Assembling Objects Through Parts Correlation

Pamela J. Vermeer
Washington and Lee University
Lexington, Virginia 24450 USA
pvermeer@wlu.edu http://www.wlu.edu/~pvermeer

1 Introduction

When designing a complex mechanism or structure, it is often easiest to construct the object component by component. The simpler individual components can then be pieced together by correlating geometric elements of pairs of objects. In this paper, we present a solution to the problem of assembling four parts by matching points and planes in pairs of parts. This problem would be simple if the parts could be assembled by fixing one part and adding the others sequentially. In the situation we consider, however, after fixing one part, the remaining three parts must be positioned simultaneously. To do this requires solving an interlocked system of nonlinear equations with up to sixteen different solutions. Since we are interested in obtaining all possible positionings of the four objects which satisfy the given constraints, an analytic approach is taken to solve the resulting system of equations.

We first discuss how our problem arises in the context of geometric constraint solving and we establish the framework of the required matching, that is, how the four base objects are related to each other. This basic framework leads to a suite of problems which must be solved, each corresponding to a different type of matching between the four parts. We next develop a general solution method for finding all the possible positionings of the four components which satisfy a particular matching. We finish with a discussion of an implementation in progress of a program to allow interactive solving of these constraint problems.

2 Problem Framework

The problem considered in this paper arises from the graph-based approach to spatial constraint solving. In the analysis phase of that method, geometric entities (or *geoms*) which can be placed relative to one another are formed into groups called *clusters*. Each geom begins with three degrees of freedom, so placing a new geom requires that it be constrained by three other previously fixed geoms. Thus to begin a cluster, there must be at least one set of three pairwise constrained geoms. These three geoms can be positioned in a standard way depending on the types of geoms involved and the constraints between them. A new geom or another cluster can be added to a cluster if there are three constraints between the existing cluster and the new geom or cluster to be added. This process continues until no new clusters can be formed and no additional geoms can be added to any of the existing clusters.

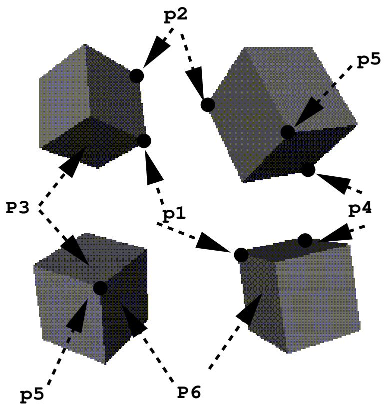

Figure 1: Four boxes to be assembled into a single unit

Once the initial clusters have been formed, *cluster merging* takes place. Having multiple constraints may have allowed a geom to be part of more than one cluster. Cluster merging combines clusters which share a geometric entity in such a way that the result is rigid in $\Re^3$, up to translation and rotation as an entire body. Further details about cluster formation and merging in spatial constraint solving can be found in[DH95].

In the problem we are working on, we begin with four objects. In order to assemble these objects into a single object, each object must be matched with every other object. That is, three different geoms of the first object must be chosen such that one of them is the same as some geom on the second object, another is the same as some geom on the third object, and the third is the same as some geom on the fourth object. Then two more geoms on the second object must be chosen so that the first matches a geom on the third object, and the third selected geom matches a geom on the fourth object. Finally, a geom on the third object must be selected to match a geom on the fourth object. With this framework, the problem becomes one of merging four clusters, where each cluster shares exactly one geom with each other cluster. Each cluster corresponds to the geoms

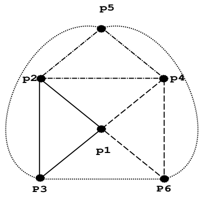

Figure 2: Cluster abstraction of cube assembly problem

selected on one of the objects.

For example, consider Figure 1. Here each unit cube is one of the objects to be assembled into a larger unit. The cube in the upper left can be reduced to the cluster $\{p_1, p_2, P_3\}$ with constraints $d_{12} = 1$, $d_{13} = 0$, and $d_{23} = -1$ (assuming outward normals on the object surfaces). The cube in the upper right is simplified to the cluster $\{p_2, p_4, p_5\}$ with constraints $d_{24} = \sqrt{3}$, $d_{25} = \sqrt{2}$, and $d_{45} = \sqrt{2}$. The cube in the lower left is represented by the cluster $\{P_3, p_5, P_6\}$ with constraints $d_{35} = 0$, $\alpha_{36} = \pi/2$, and $d_{56} = 0$. Finally, the cube in the lower right can be thought of as the cluster $\{p_1, p_4, P_6\}$ with constraints $d_{14} = 1$, $d_{16} = 0$, and $d_{46} = -1$. Solving this cluster merging problem will give positions for the three geometric entities on each cube. This is enough information to be able to rigidly place the cubes together.

The abstraction of the problem to cluster merging can be visualized as in Figure 2. Each of the four triangles in the figure with the same line style for edges represents one cluster. The lines indicate a known constraint between the geoms. The clusters are joined at exactly one geometric element; this indicates a matching between geoms in different clusters. Because of the octahedral nature of this abstract representation, we sometimes refer to the problem of matching objects in this way as *the octahedron problem*. In fact, if the geometries are all points and the constraints all positive distances between the points, the final configuration of the cluster geoms is an octahedron.

If one cluster is fixed in space, the other three clusters cannot be added sequentially because there are only two known constraints between the fixed cluster and any unfixed cluster. This means that to obtain a rigid configuration of the four clusters as a single object, the remaining three clusters must be placed simultaneously. In the rest of the paper, we present a solution to the problem of fixing the first cluster and then positioning the other three clusters with respect to that cluster.

Notation

Throughout the paper we use the following notation: g_i refers generically to either a point or a plane, while p_i is a point and P_i is a plane. Any constraint between two points or between a point and a plane is a signed distance d_{ij}, where the sign of the distance between a point and a plane is positive if the point is on the side of the plane towards which the normal points and is negative otherwise. Any constraint between two planes is an angle α_{ij}

3 A General Solution

An initial analysis of the problem in its abstract formulation as six geometric elements, each of which is either a point or a plane, comprising four clusters yields ten distinct cases which must be solved. These range from all clusters containing only points to all clusters containing only planes.

The following steps outline the solution process for any of the subcases.

1. Fix the coordinate system using the first cluster. This is done by explicitly positioning the three elements in the cluster. We now consider only the three remaining clusters.
2. For each of the three remaining unfixed elements, establish a parametric representation of the point, or of the normal direction in the case of a plane. This representation is formulated from the interactions between the unfixed element and the fixed element in each cluster in which the unfixed element appears.
3. Use the remaining constraint in each cluster between two unfixed elements to establish three algebraic equations in the three unknown parameters of the representation in step 2.
4. Solve the system of equations.
5. Substitute the parameter values into the representation in step 2 to determine the position of the three remaining elements.

In the following sections, these steps are explained in greater detail. We note that the case of six points is solved in its entirety in Nanua *et. al.* [NWM90]

3.1 Fixing the First Cluster

The first cluster is fixed by establishing a coordinate system based on its geometric elements and their relative positions. In every case in which there is at least one plane and one point, we make a cluster with a plane and a point the first cluster. We assume that this point, p_1, lies on the plane, P_3; this requires that we adjust any distance constraints between the plane P_3 and the other points in the system by $-d_{13}$. We then fix the first cluster by making P_3 the xy-plane, and p_1 the origin. The remaining point or plane completes the coordinate system.

In the last case, all six geometric elements are planes, and the only constraints are angles between the planes. This situation is underconstrained, as is the case of five planes and one point, but we can find necessary normal directions for the planes. In this case, we make the origin the intersection point of the three planes in the first cluster, and then proceed as before.

3.2 Determining the Parametric Representation

The first step of the placement process completely determines the positions of the three geometric elements g_1, g_2, and g_3 of the first cluster. The three remaining clusters have one fixed element (the element which is also in the first cluster), and two unfixed elements. Each unfixed element is in two of the remaining clusters, so has a relationship with two fixed elements. We can use the constraints between the unfixed element and these two fixed elements to establish a parametric representation of the unfixed element. For a point, the parameterization is the set of possible positions for the point as determined from the two fixed geometries. For a plane, the parameterization is the set of the possible directions of the normal vector. As an example, consider how the paramateric representation of an unfixed point relative to a fixed point and plane is determined.

Let p_1 and P_3 be the fixed entities, and p_4 be the unfixed point. The distance constraint between p_4 and P_3 means that p_4 must lie on a plane parallel to P_3, and the distance constraint between p_4 and p_1 constrains p_4 to lie on a sphere. The intersection of these two surfaces is a circle of form

$$(A + B\cos u,\ C\sin u,\ D)$$

When all the possible combinations of points and planes in the fixed and unfixed roles are explored, we find that the interactions lead to a parametric representation of the unfixed geometry which is one of three forms:

$$(A + B\cos u,\ C\sin u,\ D) \qquad (1)$$

$$(A + B\sin u,\ C\cos u,\ D + E\sin u) \qquad (2)$$

$$(A,\ u,\ B) \qquad (3)$$

3.3 Forming the Equations

In the previous step, a parametric representation was determined for each unfixed point or plane normal. We now use these representations and the constraint between the two unfixed geometries in each of the three unfixed clusters to set up three equations in the three parameters. There are three different equations, depending on whether the two unfixed geometries are both points, both planes, or one point and one plane.

The first equation comes from the distance between two points. If the two points are $p_i : (x_i, y_i, z_i)$ and $p_j : (x_j, y_j, z_j)$, then the equation is

$$(p_i - p_j) \cdot (p_i - p_j) = {d_{ij}}^2 \qquad (4)$$

The second equation comes from the angle between two planes. If the plane unit normals are $n_i : (x_i, y_i, z_i)$ and $n_j : (x_j, y_j, z_j)$, then the equation is

$$n_i \cdot n_j = \alpha_{ij} \qquad (5)$$

The final equation comes from the distance between a point and a plane. If the point is $p_i : (x_i, y_i, z_i)$ and the plane has normal $n_j : (x_j, y_j, z_j)$ and distance to the origin l_j, then the equation is

$$n_j \cdot p_i = d_{ij} + l_j \qquad (6)$$

To set up the actual equations to solve, we choose u as the parameter for the unfixed geometry g_4, v for g_5, and w for g_6. We then compute the coefficients of each parametric representation, and substitute the form into the appropriate equation above. Next, we make the standard rational parametric substitutions for the trigonometric expressions, and simplify. The result in all ten cases is that each equation has total degree four, and the degree in each of the two variables is two. Each equation has mixed terms, and so it is not possible to solve the three equations by simple substitution.

3.4 Solving the Equations

Because we desire all possible solutions to the assembly problem, the system of equations is solved symbolically as far as possible. When we perform elimination on the equations derived in the previous step, we obtain a polynomial of degree 16 in the remaining variable. We must rely on numerical methods to find solutions to this polynomial For each solution of the polynomial, there is at most one solution for the remaining two variables which satisfy all three equations. Thus there are at most 16 different positions in real space which satisfy the given constraints.

In the final step, having obtained values for each of the parameters u, v, and w, we substitute into the original parametric forms to obtain the positions of the points and the directions of the normals. In addition, except in the final two cases, we are able to determine the distance of each plane from the origin, since we either have the distance directly as a constraint or we know the distance from some fixed point to the plane and can compute the distance from the origin using that information.

4 Implementation

We have implemented a library which solves all ten cases of this problem, making distance assumptions in the last two cases to make them well-constrained. This library uses Maple to carry out the necessary numerical solving processes. We have also implemented a graphical user interface front-end to the assembler library using Open Inventor. This interface displays four objects and allows the user to interactively select a matching between points and planes of the four objects which correspond to well-constrained matching problem described in the paper. When a valid matching is selected, the program extracts the positions of the selected points and planes, and computes the distances and angles to input to the library.

The back-end of the interface uses the new positions of the geometries generated by the solver to compute transformations which take the geometries from their original positions to their new positions. The corresponding transformation is applied to each object and the assembled object will be displayed. A major advantage to our method over numerical solving of the equation system is seen by browsing through all the returned solutions: A numerical solution returns only one solution, which may be a solution in which the four objects intersect each other. Since our method returns all solutions, if there is a solution in which the objects do not intersect, it will be found and displayed.

5 Acknowledgements

Thanks to John Hills and Stephen Williard who developed and programmed the graphical user interface to the assembler library, and to Jean-Guy Speton who assisted in programming the assembler library. This work was funded in part by NSF grant number CCR-9505745, by the Washington and Lee University Glenn Grant program, and by the Washington and Lee University R. E. Lee Research Scholars program.

References

[DH95] D.-Z. Du and F. Hwang, editors. *Computing in Euclidean Geometry*, pages 266–298. World Scientific Publishing Co Pte Ltd, 2 edition, 1995.

[HV95] Christoph M. Hoffmann and Pamela J. Vermeer. A spatial constraint problem. In *Computational Kinematics Workshop*, Nice, France, Sept. 1995.

[NWM90] P. Nanua, K. J. Waldron, and V. Murthy. Direct kinematic solution of a stewart platform. *IEEE Transactions on Robotics and Automation*, 6(4):438–444, Aug. 1990.

Full paper: www.wlu.edu/~pvermeer/papers/SoCG.html