Bull. Nov. Comp. Center, Comp. Science, 26 (2007), 75–89
© 2007 NCC Publisher

Spanning-tree modeling method for geometric constraint satisfaction problem

Alexey Ershov, Alexey Kiselev, Eugene Rukoleev

Abstract. It is well known that parameterization is a powerful tool for creation and reuse of models. It allows us to construct models with predefined features, as well as to form new models by modification of parameters of already existing models. These opportunities are crucial for engineers and other CAD users, since they greatly reduce the total product operation time. There are two classical types of parametric design: the more traditional one is a hierarchical approach based on a history of model creation, and more powerful one is a variational design that expresses a model by a set of constraints. In this article we introduce a mathematical approach which combines the advantages of both methods and has no their drawbacks.

Introduction

In the hierarchical design, the order of creation of elements is quite important. This order is reflected in a tree of creation of a model. An element based on any part of another element is called to be dependent on another element. Any parametric model keeps the history of its creation, which makes it possible to modify an arbitrary element in the history by changing parameters of its dependency on other elements.

This approach is easy to implement but it prevents many actions like relative positioning of two objects that are already placed in the history of model creation. It is impossible because of arising of cycles of dependencies between elements and collapse of the tree-like hierarchical structure. Thus, if there are non-trivial dependencies between elements in the model, an essentially different approach to the model description is needed [1]. Another serious drawback of the traditional hierarchical design method is the necessity to work only with well-defined geometric models that have no inner degrees of freedom. This condition greatly restricts the editing possibilities. In order to overcome all drawbacks of the traditional parametric approach based on a history of model creation, a new *variational design* paradigm was introduced. It consists in expression of relations between constructive elements and their defining parameters by means of *constraints*. A constraint is a formal declarative description of an arbitrary relation between objects of a model [2]. Usually constraints are classified as *logical* (lines parallelism, plane-cylinder tangency, etc.) and *parametric* (like segment length, sphere

radius, angle between facets). Using constraints, someone can completely control the shape of the product to be designed. No manual calculations of relative positions of model parts are needed since CAD automatically determines positions of all objects that satisfy all the constraints defined.

The expressive power is the main advantage of the variational approach as compared to the history-based approach [3]. An engineer does not need to plan product creation step by step as computer can do it. At any moment, one can add or remove constraints or change their parameters and obtain a new model with desired properties. The more perfect geometric model description mechanism can deal with cycle dependencies and underdefined models. Nevertheless, the variational approach is subject to criticism by developers of traditional parametric design systems. They consider as a shortcoming the necessity of simultaneous resolution of the system of all constraints after modification of the model. Thus, the variational design systems can be developed only on the basis of a specialized geometric solver. The main potential weakness of such systems is a long time of reaction to model modification performed by a user.

The geometric solvers LGS 2D and LGS 3D developed by LEDAS company can efficiently solve the variational design problems. These solvers have been embedded in several worldwide CADs [4]. Although all users were satisfied with new features granted by a modern paradigm, some of them have mentioned the performance problems in processing of some models. Aiming at this imperfection, a new method dedicated to modeling the variational design problems was developed by LEDAS and implemented in the LGS 3D product. This new method is called spanning-tree modeling. It represents a geometric model with logical and parametric constraints as a graph in which all objects are vertexes and all constraints are edges. Then a spanning tree of the graph is built. It includes some constraints that make it possible to parameterize efficiently the relative object positions. After that it is sufficient to solve only the constraints that are not presented in the spanning tree (so they form closures of the cycles in the graph) in order to find the configuration of the designed product. It means that only calculations on constraints that cannot be handled by the hierarchical approach are performed. Later we describe the spanning-tree modeling method in detail.

Transformations of objects

In any industrial variational design solver, the set of geometric constraints is transformed sooner or later into a system of non-linear equations that is solved by computational methods. The main variables of the system are new coordinates of geometric objects or angles of rotations and vectors of translations that describe the process of movement of geometric objects into new positions.

3D geometric objects in CAD are complicated enough, so many basic objects (points, curves, planes, etc.) are used for their representation. That is why complex 3D objects are represented during modeling as *rigid sets* – the sets of basic objects that cannot change their relative positions. So the second way of specification of variables is more useful in practice and LGS 3D, instead of figuring out coordinates of primitive objects, tries to find the so-called *transformations* of rigid sets in a form $\{R, \vec{T}\}$, where R is a rotation matrix 3×3, and $\vec{T} = (T_x, T_y, T_z)$ are translations along axes Ox, Oy, Oz. A transformation moves a rigid set from a position X_{old} to a position X_{new}:

$$X_{new} = R \cdot X_{old} + \vec{T}. \tag{1}$$

An arbitrary rotation matrix can be considered as a composition of rotation matrices around the axes Ox, Oy, Oz with the angles α, β, γ, respectively:

$$R = R(\alpha, \beta, \gamma) = R_x(\alpha) \cdot R_y(\beta) \cdot R_z(\gamma). \tag{2}$$

The user constraints are written as equations on new positions of objects X_{new} so they are expressed through rotational α, β, γ and translational T_x, T_y, T_z variables.

In LGS 3D, the basic geometric object (a point, line, circle, etc., except curve and surface) and its spatial position are expressed by the Cartesian coordinates of its anchor point and a unit direction vector (a radius is also provided if needed). For example, a circle is completely defined by its center (the anchor point), the unit normal for its plane (the direction vector) and the radius, while a point is completely defined by its Cartesian coordinates.

Any transformation $\{R, \vec{T}\}$ moves the anchor point $O = (O_x, O_y, O_z)$ and the direction vector of an object $\vec{n} = (n_x, n_y, n_z)$ to a new position O' and $\vec{n}'$:

$$O' = R \cdot O + \vec{T},$$

$$\vec{n}' = R \cdot \vec{n}. \tag{3}$$

For spanning-tree modeling, it is better to represent the transformation T_{child} of one rigid set in a relative parameterized form based on the transformation T_{parent} of another rigid set:

$$T_{child} = T_{parent} \circ Q = T_{parent} \circ L \circ P(\alpha, \beta, \gamma, T_x, T_y, T_z) \circ F, \tag{4}$$

where $P(\alpha, \beta, \gamma, T_x, T_y, T_z) = \{R(\alpha, \beta, \gamma), \vec{T}\}$, $\vec{T} = (T_x, T_y, T_z)$, and the rotation matrix $R(\alpha, \beta, \gamma)$ has the form (2). L and F are some constant transformations which will be described in the section about canonical positions of rigid sets. So the relative transformation Q depends on six variables: $\alpha, \beta, \gamma, T_x, T_y, T_z$.

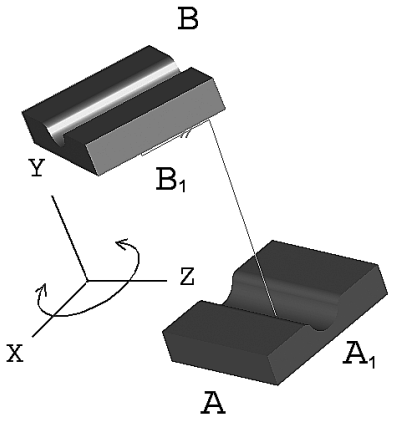

Figure 1. Relative degrees of freedom of two parts

Spanning-tree modeling description

The idea of a new spanning-tree modeling method is that instead of a full set of six variables $\alpha, \beta, \gamma, T_x, T_y, T_z$ it is possible to use a subset that fully parameterizes all possible positions of a rigid set with respect to another one if both rigid sets are connected by some constraint or a set of constraints. A similar idea was used in [5, 6] in the context of the kinematics analysis. For example, it is easy to see that, if two parts **A** and **B** of a mechanism are bounded by a parallelism constraint specified between the planes $\mathbf{A}_1$ and $\mathbf{B}_1$, the position of the part **B** can be only described by four variables. Indeed, for any position of the part **A**, the part **B** has to be placed so that its plane $\mathbf{B}_1$ is parallel to the plane $\mathbf{A}_1$, so it can only rotate around a common perpendicular to the planes $\mathbf{A}_1$, $\mathbf{B}_1$ and be translated in an arbitrary direction. So the part **B** has one rotational and three translational degrees of freedom with respect to the part **A**, see Fig. 1.

In order to treat in the same way other rigid sets of the model, the spanning-tree should be built, where the edge "parent-child" defines that possible transformations of a "child" can be expressed using the "parent" transformation by the formula (4). In this formula $P(\alpha, \beta, \gamma, T_x, T_y, T_z)$ is the parametric transformation, where some of the variables $\alpha, \beta, \gamma, T_x, T_y, T_z$ are fixed due to constraints between "parent" and "child". The number of variables used for this representation is reduced by the number of degrees of freedom for constraints between "child" and "parent". The number of equations is reduced as well, since the constraints that form the edges of the spanning-tree will always be satisfied due to the form of parametric transformations used, so they do not require generation of equations. This is the principal advantage of the proposed spanning-tree modeling method. The

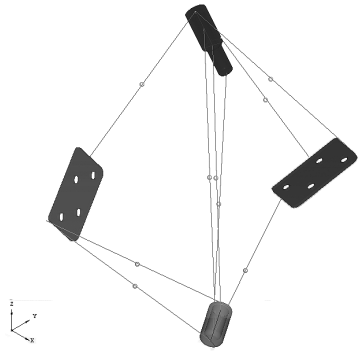

Figure 2. The initial model with 4 bodies and 9 constraints

constraints not included into the spanning-tree will be processed correctly due to the formula (4), so the spanning-tree method is a complete modeling method as compared to iterative solving proposed in [7].

Generally speaking, during the spanning-tree modeling of a geometric constraint satisfaction problem, two algorithmic steps are performed: building of an optimal spanning-tree and generation of a reduced system of equations.

Building of an optimal spanning-tree

At the first step, a multigraph is built using the geometric rigid sets as its vertexes and the constraints between objects of these rigid sets as its edges, see Figures 2 and 3. Each edge is marked by the number of degrees of freedom taken off by the corresponding constraint. E.g., the planes coincidence takes off three degrees of freedom, the plane-point distance – only one, etc. Each vertex is also marked by the number of degrees of freedom that describe positioning of the corresponding rigid set in 3D space.

Then analysis of all pairs of rigid sets and sets of constraints imposed on a particular pair is performed. For example, in Figure 3, for a pair of rigid sets "top-bottom" three constraints should be treated, while for a pair "right-bottom" only one constraint should be processed. During analysis, pattern matching is invoked based on a predefined set of *spanning-tree patterns*. Each pattern contains one, two or three constraints and defines which transformation variables $\alpha, \beta, \gamma, T_x, T_y, T_z$ of a child rigid set can be fixed. Each pattern has its weight equal to the sum of the numbers of degrees of freedom of its constraints. After completion of the analysis, the multigraph is converted to a graph (see Figure 4): for every pair of rigid sets, the set of edges representing constraints between these rigid sets is transformed to a single edge marked by the weight of the matched pattern. The whole base of all possible patterns is finite, and several dozens of the most powerful and generic spanning-tree patterns are enough in practice. If the set N of

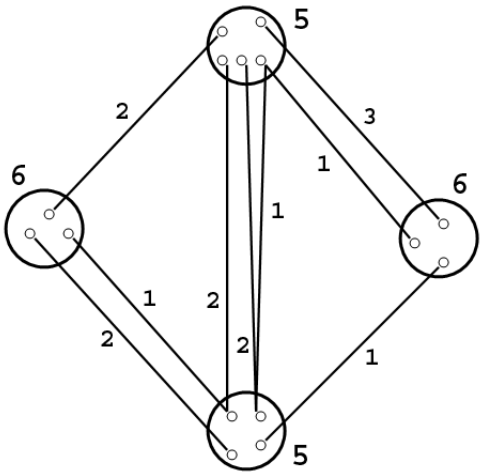

Figure 3. The multigraph with marked edges and vertexes

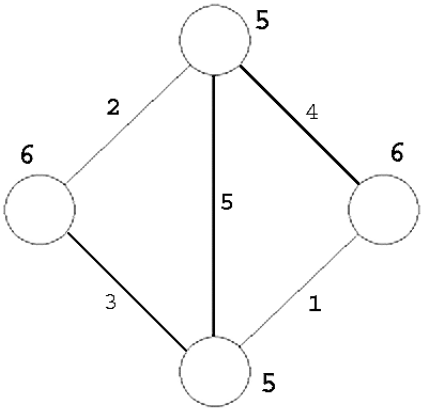

Figure 4. The maximal spanning-tree in a weighted graph

constraints between rigid sets does not match any pattern, then the pattern M that matches a subset of N is considered to be a result. This means that, during generation of the system of equations, the constraints of M do not produce any equations while the constraints from $M \backslash N$ produce equations.

In order to minimize the number of variables and equations in the generated system of equations, the spanning-tree with the maximal total weight should be found in the graph. It can be efficiently done using an algorithm with the computational time $O(n)$, n is the number of constraints in the model, because only six different weights of edges are permitted [8]. For constraints contained in the patterns included in the maximal spanning-tree, there is no need to generate equations, since they will be satisfied automatically using parametric transformations. As a result, the system of equations

will be built only for those constraints of the model that form closures of cycles in the graph. Therefore, the number of equations and variables will be much less.

Canonical positions of rigid sets

Generally speaking, in (4) the constant transformations L, F may be chosen in an arbitrary way (for example, there may be zero movement). However, it appears that for the spanning-tree modeling combined with parameterization (4) there is a more convenient choice of these transformations.

For clarity, let us introduce the notion of a *canonical position* of a rigid set Rs and for simplicity once again restrict the considered cases by the cases with only one spanning-tree constrained object in the rigid set Rs. We mean by a canonical position of the rigid set Rs such a location that the direction vector of the constrained object is equal to some constant canonical vector $\vec{n}_0$ and the anchor point of the constrained object is equal to the constant point O_0. In most cases implemented in LGS 3D the canonical position is

$$\vec{n}_0 = (\pm 1, 0, 0),\ O_0 = (x_0, 0, 0). \tag{5}$$

The transformation F moves the child object from its initial position to the canonical one, and L moves the child from the canonical position to the final position, where a spanning-tree constraint between objects is satisfied.

The transformation L can be seen from a different point of view. Let us bring into consideration the canonical position for a parent object which also has the form (5). Then the transformation L^{-1} moves the parent from the initial position to its canonical location, or equivalently, the transformation L moves the parent from the canonical position to initial one.

The meaning of the canonical position is to represent the spanning-tree constraints as simple fixations of some variables (or degrees of freedom). Other variables define the transformation P, and all equations for the constraints not included into the spanning-tree are described via these variables only. At the same time, the spanning-tree constraints will be satisfied automatically for all values of non-fixed variables.

Let us, for example, require that the distance between two planes pl1 and pl2 be equal to d, see Figure 5. Let pl1 be a parent and pl2 be a child. Remember that rigid sets containing these planes can also include other objects in addition to pl1 and pl2.

We choose the canonical position of pl1 in the form $\vec{n}_0 = (1, 0, 0),\ O_0 = (x_0, 0, 0)$, and the canonical position of pl2 in the form $\vec{n}_0 = (\pm 1, 0, 0),\ O_0 = (x_0 \pm d, 0, 0)$, x_0 is an arbitrary value and signs are defined by constraint orientations. In case of lack of user specifications for orientations, LGS 3D will define them on the basis of the initial object positions. Let F and L^{-1} be transformations providing movements to the canonical positions.

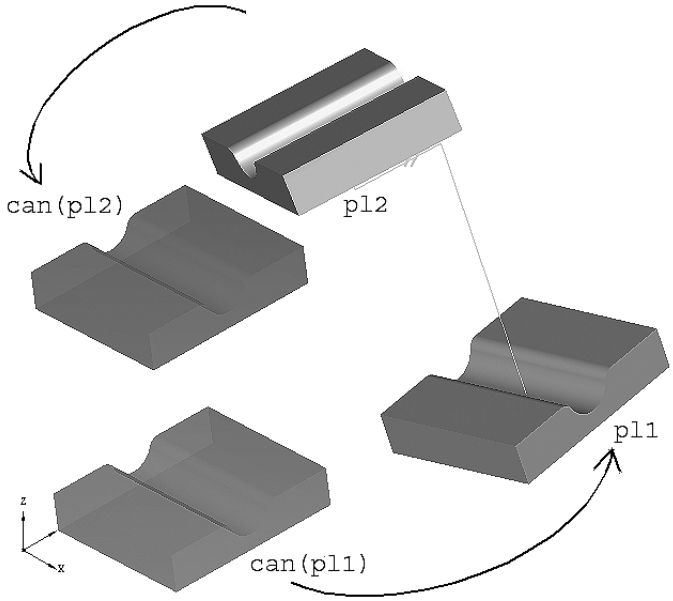

Figure 5. Computation of constant transformations

After execution of these movements, both objects are in canonical positions. Let us consider which form has possible transformations P for the plane pl2 such that the distance constraint to the canonical position of pl1 is satisfied. Obviously, every such transformation is a combination of a rotation of pl2 around the axis Ox and translations along the axes Oy and Oz. Thus, $P = P(\alpha, 0, 0, 0, T_y, T_z)$, and the number of degrees of freedom is decreased to three.

It is easy to see that the transformation of the child (4) with chosen L, P, F actually transfers it into a position, where the distance constraint is satisfied for any values of free parameters. On the contrary, each transformation which moves pl2 to the desired location is defined by some of (α, T_y, T_z).

Naturality of a solution

Note that we have a free choice for F and L, since there are infinitely many ways of transferring an object from one position to another. So, in order to choose F and L, LGS 3D is guided by considerations of naturality. Naturality of a solution is the property of final positions of objects in the model "to be close" to their sketch positions. It means that if it is possible to satisfy all constraints without additional movement of some objects, then this is to be done by the solver.

It is easy to see that zero values of free variables correspond to the transformation which is the most natural among those satisfying the spanning-tree constraints. In particular, if in the initial state the spanning-tree constraints were satisfied, then after the assignment of zero values to free variables the objects will not move. In the industrial models, many constraints are usually not violated in the initial position, so naturality is an important issue. The methods aimed at computation of more natural solutions have better performance as well, since the generated systems of equations are solved using better starting points.

After processing the requirements of naturality, the *final form* of F and L appears. Let $(O_{nat}, \vec{n}_{nat})$ be the position for the most natural solution of the problem restricted to only the spanning-tree constraints. Then the transformation F is the shortest way to move a child to the canonical position and L is also the shortest way from the canonical position to the state $(O_{nat}, \vec{n}_{nat})$.

In the above example of distance between two planes, the naturality condition requires an optimal selection of x_0 which can be equal to the initial x-coordinate of pl2. In fact, it is the requirement of the most natural movement to the canonical position which in this case is defined by the choice of the canonical position itself.

Consideration of non-spanning-tree constraints could lead to further optimization of naturality of the transformations F and L.

The case of several constraints

The case of two or more constraints imposed on two rigid sets and connecting the basic objects of these rigid sets is very popular in practice. In this case, the canonical position for a rigid set is chosen by completely similar reasons: someone should find the position such that constraints could be satisfied just by fixation of P variables. This problem is not so trivial [9] but it can be solved for all practical configurations of constraints.

Let us consider a case where two constraints are imposed on objects of two rigid sets. Let two lines $Ln1$ and $Ln2$ belong to a "child" rigid set $Rs1$, a pair of lines $Ln3$ and $Ln4$ lie inside a "parent" rigid set $Rs2$, and the coincidence constraints between $Ln1$ and $Ln3$, as well as between $Ln2$ and $Ln4$, are imposed. In addition, collinearity of the direction vectors of $Ln1$ and $Ln3$, as well as of $Ln3$ and $Ln4$, is needed. Suppose that the problem is consistent. Suppose also that $Ln1||Ln2$ and $Ln3||Ln4$, otherwise there is only one solution.

Assume that both bodies are in their canonical positions so that constraints are satisfied. Thus there remains only one degree of freedom corresponding to a shift of the pair of lines $Ln1$ and $Ln2$ along the common direction vector of all four lines.

Now the canonical position can be easily selected. For example, assume that in the canonical position $Ln1$ coincides with the axis Oy, $Ln2$ is parallel to the axis Oy, intersects the axis Oz and lies at the distance to Oy equal to the distance between these lines. The y-coordinate of the anchor point of $Ln1$ is chosen so that movement to the canonical position is the shortest one. The only remaining degree of freedom is the translational variable T_y. The matrices L and F are built according to the special rules described above.

Difficulties of finding canonical positions

Unfortunately, a suitable canonical position exists not for all cases. One of counterexamples is the incidence constraint for a point P and a line L if L is considered as a child. There are all three rotational degrees of freedom for L and only one translational along the direction vector of the line. Our parameterization combined with the transformation form (3) cannot describe this case by only fixation of some variables. This problem could be solved in two ways: by changing parameterization or by replacing the anchor point transformations form by the form

$$O' = R \cdot (O + \vec{T}). \tag{6}$$

The order of translation and rotation is changed in this form of transformations: now translation is performed before rotation. After that the canonical position $\vec{n}_0 = (1, 0, 0),\ O_0 = (0, 0, 0)$ is suitable with the remaining degrees of freedom $(\alpha, \beta, \gamma, T_x)$. This idea promises further performance growth and decrease in the size of the system of equations.

The line-plane distance constraint is another "troublesome" case. Let us first consider that a "child" is a line. Let a line Ln and a plane Pl be in their canonical positions and the constraint be satisfied. The remaining degrees of freedom relate to a movement of the line (or rather, the rigid set with the line) around itself, a rotation in a plane parallel to Pl and a displacement parallel to Pl for four degrees of freedom in total. If the line in the canonical position is directed as the axis Ox and the plane normal is along Oy, then the remaining degrees of freedom correspond to rotations around Ox and Oy. However, the transformation with the rotation matrix (2) firstly turn the line around Oy and then around Ox, breaking perpendicularity to Oy. In order to obtain a correct solution, it is necessary to direct the line along Oy and the plane normal along Ox, so the line will firstly turn around itself, then around the plane normal, and perpendicularity will hold. Thus the choice of the canonical position may depend even on the order of rotations.

It should be noted that a suitable canonical position for parameterization does not exist when a "child" is a plane for the same reason as in the point-line coincidence example discussed above. Nevertheless, using the alternative type of transformations (6) solves this problem.

Patterns implemented in LGS 3D

The spanning-tree patterns for one constraint are called single patterns; patterns for two constraints between rigid sets are called double patterns and so on.

Currently in LGS 3D only 6 single and 17 double patterns are implemented. Single patterns are implemented for the following constraints:

1. points coincidence $(0 + 3)$;
2. lines coincidence $(2 + 2)$;
3. plane-plane distance $(2 + 1)$;
4. parallelism $(2 + 0)$;
5. point-line coincidence in case when a "child" is a point $(0 + 2)$;
6. spheres coincidence $(0 + 3)$.

The numbers in parentheses are the quantities of reduced rotational and translational degrees of freedom, respectively. The sum of these numbers is equal to the number of equations that are excluded from the system of equations.

The most frequently encountered double patterns are:

1. Two line-line coincidence constraints. In the general case, lines are not parallel so the relative position of rigid sets is fully determined. In the case of parallel lines, much more often encountered in practice, only one translational degree of freedom remains.
2. Two plane-plane distances. In the general case, planes are not parallel and only one translational degree of freedom remains (along the line of intersection of planes). In the popular case of parallel planes, three degrees of freedom remain but, as compared to the pattern with a single distance constraint, this pattern may provide additional information about mutual orientation of planes.
3. Lines coincidence + planes distance. In the popular case of perpendicular lines and planes inside rigid sets, one rotational degree of freedom remains (around the common direction vector). If the lines are parallel to the planes, then only one translational degree of freedom remains (along the line itself). In the general case, the relative position of rigid sets is fully determined.
4. Lines coincidence + parallelism of two other objects. In the case when lines are parallel to other objects inside rigid sets, one translational and one rotational degree of freedom remains. In the general case, only one translational degree of freedom remains (along the line).

Generation of a reduced system of equations

A correct generation of equations for the remaining constraints requires creation of objects of the dedicated class "equation on parametric transformations". This class implements the methods for computation of discrepancy and its gradient on the basis of current values of all variables – these values are substituted to parametric transformations. Discrepancies of all equations and their gradients are then used for solving a system of non-linear equations by Newton's method.

The complexity of calculation of discrepancy and its gradient for such equations is greater than that of the standard modeling method. For every constraint it is needed to perform as many 3×3 matrix multiplications as the length of a path in the spanning-tree between objects that are arguments of these constraints. Nevertheless, all transformations and their partial derivatives for all objects can be found at once with quadratic complexity using optimization of computation by processing the vertexes in the spanning-tree from its root to leaves. The overall complexity of computation of the Jacobian matrix and right hand side vector also will be at most quadratic. While solving the non-linear system of equations by Newton's method, the most time-consuming operation is solving the linear systems, which is cubic in time, so our new spanning-tree method has an acceleration effect.

As the formula for the "child" transformation using "parent" transformation (4) requires the definition of constant matrices L and F, they are analytically found before the equation generation.

Conclusion

Experimental results for the spanning-tree modeling method are shown in Tables 1, 2. The testing framework used to estimate efficiency of the implemented method includes 3120 cases. The biggest 14 cases constitute the so-called "Big cases" group. Most of them are generated by CAD industrial problems. The average size of the models is several tens of geometric objects and around a hundred constraints on them.

In Table 1, the spanning-tree modeling is compared with 6-variable Cartesian modeling. We see that the spanning-tree modeling reduces the sizes of systems of equations and considerably speeds up the process of solving (see also Figure 6).

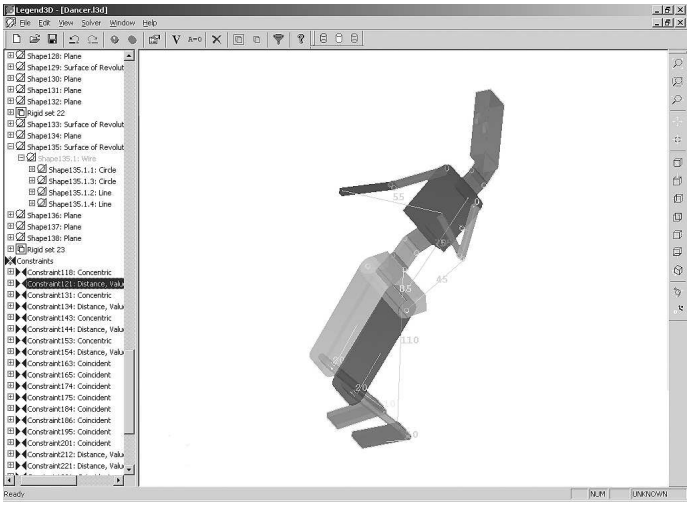

Figure 6. In LGS 3D, a model with 22 constraints is described by 9 equations only

Table 1. The comparison with 6-variable Cartesian modeling

Test base	Modeling method	Average quantity of variables	Average quantity of equations	Total computation time, sec.	Success rate
Main base	6-variable	8.48	7.12	695.05	**95.10** %
Big cases	Cartesian	29.57	37.25	242.02	**71.43**%
Total	modeling			**937.07**	
Main base	Spanning-	5.46	3.04	187.13	**97.52**%
Big cases	tree	16.46	18.25	115.12	**78.57**%
Total	modeling			**302.25**	

Along with these two modeling methods, the LGS solver uses many other techniques to increase efficiency: different kinds of geometric decomposition, geometric simplification algorithms and acceleration methods for solving non-linear equations [10]. Therefore, the overall effect of implementation of our new method is much more complex. The results of comparative testing of the LGS solver before and after plugging in the spanning-tree method showed that the total solution time dropped twice, see Table 2. Moreover, the percentage of cases successfully solved by LGS (see the column "success rate") significantly increased.

Table 2. The effect of plugging in the spanning-tree modeling method

Test base	Solver configuration	Total computation time, sec.	Success rate
Main base	Before plugging of spanning-tree modeling	77.9	**98.55**%
Big cases		478.18	**85.71**%
Total		**556.07**	
Main base	After plugging of spanning-tree modeling	52.78	**99.16**%
Big cases		235.98	**92.85**%
Total		**288.75**	

We think that this progress is due to combination of positive features of hierarchical and variational design methods in the new spanning-tree modeling method. Our plans of future development of the method include extension of the set of predefined spanning-tree patterns, minimization of the height of the spanning-tree and many other issues.

References

[1] Klein R. The Role of Constraints in Geometric Modelling // Geometric Constraint Solving and Applications / Ed. by B. Bruderlin, D. Roller. — Berlin, 1998. — P. 3–23.

[2] Ushakov D. Adding Intelligence to Software Solutions for PLM: constraint-based approach // Int. J. of Product Lifecycle Management. — 2006. — Vol. 1, N 2. — P. 181–193.

[3] Hoffmann C. M. Constraint-Based Computer-Aided Design // J. of Computing and Information Science in Engineering. — 2005. — Vol. 5, N 3. — P. 182–187.

[4] Ershov A., Ivanov I., Preis S., Rukoleev E., Ushakov D. LGS: Geometric Constraint Solver // Int. Conf. Perspectives of Systems Informatics, Novosibirsk, 2003. — Berlin, 2003. – P.423–430. — (Lect. Notes Comput. Sci.; Vol. 2890).

[5] Kim J., Kim K., Choi K., Lee J. Solving 3D Geometric Constraints for Assembly Modelling // Int. J. of Advanced Manufacturing Technology. — 2000. — Vol. 16. — P. 843–849.

[6] Kim J., Kim K., Lee J., Jeong J. Generation of Assembly Models from Kinematic Constraints // Int. J. of Advanced Manufacturing Technology. — 2005. — Vol. 26. — P. 131–137.

[7] Kim J., Kim K., Lee J., Jung H. Solving 3D Geometric Constraints for Closed-Loop Assemblies // Int. J. of Advanced Manufacturing Technology. — 2004. — Vol. 23. — P. 755–761.

[8] Kasyanov V.N., Evstigneev V.A. Graph Theory for Programmers. — Springer, 2000. — 432 p.

[9] Shi Z., Chen L. An Angular Constraints Solving Approach for Assembly Modeling Based on Spherical Geometry // Int. J. of Advanced Manufacturing Technology. — 2007. — Vol. 32. — P. 366–377.

[10] Ershov A., Ivanov I., Kazakov A., Lipski S., Markin V., Preis S., Rukoleev E., Sidorov V., Ushakov D. LEDAS Geometric Solver: Product Overview. — Novosibirsk, 2003. — 56 p. — (Prep. / LEDAS Ltd.; N 5).