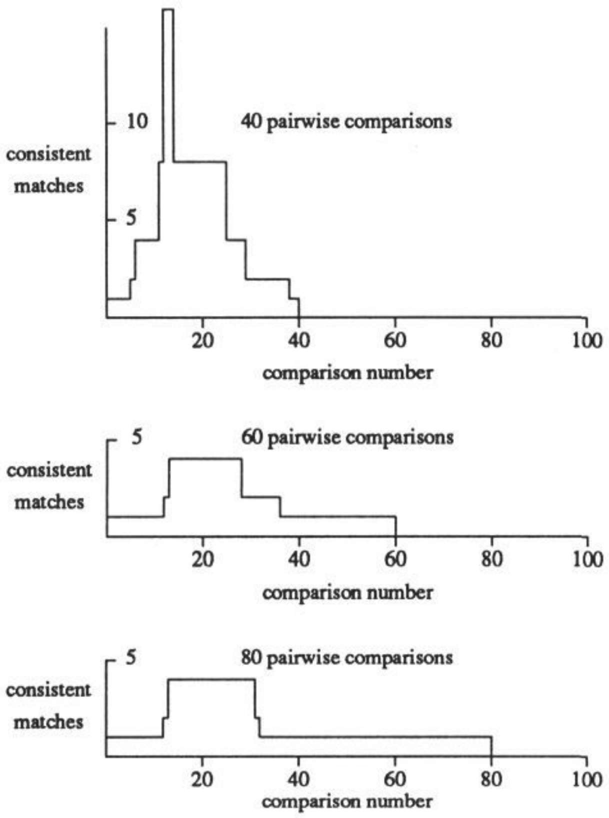

Figure 8 - The Effect of the Number of Pairwise Comparisons on the Sequence of Consistent Matches for View 1 of the Electric Plug

It can be seen that there is a trade off between the increased time required to establish the situation with regard to geometric match, when additional pairwise comparisons are appended to the data, and the reduced time needed to check the alternatives for consistency, when some of the additional data pairs for which there are few geometric matches appear near the front of the sorted list. Thus a minimum runtime of about 46.5ms is achieved with about 50 pairwise comparisons with View 1, and a minimum of about 27ms with about 40 comparisons with View 2.

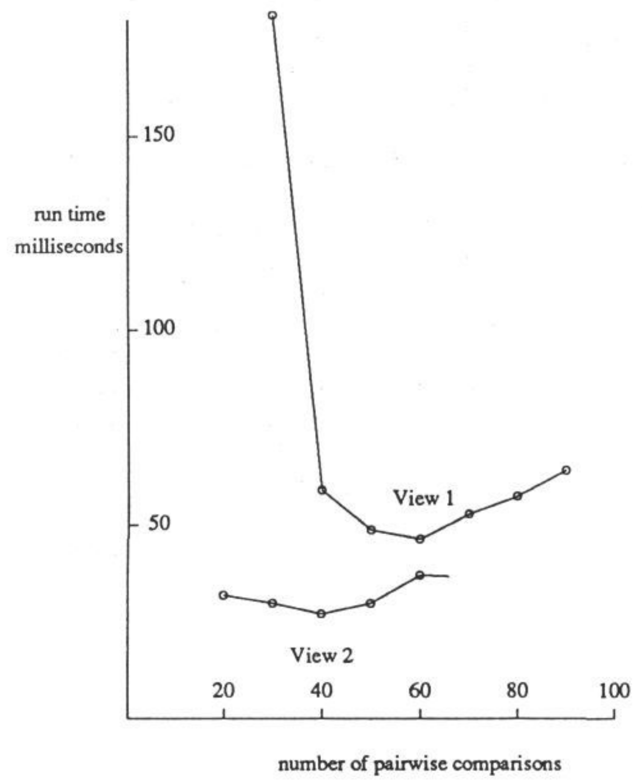

Figure 9 - The Effect of the Number of Pairwise Comparisons on Run Times for the Electric Plug

Furthermore, there is obviously scope for early termination when all of the data points have been uniquely assigned to object model faces.

CONCLUDING REMARKS

Given m data points on a polyhedral object model with n faces, we have a method that fully exploits n×n parallelism in establishing the possibilities of a geometric match. This is achieved without the escalation in the memory requirement, as m increases, of the Flynn and Harris method. The run times, with either method, inevitably increase at least linearly with m(m-1)/2, the number of pairwise comparisons between object model and data that have to be made, and the number of segments into which the look up tables have to be divided to fit onto the machine is a multiplying factor. A consistent assignment of data points to object model faces is guaranteed by Flynn and Harris, but this has to be checked within the present method. The run time for the additional work can escalate if there is a large number alternatives to be checked for consistency, but it seems that this is not a problem provided that the data pairs are sorted during the run, into ascending order of geometric match.

The electric plug model is comfortably accomodated without segmentation in the present method, on a 32×32 distributed array processor, the AMT DAP 500. With 27 faces, 14 of which are visible in the most interesting view, and with a data point at the centre of each face, there are more than 10^{20} possible interpretations. A separate processor is required by Flynn and Harris for each interpretation within a given segment, and this is clearly not a feasible proposition either on the AMT DAP or on the MIT Connection Machine. The present method accomplishes the task, with a processing time of about 50 milleseconds.

Having shortlisted the feasible interpretations the next step is to determine the most likely position and orientation and, if necessary, to choose between alternative object models. Solutions may be obtained by applying the method of least squares to the differences between a given object model and the data. Faugeras, Ayache and Faverjon[6] describe how this may be accomplished, with model surface elements and the data represented by the quaternian pairs (**n**,d), where **n** is the normal unit vector and d is the distance from the origin, in a global frame of reference.

ACKNOWLEDGEMENTS

The authors would like to thank Dr Bernard Buxton and Dr David Murray of the Hirst Research Centre, GEC Research Ltd., for the advice and assistance received from them in the course of the work described herein. Thanks are also due from the first author to numerous members of the support staff in the Department of Computer Science, especially to William Roberts for his introduction to the software at QMC, and to Kevin Smith in the Centre for Parallel Computing, for his advice with regard to the implementation of the method of Flynn and Harris.

REFERENCES

1. **Murray D.W. and Cook D.B.** "Using the Orientation of Fragmentary 3D Edge Segments for Polyhedral Object Recognition" *International Journal of Computer Vision* Vol 2 (1988) pp 147 - 163.
2. **Grimson W.E.L. and Lozano-Pérez T.** "Model-Based Recognition and Localization from Sparse Range or Tactile Data" *International Journal of Robotics Research* Vol 3 (1984).
3. **Flynn A.M. and Harris J.G.** "Recognition Algorithms for the Connection Machine" *Proc. Ninth International Joint Conference on A.I.*, Morgan and Kaufman (1985) pp 57 - 60.
4. **Murray D.W.** "Model-Based Recognition Using 3D Shape Alone" *Computer Vision Graphics and Image Processing* Vol 40 (1987) pp 250 - 266.
5. **Grimson W.E.L.** "The Combinatorics of Local Constraints in Model-Based Recognition and Localisation from Sparse Data" MIT Artificial Intelligence Laboratory A.I. Memo 763 (1984).
6. **Faugeras O.D., Ayache N. and Faverjon B.** "A Geometric Matcher for Recognising and Positioning 3-D Rigid Objects" *Proc. A.I. Applications Conference* IEEE (1984) pp 218 - 224.