GEOMETRIC CONSTRAINT SOLVING IN $\Re^2$ AND $\Re^3$

CHRISTOPH M. HOFFMANN
Department of Computer Sciences, Purdue University
West Lafayette, Indiana 47907-1398, USA

and

PAMELA J. VERMEER
Department of Computer Sciences, Washington and Lee University
Lexington, Virginia 24450, USA

ABSTRACT

Geometric constraint solving has applications in a wide variety of fields, such as mechanical engineering, chemical molecular conformation, geometric theorem proving, and surveying. The problem consists of a given set of geometric elements and a description of geometric constraints between the elements. The goal is to find all placements of the geometric entities which satisfy the given constraints. In two-dimensions, several different approaches have been examined and implemented, while the three-dimensional problem has been much less explored in previous literature. In light of the diverse applicability of the problem, we have three objectives in this paper. First, we provide a brief overview of the basic approaches to geometric constraint solving. Second, we review a specific solution to constraint solving for two-dimensional geometric problems. Finally, we present developing work in extending the solution technique for the two-dimensional problem to geometric constraint solving for elements in three-space.

1. Introduction

Geometric constraint solving is a problem with applications in many arenas, such as mechanical engineering, chemical molecular modeling, and surveying. In each of these communities the problem has been approached in a variety of ways and with differing levels of success. The problem consists of a given set of geometric elements and a description of geometric constraints between the elements. The goal is to find all placements of the geometric entities which satisfy the given constraints. For example, the set of elements might be a set of three lines, with the constraints that the first two lines must be perpendicular and the third must make a specified angle with the first line. This particular problem has infinitely many solutions, and an additional constraint such as the length of one segment between the intersections of two pairs of lines would tie down a particular solution.

A problem is *well-constrained* if there are a finite number of solutions to the problem, while a problem with an infinite number of solutions is *underconstrained*. A problem is *overconstrained* if one constraint can be deleted yet the constraint system

still has a finite number of solutions. In the example of the three lines, if the angle that the third line must make with the second line were given as another constraint, the problem would be overconstrained, since that angle is already determined by the first two angle constraints. An overconstrained problem may have a solution when the additional constraints are consistent with previous constraints, but often overconstrained problems have no solution.

One application for geometric constraint solving is in the area of Computer Aided Engineering, particularly in the branch of mechanical engineering design. A method for designing an object with the computer should be strongly visual in order to provide an intuitive interface, but must also provide a way to produce a careful, detailed description. One approach for reaching these two disparate goals is to apply geometric constraint solving to the problem of geometric data input. Through a graphical user interface, the user can sketch a rough outline of the object to be constructed. By adding constraints such as the length of an edge of the object or the angle between two edges, a precise description of the object is obtained. Such a system has been designed and implemented with points, lines, and circular arcs in the plane as allowable geometric entities, and constraints such as an angle between entities, distance from one entity to another, incidence, and tangency (in the case of circular arcs)[4]. The object obtained as the solution to the constraint problem can then be swept or rotated to obtain a three-dimensional object. Additional features can be added by sketching on a two-dimensional plane of the resulting object, and extruding the sketched region, or cutting a slot or hole in that shape through the object.

As an example of this application, consider Figures 1, 2, and 3, which demonstrates the design of a control arm. These figures were generated using the feature-based modeling system of Chen[11], which interfaces with the two-dimensional constraint solver of Fudos[5] for its geometric input. Figure 1 on the left shows a profile which has been sketched, dimensioned, and solved by the constraint solver. Once the user is satisfied with the profile, it is extruded, generating the solid on the right of the figure. To add another feature to the model, a plane of the current solid is selected in which to sketch and dimension the new feature. Here, the top face has been selected as the reference face, and the profile of a new feature sketched, dimensioned, and solved, as shown on the left of Figure 2. This new profile is then extruded, yielding the solid shown on the right of the figure. Several other features, are added in similar fashion, and the final control arm is displayed in wire frame and shaded in Figure 3.

A second application area for geometric constraint solving is in geometric theorem proving. Often the assumptions and conclusions of such theorems can be expressed in terms of constraints between the geometric elements under consideration. Solving the constraint system entails that the relationships are possible, hence constitutes a proof of the theorem. An example of this application can be found in Chapter 7 of Hoffmann[7].

Yet another application of geometric constraint solving is in molecular conformation in chemistry and biology. This entails positioning atoms, represented by

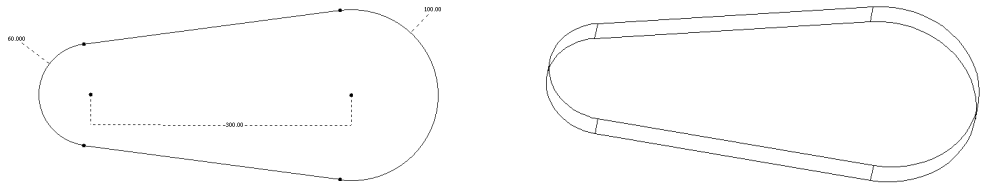

Figure 1: A two-dimensional constraint problem is solved and extruded to obtain a three-dimensional object. Not shown are the tangency constraints between the line segments and the arcs.

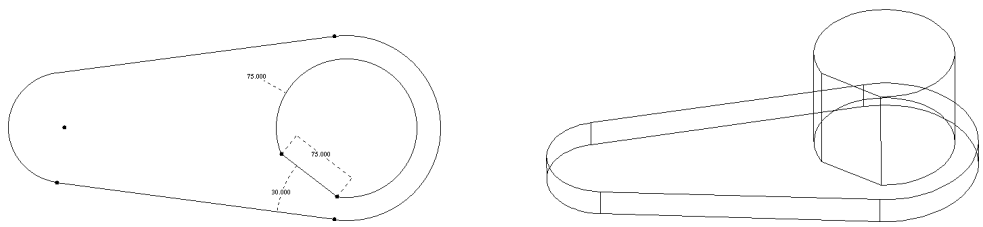

Figure 2: A feature is added using the profile sketcher and a further extrusion. An additional constraint not shown in the diagram is that the two arcs on the right are concentric.

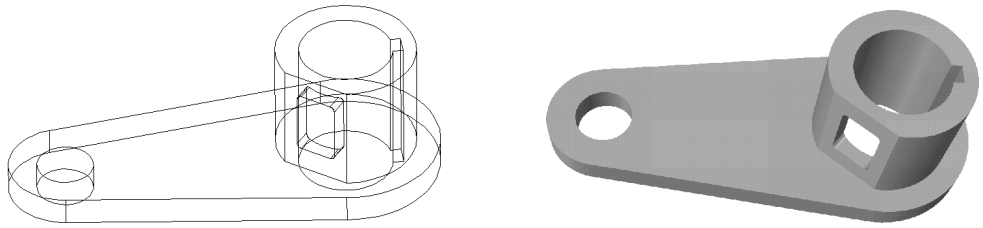

Figure 3: Final control arm

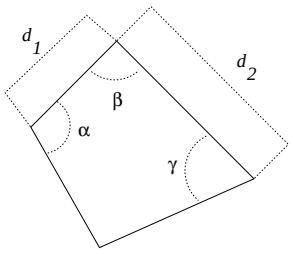

Figure 4: A well-constrained sketch for a generic solver

points in three-space, so that they satisfy certain distance relationships. Clearly, this involves solving a three-dimensional geometric constraint problem.

As these examples of applications demonstrate, there is a wide variety of uses for geometric constraint solving in two and three dimensions. In light of the diverse applicability of the problem, we have three objectives in this paper. First, we provide a brief overview of the basic approaches to geometric constraint solving. Second, we review a specific solution to constraint solving for two-dimensional geometric problems. Finally, we present developing work in extending the solution technique for the two-dimensional problem to geometric constraint solving for elements in three-space.

2. Basic Approaches to Constraint Solving

Beginning with a set of geometric elements and certain constraints between the elements, there are two basic strategies for solving the problem. The first, an *instance solver*, immediately uses the explicit values of the given constraints to determine the possible geometric configurations which satisfy the constraints. The second, a *generic solver*, determines whether the given geometric elements can be placed using the given constraints, independent of the values which are assigned to the constraints. That is, the constraints have a symbolic rather than numerical value. The determination of specific placement of the geometric elements in a generic solver takes place only after a decision has been made about whether or not the problem is generically well-constrained.

As an example of the two different approaches, consider the sketch of Figure 4. Here the constraints are given symbolically, rather than with actual values. A generic solver would be able to report that the configuration is well-constrained, and would be able to determine a method for constructing the possible configurations without needing the actual values of the constraints. An instance solver requires that the values of the constraints be given before it makes any solution determination. Generic solvers are often more elegant and more efficient than instance solvers. They also allow more flexibility in the choice of the underlying method applied to determine the actual positions of the geometric elements. However, generic solvers usually are not able to handle the case of overconstrained but consistent

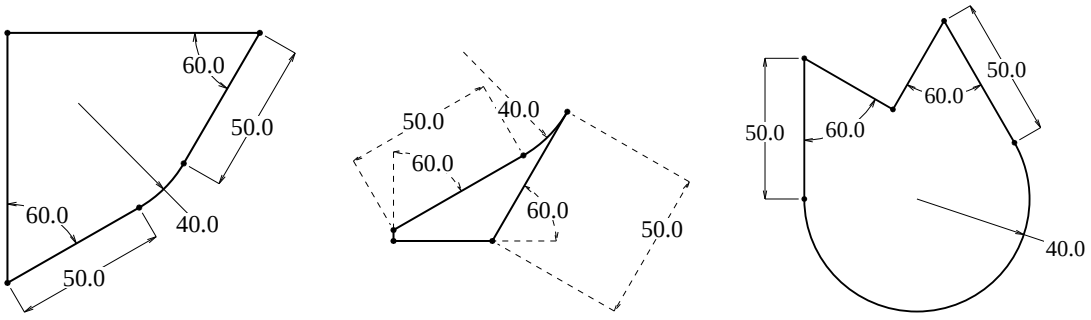

Figure 5: A well-constrained sketch may yield fundamentally different solutions.

systems, while most instance solvers can find the solution. In the example above, a generic solver may be able to determine that there is a solution, yet may not be able to construct it, in the special case that the values of the constraints force the configuration to be an (overconstrained) triangle.

An underlying principle fundamental to most constraint solvers is the fact that the position of the geometric elements can be expressed as (nonlinear) algebraic equations, with the constraints as parameters in the equations. This means that a well-constrained profile may have an exponential number of distinct solutions, dependent on the number of geometric elements. For example, Figure 5 shows three possible solutions of a well-constrained profile that differ in the interpretation of the function of the arc, the tangency type, and the right angle constraint between two segments. From the user's perspective, only one solution will be correct.

Constraint solvers select what they consider the intended solution by deducing certain topological and metric properties from the user's sketch of the profile. The deductions are based on a few heuristic rules that succeed under normal circumstances with high probability. These rules are appropriate when the user sketch is, in a technical sense, "close" to the intended solution. This may or may not be the case, and is often not the case when the dimensional constraints of a profile are changed in value, a common occurrence in redesign.

Surprisingly, most solvers have almost no provisions that would allow the user to select a different solution if the solver's heuristics fail. Developing effective paradigms for redirecting a solver interactively is an important problem, and is addressed in papers by Hoffmann and Fudos[4,5].

In the following sections, we consider four different methods for constraint solving, with special emphasis on the graph-based method, the approach we take in our solver, and which we describe in greater detail in Section 3.

2.1. Numerical Algebraic Computation

Numerical constraint solvers function by first translating the constraints into a system of algebraic equations. This system is then solved using an iterative

technique such as the Newton-Raphson method. Clearly, numerical methods are an example of instance solvers. A positive feature of this approach is that it is able to handle overconstrained but consistent problems which other techniques may not be able to solve, assuming convergence. Moreover, the solvers are very general. For this reason, many constraint solvers fall back on iterative techniques when the native method is not sufficient to solve a given configuration.

However, there are some serious drawbacks with the numerical approach. First is the problem that of the potentially exponential number of solutions, iterative methods can produce only a single solution. Also, the solution to which it converges depends strongly on the initial configuration. Furthermore, because of the multiple solutions and the large number of parameters, the constraint solving problem is often ill-conditioned, making convergence difficult or impossible.

2.2. Symbolic Algebraic Computation

Once again, the constraints are formulated as a system of algebraic equations. However, instead of applying numerical techniques to determine a solution, general symbolic computations are undertaken to find the solution to the system of equations. Methods such as Gröbner basis[3] or Wu-Ritt[10] techniques can be applied to find symbolic expressions for the solutions. This approach is an instance solver if numerical coefficients are used in the system of equations. However, if the system can be solved with symbolic coefficients, a generic solution to the constraint system is found. The generic solution can be evaluated with specific constraint values to find the actual physical configurations possible for the given constraint problem.

One potential problem with this method is that certain equations in the basis may be algebraically dependent on one another when evaluated with specific constraints values. Thus at the generic solver level, the solver may determine that a solution exists, yet it will not be able to find any of the specific configurations satisfying the constraints. A further handicap of this method is that solving symbolic systems of equations can be extremely compute-intensive. For this reason, restrictions are often placed on the types of geometric entities allowed, as well as the types of constraints between them which may be specified.

2.3. Logical Inference and Term Rewriting

This approach applies general logical reasoning techniques to the geometric problem of constraint solving. This approach has been taken by Aldefeld[1] and Bruderlin[2], among others. As an example of this method, consider the system described by Bruderlin. Geometric entities are restricted to points, lines, vectors, and triangles, and the constraints allowed are distances between points, angles between lines, or two angles of a triangle. These geometries and constraints are incorporated into a set of predicates for the system. A set of allowable congruence relationships for the geometries used are established, and then rules of Euclidean geometry for ruler and compass constructions are applied. These rules are set up

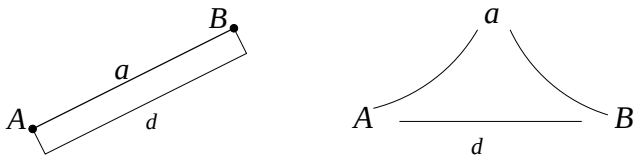

Figure 6: A set of geometric elements with constraints, and the corresponding constraint graph; d denotes a distance constraint.

in Prolog, and Prolog rewrite-rules are used to solve the system. The result is a construction technique for solving the input constraint system. All possible physical solutions can be found using the Prolog backtracking mechanism.

While this approach has the potential to be a generic solver, as implemented the system of Bruderlin is an instance solver, since the predicates and rules use the actual constraint values throughout the deductive process. The major advantage of this method is that it avoids translation of the system into complex algebraic equations. The limitations are that only constructive geometries can be handled, and that the method is not very efficient for large systems of constraints. These disadvantages are common among this type of solver, and hence they are not often applied in commercial constraint solvers.

2.4. Graph-based Construction Sequences

Graph-based algorithms for solving geometric constraint problems have two phases, the first an analysis phase and the second a construction phase. The graph-based approach begins by first constructing a graph representation of the problem. Each node in the graph represents a single geometric element, so that a line segment a of length d delimited by two points A and B would have three nodes. An edge between two geometric entities indicates a constraint between the elements. The type of constraint is indicated by a label on the edge. This simple example is shown in Figure 6, where edges which assert incidence are unlabeled in the graph.

Once a constraint graph has been obtained from the given geometric entities and the constraints, the graph is analyzed to determine whether the problem is well-constrained. If the graph is well-constrained, this phase also determines a sequence of steps for solving the problem. The second phase of the graph method takes the construction sequence determined from the first phase and performs the necessary construction steps to actually place the geometric elements. Since the first phase does not depend on the values of the constraint but only on the number and type of constraints between the geometric elements, this is a generic method of constraint solving. The actual values of the constraints only come into play in the second phase when the construction steps are carried out.

There are a variety of ways to handle the analysis phase of the graph-based method.[12,13] One approach is to look for a sequence of construction steps such that the next construction step depends only on previously placed elements. Not all

configurations can be handled in this way, however. A different approach looks for collections of geometric elements whose members can be placed with respect to one another based on constraints between them. These collections are then placed relative to one another, thus forming new, larger collections of elements, until all constraints have been processed and the locations of all the elements are known.

The approach to constraint solving which we have implemented in two dimensions, and which we are developing in three dimensions, is a graph-based technique which uses the recursive analysis phase just sketched. In the next section we provide more details of our two-dimensional constraint solver, as a basis for our discussion of the extension of the method to three-dimensional constraint solving.

3. Two-Dimensional Constraint Solving

Our approach to geometric constraint solving is a recursive analysis, graph-based method. This approach is favored for two reasons. First, it allows determination of whether the problem is well-constrained or not in quadratic timein the worst case. Second, it decouples the constraint solving problem into groups of smaller systems of equations which can be solved independently and then merged, rather than framing the problem as a single, large algebraic system to be solved.

In this section, we provide more in-depth explanation of the method as implemented for the two-dimensional problem, upon which our extension to the three-dimensional problem is based, by summarizing recent work of Bouma, Cai, Fudos, Paige and Hoffmann. Complete details about the two-dimensional constraint solver and references to further works can be found in.[4,5]

3.1. Geometric Entities Considered

The geometric elements considered in the two-dimensional constraint solver are points, lines and circles of fixed radius. A point is represented by two coordinates (p_x, p_y). A line is represented by a signed, unit normal and the distance of the line from the origin, (n_x, n_y, d). Equationally, the line can be represented as the set of points (x, y) satisfying $n_x x + n_y y - d = 0$ and $n_x^2 + n_y^2 = 1$. A circle is represented by its center (c_x, c_y) and its radius c_r. For simplification purposes, we require that the radius of a circle be fixed, that is, the radius cannot be varied to satisfy constraints.

The constraints between these geometric entities are incidence of two entities, distance between two points, distance between a point and a line, distance between two parallel lines, angle between two lines, tangency between a circle and a line, and concentricity of circles. However, because the circles are restricted to having fixed radius, they can be treated as points by transforming the constraints in which they are involved into distance and incidence constraints of their center points only. In fact, all constraints can be transformed into distance and angle constraints only, which greatly simplifies the placement problem.

Since the problem is reduced to placing points and lines, any geometric element is fixed (up to finitely many positions) by knowing its relationship to two

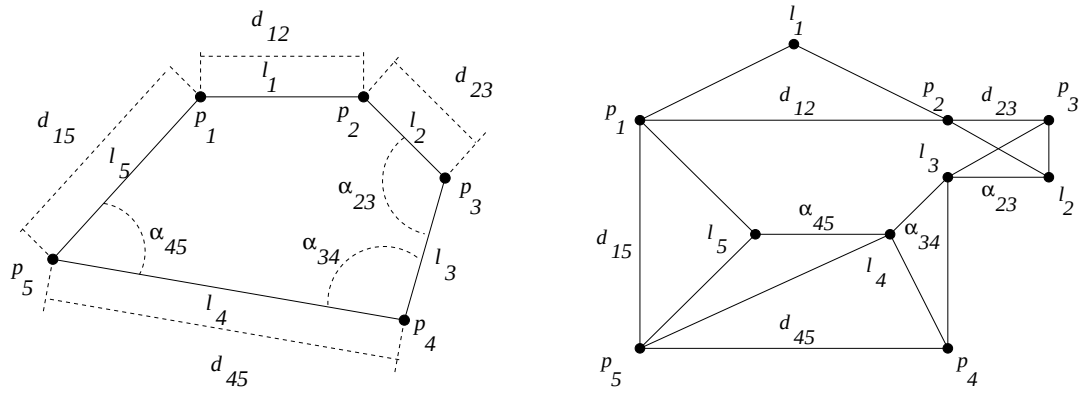

Figure 7: A well-constrained sketch and its constraint graph

other previously fixed objects. This fact plays an important role in the analysis phase of the algorithm, which we describe in the next section.

3.2. Initial Cluster Formation

The first phase of the graph-based method for constraint solving involves analysis of the constraint graph. This analysis determines if the problem is generically well-constrained or not, and it determines a sequence of steps for placing the geometric elements if the problem is well-constrained. The basic idea, as described above, is to build up collections, or *clusters* of geometric elements which can be placed relative to one another, and then to merge these clusters into larger collections using rigid body transformations.

Cluster formation begins by selecting any two nodes of the graph which are connected by a constraint edge. These two entities can be placed in some generic position, depending on the type of geometries and the type of constraint between them. These two entities are then considered *known*. The cluster is then made as large as possible by adding to the cluster any node which is connected by a constraint edge to exactly two nodes already in the cluster. There must be at least two known nodes to which the unknown node is related because each of the geometric elements has two degrees of freedom. There cannot be more than two, because otherwise the problem is overconstrained.

When no more nodes in the graph can be added to the cluster, the cluster is considered complete. All constraint edges used in forming the cluster are deleted from original graph, and a search for a new cluster is carried out in the subgraph. This process continues until there are no more constraints from which to make any clusters.

For example, consider the sketch on the left in Figure 7. Its constraint graph is shown on the right of the figure, where incidence is shown by unlabeled edges. If we start the first cluster with p_1 and l_1, we can add p_2 to the cluster, and then can go no further, since no other node in the graph is connected to two nodes of

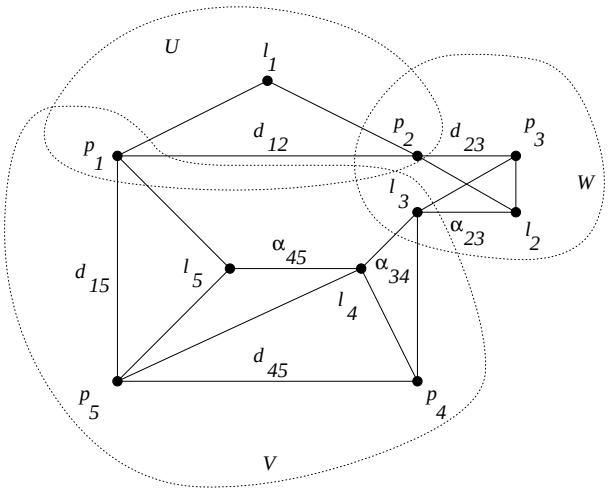

Figure 8: The constraint graph forms three clusters.

the cluster. We delete the three constraint edges used to form this cluster, and look for a new cluster, perhaps beginning this time with p_2 and l_2. To this new cluster we can add p_3, and subsequently l_3, since it is connected to l_2 and p_3. This cluster is now complete. We begin our third cluster with p_4 and l_3, and add l_4, p_5, and p_1, in that order. All constraint edges have now been used, so cluster formation is complete. The constraint graph is shown again in Figure 8, with the three clusters indicated.

Note that clusters may have nodes in common; in fact, that property is essential for the next step of the analysis phase. In order for the problem to be well-constrained, the clusters must be able to be merged together in some way so that a single structure results. Geometrically, this amounts to using rigid body transformations to bring the clusters into correct relationship with respect to one another.

Three clusters, each of which share exactly one node with each of the other two clusters, can be brought into alignment with one another using the shared elements. In our example, the points p_1 and p_2 and the line l_3 are shared in this way between the clusters. We can compute the distance from p_1 to l_3 within cluster V and the distance from p_2 to l_3 within cluster W. The distance between p_1 and p_2 is already known, so these three elements can be placed relative to one another, thus merging the three clusters into one larger cluster. If other clusters have two elements in common with the new cluster, they can be merged into it as well. When the merged cluster can be grown no longer, the clusters are searched for another set of three clusters which can be merged.

This process continues until all the clusters have been merged into a single cluster, or until no more clusters can be merged. If there is a single cluster at the end of the merging stage, the problem is well-constrained. In that case, the steps for constructing the configuration are detailed by the order of the cluster formation.

If multiple clusters are obtained, then the algorithm cannot solve the constraint problem. In that case, the problem may be well-constrained but requires coonstruction steps the algorithm cannot perform, or the problem is not well-constrained. A complete theoretical characterization of generically well-constrained point sets with distance constraints exists.[14] It leads to a nondeterministic algorithm, and no variant is known that achieves efficient running times. Consequently, efficient constraint solving algorithms require restricting the class of constraint problems.

An important point about the cluster formation process is that it is not unique. Any two nodes with a constraint between them can be chosen to begin a cluster, and if more than one node could be added to a cluster at a given step, any of them can be selected and added. However, for a well-constrained problem, it has been shown that no matter what clusters are formed, the final geometric solutions determined by the order of construction from the clusters are congruent[6].

The cluster formation phase of the solution does not do any actual placement of geometric elements. Rather, its function is to analyze the structure of the relationships between the geometric elements based on the constraints between them. The placement of the elements itself is done in the second phase of the solution. In the next section, we describe geometrically how to find the position of an unknown geometric element, given its relationship to two known elements.

3.3. Basic Construction Steps

The placement of one geometric element relative to two others is accomplished by solving small systems of algebraic equations. Because of the restriction on geometries and constraints, these equations have degree at most two. We use the following notation throughout the discussion. The Euclidean distance norm of a vector $v = (v_x, v_y)$ is denoted $\| v \| = \sqrt{v_x^2 + v_y^2}$. Let p_1 and p_2 be two points and $l_1(n_1, r_1)$ and $l_2(n_2, r_2)$ be two lines, where n_i is the unit normal of l_i, and r_i is the signed distance from l_i to the origin. We then can write algebraic equations for the geometric constraints between two entities in the following manner:

- The distance between the two points p_1 and p_2 is d_{12}:

$$\| p_1 - p_2 \| = d_{12}$$

- The signed distance between the point p_1 and the line l_2 is d_{12}:

$$p_1 \cdot n_2 = r_2 + d_{12}$$

- The signed angle between the two lines l_1 and l_2 is α_{12}:

$$n_2 = (n_x \cos \alpha_{12} - n_y \sin \alpha_{12},\ n_y \cos \alpha_{12} + n_x \sin \alpha_{12})$$

  where $n_1 = (n_x, n_y)$

The sign of the distance and angle measures are determined from the way in which the user inputs the data. For example, a line segment has an orientation in the direction from the first end point to the second. When a point is input and a distance to the line segment assigned as a constraint, the sign is determined from the side of the line on which the point is initially placed by the user. Again for angle constraints, the orientations of the two line segments are used to determine which region between the segments should be affected by the constraint. Having this algebraic understanding of the geometry of the constraints allows us to evaluate the following cases:

Case 1 : $(p_1, p_2) \Rightarrow p_3$

The point p_3 is to be constructed from two given points p_1 and p_2, where p_i has distance d_{i3} from p_3. The coordinates of point p_3 must satisfy two quadratic equations arising from the two distance constraints. Geometrically this corresponds to intersecting two circles, one centered at p_1 of radius d_{13}, the other centered at p_2 with radius d_{23}, as shown in Figure 9. The circles either intersect transversally, resulting in two solutions, tangentially, resulting in one solution, or not at all, resulting in no real solutions. In the figure, a situation with two solutions is shown, with both possible solutions labeled p_3. Notice that we can assume point p_1 is at the origin and that p_2 lies on the positive x-axis a distance d_{12} from p_1, since any other valid configuration could be obtained as a rigid motion of this configuration.

Case 2 : $(p_1, l_2) \Rightarrow p_3$

The point p_3 is to be constructed from the given point p_1 and the given line l_2, where p_3 has distance d_{13} from p_1 and signed distance d_{23} from l_2. Without loss of generality, assume that l_2 coincides with the x-axis and that p_1 lies distance d_{12} from the origin along the positive y-axis, as shown in Figure 10. Then the point p_3 must lie on a circle centered at p_1 of radius d_{13} and must lie on a line parallel to l_2 and distance d_{23} away from l_2. Since the distance between p_3 and l_2 is signed, there is exactly one line satisfying these conditions. The intersection of the circle and the line provide the possible locations for p_3. As before, there can be two, one, or no solutions depending on the type of intersection. Algebraically, the coordinates of p_3 can be found by solving a pair of equations, one of which is linear and one of which is quadratic.

Case 3 : $(l_1, l_2) \Rightarrow p_3$

The point p_3 is to be constructed from the given lines l_1 and l_2, where p_3 has signed distance d_{i3} from l_i. Assume that l_1 coincides with the x-axis and that the angle between l_1 and l_2 is α_{12}, as shown in Figure 11. Since the distances between p_3 and the lines are signed, p_3 must be the intersection of two lines, one offset parallel to l_1 by d_{13} and the other offset parallel to l_2 by d_{23}. If we ignore the case of parallel lines, there is exactly one solution in this case. Algebraically, this is equivalent to solving a pair of linear equations in the coordinates of p_3.

Case 4 : $(p_1, p_2) \Rightarrow l_3$

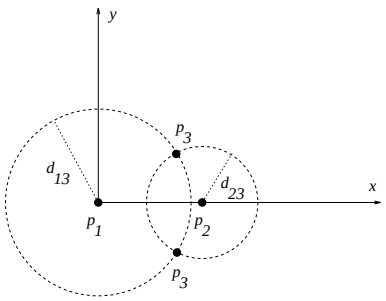

Figure 9: Placing a point relative to two known points

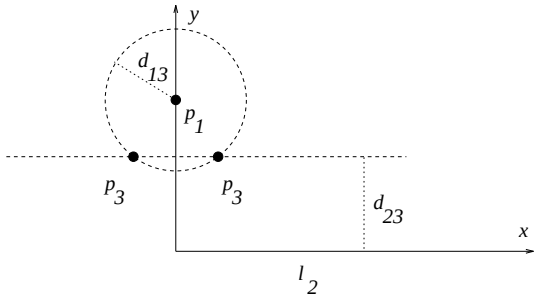

Figure 10: Placing a point relative to a known point and line

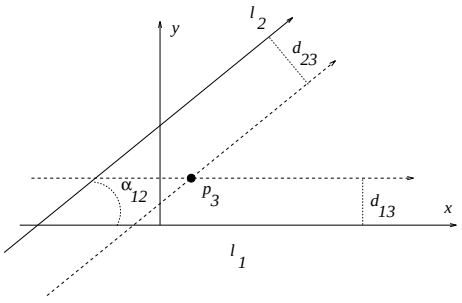

Figure 11: Placing a point relative to two known lines

Figure 12: A graph transformation in this case would limit the solutions to the problem.

Case 5 : $(p_1, l_2) \Rightarrow l_3$

These two cases are identical to cases 2 and 3, respectively, since pairwise constraints exist between all three entities. We can therefore easily transform these cases to the previous cases by changing the roles of the geometric entities from fixed to unfixed, and *vice versa*, as necessary.

Case 6 : $(l_1, l_2) \Rightarrow l_3$

In this case, the angle between each pair of lines must be given as constraints. Since the three lines must define a triangle, the three angles will be either consistent, and determine an infinite family of triangles, or redundant, and have no solutions. We consider this case to be overconstrained because the constraints are not independent.

3.4. Graph Transformations

What we have presented above are the basic elements of a two-dimensional constraint solver. There are many extensions possible, some of which we have already considered, and others which remain to be explored. One extension which has proved useful is graph transformations which may increase the constraint information available, thus allowing alternative clusters, possibly with easier constructions. For example, if two angle constraints α and β are given between three lines, a third angle constraint of $180° - \alpha - \beta$ can be imposed between the pair of lines not involved together in the first two constraints.

Graph transformations must be applied judiciously, however, as some transformations may limit the generality of the solution. Consider the example from[5] shown in Figure 12. If the incidences shown are required, and in addition point A is constrained to lie on line a, and point C on line c, the constraint graph is as shown on the right in the figure. This implies that either a and c are coincident, or A and C are. However, adding either of these incidence relationships to the constraint graph would eliminate the other possibility, and therewith some of the solutions to the original problem.

4. Three-Dimensional Constraint Solving

Our basic approach to constraint solving in three-dimensional space is analogous to that of constraint solving in two-dimensional space. We begin by construct-

ing a graph which specifies the entire constraint system, with the nodes representing the geometric elements and an edge between two nodes describing a constraint between the two geometric entities. Based on the information encoded in the graph, the geometric elements are grouped into clusters which are placements of a subset of the elements relative to one another. As we shall see below, forming a cluster in the three-dimensional case requires as a starting configuration three geometric entities which are mutually constrained. Thus it is possible that not all constraints and geometric elements are used in the cluster formation stage. These remaining nodes and edges form *degenerate clusters*. The clusters and degenerate clusters are then combined using a recursive technique, resulting in a valid placement for all the geometric elements. As before, there are in general multiple solutions to a given well-constrained problem.

4.1. Geometric Entities Considered

In two-dimensional constraint solving, the geometric entities which were considered were points, lines, and circles of fixed radius. All three of these types of elements need two constraints to be completely fixed. This property is fundamental to the cluster formation and combination method used in the two-dimensional case. The obvious generalization of geometric entities for three-dimensional constraint solving would be points, lines, planes, and spheres. Note, however, that points, planes, and spheres of fixed radius all require exactly three constraints to be placed, while lines require four constraints. In order to keep the construction of clusters as simple as possible, we do not consider lines at this time, and we further simplify matters by eliminating spheres from our current consideration. Thus the geometric entities which are allowed are points and planes. A point is represented by its Cartesian coordinates, $p : (p_x, p_y, p_z)$. A plane P is represented by the direction cosines of the unit normal and the signed distance from the origin, $P : (n_x, n_y, n_z, d)$, where $n_x^2 + n_y^2 + n_z^2 = 1$. The constraints allowed are distance between two points, distance between a point and a plane, and angle between two planes. Note that fixed-radius spheres can be used nevertheless as geometric primitives because constraints on them can be translated into equivalent constraints on their centers.

4.2. Initial Cluster Formation

The first step of the construction is to form clusters of geometric elements which are placed with respect to one another. Because each geometric element has three degrees of freedom, placing a new element requires that it be constrained by three known elements. Thus to begin a cluster, a set of three pairwise constrained nodes is necessary. These three geometric elements are placed into a standard position and the resulting configuration is fixed up to a rigid motion in space. Subsequently, a node is added to the cluster if it is incident to three nodes already in the cluster.

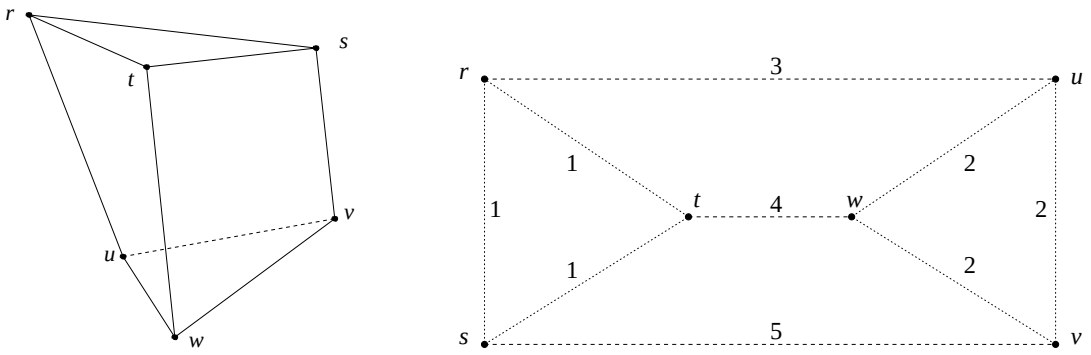

Figure 13: A three-dimensional figure and its constraint graph.

When there are no further nodes to be added to the cluster, the edges belonging to the cluster, *i.e.* the constraints between nodes in the cluster, are deleted from the original graph. Cluster formation is then applied recursively to this subgraph. That is, the subgraph is searched for three nodes which are pairwise constrained to start a new cluster, the cluster is grown as far as possible, and then the edges of the cluster are deleted from the subgraph, resulting in a smaller subgraph. This process of forming a cluster and subsequently deleting the cluster's edges from the remaining constraint graph is carried out as long as possible. Because the origination of a cluster requires three pairwise constrained elements, there may eventually be unused constraints in the remaining subgraph, yet no new cluster can be started. When this point is reached, any remaining constraint and its two incident nodes forms a degenerate cluster.

For example, consider the problem of placing the six vertices of the three-dimensional object shown on the left in Figure 13, if the lengths of the edges between the vertices are the constraints. We begin the first cluster using nodes r, s, and t. No other node of the graph is incident to three elements of this cluster, so cluster 1 is completed. Its edges are labeled 1 in the constraint graph shown on the right in Figure 13, and displayed in dotted line. We then form a second cluster with nodes u, v, and w. Again, this is a complete cluster, labeled 2 in the figure. Now none of the remaining constraints are part of an initial cluster, so they must each produce a degenerate cluster, labeled 3, 4, and 5 in the figure, and displayed in dashed line. Thus cluster formation is completed with two full clusters and three degenerate clusters.

In order to build these clusters, we need to be able to place a geometric element from three known elements. In the next section, we present a case-by-case analysis of how these placements are executed. Then, with clusters formed, the final configuration will be determined by recursively merging the clusters. The issues and techniques involved in that process are detailed in Section 4.4.

Throughout this section, points are denoted p_i and planes $P_i(n_i, r_i)$, or P_i for short, where n_i is the unit normal of P_i, and r_i is the signed distance from P_i to the origin. We assume throughout that distances between points are non-zero. The

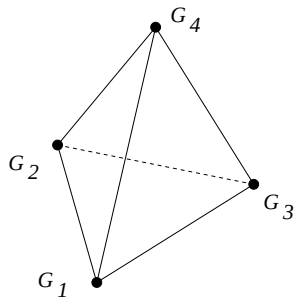

Figure 14: Cluster growth entails a tetrahedral structure in the constraint graph.

distance between a point and a plane, however, may be zero, meaning the point lies on the plane.

4.3. Basic Construction Steps

As discussed in the previous section, the three degrees of freedom of the points and planes under consideration means that the placement of a new point or plane can be made relative to three already placed points and planes. The structure of a region of the constraint graph containing an element which is being added to a cluster is tetrahedral, for the original three elements must be pairwise constrained, forming the base of the tetrahedron, and the new element must have a constraint edge between itself and each of the first three nodes, forming the apex of the tetrahedron. This subgraph structure is shown in Figure 14, where each geometric element G_i, $i = 1, ..., 4$ is either a point or a plane. Because of the symmetry of the tetrahedron, any three elements can be selected as the base, or known elements, and the remaining apex then is considered to be the unknown element. This allows flexibility in choosing the easiest method for placing the next element and reduces the number of placement problems which must be considered.

This interchangability of known and unknown elements may be possible directly only in the early stages of forming a cluster. For example, consider the following construction: The beginning of the cluster is a triangle with elements (G_1, G_2, G_3), to which the an additional vertex is added to form a tetrahedron (G_1, G_2, G_3, G_4). Three more elements G_5, G_6, and G_7 are added as three tetrahedrons whose bases are faces of the original tetrahedron. Then a new element G_8 could be added to the cluster by having constraints between G_8 and each of G_5, G_6, and G_7. However, there need be no direct relationship given between elements G_5, G_6, and G_7, so that a choice of which triple of elements to begin with for adding G_8 is not possible. This problem can be overcome by using the information implicit in the cluster formed so far to determine the relationships between G_5, G_6, and G_7. For example, if these three elements were points, the pairwise distances between them could be computed from within the cluster. Subsequently, adding G_8 could be handled by choosing any three elements of the G_5, G_6, G_7, and G_8

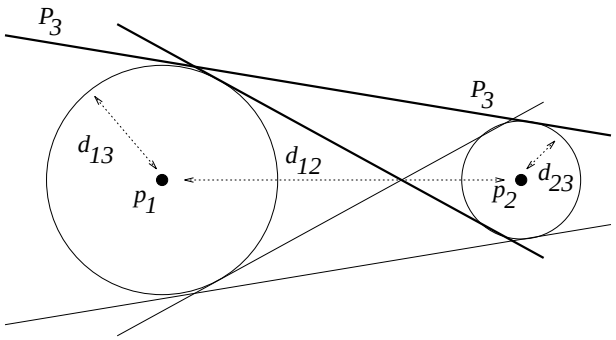

Figure 15: There are two generic configurations for two points and one plane.

as the known elements and placing the remaining element with respect to the chosen three. The tetrahedron constructed then must be brought into line with the previously constructed component of the cluster via a rigid body transformation.

Just as in the procedure of placement of geometric entities in the two-dimensional case, the generic positioning of the initial elements depends on the orientation of the geometric elements determined from the user input. The measurement of a signed distance from a plane is made in the direction of the plane normal if the sign is positive, and in the opposite direction if it is negative. The region of measure of an angle is determined by the mutual orientation of the two planes in the region. By incorporating this orientation information, a generic initial position is determined from each three element configuration.

For three points p_1, p_2, and p_3 with pairwise distance constraint d_{ij} between p_i and p_j, $i < j$, no choice of generic position is necessary, since only a single generic position exists. We place p_1 at the origin, p_2 distance d_{12} along the positive x-axis, and p_3 at either intersection point of the xy-plane and the spheres centered at p_1 and p_2 with radii d_{13} and d_{23}, respectively. This is essentially the point placement routine for two-dimensions when two points are known, but because we are in $\Re^3$, the two solutions of $\Re^2$ are equivalent up to a rigid body motion.

For two points p_1 and p_2, and a plane P_3 with pairwise distance constraint d_{ij} between entities i and j, $i < j$, two generic positions exist. The plane can be placed as the xy-plane and p_1 at $(0, 0, d_{13})$. The second point is then placed at $(l, 0, d_{23})$, where $l = \sqrt{d_{12}^2 - (d_{13} - d_{23})^2}$. The sign of d_{23} will determine whether p_1 and p_2 lie on the same side of P_3 or on opposite sides.

Geometrically, the distance constraint between the plane and the two points implies that the plane must be tangent to the spheres centered at p_1 and p_2 with radii d_{13} and d_{23}, respectively. The envelope of all such planes consists of two cones. The normals to the cones are the possible normals to the desired plane. If the points are input oriented oppositely with respect to the plane, the generic position is given by a plane tangent to the cone which crosses between the two spheres. If the points are input oriented the same with respect to the plane, the generic position is given by a plane tangent to the cone exterior to the two spheres. In Figure 15 a projection

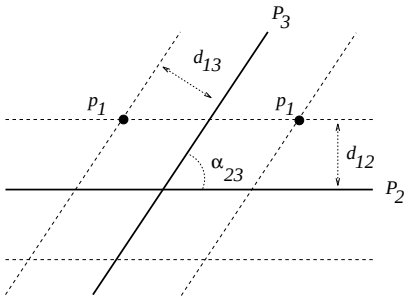

Figure 16: Two planes and a point have two generic positions.

of the situation is shown, with the two possible locations of the planes relative to the points highlighted. As before, any other potential solution can be found by a rigid body motion of the relevant one of these two generic configurations.

Again, for one point p_1 and two planes P_2 and P_3, with distance constraints d_{12} and d_{13} between the point and the two planes, respectively, and angle constraint $0 < \alpha_{23} < 180°$ between the two planes, two generic position are possible, depending on the input orientations. Plane P_2 can be placed as the xy-plane, and plane P_3 positioned so that the intersection of P_2 and P_3 coincides with the y-axis. Then if the input orientation of p_1 with respect to P_2 and P_3 coincides with the input orientation of the angle between P_2 and P_3, p_1 will be placed in the region containing α_{23}. Otherwise, p_1 will be placed in the complementary region. In either case, p_1 can be placed at the intersection of the two signed offset planes, with $y = 0$. These configurations are shown projected into the xz-plane in Figure 16. The four planes shown in dashed line are the four possible offset planes, depending on orientation of the distance between p_1 and the given planes. The two intersections below P_2 relate to other combinations of orientations of the signed distances and angles and can be obtained by a rigid body motion of one of the shown cases, thus are not considered generic positions.

Finally, when three planes are given, and the pairwise angles α_{ij} between them, two generic positions exist. In this case, P_1 is placed as the xy-plane and P_2 is placed so that the intersection with P_1 is the y-axis. Then P_3 can first be placed with respect to P_1 so that the angle between P_1 and P_3 is α_{13}, and so that the intersection between P_1 and P_3 is the x-axis. This makes the intersection of the three planes coincide with the origin. This first placement of P_3 results in a plane with normal $(0, n_{32}, n_{33})$, and the four possible combinations of signs of n_{32} and n_{33} yield two distinct positions for P_3. The appropriate plane is chosen depending on the input orientation of the planes. Once selected, P_3 can be rotated about the z-axis until the angle between P_2 and P_3 is α_{23}.

This derivation of the two generic positions can also be obtained algebraically: Let the normal of P_1 be $n_1 = (0, 0, 1)$, and the normal of P_2 be $n_2 = (n_{21}, 0, n_{23})$. These are fixed by the input orientation of P_1 and P_2. Let the normal of P_3 be

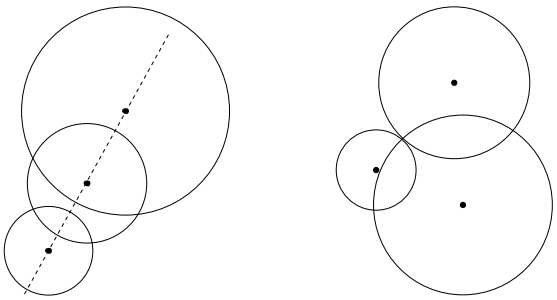

Figure 17: Degenerate cases for placing a point relative to three known points.

$n_3 = (n_{31}, n_{32}, n_{33})$. Then n_3 must satisfy

$$\begin{aligned} n_{33} &= \cos\alpha_{13} \\ n_{31}n_{21} + n_{33}n_{23} &= \cos\alpha_{23} \\ n_{31}^2 + n_{32}^2 + n_{33}^2 &= 1 \end{aligned}$$

Since n_{33} and n_{31} are completely determined by the first two equations, there are two different values fo n_{32} from the third equation corresponding to the two different orientations of P_3.

We now assume that three known elements have been put into the single possible generic position, based on their signed constraints. Based on these generic positions, we can now proceed to place a fourth element with respect to three known elements.

Case 1 : $(p_1, p_2, p_3) \Rightarrow p_4$

The point p_4 is to be constructed from three given points p_1, p_2 and p_3, where p_k has distance d_{k4} from p_4. The coordinates of point p_4 must satisfy three quadratic equations arising from the distance constraints. Geometrically this corresponds to intersecting three spheres. The intersection of two spheres is a circle, and is equal to the intersection of a sphere with a certain plane. If the two spheres have the equations

$$\begin{aligned} (x-u_1)^2 + (y-v_1)^2 + (z-w_1)^2 &= d_{14}^2 \\ (x-u_2)^2 + (y-v_2)^2 + (z-w_2)^2 &= d_{24}^2 \end{aligned}$$

then this plane is

$$2(u_2-u_1)x + 2(v_2-v_1)y + 2(w_2-w_1)z + U = d_{24}^2 - d_{14}^2$$

where $U = u_1^2 + v_1^2 + w_1^2 - u_2^2 - v_2^2 - w_2^2$. Thus, we can intersect two planes and a sphere instead. There will be two solutions in general.

The degenerate situations in this case are shown in Figure 17. On the left of the figure is a projection of the case where the three points are collinear. In this case two of the spheres determine a circle on which p_4 must lie, and the third sphere is either redundant or inconsistent. Thus this case either is overconstrained or has no solutions. In the figure is shown a case where there is no solution.

The other degenerate situation, shown in projection on the right in Figure 17, occurs when two of the spheres are tangent. In this case again the third sphere is either redundant or inconsistent. Shown is the case where it is redundant. Note that both degenerate cases can occur simultaneously as well, that is, the three points are collinear and two spheres, or possibly all three, are tangent to each other.

Case 2 : $(p_1, p_2, P_3) \Rightarrow p_4$

The point p_4 is to be constructed from two given points p_1 and p_2, and from the given plane P_3, at respective distances d_{k4} from entity k. The coordinates of point p_4 must satisfy one linear and two quadratic equations which geometrically corresponds to intersecting two spheres and a plane. Note that the plane is the locus of points at signed distance d_{34} from P_3. As before we intersect instead two planes and a sphere, obtaining two solutions in general.

There is only one degenerate case: If the two spheres meet tangentially, then as in Case 1 above, the problem is either overconstrained with a single solution, or there is no solution because the constraints are inconsistent.

Case 3 : $(p_1, P_2, P_3) \Rightarrow p_4$

The point p_4 is to be constructed from a given point p_1, and from two given planes P_2 and P_3, at respective distances d_{k4} from entity k. The coordinates of point p_4 must satisfy one quadratic and two linear equations, corresponding to intersecting a sphere and two planes. Because the distances between points and planes are signed, there are two solutions in general. A special case arises if the sphere meets both planes tangentially, as then there must be only a single solution.

This case is degenerate when the two planes are parallel or coincident. If P_2 and P_3 are parallel, then p_1 is only constrained to lie on a plane also parallel to the original two planes. Assuming that the original three entities are consistently constrained, so that such a plane exists, the distance constraints between p_4 and the original three entities are either redundant or inconsistent. For if the distances to P_2 and P_3 are consistent, then they determine a plane on which p_4 must lie, and the intersection of this plane with the sphere centered at p_1 of radius d_{14} determines an entire circle on which p_4 may lie. This occurs because of the redundancy in the distance constraints between P_2 and p_4 and between P_3 and p_4. If P_2 and P_4 are coincident, again the constraints between p_4 and the three given entities determine either a circle of points for p_4 or no points at all.

Case 4 : $(P_1, P_2, P_3) \Rightarrow p_4$

The point p_4 is to be constructed from three given planes P_1, P_2 and P_3, where P_k has distance d_k from p_4. The coordinates of point p_4 must satisfy three linear equations corresponding to the intersection of three planes. There will be one solution in general, because of the orientation of the planes. As in the previous case, degeneracies occur when two or more of the planes are parallel or coincident. Unlike before, however, in this case there are no consistent overconstrained solutions possible.

Case 5 : $(p_1, p_2, p_3) \Rightarrow P_4$

Case 6 : $(p_1, p_2, P_3) \Rightarrow P_4$

Case 7 : $(p_1, P_2, P_3) \Rightarrow P_4$

These three cases can be converted to cases 2, 3, and 4, respectively, by swapping roles of known vs. unknown between appropriate elements, as discussed earlier. As pointed out in that discussion, this role swapping may require a rigid body motion to bring the new elements in line with previously placed elements of the cluster.

Case 8 : $(P_1, P_2, P_3) \Rightarrow P_4$

This case is always an underdetermined situation. Here, the angles from two planes determine the direction of the sought plane (two possible solutions). The third angle constraint is redundant or inconsistent. For consistent constraints, the plane has an additional degree of freedom that remains undetermined.

4.4. Cluster Merging

Once initial clusters have been formed as described above, clusters which share geometric elements can be placed relative to one another. The goal in cluster merging is to combine clusters in such a way as to form a rigid body, unique up to rotation and translation in space. In the two-dimensional setting, the general merging rule is to combine any three clusters each of which shares a geometric element with the other two. In the three-dimensional case, the necessary relationships between clusters is considerably more complicated.

A cluster has in general six degrees of freedom, three rotational and three translational. Exceptions include degenerate clusters such as a plane with an incident point, which has only five degrees of freedom, since one degree of freedom is lost by symmetry. To fix a cluster in space, we must determine how to place certain elements in the cluster with respect to other known clusters based on elements shared between the cluster being placed and the known clusters. However, it is not sufficient to place the cluster based on only one or two elements in the cluster. Fixing a plane alone in the cluster leaves two translational and one rotational degree of freedom, while fixing a point alone leaves three rotational degrees of freedom. Furthermore, fixing any combination of two points or planes in a cluster is also insufficient to fix the cluster itself. Fixing two distinct points or a point and a plane leave one rotational degree of freedom, while fixing two planes in general position leaves one translational degree of freedom. Therefore, three *separate* geometric entities in the cluster must be placed with respect to other known clusters in order to fix the cluster itself. In the case of degenerate clusters, the two geometric elements of the cluster must be shared with another cluster in order to fix the cluster. Note that degenerate clusters can only be fixed up to symmetry, because they only contain two elements.

Now, if two clusters A and B share two geometric elements between them, then the clusters are overconstrained, because the relative position of the shared elements is determined independently in each cluster. Therefore, the three elements in the cluster to be fixed must belong to three *separate* known clusters. For degenerate clusters, this entails that the two elements in the cluster are each shared with a different cluster.

If we restrict our discussion to points and distances between them, visualization of the required combinations of clusters and degenerate clusters needed to obtain a rigid body is greatly simplified. In this consideration of cluster merging, we begin with a specified number of full clusters. Our goal is to determine the minimum number of additional degenerate clusters needed to make the configuration stable, up to rigid motions. We ensure the stability by requiring that the outcome figures are polyhedra with triangular faces. We will begin with no full clusters, and work up to a situation which requires no degenerate clusters. For simplicity of the discussion, we consider full clusters as three points with mutual distance constraints between the points, and degenerate clusters as two points with a distance constraint between them. This is sufficient since placement of only three elements of a full cluster is necessary to place the complete cluster.

Case 1 : 0 full clusters

Six degenerate clusters are necessary to fix the geometry of the points and fulfill the requirements that the two elements of any one degenerate cluster are shared with two different clusters. These degenerate clusters share four points between them, in a tetrahedral formation. Explicitly, the clusters can be given by (p_1, p_2), (p_2, p_3), (p_3, p_1), (p_1, p_4), (p_2, p_4), and (p_3, p_4). If the constraints between these points were all given explicitly, this combination of degenerate clusters would have been formed into a cluster in the earlier phase of cluster formation. However, since constraints can be added to the constraint graph through graph transformations, some of these may have been implicit constraints not present in the graph during initial cluster formation, preventing formation of a full cluster then.

Case 2 : 1 full cluster

Three degenerate clusters are required to fix a single full cluster. Again, there will be four points involved, and a tetrahedron the result. If the full cluster is (p_1, p_2, p_3), the degenerate clusters are (p_1, p_4), (p_2, p_4), and (p_3, p_4). As in the previous case, if the constraints in the degenerate cluster were all given explicitly, then this combination would not occur since the four points would have been combined into the same cluster in initial cluster formation.

Case 3 : 2 full clusters

When we consider the case of two full clusters, there are two subcases to be handled, based on whether the full clusters share an element or not.

Subcase 3.1 : Shared element

If the two clusters share an element, their initial configuration is as seen on the left in Figure 18. If two degenerate clusters are added by adding constraints between pairs of elements in the two clusters, the figure is still not rigid. A third degenerate cluster must be added between the two clusters to make the configuration immobile. The necessary degenerate clusters are shown in the diagram on the right of Figure 18. The first two additional constraints are shown in dashed line, and the third in dotted line. The configuration is a double tetrahedron

Subcase 3.2 : No shared element

If the two clusters do not share an element, then six degenerate clusters are needed to connect the two cluster. This is the case shown in Figure 13, but with only three degenerate clusters. However, these three constraints are insufficient to fix the points with respect to each other. A constraint running diagonally across each of the four-sided faces is needed to make the configuration rigid. For example, constraints between (t, u), (t, v), and (s, u) would make the configuration stable. Structurally, this configuration is an octahedron.

Case 4 : 3 full clusters

There are a variety of combinations possible when none of the clusters shares an element or when two cluster share an element but neither of these two shares an element with the third. These are most easily stabilized by adding enough degenerate constraints to make a triangulated figure. The three cases we consider in more detail are when the three clusters share elements between each other. This can happen in three different ways, which we call the *triangle*, *chain*, and *star* connections. These three connections are shown in Figure 19, with the clusters shaded.

Subcase 4.1 : Triangle

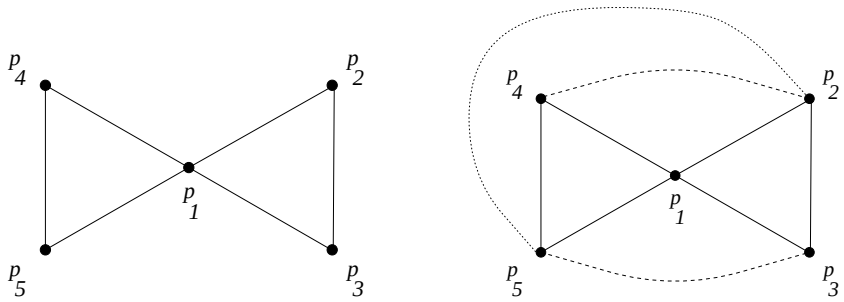

Figure 18: Two full clusters sharing an element need three degenerate clusters to make the figure stable.

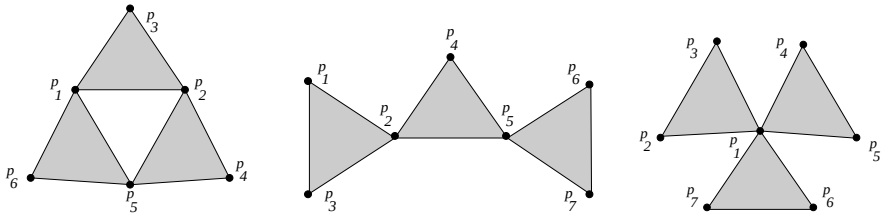

Figure 19: Triangle, chain, and star connections between three full clusters

The triangle initial configuration has six distinct vertices, and can be stabilized by adding three degenerate clusters. Constraints are added pairwise between p_3, p_4, and p_6. This makes an octahedral figure.

Subcase 4.2 : Star

The star in Figure 19 will be stable if the following six degenerate clusters exist: (p_3, p_4), (p_5, p_6), (p_2, p_7), (p_2, p_4), (p_4, p_6), and (p_2, p_6). The configuration can then be built by first placing one of the original clusters, say cluster (p_1, p_6, p_7). Then point p_2 can be placed using information from the full cluster (p_1, p_2, p_3) and the two degenerate clusters (p_2, p_7) and (p_2, p_6). Point p_4 can be placed in a similar fashion. The remaining points p_3 and p_5 can now be fixed using the relationships between them and the five fixed points. The resulting figure is a *decahedron.*

Subcase 4.3 : Chain

The chain configuration can be stabilized by repeated application of Case 3.1 above, requiring six degenerate clusters and resulting in a decahedron. In this case, the three clusters are not inter-related in the way that the triangle and the star are, where each cluster shares an element with both of the other two clusters. Because of the sequential nature of the connection between the clusters, it is not obvious that there is a collection of degenerate clusters which would stabilize the chain without having Case 3.1 as a component of a sequential solution.

Case 5 : 4 full clusters

With four full clusters, it is possible to have elements shared between the clusters so that no degenerate clusters are necessary. This is the case if three cluster meet in the triangular configuration, and the fourth cluster shares one vertex with each of the first three. This is similar to the solution of the case 4.1, except that a full cluster exists to join the first three clusters together, instead of degenerate clusters.

Table 1 summarizes these results, with the subcases for the cases of two and three full clusters denoted by subscripts.

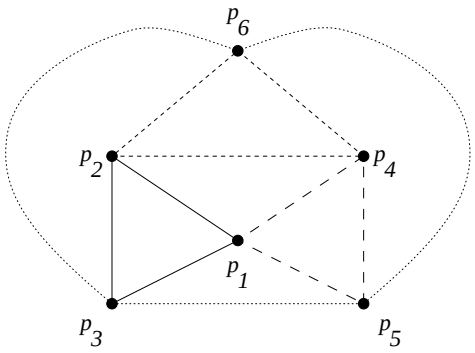

Figure 20: The constraint graph for six points in four clusters.

4.5. Examples

As an example of three-dimensional constraint solving, we consider the case in Table 1 above of four clusters each with exactly three elements. Each element in a given cluster is shared with exactly one of the other three clusters, thus there are six geometric elements to be placed. Note that a solution to this problem also entails a solution to Problems 2_2 and 3_1 of the table. The first example discusses a solution from robotics for the case when all six geometric elements are points. The second example demonstrates a solution for the placement when five elements are points and the sixth is a plane.

Example 1: Six points in four clusters

The four clusters for the problem of six points as the geometric entities are (p_1, p_2, p_3), (p_1, p_4, p_5), (p_2, p_4, p_6), and (p_3, p_5, p_6). Note that this arrangement satisfies the criterion that each element of a given cluster is in exactly one of the other three clusters. The constraints are the distances between each pair of points in a cluster. A graphical representation of these clusters is shown in Figure 20, with the clusters distinguished by the type of line of the constraint edges. Each edge in

Table 1: Degenerate and complete cluster combinations necessary for merging of clusters

Complete	Degenerate	Configuration
0	6	Tetrahedron
1	3	Tetrahedron
2_1	3	Double Tetrahedron
2_2	6	Octahedron
3_1	3	Octahedron
3_2	6	Decahedron
4	0	Octahedron

the graph represents a distance constraint.

Physically, the problem is that of positioning the vertices of an octahedron, given the lengths of the edges of the octahedron. This problem has been extensively explored in robotics as a special case of the Stewart platform problem. A Stewart platform has two platforms, not necessarily planar, connected by six legs of variable length. The problem consists of finding the coordinates of vertices of the upper platform relative to the lower platform based solely on the lengths of the legs. If the two platforms are triangles, as shown in Figure 21, the solution to the Stewart platform problem is a solution to the six point constraint problem, since the lengths of the platforms' edges are also given. The four original clusters are the lower plate and the three triangles with a single vertex on the lower platform and an edge on the upper platform.

A solution to this problem is given by Nanua, *et. al.*[8], where it is shown that there are at most 16 possible configurations, including complex solutions.

Example 2: Five points and one plane in four clusters

The four clusters have the same topological structure as in the previous example, except that in this case one of the points is replaced with a plane, *e.g.* (p_1, p_2, P_3), (p_1, p_4, p_5), (p_2, p_4, p_6), and (P_3, p_5, p_6). Since there is only one plane, the only constraints are again distances between each pair of elements in a cluster. A diagram of this situation is shown in Figure 22. The open circles in P_3 represent the points in P_3 which satisfy the distance constraints between P_3 and the points p_2, p_5, and p_6. Again, the elements of each cluster are connected by the same type of line.

Suppose that the distance between any two geometric elements g_i and g_j, where g is a point or a plane, is given by d_{ij}. If we assume the distances between a point and the plane are signed, then we can modify the problem so that the point p_1 lies in the plane P_3. If we can find a configuration where p_1 lies in P_3 and the respective distances between each of the three points p_2, p_5, and p_6 and P_3 are reduced by the given distance from p_1 to P_3, d_{13} , the actual configuration which satisfies the given constraints can be found by offsetting P_3 in the found configuration by d_{13}.

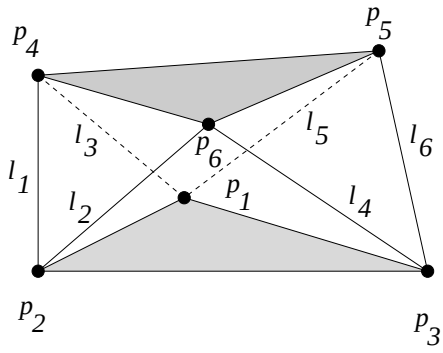

Figure 21: A Stewart platform with triangular platforms

We first position p_1 at the origin, make P_3 coincide with the xy-plane, and place p_2 so that it lies at $(l_1, 0, d_{23})$, where $l_1^2 + d_{23}^2 = d_{12}^2$. The distance constraints between p_1 and p_4 and between p_2 and p_4 force p_4 to lie on a circle C_4 whose center is on $\overline{p_1 p_2}$. The circle lies in a plane whose normal is in the direction of $\overline{p_1 p_2}$. Suppose for a moment that p_2 also lies in P_3. Because the lengths of the edges of the triangle $p_1 p_2 p_4$ are all given, the center $\hat{p}_4 = (h_1, 0, 0)$ of C_4 can be easily computed, as can the radius r_4. The circle C_4 can then be written $(h_1, r_4 \cos u, r_4 \sin u)$. However, since p_2 actually may not lie in the plane, this circle must be rotated about the y-axis by θ, the angle $\overline{p_1 p_2}$ makes with P_3. Thus the circle C_4 on which p_4 must lie is given by

$$C_4 \; : \; (\cos\theta\, h_1 + \sin\theta\, r_4 \sin u,\; r_4 \cos u,\; -\sin\theta\, h_1 + \cos\theta\, r_4 \sin u)$$

Now, p_5 lies in a plane parallel to P_3 and also on a sphere of radius d_{15} centered at p_1. Thus it must lie on the circle C_5 given by

$$C_5 \; : \; (r_5 \cos v, r_5 \sin v,\; d_{35})$$

Similarly p_6 lies in a plane parallel to P_3 and also on a sphere of radius d_{26} centered at p_2. Since the distance from p_1 to the projection of p_2 into P_3 is l_1 (from the earlier positioning of p_2), p_6 must lie on the circle C_6 given by

$$C_6 \; : \; (l_1 + r_6 \cos w, r_6 \sin w,\; d_{36})$$

The parameters h_1, r_4, l_1, $\cos\theta$, $\sin\theta$, r_5, and r_6 are all dependent only on the distances given between elements of the clusters. Specifically, we have the following:

$$\begin{aligned} h_1 &= \frac{d_{24}^2 - d_{14}^2 - d_{12}^2}{2 d_{12}} \\ r_4 &= \sqrt{d_{14}^2 - h_1^2} \\ l_1 &= \sqrt{d_{12}^2 - d_{23}^2} \end{aligned}$$

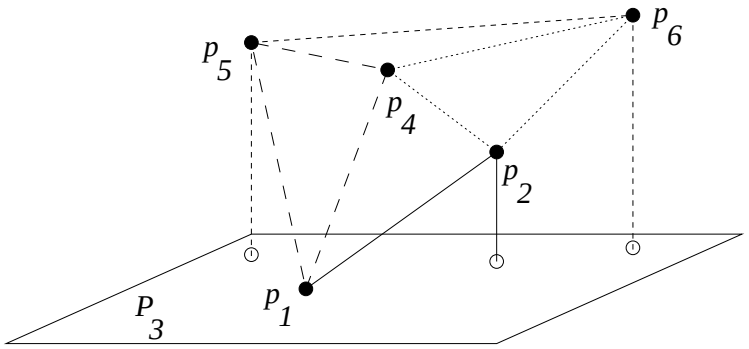

Figure 22: Five points and one plane with distance constraints.

$$\begin{aligned}
\cos\theta &= \frac{l_1}{d_{12}} \\
\sin\theta &= \frac{d_{23}}{d_{12}} \\
r_5 &= \sqrt{d_{15}^2 - d_{35}^2} \\
r_6 &= \sqrt{d_{26}^2 - (d_{36} - d_{23})^2}
\end{aligned}$$

We must now find all possible positions of the triangle $p_4p_5p_6$ such that the vertices lie on C_4, C_5, and C_6, respectively. That is, we need to find values of u, v, and w which satisfy the three equations

$$\begin{aligned}
(ch_1 + sr_4\sin u - r_5\cos v)^2 + (r_4\cos u - r_5\sin v)^2 + \\
(-sh_1 + cr_4\sin u - d_{35})^2 &= d_{45}^2 \qquad (1) \\
(ch_1 + sr_4\sin u - l_1 - r_6\cos w)^2 + (r_4\cos u - r_6\sin w)^2 + \\
(-sh_1 + cr_4\sin u - d_{36})^2 &= d_{46}^2 \qquad (2) \\
(r_5\cos v - r_6\cos w - l_1)^2 + (r_5\sin v - r_6\sin w)^2 + \\
(d_{35} - d_{36})^2 &= d_{56}^2 \qquad (3)
\end{aligned}$$

where $c = \cos\theta$ and $s = \sin\theta$. To solve these three equations, we follow the technique used in[8], in which three equations similar to these are solved to determine solutions for the six point problem.

Making the standard substitutions $\cos u = \frac{1-q_4^2}{1+q_4^2}$ $\sin u = \frac{2\,q_4}{1+q_4^2}$, $\cos v = \frac{1-q_5^2}{1+q_5^2}$ $\sin v = \frac{2\,q_5}{1+q_5^2}$, $\cos w = \frac{1-q_6^2}{1+q_6^2}$, and $\sin w = \frac{2\,q_6}{1+q_6^2}$, we obtain three equations with the following structure:

$$(A_1\,q_5^2 + A_2\,q_5 + A_3)\,q_4^2 + (A_4\,q_5^2 + A_6)\,q_4 + (A_7\,q_5^2 + A_8\,q_5 + A_9) = 0 \qquad (4)$$

$$(B_1\,q_6^2 + B_2\,q_6 + B_3)\,q_4^2 + (B_4\,q_6^2 + B_6)\,q_4 + (B_7\,q_6^2 + B_8\,q_6 + B_9) = 0 \qquad (5)$$

$$(D_1\,q_5^2 + D_3)\,q_6^2 + D_5\,q_5\,q_6 + (D_7\,q_5^2 + D_9) = 0 \qquad (6)$$

Here the coefficients A_i, B_j, and D_k are functions only of the constants c, s, l_1, h_1, d_{35}, d_{36}, d_{45}, d_{46}, d_{56}, r_4, and r_5.

We can eliminate q_4 from the first two equations using Bezout's method[9]. This yields an equation of the form

$$E_1\,q_6^4 + E_2\,q_6^3 + E_3\,q_6^2 + E_4\,q_6 + E_5 = 0$$

where each E_i is polynomial of degree four in q_5. The coefficients of these polynomials again are functions only of the constants of the problem. If we now rewrite Eq. (6) as

$$G_1 q_6^2 + G_2 q_6 + G_3 = 0$$

we can again apply Bezout's method, obtaining the single equation

$$\begin{aligned}0 = &-E_4^2G_1^3G_3 - E_1G_2^4E_5 - E_3^2G_1^2G_3^2 - E_2^2G_1G_3^3 - E_1^2G_3^4 - G_1^4E_5^2 + E_2G_1G_3^2E_3G_2 + \\ &2E_2G_1^2G_3^2E_4 - E_2G_1G_3E_4G_2^2 - 3E_2G_1^2E_5G_2G_3 + E_2G_1E_5G_2^3 - E_1G_2^2G_3^2E_3 + \\ &E_1G_2G_3^3E_2 - 3E_1G_2G_3^2E_4G_1 + E_1G_2^3G_3E_4 + 4E_1G_2^2G_1E_5G_3 + 2E_3G_1G_3^3E_1 + \\ &E_3G_1^2G_2E_4G_3 - E_3G_1^2E_5G_2^2 + 2E_3G_1^3G_3E_5 - 2E_1G_3^2G_1^2E_5 + E_4G_1^3E_5G_2\end{aligned}$$

Some of the terms in this equation are of degree 16 in q_5, since each E_i is degree four and G_1 and G_3 are degree two. Thus this polynomial has 16 roots. Furthermore, when the polynomial is expanded in terms of the d_{ij}, the odd-power terms vanish, so that a polynomial in q_5^2 of degree eight results. The positive roots of this polynomial yield the potential solutions to the problem, since we are interested only in configurations in real space. The values of q_5 can then be back-substituted into the formulas for $\cos v$ and $\sin v$, and the positions of point p_5 can be found directly from these values.

For each set of values $\{\cos v, \sin v\}$, the corresponding values of of $\cos u$, $\sin u$, $\cos w$, and $\sin w$ must be computed. Substitution of $\cos v$ and $\sin v$ into Eq. (1) and Eq. (3) yields two equations, which can be written in the form

$$L_i \cos\alpha_i + M_i \sin\alpha_i + N_i = 0$$

where $\alpha_1 = u$ and $\alpha_2 = w$. These can be solved as

$$\cos\alpha_i = \frac{-L_iN_i + \sigma M_i(L_i^2 + M_i^2 - N_i^2)^{1/2}}{L_i^2 + M_i^2}$$

$$\sin\alpha_i = \frac{-M_iN_i - \sigma L_i(L_i^2 + M_i^2 - N_i^2)^{1/2}}{L_i^2 + M_i^2}$$

where $\sigma = \pm 1$. Of the four possible combinations of solutions, only one will satisfy the remaining equation, Equation (2). These values can then be substituted into the equations of the circles C_4 and C_6 to determine the corresponding points p_4 and p_6.

A Numerical Example

To verify the above process, we consider a numerical example with one predetermined solution[1] The initial configuration is

$$\begin{aligned}p_1 &= (0,0,0)\\ p_2 &= (3,0,4)\\ P_3 &= xy-\text{plane}\\ p_4 &= (3,4,0)\\ p_5 &= (2,2,2)\\ p_6 &= (6,1,1)\end{aligned}$$

[1]This example, as well as the general solution above, was computed using Maple.

This means the input to the problem is four clusters with the following distance constraints:

Cluster 1	Cluster 2	Cluster 3	Cluster 4
$d(p_1, P_3) = 0$	$d(p_1, p_4) = 5$	$d(p_2, p_4) = 4\sqrt{2}$	$d(p_5, p_6) = 3\sqrt{2}$
$d(p_2, P_3) = 4$	$d(p_1, p_5) = 2\sqrt{3}$	$d(p_4, p_6) = \sqrt{19}$	$d(P_3, p_5) = 2$
$d(p_1, p_2) = 5$	$d(p_4, p_5) = 3$	$d(p_2, p_6) = \sqrt{19}$	$d(P_3, p_6) = 1$

From these distances the parameters of the three equations are computed:

$$\begin{aligned} h_1 &= \frac{9}{5} \\ r_4 &= \frac{4}{5}\sqrt{34} \\ \cos\theta &= \frac{3}{5} \\ \sin\theta &= \frac{4}{5} \\ l_1 &= 3 \\ r_5 &= 2\sqrt{2} \\ r_6 &= \sqrt{10} \end{aligned}$$

With these values, Eqs. (1), (2), and (3) become

$$\left(27/25 + (16\sqrt{34}\sin u)/25 - 2\sqrt{2}\cos v\right)^2 + \left((4\sqrt{34}\cos u)/5 - 2\sqrt{2}\sin v\right)^2 + \left(-36/25 + (12\sqrt{34}\sin u)/25 - d_{35}\right)^2 = d_{45}^2$$

$$\left(-48/25 + (16\sqrt{34}\sin u)/25 - \sqrt{10}\cos w\right)^2 + \left((4\sqrt{34}\cos u)/5 - \sqrt{10}\sin w\right)^2 + \left(-36/25 + (12\sqrt{34}\sin u)/25 - d_{36}\right)^2 = d_{46}^2$$

$$\left(2\sqrt{2}\cos v - \sqrt{10}\cos w - 3\right)^2 + \left(2\sqrt{2}\sin v - \sqrt{10}\sin w\right)^2 + (d_{35} - d_{36})^2 = d_{56}^2$$

Following the solution procedure detailed above, we found six solutions for q_5^2, two of which were positive. The corresponding values of q_5, $\cos v$, $\sin v$, $\cos u$, $\sin u$, $\cos w$, and $\sin w$ are shown in Table , rounded to six digits. The equations for the circles C_4, C_5, and C_6 were evaluated at these four solutions, with the following results:

Table 2: Real solutions to the numerical example

q_5	$\cos v$	$\sin v$	$\cos u$	$\sin u$	$\cos w$	$\sin w$
0.266489	0.867385	0.497638	-0.0735262	0.997293	-0.631411	-0.775448
-0.266489	0.867385	-0.497638	0.0735262	0.997293	-0.631411	0.775448
0.414214	0.707107	0.707107	0.857493	0.514496	0.948683	0.316228
-0.414214	0.707107	-0.707107	-0.857493	0.514496	0.948683	-0.316228

Solution 1

$$\begin{aligned} p_4 &= (4.801708264, -0.3429823033, 1.351281198) \\ p_5 &= (2.453334509, 1.407533228, 2.0) \\ p_6 &= (1.003302954, -2.452182886, 1.0) \end{aligned}$$

Solution 2

$$\begin{aligned} p_4 &= (4.801708264, 0.3429823033, 1.351281198) \\ p_5 &= (2.453334509, -1.407533228, 2.0) \\ p_6 &= (1.003302954, 2.452182886, 1.0) \end{aligned}$$

Solution 3

$$\begin{aligned} p_4 &= (3.000000039, 3.999999971, 0.000000029) \\ p_5 &= (2.000000071, 1.999999927, 2.0) \\ p_6 &= (6.000000050, 0.9999998449, 1.0) \end{aligned}$$

Solution 4

$$\begin{aligned} p_4 &= (3.000000039, -3.999999971, 0.000000029) \\ p_5 &= (2.000000071, -1.999999927, 2.0) \\ p_6 &= (6.000000050, -0.9999998449, 1.0) \end{aligned}$$

Solution 3 is the predetermined solution, which corroborates the correctness of the solution procedure. Note that Solution 1 and Solution 2 are mirror-images of each other through the yz-plane, as are Solution 3 and Solution 4.

5. Acknowledgements

Support for this work has been provided by the Office of Naval Research Contract N00014-90-J-1599, by the National Science Foundation under Grant ECD 92-23502, by the AT&T Foundation, and by the G.E. Foundation.

5. References

1. B. Aldefeld, *CAD*, **20** (1988) 117.
2. B. Bruderlin, in *Advances in Design and Manufacturing Systems*, (1990).
3. B. Buchberger, in *Multidimensional Systems Theory*, (1985) 184.
4. W. Bouma, I. Fudos, C. Hoffmann, J. Cai, and R. Paige, A Geometric Constraint Solver, To Appear in *CAD* (1994).
5. I. Fudos, *Editable Representations for 2D Geometric Design*, Master's Thesis (Purdue University, 1993).
6. I. Fudos and C. Hoffmann, *Correctness of a Geometric Constraint Solver*, CSD-TR-93-076 (Purdue University, 1993).
7. C. M. Hoffmann, *Geometric and Solid Modeling* (Morgan Kaufmann Publishers, 1989) 285.
8. P. Nanua, K. J. Waldron, and V. Murthy, *IEEE Trans. Robotics and Automation* **6** (1993) 438.
9. G. Salmon, *Modern Higher Algebra*, (Dublin, 1866).
10. Wu Wen-Tsun, *J. Automated Reasoning* **2** (1986) 221.
11. X. Chen and C. Hoffmann, *Towards Feature Attachment*, CSD-TR-94-010 (Purdue University, 1994).
12. J. Owen, *ACM Symp. Found. of Solid Modeling*, Austin, Texas 1991, 397–407.
13. A. Verroust, *Etude de problemes lies a la definition, la visualisation et l'animation d'objets complexes en informatique graphique*, PhD thesis, Universite Paris-Sud, France, 1990.
14. G. Crippen and T. Havel, *Distance Geometry and Molecular Conformation*, John Wiley & Sons, 1988.