MM Research Preprints, 94–111
MMRC, AMSS, Academia, Sinica, Beijing
No. 21, December 2002

Geometric Constraint Solving via C-tree Decomposition[1)]

Xiao-Shan Gao[1] and Gui-Fang Zhang[1,2]
Institute of System Science, AMSS, Chinese Academy of Sciences[1]
Beijing 100080, China
Department of Mathematics, Capital Normal University[2]
Beijing 100037, China
(xgao,gfzhang)@mmrc.iss.ac.cn

Abstract. This paper has two parts. First, we propose a method which can be used to decompose a geometric constraint graph into a c-tree. With this decomposition, solving for a well-constrained problem is reduced to the solving for smaller rigids if possible. Second, we give the analytical solutions to one of the basic merge patterns used to solve a c-tree: the 3A3D general Stewart platform, which is to determine the position of a rigid relative to another rigid when we know three angular and three distance constraints between the two rigids.

Keywords. Geometric constraint solving, parametric CAD, general construction sequence, decomposition tree, generalized Stewart platform, assembly.

1. Introduction

Geometric constraint solving (GCS) is one of the key techniques in parametric CAD, which allows the user to make modifications to existing designs by changing parameter values. There are four major approaches to GCS: the numerical approach[17, 20], the symbolic approach[6, 14], the rule-based approach[1, 5, 15, 23] and the graph-based approach [4, 9, 11, 19]. This paper will focus on using graph algorithms to decompose large constraint problems into smaller rigids and how to find the analytical solutions to one class of constraint problems.

In [22], Owen proposed a method based on the tri-connected decomposition of graphs, which may be used to reduce a class of constraint problems into constraint problems consisting of three primitives. In [8, 4], Hoffmann et al proposed a method based on cluster formation to solve 2D and 3D constraint problems. An algorithm is introduced by Joan-Arinyo et al in [12] to decompose a 2D constraint problem into an s-tree. This method is equivalent to the methods of Owen and Hoffmann, but is conceptually simpler.

The above approaches use special constraint problems, i.e. triangles, as basic patterns to solve geometric constraint problems. In [19], Latham and Middleditch proposed a connectivity analysis algorithm which could be used to decompose a constraint problem into what we called the *general construction sequence* (definition in Section 2). A similar method based

[1)]Partially supported by a National Key Basic Research Project of China (NO. G1998030600) and by a USA NSF grant CCR-0201253.

on maximal matching of bipartite graphs was proposed in [18]. In [9], Hoffmann et al gave an algorithm to find rigids in a constraint problem. From this, a general approach to GCS is proposed [10].

In this paper, we propose a method which can be used to decompose a constraint graph into a c-tree by combining the idea of s-tree of Joan-Arinyo et al in [12] and Latham-Middleditch's algorithm [19]. The general construction sequence obtained with Latham-Middleditch's algorithm is used to find a rigid in the constraint graph. With this rigid, we can split the constraint problem into two smaller problems. A c-tree is a binary tree. For each node in the tree, its left child is a rigid which will be solved first. After the left child is solved, we may use the information from the left child to solve the right child and to merge the left and right children to solve the constraint problem represented by the node.

The basic idea for all graph based methods is to reduce a large constraint problem into several smaller ones. Among these smaller problems, the largest (with largest degrees of freedom) is called the *controlling problem*. The controlling problem can be used to measure the effect of the decomposition method. The main reason for introducing the c-tree is that we may obtain smaller controlling problems than Latham-Middleditch's algorithm. We may obtain the smallest controlling problem for a constraint problem if we do an exhaust search. But this will increase the complexity of the method. As pointed out in [9], finding the smallest rigid is NP-hard. We generally satisfy if a constraint problem can be divided into smaller rigids. Also, our method is easy to understand and implement.

In Section 2., we show how to use Latham-Middleditch's algorithm to find a general construction sequence for a constraint problem. We also give a classification of the general construction sequence. In particular, we propose the concept of generalized Stewart platform.

In Section 3., the concept of c-tree is introduced and an algorithm to generate the c-tree is proposed. The basic step to solve a constraint problem with a c-tree is to solve some basic merge patterns, which are generalizations of the explicit constructions like to find the intersection of three planes or three spheres. To solve a basic merge pattern, we need to determine the position for several geometric primitives simultaneously.

In Section 4., we give the analytical solutions to one of the basic merge patterns: the **3A3D** generalizes Stewart platform, which is to determine the position of one rigid R_1 relative to the other rigid R_2 when we know three angular and three distance constraints between R_1 and R_2. This case is relatively easy to solve, because we may impose the angular constraints and the distance constraints separately, an idea first proposed in [16].

2. Classification of General Construction Sequence

2.1. Basic Concepts about Constraint Graphs

We consider three types of *geometric primitives*: points, planes and lines in the three dimensional Euclidean space and two types of constraints: the distance constraints between point/point, point/line, point/plane, line/line and the angular constraints between line/line, line/plane, plane/plane. A *geometric constraint problem* consists of a set of geometric primitives and a set of geometric constraints among these primitives. Angular and distance constraints between two primitives o_1 and o_2 are denoted by $\mathrm{ANG}(o_1, o_2)$ and $\mathrm{DIS}(o_1, o_2)$ respectively. We will use p_i, h_i and l_i to represent points, planes and lines respectively.

We use a *constraint graph* to represent a constraint problem. The vertices of the graph represent the geometric primitives and the edges represent the constraints. For a constraint

graph G, we use $\mathbf{V}(G)$ and $\mathbf{E}(G)$ to denote its sets of vertices and edges respectively.

For an edge e in the graph, let $\mathrm{DOC}(e)$ be the valence of e, which is the number of scalar equations required to define the constraint represented by e. Most constraints considered by us have valence one. There are several exceptions: (1) Constraint $\mathrm{DIS}(p_1, p_2) = 0$ has valence three. In this case, $p_1 = p_2$. We assume that this case does no occur. (2) Constraint $\mathrm{DIS}(p_1, l_1) = 0$ has valence two. (3) Constraint $\mathrm{ANG}(h_1, h_2) = 0$ has valence two. (4) Constraint $\mathrm{ANG}(l_1, l_2) = 0$ has valence two. These constraints are *degenerate cases*. For a geometric primitive o, let $\mathrm{DOF}(o)$ be the degrees of freedom for o. For a constraint graph G, let $\mathrm{DOF}(G) = \sum_{v \in \mathbf{V}(G)} \mathrm{DOF}(v), \mathrm{DOC}(G) = \sum_{e \in \mathbf{E}(G)} \mathrm{DOC}(e)$.

A constraint graph G is called *structurally well-constrained* if $\mathrm{DOC}(G) = \mathrm{DOF}(G) - 6$ and for every subgraph H of G, $\mathrm{DOC}(H) \leq \mathrm{DOF}(H) - 6$. A constraint graph G is called *structurally over-constrained* if there is a subgraph H of G satisfying $\mathrm{DOC}(H) > \mathrm{DOF}(H) - 6$. A constraint graph G is called *structurally under-constrained* if G is not over-constraint and $\mathrm{DOC}(G) < \mathrm{DOF}(G) - 6$. A structurally well-constrained graph defined a rigid in most cases. But, in some special cases the constraint problem represented by a structurally well-constrained graph may have no solutions or an infinite number of solutions. In this paper, we will concern the structure solvability of the constraint problem only. Therefore, when we say rigids, we mean structurally well-constrained problems.

2.2. Latham-Middleditch's Connectivity Algorithm

Before using Latham and Middleditch's algorithm, we need to add six more degrees of freedom to a set of primitives called *base primitives*. The geometric meaning of this step is as follows: a rigid in the space has six degrees of freedom. By selecting the base primitives, we can find the absolute position of the rigid in the space. After this step, a structurally well-constrained problem G will satisfy $\mathrm{DOC}(G) = \mathrm{DOF}(G)$, which is called *strictly well-constrained*.

In [19], Latham and Middleditch proposed an algorithm which may be used to decompose a strictly well-constrained problem into a *general construction sequence* (GCS): $C_1, C_2, \cdots, C_n$ where each C_i is a set of geometric primitives, such that

1. The geometric primitives in C_i only have constraints with primitives in $C_1, \cdots, C_{i-1}$.
2. The subgraph induced by $\cup_{k=1}^{i} C_k$ is strictly well-constrained for each $1 \leq i \leq n$.
3. No proper subsets of C_i satisfy conditions 1 and 2.

If the modified constraint graph is not strictly well-constrained, the connectivity algorithm of Latham and Middleditch may be used to add or delete a proper number of constraints to obtain a strictly well-constrained problem.

To find base primitives, we first try to find a rigid consisting of less than four primitives in the constraint problem. The four graphs in Figure 1 represent such rigids. To find one of these rigids, we will search all the edges $e = (o_1, o_2)$ of the constraint graph to see whether there exists a primitive o such that o has a proper number of constraints with both o_1 and o_2. If such a rigid exists, we may treat it as the base primitives by fixing its three translational degrees of freedom and its three directional degrees of freedom.

If such a rigid does not exist, we try to find three points p_1, p_2, p_3 such that $d_1 = \mathrm{DIS}(p_1, p_2)$ and $\mathrm{DIS}(p_2, p_3)$ are known. We select p_1, p_2, p_3 as the base primitives by adding the following constraints: $p_1 = (0, 0, 0)$, $p_2 = (d_1, 0, 0)$, $p_3 = (x, y, 0)$, where x and y are

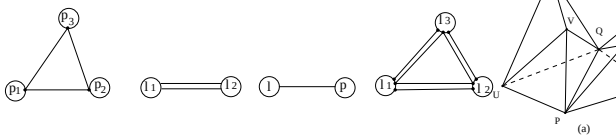


Fig. 1. Rigids with two or three primitives

Fig. 2. A constraint problem and its c-tree

variables to be determined. Other cases can be treated similarly. Let e be the number of edges, the above procedure of selecting base primitives has complexity of $O(e)$ if properly implemented.

Let us look at the constraint problem in Figure 2(a), where each line represents a distance constraint between two points. We need only to determine the position of the points.

For this problem, there are three essentially different GCSs, the geometric meaning of which is self evident. It is clear that the GSC depends on the base primitives. Note that according to our methods of selecting base primitives, $\mathcal{C}_3$ will not be generated.

$\mathcal{C}_1$: $P; Q; A; B; C; D; \{U, V, W\}$; $\mathcal{C}_2$: $P; Q; U; V; W; \{A, B, C, D\}$; $\mathcal{C}_3$: $W; D; \{P, Q, A, B, C, D, U, V\}$

It is clear that the generated GSCs depends on the base primitives. The base primitives for GCSs $\mathcal{C}_1$, $\mathcal{C}_2$, and $\mathcal{C}_3$ are $\{P, Q, A\}$, $\{P, Q, U\}$, $\{W, D, C\}$. Actually, according to our methods of selecting base primitives, only $\mathcal{C}_1$ and $\mathcal{C}_2$ could be generated. $\mathcal{C}_3$ will not be generated.

2.3. Classification of GCS

Suppose that a constraint graph G is decomposed into a GCS:

$$\mathcal{C}: \quad C_1, C_2, \cdots, C_n \tag{1}$$

The maximal of $\mathrm{DOF}(C_i)$ for $i = 1, \cdots, n$ is the maximal number of simultaneous equations to be solved in order to solve $\mathcal{C}$. This number is called the *controlling degree of freedom* of $\mathcal{C}$ and denoted by $\mathrm{MDOF}(\mathcal{C})$.

We call the type of dependency of C_i on $C_1, \cdots, C_{i-1}$ a *basic merge pattern*. Let

$$\mathcal{B}_i = \bigcup_{k=1}^{i} C_k, \qquad \mathcal{U}_i = C_{i+1}.$$

We call $\mathcal{B}_i$ and $\mathcal{U}_i$ *the base* and *the dependent object*, respectively. To solve a GCS, we need to determine $\mathcal{U}_i$ assuming that $\mathcal{B}_i$ is known. The sum of $\mathrm{DOC}(e)$ for all edges e between $\mathcal{B}_i$ and $\mathcal{U}_i$ describes an important natural of the merging step, and is called *the connection number*, denoted by $\mathrm{CN}(\mathcal{B}_i, \mathcal{U}_i)$.

Theorem 2.1 *Let V_3 be the set of points and planes in $\mathcal{U}_i$, V_4 the set of lines in $\mathcal{U}_i$, and $V = V_3 \cup V_4$. We have $3 \leq CN(\mathcal{B}_i, \mathcal{U}_i) \leq DOF(V) - |V| = 2|V_3| + 3|V_4|$.*

Proof: Since $\mathcal{U}_i$ contains at least one geometric primitive and $\mathcal{B}_i \cup \mathcal{U}_i$ is a rigid, $\mathrm{CN}(\mathcal{B}_i, \mathcal{U}_i)$ should be greater than or equal to the degree of freedom for one primitive. Hence $\mathrm{CN}(\mathcal{B}_i, \mathcal{U}_i) \geq 3$. From [19], $\mathcal{U}_i$ can be changed to a strongly connected directed graph. Since a strongly

connected graph with n vertices has at least n edges, $\mathcal{U}_i$ contains at least $|V|$ constraints, i.e. $\mathrm{DOC}(V) \geq |V|$. Since both $\mathcal{B}_i$ and $\mathcal{B}_i \cup \mathcal{U}_i$ are rigids, we need exactly $\mathrm{DOF}(V) = 3|V_3| + 4|V_4|$ constraints to determine $\mathcal{U}_i$. In other words, we have

$$\mathrm{CN}(\mathcal{B}_i, \mathcal{U}_i) + \mathrm{DOC}(V) = \mathrm{DOF}(V) = 3|V_3| + 4|V_4|.$$

Thus $\mathrm{CN}(\mathcal{B}_i, \mathcal{U}_i) \leq \mathrm{DOF}(V) - |V| = 2|V_3| + 3|V_4|$.

The computation of $\mathcal{U}_i$ with respect to $\mathcal{B}_i$ can be divided into the following cases.

1. If $\mathrm{CN}(\mathcal{B}_i, \mathcal{U}_i) = 3$, $\mathcal{U}_i$ consists of a point or a plane, which can be constructed explicitly.
2. If $\mathrm{CN}(\mathcal{B}_i, \mathcal{U}_i) = 4$, $\mathcal{U}_i$ consists of a line, which can be constructed explicitly.
3. If $\mathrm{CN}(\mathcal{B}_i, \mathcal{U}_i) = 5$, $\mathcal{U}_i$ consists of a line l and several points on l. Suppose that there are m points $p_i, i = 1, \cdots, m$ on l. After renaming the points, $\mathrm{DIS}(p_i, p_{i+1}), i = 1, \cdots, n-1$ must be known. Otherwise, $\mathrm{DOF}(U_i) > 5$ which is contradict to the fact that $\mathrm{CN}(\mathcal{B}_i, \mathcal{U}_i) = 5$.
4. If $\mathrm{CN}(\mathcal{B}_i, \mathcal{U}_i) = 6$, there exist $\mathrm{DOF}(\mathcal{U}_i) - 6$ constraints between primitives in $\mathcal{U}_i$. Hence $\mathcal{U}_i$ is a rigid. It may be considered as to assembly two rigids according to six constraints.
5. If $\mathrm{CN}(\mathcal{B}_i, \mathcal{U}_i) > 6$, the problem becomes more complicated. Now $\mathcal{U}_i$ is not a rigid anymore. We need to use the constraints inside $\mathcal{U}_i$ and those between $\mathcal{U}_i$ and $\mathcal{B}_i$ to determine $\mathcal{U}_i$.

2.4. Generalized Stewart Platform

When $\mathrm{CN}(\mathcal{B}_i, \mathcal{U}_i) = 6$, both $\mathcal{B}_i$ and $\mathcal{U}_i$ are rigids. Hence, it may be considered as an *assembly problem*. We need to assemble two rigids according to six constraints. This problem can be divided into four cases:

3D3A: There are three distance and three angular constraints.
4D2A: There are four distance and two angular constraints.
5D1A: There are five distance and one angular constraints.
6D: All the six constraints between $\mathcal{B}_i$ and $\mathcal{U}_i$ are distance constraints.

This case deserves special attention because it is closely related to the famous *Stewart Platform* [2], which is a **6D** problem where all distance constraints are between points. This platform is extensively studied because it has many important applications. For a survey, please consult [2]. Most of the work on Stewart platform is focused on the *forward displacement* problem: for a given position of $\mathcal{B}_i$ and a set of values of the distances, to determine the position of $\mathcal{U}_i$. This is exactly what we are trying to do in geometric constraint solving. The system $\mathcal{B}_i, \mathcal{U}_i$ satisfying $\mathrm{CN}(\mathcal{B}_i, \mathcal{U}_i) = 6$ will be called a *generalized Stewart platform* (GST). In Section 4., we will give analytic solutions to the **3D3A** case. More discussions about GSTs may be found in Section 4..

3. A Decomposition Algorithm

3.1. A New Decomposition Tree: C-tree

The concept of s-tree is introduced by Joan-Arinyo et al in [12] to decompose a 2D constraint problem into triangles and tri-connected components. The s-tree is equivalent to the decomposition tree of Owen [22] and Fudos and Hoffmann's method [4], but is conceptually simpler. Basically speaking, the s-tree handles problems that can be decomposed into triangles. We will introduce a new decomposition tree, c-tree, that can be used to handle more complicated rigids.

Let G be a structurally well-constrained graph and H a structurally well-constrained subgraph of G. Let I be the set of vertices $u \in H$ such that there exists at least one constraints between u and a vertex in $\mathbf{V}(G) - \mathbf{V}(H)$. If $I \neq \mathbf{V}(H)$, H is called a *faithful subgraph*.

Let H be a faithful subgraph of G. We may construct a *split subgraph* G' of G with H as follows. $\mathbf{V}(G') = (\mathbf{V}(G) - \mathbf{V}(H)) \cup I$. All the edges in $\mathbf{E}(G)$ between two vertices in $\mathbf{V}(G')$ will be in $\mathbf{E}(G')$. If G' is structurally well-constrained, G' is the split subgraph. Otherwise, we add $\mathrm{DOF}(\mathbf{V}(G')) - \mathrm{DOC}(\mathbf{E}(G')) - 6$ *auxiliary constraints* between vertices in I to make the new graph G' structurally well-constrained. This can be done with the algorithm in [19]. This new graph is the split graph.

Definition 3.1 *A c-tree for a constraint graph G is a binary tree. The root of the tree is G. For each node n in the tree, its left child L and right child R are as follows.*

1. *L is either a GCS or a basic merge pattern $L = [\mathcal{B},\mathcal{U}]$, where $\mathcal{B}$ is a rigid whose positions are known, $\mathcal{U}$ is a set of primitives to be determined by $\mathcal{B}$.*

2. *The graph induced by L is a faithful subgraph of G and the right child R is the split graph of G with H.*

All leaves are either base primitives or basic merge patterns.

Let us consider the constraint problem in Figure 2(a). The subgraph induced by $\{P, Q, U, V, W\}$ is a faithful subgraph. The split subgraph induced by $\{P, Q, U, V, W\}$ is Figure 2(b) where two auxiliary edges (W, P) and (W, Q) are added.

After a c-tree is obtained, we may use it to solve the constraint problem as follows. We do a left to right depth-first search of the c-tree and consider the following three cases.

1. The current node is a set of base primitives. We need assign six more degrees of freedom and compute the coordinates of these primitives.

2. The current node is a basic merge pattern. We need to solve this merge pattern.

3. The current node is not a leaf. In this case, we already solved the left child which is a rigid. From this rigid, we may compute the values for the auxiliary constraints in the right child. Now the right child becomes a structurally well-constrain problem. We may solve the right child recursively. The merging of the left and right children is easy, since they are connected by sharing several geometric primitives.

It is clear that the computation of a c-tree is reduced to the computation of basic merge patterns.

3.2. Finding the C-tree with GCS

After a GCS $\mathcal{C}$ for a constraint problem G is generated, we may use $\mathcal{C}$ to solve the constraint problem. For the problem in Figure 2(a), if using GCSs $\mathcal{C}_1$, $\mathcal{C}_2$ and $\mathcal{C}_3$ in Section 2.2. to solve the problem, we have $\text{MDOF}(\mathcal{C}_1) = \text{DOF}(\{U, V, W\}) = 9$, $\text{MDOF}(\mathcal{C}_2) = \text{DOF}(\{A, B, C, D\}) = 12$,$\text{MDOF}(\mathcal{C}_3) = \text{DOF}(\{A, B, C, D, U, V, W\}) = 21$. It is clear that $\mathcal{C}_1$ is the best GCS. To further simplify the computation, we may first find the position for the rigid $T_1 = \{P, Q, U, V, W\}$ by solving the following GCS:

$$\mathcal{C}_4 : \quad P; Q; U; V; W.$$

From this rigid, we may compute the distances $\text{DIS}(W, P)$ and $\text{DIS}(W, Q)$. The constraint problem in Figure 2(a) is reduced to the one in Figure 2(b), which can be solved with the following GCS.

$$\mathcal{C}_5 : \quad P; Q; A; B; C; D; W.$$

Now to solve the problem, we need to solve two GCSs: $\mathcal{C}_4$ and $\mathcal{C}_5$. Since $\text{MDOF}(C_4) = \text{MDOF}(C_5) = 3$, which is much better. The solution procedure is represented by the c-tree in Figure 2(c). This solution is the same as that obtained with the cluster formation algorithm of Hoffmann et al [8]. It is an interesting problem to prove that all the problems could be solved with the cluster formation algorithm can be solved similarly with our method.

Based on this idea, we will give an algorithm of geometric constraint solving. First let us note that a GCS $\mathcal{C}$ can be easily transformed into a c-tree. Let $\mathcal{C}$ be given in (1). Let $\mathcal{C}_i = C_1, \cdots, C_i$ and $\mathcal{B}_i = \cup_{j=1}^{i} C_i$. Then the first left child is $[\mathcal{B}_{n-1}, C_n]$ which is a leaf of the c-tree. The first right child is $\mathcal{C}_{n-1}$. The second left child is $[\mathcal{B}_{n-2}, C_{n-1}]$ which is a leaf. The second right child is $\mathcal{C}_{n-1}$, and so on. At the n-th step, we have a leaf C_1 which is the set of base primitives.

Algorithm 3.2 *The input is a structurally well-constraint constraint graph G. The output is a c-tree for G.*

S1 Select a set of base primitives for G to generate a new graph H.

S2 With Latham-Middleditch's connectivity algorithm [19], we may find a GCS for H

$$\mathcal{C} : \quad C_1, \cdots, C_m.$$

S3 Let $\mathcal{B}_i = \cup_{j=1}^{i} C_j$. If $\text{CN}(\mathcal{B}_i, C_{i+1}) < 5$, C_{i+1} is a point, a plane or a line. These cases are relatively easy to solve. Merge C_i and C_{i+1} into one set. After all such merges, we obtained a *reduced GCS*:

$$\mathcal{C}' : \quad C'_1, \cdots, C'_s.$$

S4 Let $\mathcal{B}'_i = \cup_{j=1}^{i} C'_j$. If $s = 1$, then H can be solved by explicit constructions and $\mathcal{C}$ is a construction sequence for H. We may generate a c-tree from $\mathcal{C}$. The algorithm terminates.

S5 Let k be the minimal number such that there exists at least one primitive $o \in \mathcal{B}'_i$ such that there are no constraints between o and primitives in $C'_i, i = i + 1, \cdots, s$.

S6 If $k = s$, we cannot decompose H into smaller rigids. Find a set of new base primitives for G to generate a new H and goto S2. If no new base primitives exist, we have to solve G with the GCS $\mathcal{C}$. We may generate a c-tree with $\mathcal{C}$. The algorithm terminates.

S7 Otherwise, $k < s$. We build the c-tree as follows. The left child is the c-tree generated by GCS: $\mathcal{C}'' = C'_1, \cdots, C'_k$. The right child is the split subgraph of G with the faithful subgraph induced by $\cup_{j=1}^{k} C'_j$. Set $G = G'$ goto S1.

Let N and E be the number os vertices and edges in G. Latham-Middleditch's algorithm using maximal b-matching from graph theory has complexity $O(N^3)$. At worst case, the loop started by S6 could run for $O(E)$ times and the loop started at S7 could run N times. So the total complexity of the algorithm is $O(N^4 * E)$.

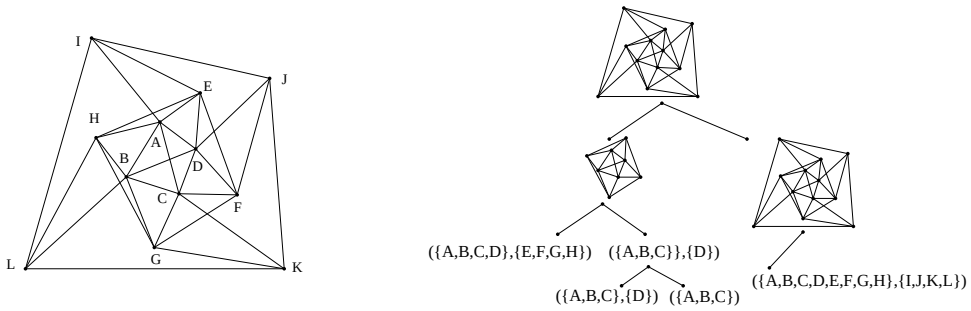


Fig. 3. A constraint problem about twelve points

With Algorithm 3.2, the c-tree in Figure 2(c) for the problem in Figure 2(a) can be generated automatically. Figure 3(a) is a more difficult constraint problem, where each edge represents a distance between two points. Figure 3(b) is the c-tree for it. Note that the problem in Figure 2(a) could be solved with the cluster formation method proposed in [8]. Another possible way to solve the problem in Figure 2(a) is to decompose the constraint graph into 4-connected components with method from [13]. But the problem in Figure 3 cannot be simplified with both methods.

4. Analytic Solution to 3D3A Case

The basic step to solve a constraint problem with a c-tree is to solve basic merge patterns. Basic merge patterns with connection number less than six are relatively easy to solve. Basic merge patterns with connection number greater than six are generally very difficult to solve. Basic merge patterns with connection number six, i.e., the GSTs, are difficult to solve and have been studied extensively [2].

We could simplify a GST $(\mathcal{B},\mathcal{U})$ as follows. We may solve the rigid $\mathcal{U}$ separately with Algorithm 3.2. Let $\mathcal{U}'$ be the set of vertices in $\mathcal{U}$, such that each vertex in $\mathcal{U}'$ has a constraint with a vertex in $\mathcal{B}$. Since $\mathcal{U}$ has been solved, we may add a reasonable number of constraints to $\mathcal{U}'$ so that $\mathcal{U}'$ becomes a rigid. Then the basic merge pattern $(\mathcal{B},\mathcal{U})$ could be simplified to $(\mathcal{B},\mathcal{U}')$. As a consequence, we may assume that $|\mathcal{U}| \leq 6$. For instance, the basic pattern in Figure 4(a) could be reduced to the one in Figure 4(b).

In this section, we try to give the analytical solution to the **3D3A** Stewart platform. This platform is easier because we may impose the angular constraints first and then the distance constraints.

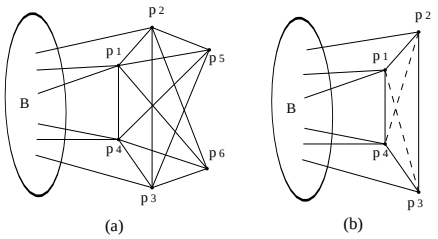


Fig. 4. A GST and its simplified form

We use Wu-Ritt's zero decomposition method [24] to find the analytical solutions. This method may be used to represent the zero set of a polynomial equation system as the union of zero sets of equations in *triangular form*, that is, equation systems like

$$f_1(u, x_1) = 0, f_2(u, x_1, x_2) = 0, \ldots, f_p(u, x_1, \ldots, x_p) = 0$$

where the u could be considered as a set of parameters and the x are the variables to be determined. As shown in [24], solutions to an equation system in triangular form are well-determined. For instance, the number of solutions to an equation system in triangular form can be easily computed.

In what below, assume that $(\mathcal{B},\mathcal{U})$ is a GST with base object $\mathcal{B}$ and dependent object $\mathcal{U}$.

4.1. Imposing Angular Constraints

Since a vector can be used to represent both the orientation of a line and the normal of a plane, there are in fact only two types of angular constraints. One is the angular constraint of valency 1 and the other is the parallelism constraint of valency 2 between two lines or two planes. There are only two cases to remove the three directional degrees of freedom: imposing a parallelism constraint of valency 2 and an angular constraint of valency 1; imposing three angular constraints of valency 1.

It is obvious that the first case can be reduced to solving two linear and one quadratic equations. Therefore, the problem could have two solutions.

Now we consider the case of imposing three angular constraints of valency 1. Let $\mathbf{l_{11}}$, $\mathbf{l_{12}}$ and $\mathbf{l_{13}}$ be three lines in $\mathcal{B}$, the base object set, and $\mathbf{l_{21}}$, $\mathbf{l_{22}}$ and $\mathbf{l_{23}}$ three lines in $\mathcal{U}$, the dependent object set. Let the parametric equations of line $\mathbf{l_{1i}}$ be $\mathbf{p} = \mathbf{p_{1i}} + u_{1i}\mathbf{s_{1i}}$, where $\mathbf{s_{1i}} = (l_i, m_i, n_i)$, $|\mathbf{s_{1i}}| = 1$, $i = 1, 2, 3$. Assume that after imposing three angular constraints the parametric equations of the three lines in the dependent object are $\mathbf{l_{2i}}$ $\mathbf{p} = \mathbf{p_{2i}} + u_{2i}\mathbf{s_{2i}}$, where $\mathbf{s_{2i}} = (x_i, y_i, z_i)$, $|\mathbf{s_{2i}}| = 1$, $i = 1, 2, 3$.

Let the three angular constraints be $\mathrm{ANG}(\mathbf{l_{11}}, \mathbf{l_{21}}) = \alpha_1$, $\mathrm{ANG}(\mathbf{l_{12}}, \mathbf{l_{22}}) = \alpha_2$, $\mathrm{ANG}(\mathbf{l_{13}}, \mathbf{l_{23}}) = \alpha_3$ and $d_i = \cos\alpha_i (i = 1, 2, 3)$. Since $\mathcal{U}$ is a rigid, the angles between three lines in $\mathcal{U}$ are also known: $\mathrm{ANG}(\mathbf{l_{21}}, \mathbf{l_{22}}) = \beta_1$, $\mathrm{ANG}(\mathbf{l_{21}}, \mathbf{l_{23}}) = \beta_2$, $\mathrm{ANG}(\mathbf{l_{22}}, \mathbf{l_{23}}) = \beta_3$. Let $\cos\beta_j = d_{j+3} (j = 1, 2, 3)$. We need to determine three unit vectors $\mathbf{s_{21}}$, $\mathbf{s_{22}}$, $\mathbf{s_{23}}$ by solving

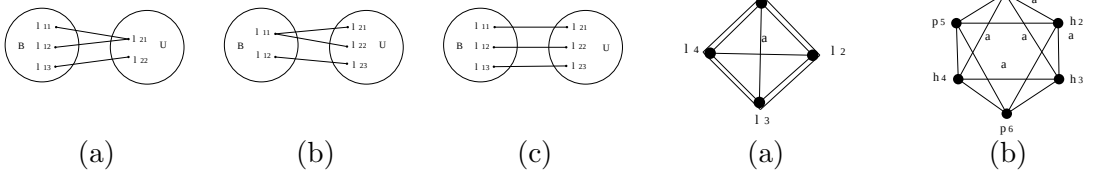
the following nine equations

$$\begin{cases} l_1x_1 + m_1y_1 + n_1z_1 - d_1 = 0 \\ l_2x_2 + m_2y_2 + n_2z_2 - d_2 = 0 \\ l_3x_3 + m_3y_3 + n_3z_3 - d_3 = 0 \\ x_1x_2 + y_1y_2 + z_1z_2 - d_4 = 0 \\ x_1x_3 + y_1y_3 + z_1z_3 - d_5 = 0 \\ x_2x_3 + y_2y_3 + z_2z_3 - d_6 = 0 \\ x_1^2 + y_1^2 + z_1^2 - 1 = 0 \\ x_2^2 + y_2^2 + z_2^2 - 1 = 0 \\ x_3^2 + y_3^2 + z_3^2 - 1 = 0 \end{cases} \tag{2}$$

Because a line has two rotational degrees of freedom, the problem can be classified into three cases shown in Figure 5.

For the case in Figure 5-(a), we could set $(l_1, m_1, n_1) = (0, 0, 1)$ and $m_2 = 0$. For the case in Figure 5-(b), we could set $(l_1, m_1, n_1) = (0, 0, 1)$ and $m_3 = 0$. With Wu-Ritt's method [24], we could reduce the equation system (2) into a triangular set with three linear and two quadratic equations, respectively. Then each case has at most four solutions.

For the case shown in Figure 5-(c), it is difficult to transform the corresponding equation system into triangular form. But, we may compute the m-Bezout number for the system as follows [21]. Let $T_1 = \{x_1, y_1, z_1\}, T_2 = \{x_2, y_2, z_2\}, T_3 = \{x_3, y_3, z_3\}$. The degree vectors of the nine equations in (2) for T_1, T_2, T_3 are $(1, 0, 0)$, $(0, 1, 0)$, $(0, 0, 1)$, $(1, 1, 0)$, $(1, 0, 1)$, $(0, 1, 1)$, $(2, 0, 0)$, $(0, 2, 0)$, $(0, 0, 2)$. Then the m–Bezout number is the coefficient of $T_1^3T_2^3T_3^3$ in the polynomial $T_1T_2T_3(T_1+T_2)(T_1+T_3)(T_2+T_3)(2T_1)(2T_2)(2T_3)$, which is 16. As a consequence, we know that the above equation system has at most 16 solutions.



(a) (b) (c) (a) (b)

Fig. 5. Three Cases of Three Angle Constraints

Fig. 6. Examples for case 3D3A

Example 4.1 *The constraint graph shown in Figure 6-(a) is a basic configuration with four lines from [7]. Lines labelled by a represent angular constraints. If there are two lines between two vertices, one line represents an angular and the other represents a distance constraint. Let l_3l_4 be the base and l_1l_2 the dependent object. Imposing three angular constraints to l_1l_2 is just corresponding to the case shown in Figure 5-(a). So there are four solutions at most.*

The constraint graph shown in Figure 6-(b) is a basic configuration from [3]. $\mathbf{h_i}(i = 1, \cdots, 4)$ *are planes and* $\mathbf{p_5}$ *and* $\mathbf{p_6}$ *are points. Let triangle $h_3h_4p_6$ be the base object and $h_1h_2p_5$ the dependent object. The three angular constraints are imposed on h_1, h_1, h_2. This problem corresponds to the case shown in Figure 5-(a). So there are four solutions at most.*

4.2. Imposing Distance Constraints

There are two ways to remove the three translational degrees of freedom: (1) imposing a distance constraint of valency two, which is line-line or point-line coincident and a distance constraint of valency one; (2) imposing three distance constraints of valency one. We discuss the second case first. The problem of imposing three distance constraints of valency one can be classified into five cases shown in Figure 7.

We will make use of the following definition and theorem from Kumar et al[16]. This theorem is also proposed in [5], under the name of 'translational' transformation.

Definition 4.2 *The translation space of a primitive on the dependent object with respect to a distance constraint is the set of positions to which the primitive can be moved to by translating the dependent object without violating this constraint.*

Theorem 4.3 *If $\mathbf{R}^X(u)$ is the translation space of $\mathbf{p_0}$ on the dependent object with respect to a constraint X, the translation space of any other point $\mathbf{p}$ on the dependent object with respect to this constraint is ${\mathbf{R}_\mathbf{p}}^X(u) = \mathbf{R}^X(u) + (\mathbf{p} - \mathbf{p_0})$, assuming that previously imposed angular constraints have eliminated all the directional degrees of freedom of the dependent object.*

Constraint type	Space type	Parametric equation for translation space
LL	plane	$\mathbf{R}^p(u,v) = \mathbf{p_l} + r_l\mathbf{m} + u\mathbf{I_1} + v\mathbf{I_2}$
PH	plane	$\mathbf{R}^p(u,v) = \mathbf{p_0} + r_p\mathbf{m} + u\mathbf{a} + v\mathbf{b}$
PP	sphere	$\mathbf{R}^s(\phi,\theta) = \mathbf{C_0} + r_s(\sin\phi\cos\theta\mathbf{i} + \sin\phi\sin\theta\mathbf{j})$
PL	cylinder	$\mathbf{R}^c(\rho,\phi) = \mathbf{p_a} + \rho\mathbf{I} + r_\rho(\cos\phi\mathbf{m} + \sin\phi\mathbf{n})$

Table 1 Parametric Equations for Translations Space

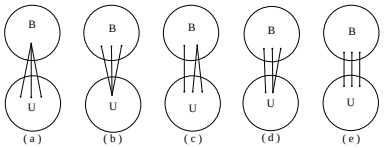


Fig. 7. Five Cases of Three distance Constraints

After removing three directional degrees of freedom for $\mathcal{U}$, there are four basic types of translation spaces, shown in Table 1, where **LL** means that the distance constraint is between two lines; **PH** means that the distance constraint is between a point and a plane, etc. In case **LL**, assuming that line $\mathbf{l_1} \in \mathcal{B}$ and line $\mathbf{l_2} \in \mathcal{U}$, $\mathbf{p_l}$ is a point on $\mathbf{l_1}$. Then $\mathbf{I_1}$ and $\mathbf{I_2}$ are unit vectors parallel to $\mathbf{l_1}$ and $\mathbf{l_2}$ respectively; $\mathbf{m}$ is a unit vector perpendicular to $\mathbf{l_1}$ and $\mathbf{l_2}$; and r_l is the distance. In case **PH**, if the point is in $\mathcal{B}$, then $\mathbf{p_0}$ is the corresponding point. Otherwise it is a point on the plane. $\mathbf{a}$ and $\mathbf{b}$ are mutually perpendicular unit vectors parallel to this plane; $\mathbf{m}$ is a unit vector perpendicular to $\mathbf{a}$ and $\mathbf{b}$; and r_p is the distance. In case **PP**, $\mathbf{C_0}$ is a point in $\mathcal{B}$ and r_s is the distance. In case **PL**, $\mathbf{I}$ is a unit vector parallel to the line and $\mathbf{m}$ and $\mathbf{n}$ are mutually perpendicular unit vectors perpendicular to $\mathbf{I}$. If the point is in the base object, $\mathbf{p_a}$ is the corresponding point. Otherwise it is a point on the line.

We can always assume that the primitives in $\mathcal{U}$ are three points. If the given vertex in the dependent object is a plane or a line, we can take a point on the plane or the line. Proofs for the correctness of this are omitted. Let $\mathbf{p_1}$, $\mathbf{p_2}$ and $\mathbf{p_3}$ be the points in $\mathcal{U}$, d_1, d_2 and d_3 the corresponding distances. The translation spaces of constraints d_1, d_2 and d_3 are $\mathbf{R}^{d_1}(u_1)$, $\mathbf{R}^{d_2}(u_2)$ and $\mathbf{R}^{d_3}(u_3)$ respectively. Imposing constraints d_1, d_2 and d_3 means that we must find points ${\mathbf{p_1}}^*$ in $\mathbf{R}^{d_1}(u_1)$, ${\mathbf{p_2}}^*$ in $\mathbf{R}^{d_2}(u_2)$ and ${\mathbf{p_3}}^*$ in $\mathbf{R}^{d_3}(u_3)$, such that segments ${\mathbf{p_1}}^*{\mathbf{p_3}}^*$ and ${\mathbf{p_2}}^*{\mathbf{p_3}}^*$ are parallel and equal to segments $\mathbf{p_1p_3}$ and $\mathbf{p_2p_3}$, respectively. The parametric equation of line $\mathbf{p_1^*p_3^*}$ is $\mathbf{p} = {\mathbf{p_1}}^* + v_1\mathbf{s_1}$, where v_1 is a free parameter and $\mathbf{s_1}$ is a unit vector. The parametric equation of line $\mathbf{p_2^*p_3^*}$ is $\mathbf{p} = {\mathbf{p_2}}^* + v_2\mathbf{s_2}$, where v_2 is a free parameter and $\mathbf{s_2}$ is a unit vector. Let $t_1 = |\mathbf{p_1p_3}|$ and $t_2 = |\mathbf{p_2p_3}|$. Then we have ${\mathbf{p_3}}^* = {\mathbf{p_1}}^* + t_1\mathbf{s_1}$ and ${\mathbf{p_3}}^* = {\mathbf{p_2}}^* + t_2\mathbf{s_2}$.

After obtaining the analytical solutions to point $\mathbf{p_3^*}$, we can translate $\mathcal{U}$ along the vector $\mathbf{t} = \mathbf{p_3^*} - \mathbf{p_3}$ and get the analytic solutions to point $\mathbf{p_1^*}$ according to $\mathbf{p_3^*} = \mathbf{p_1^*} + t_1\mathbf{s_1}$, point $\mathbf{p_2^*}$ according to $\mathbf{p_3^*} = \mathbf{p_2^*} + t_2\mathbf{s_2}$, and hence the position of $\mathcal{U}$.

According to Theorem 4.3, it's obvious that point $\mathbf{p_3}^*$ satisfies the following equations

$$\begin{cases} \mathbf{p} = \mathbf{R}^{d_1}(u_1) + t_1\mathbf{s_1} \\ \mathbf{p} = \mathbf{R}^{d_2}(u_2) + t_2\mathbf{s_2} \\ \mathbf{p} = \mathbf{R}^{d_3}(u_3) \end{cases} \tag{3}$$

1. If $t_1 \neq 0$ and $t_2 \neq 0$ in (3), that is $\mathbf{p_1}$, $\mathbf{p_2}$ and $\mathbf{p_3}$ are three different points, equation system (3) corresponds to cases (a), (c) and (e) in Figure 7.
2. If $t_1 = 0$ and $t_2 = 0$, that is $\mathbf{p_1} = \mathbf{p_2} = \mathbf{p_3}$, equation system (3) corresponds to cases (b) in Figure 7.
3. If $t_1 = 0$ and $t_2 \neq 0$, that is point $\mathbf{p_1} = \mathbf{p_2}$ and $\mathbf{p_3} \neq \mathbf{p_1}$, equation system (3) corresponds to case (d) shown in Figure 7.

Since we'll use implicit equations during computation, in what below, we will list the implicit equations for $\mathbf{R}^d(u) + t\mathbf{s}$, where $\mathbf{R}^d(u)$ is the translation space in Table 1 and $t\mathbf{s}$ is a constant vector.

Case **LL:**The parametric equation is

$$\mathbf{p} = \mathbf{p_l} + r_l\mathbf{m} + u\mathbf{I_1} + v\mathbf{I_2} + t\mathbf{s}$$

The implicit equation is

$$(m_2n_1 - n_2m_1)(x - x_1 - r_ll_3) + (l_1n_2 - n_1l_2)(y - y_1 - r_lm_3) + (m_1l_2 - l_1m_2)(z - z_1 - r_ln_3) = 0$$

where $\mathbf{p} = (x, y, z)$, $\mathbf{p_l} = (x_{p_l}, y_{p_l}, z_{p_l})$, $\mathbf{s} = (l_s, m_s, n_s)$, $\mathbf{I_1} = (l_1, m_1, n_1)$, $\mathbf{I_2} = (l_2, m_2, n_2)$, $\mathbf{m} = (l_3, m_3, n_3)$, $\mathbf{m} = \pm\mathbf{I_1} \times \mathbf{I_2}$, $x_1 = x_{p_l} + tl_s$, $y_1 = y_{p_l} + tm_s$, $z_1 = z_{p_l} + tn_s$.
The implicit equation can be simplified as

$$d_1x + d_2y + d_3z + d4 = 0$$

where $d_1 = (m_2n_1 - n_2m_1)$, $d_2 = (l_1n_2 - n_1l_2)$, $d_3 = (m_1l_2 - l_1m_2)$, $d_4 = -(m_2n_1 - n_2m_1)(x_1 + r_ll_3) - (l_1n_2 - n_1l_2)(y_1 + r_lm_3) - (m_1l_2 - l_1m_2)(z_1 + r_ln_3)$.

Case **PH:** The parametric equation is

$$\mathbf{p} = \mathbf{p_0} + r_p\mathbf{m} + u\mathbf{a} + v\mathbf{b} + t\mathbf{s}$$

The implicit equation is

$$(m_2n_1 - n_2m_1)(x - x_1 - r_pm_3) + (l_1n_2 - n_1l_2)(y - y_1 - r_pm_3) + (m_1l_2 - l_1m_2)(z - z_1 - r_pn_3) = 0$$

where $\mathbf{p} = (x, y, z)$, $\mathbf{p_0} = (x_{p_0}, y_{p_0}, z_{p_0})$, $\mathbf{s} = (l_s, m_s, n_s)$, $\mathbf{a} = (l_1, m_1, n_1)$, $\mathbf{b} = (l_2, m_2, n_2)$, $\mathbf{m} = (l_3, m_3, n_3)$, $\mathbf{m} = \pm\mathbf{a} \times \mathbf{b}$, $x_1 = x_{p_0} + tl_s$, $y_1 = y_{p_0} + tm_s$, $z_1 = z_{p_0} + tn_s$.
The implicit equation can be simplified as

$$d_1x + d_2y + d_3z + d4 = 0$$

where $d_1 = (m_2n_1 - n_2m_1)$, $d_2 = (l_1n_2 - n_1l_2)$, $d_3 = (m_1l_2 - l_1m_2)$, $d_4 = -(m_2n_1 - n_2m_1)(x_1 + r_pm_3) - (l_1n_2 - n_1l_2)(y_1 + r_pm_3) - (m_1l_2 - l_1m_2)(z_1 + r_pn_3)$.

Case **PP:** The parametric equation is

$$\mathbf{p} = \mathbf{C_0} + r_s(\sin\phi\cos\theta\mathbf{i} + \sin\phi\sin\theta\mathbf{j}) + t\mathbf{s}$$

The implicit equation is

$$(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 - r_s^2 = 0$$

where $\mathbf{p} = (x, y, z)$, $\mathbf{C_0} = (x_{\mathbf{C_0}}, y_{\mathbf{C_0}}, z_{\mathbf{C_0}})$, $\mathbf{s} = (l_s, m_s, n_s)$, $x_1 = x_{C_0} + tl_s$, $y_1 = y_{C_0} + tm_s$, $z_1 = z_{C_0} + tn_s$.

The implicit equation can be simplified as

$$x^2 + y^2 + z^2 + d_1x + d_2y + d_3z + d_4 = 0$$

where $d_1 = -2x_1$, $d_2 = -2y_1$, $d_3 = -2z_1$, $d_4 = x_1^2 + y_1^2 + z_1^2 - r_s^2$.

Case **PL:** The parametric equation is

$$\mathbf{p} = \mathbf{p_a} + \rho\mathbf{I} + r_\rho(\cos\phi\mathbf{m} + \sin\phi\mathbf{n}) + t\mathbf{s}$$

The implicit equation is

$$(x - x_1 - \rho l)^2 + (y - y_1 - \rho m)^2 + (z - z_1 - \rho n)^2 - r_\rho^2 = 0$$

where $\rho = \frac{(m_1n_2 - n_1m_2)x + (n_1l_2 - l_1n_2)y + (l_1m_2 - m_1l_2)z}{(l_1m_2 - n_1m_2 - m_1l_2 + m_1n_2)l + (n_1l_2 - l_1n_2)m} + \frac{(n_1m_2 - n_2m_1)x_1 + (n_2l_1 - l_2n_1)y_1 + (l_2m_1 - l_1m_2)z_1}{(l_1m_2 - n_1m_2 - m_1l_2 + m_1n_2)l + (n_1l_2 - l_1n_2)m}$, $(l_1m_2 - n_1m_2 - m_1l_2 + m_1n_2)l + (n_1l_2 - l_1n_2)m \neq 0)$.

The implicit equation can be simplified as

$$d_1x^2 + d_2y^2 + d_3z^2 + (d_4y + d_5z + d_6)x + (d_7z + d_8)y + d_9z + d_{10} = 0$$

where $\mathbf{p} = (x, y, z)$, $\mathbf{p_a} = (x_{\mathbf{p_a}}, y_{\mathbf{p_a}}, z_{\mathbf{p_a}})$, $\mathbf{s} = (l_s, m_s, n_s)$, $\mathbf{I} = (l, m, n)$, $\mathbf{m} = (l_1, m_1, n_1)$, $\mathbf{n} = (l_2, m_2, n_2)$, $x_1 = x_{\mathbf{p_a}} + tl_s$, $y_1 = y_{\mathbf{p_a}} + tm_s$, $z_1 = z_{\mathbf{p_a}} + tn_s$. $d_1 = 1 - 2lf_1 + 2f_1^2$, $d_2 = 1 - 2mf_2 + 2f_2^2$, $d_3 = 1 - 2nf_3 + 2f_3^2$, $d_4 = 4f_1f_2 - 2lf_2 - 2mf_1$, $d_5 = 4f_1f_3 - 2lf_3 - 2nf_1$, $d_6 = 4f_1f_4 - 2x_1 + (2x_1f_1 - 2f_4)l + 2z_1 1nf_1 + 2y_1mf_1$, $d_7 = 4f_2f_3 - 2nf_2 - 2mf_3$, $d_8 = 2y_1mf_2 + 2x_1lf_2 - 2y_1 - 2mf_4 + 2 * z1 * n * f2 + 4 * f2 * f4$, $d_9 = 2y_1mf_3 + 2z_1nf_3 + 4f_3f_4 + 2x_1lf_3 - 2n_f4 - 2z_1$, $d_{10} = 2z_1nf_4 + 2y_1mf_4 + 2x_1lf_4 + 2f_4^2 - r_\rho^2 + x_1^2 + y_1^2 + z_1^2$, $f_1 = \frac{(m_1n_2 - n_1m_2)}{(l_1m_2 - n_1m_2 - m_1l_2 + m_1n_2)l + (n_1l_2 - l_1n_2)m}$, $f_2 = \frac{(n_1l_2 - l_1n_2)}{(l_1m_2 - n_1m_2 - m_1l_2 + m_1n_2)l + (n_1l_2 - l_1n_2)m}$, $f_3 = \frac{(l_1m_2 - m_1l_2)}{(l_1m_2 - n_1m_2 - m_1l_2 + m_1n_2)l + (n_1l_2 - l_1n_2)m}$, $f_4 = \frac{(n_1m_2 - n_2m_1)x_1 + (n_2l_1 - l_2n_1)y_1 + (l_2m_1 - l_1m_2)z_1}{(l_1m_2 - n_1m_2 - m_1l_2 + m_1n_2)l + (n_1l_2 - l_1n_2)m}$.

There are three special cases while $(l_1m_2 - n_1m_2 - m_1l_2 + m_1n_2)l + (n_1l_2 - l_1n_2)m = 0$

1. $\mathbf{I} = (0, 0, 1)$, $\mathbf{p_a} + t\mathbf{s} = (0, 0, 0)$
   The implicit equation is $x^2 + y^2 - r_\rho^2 = 0$, $\rho = z$;
2. $\mathbf{I} = (0, 1, 0)$, $\mathbf{p_a} + t\mathbf{s} = (0, 0, 0)$
   The implicit equation is $x^2 + z^2 - r_\rho^2 = 0$, $\rho = y$;
3. $\mathbf{I} = (1, 0, 0)$, $\mathbf{p_a} + t\mathbf{s} = (0, 0, 0)$
   The implicit equation is $z^2 + y^2 - r_\rho^2 = 0$, $\rho = x$.

Thus imposing three distance constraints can be classified into solving the following four types of equation system.

1. Three Linear Equations The three translation spaces are all plane. It is easy to be solve.

2. Two Linear Equations and One Quadratic Equation It is easily to solve two linear equations and a quadratic equation. The types of translation space should be two planes and a sphere or a cylinder. The equation system has two solutions at most. As a consequence, we know that this case is ruler and compass constructible.

3. One Linear Equation and Two Quadratic Equations The types of translation space should consist a plane and two spheres, a plane and two cylinders and a plane, a sphere and a cylinder. The detailed equations are shown as follows.

Case 1. One Plane and Two Spheres Equation (3) can be represented as

$$\begin{cases} d_1x + d_2y + d_3z + d_4 = 0 \\ x^2 + y^2 + z^2 + f_5x + f_6y + f_7z + f_8 = 0 \\ x^2 + y^2 + z^2 + d_9x + d_{10}y + d_{11}z + d_{12} = 0. \end{cases} \tag{4}$$

The equation system can be simplified as

$$\begin{cases} d_1x + d_2y + d_3z + d_4 = 0 \\ d_5x + d_6y + d_7z + d_8 = 0 \\ x^2 + y^2 + z^2 + d_9x + d_{10}y + d_{11}z + d_{12} = 0. \end{cases} \tag{5}$$

where $d_i = f_i - d_{i+4}$ $(i = 5, \cdots, 8)$.

Above equation system consists of two linear and one quadratic equations and can be solved easily.

Case 2. One Plane, One Sphere and One Cylinder Equation (3) can be represented as

$$\begin{cases} d_1x + d_2y + d_3z + d_4 = 0 \\ x^2 + y^2 + z^2 + d_5x + d_6y + d_7z + d_8 = 0 \\ d_9x^2 + d_{10}y^2 + d_{11}z^2 + (d_{12}y + d_{13}x + d_{14})z + (d_{15}z + d_{16})y + d_{17}x + d_{18} = 0. \end{cases} \tag{6}$$

The above equation system can be reduced into the following triangular form.

$$\begin{cases} (z_{411}x + z_{412})z + z_{413}x^2 + z_{414}x + z_{415} = 0 \\ (z_{421}x + z_{422})y + z_{423}x^2 + z_{424}x + z_{425} = 0 \\ z_{431}x^4 + z_{432}x^3 + z_{433}x^2 + z_{434}x + z_{435} = 0 \end{cases} \tag{7}$$

Case 3. One Plane and Two Cylinders Equation (3) can be represented as

$$\begin{cases} d_1x + d_2y + d_3z + d_4 = 0 \\ d_5x^2 + d_6y^2 + d_7z^2 + (d_8y + d_9z + d_{10})x + (d_{11}z + d_{12})y + d_{13}z + d_{14} = 0 \\ d_{15}x^2 + d_{16}y^2 + d_{17}z^2 + (d_{18}y + d_{19}z + d_{20})x + (d_{21}z + d_{22})y + d_{23}z + d_{24} = 0. \end{cases} \tag{8}$$

This problem is similar to the case *One Plane, One Sphere and One Cylinder* case.

Three Quadratic Equations

The types of translation space should consist of three spheres, two spheres and one cylinder, one sphere and two cylinders and three cylinders. The detailed equations are shown as follows.

Case 1. Three Spheres Equation (3) can be represented as

$$\begin{cases} x^2+y^2+z^2+f_1x+f_2y+f_3z+f_4=0 \\ x^2+y^2+z^2+f_5x+f_6y+f_7z+f_8=0 \\ x^2+y^2+z^2+d_9x+d_{10}y+d_{11}z+d_{12}=0. \end{cases} \tag{9}$$

The equation system can be simplified as

$$\begin{cases} d_1x+d_2y+d_3z+d_4=0 \\ d_5x+d_6y+d_7z+d_8=0 \\ x^2+y^2+z^2+d_9x+d_{10}y+d_{11}z+d_{12}=0. \end{cases} \tag{10}$$

where $d_i=f_i-d_{i+8}$ $(i=1,\cdots,8)$.

The equation system can be treated similarly to the case *Two Planes and One Sphere* case.

Case 2. Two Spheres and One Cylinder Equation (3) can be represented as

$$\begin{cases} x^2+y^2+z^2+f_1x+f_2y+f_3z+f_4=0 \\ x^2+y^2+z^2+d_5x+d_6y+d_7z+d_8=0 \\ d_9x^2+d_{10}y^2+d_{11}z^2+(d_{12}y+d_{13}x+d_{14})z+(d_{15}z+d_{16})y+d_{17}x+d_{18}=0. \end{cases} \tag{11}$$

The equation system can be simplified as

$$\begin{cases} d_1x+d_2y+d_3z+d_4=0 \\ x^2+y^2+z^2+d_5x+d_6y+d_7z+d_8=0 \\ d_9x^2+d_{10}y^2+d_{11}z^2+(d_{12}y+d_{13}x+d_{14})z+(d_{15}z+d_{16})y+d_{17}x+d_{18}=0. \end{cases} \tag{12}$$

where $d_i=f_i-d_{i+4}$ $(i=1,\cdots,4)$.

The above equation system can be treated similar to the case *One Planes, One Sphere and One cylinder* case.

case 3. One Sphere and Two Cylinders Equation (3) can be represented as

$$\begin{cases} x^2+y^2+z^2+d_1x+d_2y+d_3z+d_4=0 \\ d_5x^2+d_6y^2+d_7z^2+(d_8y+d_9z+d_{10})x+(d_{11}z+d_{12})y+d_{13}z+d_{14}=0 \\ d_{15}x^2+d_{16}y^2+d_{17}z^2+(d_{18}y+d_{19}z+d_{20})x+(d_{21}z+d_{22})y+d_{23}z+d_{24}=0. \end{cases} \tag{13}$$

The above equation system can be reduced into the following triangular form.

$$\begin{cases} (z_{511}z^2+z_{512}z+z_{513})x+z_{514}z^4+z_{515}z^3+z_{516}z^2+z_{517}z+z_{518}=0 \\ (z_{521}z^2+z_{522}z+z_{523})y+z_{524}z^4+z_{525}z^3+z_{526}z^2+z_{527}z+z_{528}=0 \\ z_{531}z^8+z_{532}z^7+z_{533}z^6+z_{534}z^5+z_{535}z^4+z_{536}z^3+z_{537}z^2+z_{538}z+z_{539}=0 \end{cases} \tag{14}$$

Case 4. Three Cylinders Equation (3) can be represented as

$$\begin{cases} g_1x^2+g_2y^2+g_3z^2+(g_4y+g_5z+g_6)x+(g_7z+g_8)y+g_9z+g_{10}=0 \\ g_{11}x^2+g_{12}y^2+g_{13}z^2+(g_{14}y+g_{15}z+g_{16})x+(g_{17}z+g_{18})y+g_{19}z+g_{20}=0 \\ g_{21}x^2+g_{22}y^2+g_{23}z^2+(g_{24}y+g_{25}z+g_{26})x+(g_{27}z+g_{28})y+g_{29}z+g_{30}=0. \end{cases} \tag{15}$$

It is difficult to get the solutions of above equations directly, so we make a transformation according to the fact that above three translation spaces are all cylinders. The equation system can be represented as

$$\begin{cases} f_1x^2+f_2y^2+f_3z^2+(f_4y+f_5z+f_6)x+(f_7z+f_8)y+f_9z+f_{10}=0 \\ f_{11}x^2+f_{12}y^2+f_{13}z^2+(f_{14}y+f_{15}z+f_{16})x+(f_{17}z+f_{18})y+f_{19}z+f_{20}=0. \\ x^2+y^2-d_1=0 \end{cases} \tag{16}$$

The above equation system can be reduced into the following triangular form.

$$\begin{cases} (z_{611}x^4+z_{612}x^3+z_{613}x^2+z_{614}x+z_{615})z+z_{616}x^5+z_{617}z^4+z_{618}z^3+z_{619}z^2+z_{6110}z+z_{6111}=0 \\ (z_{621}x^3+z_{622}x^2+z_{623}x+z_{624})y+z_{625}x^4+z_{626}x^3+z_{627}x^2+z_{628}x+z_{629}=0 \\ z_{631}x^8+z_{632}x^7+z_{633}x^6+z_{634}x^5+z_{635}x^4+z_{636}x^3+z_{637}x^2+z_{638}x+z_{639}=0. \end{cases} \tag{17}$$

From these equations, we see that the translation spaces are planes, spheres or cylinders which are denoted by **P**, **S** and **C**. For instance, **PPS** means that we need to find the intersection points of two planes and one sphere. To satisfy three distance constraints can be classified into the following cases, which are all solved with Wu-Ritt's method [24]. For the explicit formulas and expressions of z_{ijk}, please refer to http://www.mmrc.iss.ac.cn/~xgao/publ.html.

Here is a summery.

1. The equation system for **PPP** consists of three linear equations. Then it has one solution.
2. The equation system for **PPC** or **PPS** consists of two linear and one quadratic equations. Then they have two solutions at most.
3. The equation system for **SSS** or **PSS** can be reduced into a triangular set consisting of one quadratic and two linear equations. Then they have two solutions at most.
4. The equation system for **PCC**, **SSC** or **PSC** can be reduced into a triangular set consisting of one quartic and two linear equations. Then they have four solutions at most.
5. The equation system for **CCC** or **SCC** can be reduced into a triangular set consisting of a degree eight equation and two linear equations. They have eight solutions at most.

Example 4.4 *Continue from Example 4.1. Now impose three distance constraints to the problems in Figure 6, respectively. It is obviously that each equation system consists of three linear equations, and only has one solution. Thus each problem shown in Figure 6 has four solutions at most. For the problem in Figure 6(a), this result is better than that in [7]. For the problem in Figure 6(b), this is the same as that in [3].*

For the case of imposing a distance constraint of valency 2 and a distance constraint of valency 1, it is obvious that the corresponding equations for both line-line coincident and point-line coincident are two linear equations. From the above discussion, we know the equation for the distance constraint of valency one is linear or quadratic after removing three rotational degrees. So this case has one or two solutions.

Theorem 4.5 *From Section 4.1., we generally could have 4, 16 solutions when imposing the angular constraints. From Section 4.2., we generally could have 1,2,4, 8 solutions when imposing the distance constraints. Therefore, the 3D3A problem generally could have* $2^k, k = 2, \cdots, 7$ *solutions depending on the types of the constraints imposed on it.*

5. Conclusion

A geometric constraint solving procedure usually consists of two phases: the analysis phase, which is to reduce a large geometric constraint problem into several subproblems, and the computation phase, which is to merge the subproblems by numerical or symbolic computation. In this paper, we propose an analysis method which may be used to decompose any constraint problem into smaller rigids if possible. Comparing to other decomposition methods, our method can be used to handle general constraint problems and is easier to understand and implement.

The computation phase could be very difficult. This is due to the intrinsic difficulty of the constraint problem: there exist constraint problems of any size which cannot be decomposed into smaller rigids. For these problems, we have to solve them with brutal force computation methods. For some problems, we are lucky in terms that we can find their analytical solutions. In this paper, we showed that the 3A3D Stewart platform is such a problem.

It is an interesting problem to compare the scope and performance for the existing decomposition algorithms for 3D constraint problems, which needs detailed implementations. We may ask, for example, whether all the problems that could be solved with the cluster formation method in [8] can also be solved similarly with our method. The idea of using k-connectivity algorithms for $k > 3$ [13] to geometric constraint solving is also worth considering.

References

[1] B. Brüderlin, Using Geometric Rewriting Rules for Solving Geometric Problems Symbolically, *Theoretical Computer Science*, **116**, 291-303, 1993.

[2] B. Dasgupta and T. S. Mruthyunjaya, the Stewart Platform Manipulator: A Review, *Mechanism and Machine Theorey*. **35**, 15-40, 2000.

[3] C. Durand and C. M. Hoffmann, A Systematic Framework for Solving Geometric Constraints Analytically, *J. of Symbolic Computation*, **30**(5), 493-529, 2000.

[4] I. Fudos and C.M. Hoffmann, A Graph-Constructive Approach to Solving Systems of Geometric Constraints, *ACM Trac. on Graphics*, **16**(2), 179-216, 1997.

[5] X. S. Gao and S. C. Chou, Solving Geometric Constraint Systems I. A Global Propagation Approach, *Computer Aided Design*, **30**(1), 47-54, 1998.

[6] X. S. Gao and S. C. Chou, Solving Geometric Constraint Systems II. A Symbolic Approach and Decision of Rc-constructibility, *Computer-Aided Design*, **30**(2), 115-122, 1998.

[7] X. S. Gao, C. M. Hoffmann and W. Q. Yang, Solving Basic Gometrci Constraint Configurations with Locus Intersection, *Proc. ACM SM02*, 95-104, ACM Press, New York, 2002.

[8] C. M. Hoffmann and P. J. Vermeer, Geometric Constraint Solving in R^2 and R^3, in *Computing in Euclidean Geometry*, D. Z. Du and F. Huang (eds), World Scientific, Singapore, 266-298, 1995

[9] C. M. Hoffmann, A. Lomonosov and M. Sitharam, Finding Solvable Subsets of Constraint Graphs, in *LNCS*, NO. 1330, Springer, Berlin Heidelberg, 163-197, 1997.

[10] C. M. Hoffmann, A. Lomonosov and M. Sitharam, Decomposition Plans for Geometric Constraint Systems, I: Performance Measures for CAD, **31**, 367-408; II: New Algorithms, **31**, 409-427, *J. of Symbolic Computation*, 2001.

[11] R. Joan-Arinyo and A. Soto, A Correct Rule-Based Geometric Constraint Solver, *Computers and Graphics*, **21**(5), 599-609, 1997.

[12] R. Joan-Arinyo A. Soto-Riera, S. Vila-Marta, J. Vilaplana-Pasto, Revisiting Decomposition Analysis of Geometric Constraint Graphs, *Proc. ACM SM02*, 105-115, ACM Press, New York, 2002.

[13] A. Kanevsky and V. Ramachandran, Improved Algorothms for Graph Four-connectivity, *Proc. 28th IEEE Symp. Foundations of Comp. Sci.*, Los Angeles, 252-259, 1987.

[14] K. Kondo, Algebraic Method for Manipulation of Dimensional Relationships in Geometric Models, *Computer Aided Design*, **24**(3), 141-147, 1992.

[15] G. A. Kramer, Solving Geometric Constraints Systems: A Case Study in Kinematics, MIT Press, Cambridge Massachusetts, 1992.

[16] A. V. Kumar and L. Yu, Sequential Constraint Imposition for Dimension-driven Solid Models, *Computer Aided Design*, **33**, 475-486, 2001

[17] H. Lamure and D. Michelucci, Solving Geometric Constraints By Homotopy, *IEEE Trans on Visualization and Computer Graphics*, **2**(1):28-34, 1996.

[18] H. Lamure and D. Michelucci, Qualitative Study of Geometric Constraints, in *Geometric Constraint Solving and Applications*, 234-258, Springer, Berlin, 1998.

[19] R. S. Latham and A. E. Middleditch, Connectivity Analysis: a Tool for Processing Geometric Constraints, *Computer Aided Design*, **28**(11), 917-928, 1994.

[20] V. C. Lin, D. C. Gossard and R. A. Light, Variational Geometry in Computer-Aided Design, *Computer Graphics*, **15**(3), 171-177, 1981.

[21] A. Morgan and A. Sommese, A Homotopy for Solving General Polynomial Systems That Respect m-homogeneous Structures, *Appl. Math. Comput.*, **24**, 95-114, 1987.

[22] J. Owen, Algebraic Solution for Geometry from Dimensional Constraints, in *ACM Symp., Found of Solid Modeling*, ACM Press, New York, 397-407, 1991.

[23] A. Verroust, F. Schonek and D. Roller, Rule-Oriented Method for Parameterized Computer-Aided Design, *Computer Aided Design*, **24**(10), 531-540, 1992.

[24] W.T. Wu, Basic Principles of Mechanical Theorem Proving in Geometries, Science Press, Beijing, 1984; English Version, Springer-Verlag, Berlin Heidelberg, 1994.