REASONING ABOUT THE SPATIAL RELATIONSHIPS DERIVED FROM A RAPT PROGRAM FOR DESCRIBING ASSEMBLY BY ROBOT

Corner ,D.F., Anbler,A.P. and Popplestone ,R.J.
Department of Artificial Intelligence,
University of Edinburgh

ABSTRACT

The network reduction systen of RAPT, an offline object level language for programming robot assembly tasks is described in this paper. The task description is converted into a relational network with the positions of bodies as nodes, and relationships between positions as links. Rewrite rules based on the intersection and conposition of relations are used to simplify the network.

1 Introduction.

As versatile industrial robots come into greater use, particularly for snail batch assembly, the ability to program then offline becomes more desirable. The RAPT language has been developed at the University of Edinburgh for such prograrning of assembly tasks.

The language allows the user to describe the task in a natural way, in terns of surface features being In contact, and of notions of objects relative to each other. The gripping, parts of manipulators can be included simply as objects used in the task.

The transformation of this relational information into the positional information necessary to drive a manipulator is not a trivial task, and this paper is a description of the current system for doing it. This involves the reduction of a relational network, and was described in (Popplestone, Ambler & Bellos 1980b). An earlier system used algebraic manipulation, (Popplestone, Ambler & Bellos 1930a), and conparison will be made with it.

Our system as implemented at present is completely determinate - all positions are specified at compile time. Another paper at this conference (Yin 1933) describes how sensory information, in particular vision verification, can be incorporated.

II The RAPT input language

This section can only give a brief outline of the input language; for a complete description see (Popplestone, Ambler & Bellos 1978).

This paper describes work done in the Department of Artificial Intelligence at the University of Edinburgh supported by contract N2B IR1019 from the Science and Engineering Research Council.

In RAPT, each object is described in terms of a number of surface features, e.g. planar faces, shafts (cylindrical faces) and holes.

At any stage of the assembly process, we have a situation in which we can describe spatial relations between objects, e.g.

```
AGAINST / FACE1 OF B0DY1, FACE2 OF B0DY2
FITS    / SHAFT OF B0DY1, HOLE OF BODY2
```

An action statement describes the movement between one situation and the next, e.g.

```
MOVE / B0DY1
MOVE / BODY1, PARLEL , SHAFT OF B0DY1
MOVE / BODY1, PARLEL , SHAFT OF B0DY1, 100
```

Actions may constrain the position of the body, e.g., the second and third actions constrain the body position to be along the axis of the shaft from its previous position, the third action completely determines the final position relative to the start position. The first action imposes no constraints on the final position; it must be determined from any subsequent spatial relations.

III The RAPT Inference system.

The specification of a RAPT progran consists of a number of relationships between features of objects in different situations, and a number of actions. We must solve for the position of each body in each situation. We call this set of positions a set of body instances, and we have a relational network linking them. Within any situation, body instances are linked by spatial relations; between situations the links are either action relations, or identity relations for stationary objects. The form of each link is identical, whichever type of constraint, and therefore situation and action relations can be mixed in any way to form the network. The task of the inference system is to reduce the network, and to determine the positions of each object.

A. Position Representation.

We represent the position of each body instance as a 4x4 matrix which holds the origin and the orientation of the X,Y, and Z axes of each body. Body positions are defined relative to a world frame of reference.

We use the post-multiplication convention of transformations to define positions transformed from the body position. Each feature of a body can be represented by a transformation from the

body position, and therefore can be expressed relative to the world frame of reference by a t ransformat ion,

f ** p

where ** denotes matrix multiplication, f is a feature transformation, and p a body position.

B. The Representation of Relationships.

A relationship links features of two bodies. Corresponding to each relation type is a transformation function with a number of variable parameters, representing the degrees of freedom implicit in the relation. The relationship states that the world position of one feature, transformed by this function, with unknown values for the variable parameters, equals the world position of the other feature.

f2 ** p2 = Rel(v) ** fl ** pl

is an example of a relationship linking features f1, f2 of bodies in positions p1, p2 respectively, where v is a vector of n variables unique to this instance, and Rel is one of a set of relation functions. An example of such a relation is

ACPP(x,y,theta)

represent ing the against relationship between two plane faces, with two spatial and one rotational degree of freedom.

Actions are represented by similar equations. If an action operates on a body to move it from position p1 to position p2, and the action is relative to a feature f, it can be represented by the equation

f ** p2 = Rel(v) ** f ** p1

This is an equation of the same form as the equation derived from a relationship, and the RAPT inference system does not distinguish between them.

It is implicit in the description of actions that some bodies in the system do not move during the action. This is encoded by making the positions of the body before and after the action equal.

C. Reducing the Set of Relations.

The fundamental reduction step takes a pair of equations relating the same two positions

p2 - expressionl(v1) ** p1
p2 ″ expression2(v/2) ** p1

and combines them to form a new equation

p2 ■ expression3(v3) ** p1

Each expression Is the product of one or more relation functions and feature transformations, and the vectors represent the variables. Expression is derived from expressionl and expression2, and v3 has at most the number of elements of the smaller of v1 and *v2,* where v3 represents the remaining degrees of freedom.

The equation solving method solved the set of equations from first principles, but it was realised that since for most pairs of the relation functions Rel(v), there are standard solutions, this was inefficient. The present method uses these standard solutions, and dors not form the equations; these are implied by the form of the relation. Thus, when a suitable pair of relations is found, the standard solution is applied, e.g., subject to certain constraints, from the relations

AGPP between FACK11 of B0DY1 ,FACE;21 of B0L)Y2
AGPP between FACE12 of B0DY1,FACK22 of BODY2

the standard solution gives

LTN between FEAT13 of B0DY1,FEAT23 of BODY2

LIN is a relation with only one degree of spatial freedom, *and* FEAT13, FEAT23 are generated as part of the standard solution.

Not all pairs of relations have a solution. In some cases there are two results, in others the relation is not expressible as one of our set of relations.

D.. FIX Relations.

A FIX relation is a special case, because the linking function is the identity function, with no variables, i.e. the equation becomes:

f2 ** p2 - fl ** p1

We have determined the position p2 relative to p1, so that whenever we have an equation in p1, we could transform it to one in p2, or vice-versa.

We use a FIX relation to do a merge operation. We choose one of the positions, say p2, and remove it from our set of unresolved positions. We also transform any equation involving p2 into one involving p1. When, at a later time, we have solved for the position p1, we can transform it and so produce the position p2.

E. Chaining relations.

We can only use our standard solutions when we have two relations between the same pair of body instances. The method of chaining relations allows a solution in more general cases.

Given a relation involving body instances A and B, and another involving body instances B and C, the two may be chained together to form a new relation between A and C.

The equation solver placed no limitation on chaining, since it could combine

f21 ** p2 = Rell(vl) ** fll ** p1
f31 ** p3 = Rel2(V2) ** f22 ** p2

to form the equation

f31 ** p3 - Rel2(v2) ** f22 ** inv(f21) ** Rell (vl) ** fll ** p1

In a number of cases, this can be simplified to the form

f32 ** p3 = Rel3(v3) ** f12 ** pl

where Rel3 is one of the set of relation functions.

When this simplification is possible, we can apply a standard chaining solution to the pair of relations to form the new one.

Not all pairs of relations can be chained; for some, the result is not expressible as one of the set of relations, for others, there is no relation - the input relations are too weak and the two bodies are unconstrained.

F. Method of Application.

We could consider our operations, solution, chaining *and* merging as a set of rewrite rules. The solution operation is obviously a terminating operation, since it reduces the number of relations (equations) in the system. Merging is also terminating, since the number of unresolved positions is reduced.

Chaining by itself does not appear to be a terminating operation, since it adds a new relation to the set of relations. (It does not add any new information, since the Information in it is contained in the relations it was derived from). Because of this, we only use chaining to create temporary relations.

When we have a cycle of relationships:
Relation between Bl and B2
Relation between B2 and 33
... ...
Relation between Bn-1 and 3n
Relation between Bn and 31
we can chain the first two to form a relation between Bl and B3, use this to form one between *Hi* and B4, and so on until we have chained all except the last together to form a relation between B1 and Bn. We can now apply our solution method to this and the final relation; if it gives a new relation, we can replace the relation between Bn and Bl with this new relation.

Since the result of our solution method has the same number or fewer degrees of freedom than either input relation, and we only accept the solution from the combination of a relation formed by chaining with another if we have reduced the number of degrees of freedom, the net effect of the operation is to reduce the number of degrees of freedom in the relations left in the problem.

With a complex network of relations there can be a large number of cycles, many of which will be large, and for each cycle we can choose any of the body instances as a start point, and work in either direction. There is therefore a large search space. In practise, we restrict the size of the cycles to which we apply this method to a maximum of four. This has been found to be adequate. In addition, we defer cycle finding until there re are no more solution or merge operations possible. If the processing of cycles has resulted in any FIX relations, merge operations can be performed, with the consequent possibility tahe more pairs of relations appear.

In general, our process consists of alternate application of the solution and merge operations, and the cycle finding operations, until all the positions have been resolved, or until no further progress is possible.

C. Limitations of the System.

The cycle-finding system is very much more efficient than the equation solving system (speeds have increased 100-fold) but it is less powerful. The restricted set of relations to which the standard solutions can be applied means that some simplifications cannot be made that the equation solving system would manage. The pairwise combination of relationships and the restriction to unique solutions sometimes means that a solution within the theoretical capability of the system is missed: three constraints taken a pair at a time might always produce two solutions; the third constraint would eliminate one of them. There are some specifications which should produce multiple solutions (eg 3 revolute joints): at the moment the cycle finder does not solve them. There are some constraints which cannot he easily represented in the simple relational form. These involve variables: for example the jaws of a gripper may be constrained to move in equal *and* opposite directions by the same amount.

Given these inherent limitations in the system, we feel that the cycle finder should be used as front end to some more powerful inference system - the cycle finder will be used to solve quickly and efficiently those problems it is capable of, *and* pass the residue on the the next system. In our experience most of the problems we come across can be completely solved by the cycle finder.

Some linkages form standard, but large cycles. (Duffy 1980) has provided solutions for them. We anticipate including such solutions in the final inference system.

REFERENCES

(1) Duffy,J., Analysis of mechanisms and Robot Manipulators London: Edward Arnold, 1980

(2] Popplestone, R.J. , Ambler, A.P. , and Bellos, I.M., "RAPT: A Language for describing Assemblies" in Industrial Robot Sept. 1978, pp.131-137

(3] Popplestone, R.J., Ambler, A.P., and Bellos, I.M., "RAPT: An interpreter for a language describing assemblies" in Artificial Intelligence 14(1980) pp.79-107

(4] Popplestone, R.J., Ambler, A.P., and Bellos, I.ii., "An Efficient and Portable Implementation of RAPT" in Proc. 1st ICAA. Bedford, UK: IFS(Publications), March 1980, "pp 411-422

[5] Yin,B., "A Framework for Handling Vision Data in an Object Level Robot Language" In this conference proceedings.