DISTANCE CONSTRAINT SATISFACTION BASED ON GEOMETRIC RELAXATION

Lluís Solano

Polytechnical University of Catalunya

Computer Graphics Section. Software Department

Barcelona. Spain

solano@lsi.upc.es

Pere Brunet

Polytechnical University of Catalunya

Computer Graphics Section. Software Department

Barcelona. Spain

pere@lsi.upc.es

Abstract In this paper, we characterize the family of 2D models with distance and angle constraints that can be transformed into equivalent models defined only in terms of distance constraints. The transformation process will be called the *d-reduction* process. Different geometric relaxation strategies for solving 2D models with only distance constraints are also presented and discussed.

Keywords: Geometric constraints, angle and distance constraints conversion, constraint solver, geometric relaxation.

Introduction

Constraint-based modeling allows defining families of objects (generic objects) that can be subsequently converted into specific objects by giving the values of the generic object parameters and by solving the defined constraints (Rossignac et al. 1989, van Emmerick 1990).

In the literature, there are different approaches to solve geometric constraints which can be classified into two main strategies: the constructive one and the equational or numerical one. For more details, see Bouma et al. (1995) or Solano and Brunet (2000).

In this paper, we deal with 2D models with n points and m geometric constraints that can be distances or angles. In order to have a basic and single

constraint solver, we define a process to obtain the 2D model in terms of only distance constraints. In the following section, some definitions are made and the transformation process is defined in sections 3 to 5. Section 6 presents the iterative relaxation techniques that have been used for solving *d-reduced* constraint models. Finally, some examples of the transformation process are shown.

1. The *d-reduced* Model and the *d-reduction* Process

In the case of 2D geometric models with points, the necessary condition for the system to be well-constraint is (Owen 1996)

$$2n - 3 \geq e, \text{ for all subgraphs.} \quad (1)$$

where n is the number of nodes (points) and e denotes the number of edges (constraints).

This condition is also sufficient (Laman 1970).

Definition 1.1. A *3P-angle* model is a model whose geometric elements are points, and the geometric constraints are either distances between points or angles between two straight segments with a common endpoint.

Definition 1.2. A *d-reduced* model is a model whose geometric elements are points, and the geometric constraints are distances between them.

Definition 1.3. Let $\mathcal{M}$ be a *3P-angle* model. A *reduction triangle* in $\mathcal{M}$ is a triangle defined by the three different endpoints of two segments with an angle constraint between them.

In what follows in sections 2, 3 and 4, we consider only *3P-angle* models whose reduction triangles have only one angle constraint.

Definition 1.4. Let $\mathcal{T}$ be a reduction triangle of a *3P-angle* model $\mathcal{M}$. A *reduction distance* of $\mathcal{T}$ is an edge between two different vertices of $\mathcal{T}$ on which no distance constraint is defined.

For instance, in Figure 1, the pairs of points $\{P_5, P_6\}$ and $\{P_6, P_4\}$ in the reduction triangle $\mathcal{T} = P_4P_5P_6$ defined by the angle constraint α_3^c , are reduction distances.

Definition 1.5. Let $\mathcal{M}$ be a well-constraint *3P-angle* model and let $\mathcal{T}$ be a reduction triangle in $\mathcal{M}$. A *d-reduction step* is defined as the process that replaces the angle constraint in $\mathcal{T}$ with one reduction distance of $\mathcal{T}$ and obtains a reduced triangle $\mathcal{T}'$. The replacement is performed in such a way that the solution of $\mathcal{T}'$ is also a solution for $\mathcal{T}$.

Now, we define the d -reduction process.

Definition 1.6. A d -reduction process is defined as a sequence of d -reduction steps.

Definition 1.7. Let $\mathcal{M}$ be a $3P$ -angle model. $\mathcal{M}'$ is a d -reduced model of $\mathcal{M}$ iff there exists a reduction process that generates $\mathcal{M}'$ from $\mathcal{M}$. In this case, we say that $\mathcal{M}$ is d -reducible.

Figure 1. Left: A $3P$ -angle model. Right: A derived d -reduced model.

Figure 1 illustrates these concepts. The model $\mathcal{M}'$ on the right side is a d -reduced model of the model $\mathcal{M}$ on the left side. The constraints in the model $\mathcal{M}'$ are the d_i^c and d_i^r distances. The Figure also includes the initial α_j^c for clarification.

2. d -reducible $3P$ -angle Models

Let us first prove the following property.

Property 2.1. Let $\mathcal{M}$ be a well-constraint $3P$ -angle model. Then, every reduction triangle $\mathcal{T}$ in $\mathcal{M}$ has at least one reduction distance.

Proof. For a contradiction, assume that reduction triangle $\mathcal{T}$ has no reduced distances. This means that there are three distance constraints between the points that define $\mathcal{T}$, one on each triangle edge, plus an angle distance. Hence, $\mathcal{T}$ is over-constraint and so is $\mathcal{M}$. See Figure 2. $\square$

Next we introduce the concept of *reduction-graph* induced by a well-constraint $3P$ -angle model $\mathcal{M}$.

Figure 2. Reduction triangles with no reduction distances are over-constrained.

Definition 2.1. Let $\mathcal{M}$ be a well-constraint 3P-angle model. With $\mathcal{M}$ we associate the graph $R(V, E)$, where the set of vertices V is the set of reduction triangles in $\mathcal{M}$, and the set of edges is the union of two sets, $E_V = E_T \cup E_\emptyset$. E_T is the set of pairs $\{\mathcal{T}_i, \mathcal{T}_j\}$, with $i \neq j$, such that the reduction triangles $\mathcal{T}_i$ and $\mathcal{T}_j$ share a reduction distance. $E_\emptyset$ is the set of pairs $\{\mathcal{T}_i, \mathcal{T}_i\}$ where $\mathcal{T}_i$ is a reduction triangle whose reduction distance is not shared with any other reduction triangle in V .

We will label each reduction triangle $\mathcal{T}$ in V with the angle constraint associated with $\mathcal{T}$, and the edges in E with the reduction distance constraint that defines the edge. A reduction triangle whose reduced distance is not shared with any other reduction triangle produces a one-edge cycle in the vertex graph. In Figure 3 there is an example of the reduction-graph associated to a model.

We can now prove the following theorem.

Theorem 2.1. Let $\mathcal{M}$ be a well-constraint 3P-angle model and let $R(V, E)$ be its reduction-graph. If $R(V, E)$ is a set of disconnected graphs where each graph is cycle-free except one one-edge cycle, then there exists a sequence of d -reduction steps that transforms $\mathcal{M}$ into a d -reduced model, $\mathcal{M}'$. That is, $\mathcal{M}$ is d -reducible.

Proof. If each connected component of R has just one one-edge cycle, it can be assumed that the graph is acyclic. There are no sequences of different graph nodes that form a cycle. Then each connected component can be represented as a tree. In each one of the trees associated to the reduction-graph R , the root node is represented by the node with the one-edge cycle.

In each of these trees, every node except the root has one and only one parent, and at most two descendents, since every node can have a maximum of three edges corresponding to its reduction distances.

Let us assume that a postorder tree traversal is performed. In terminal nodes, the reduction triangle must have two known distances d^k . The reduction dis-

Figure 3. A 3P-angle model and the induced reduction graph.

tance represented by the edge to the parent node can be computed (and the node can be reduced) by a trigonometric rule:

$$d^r = f(\alpha^c, d_i^k, d_j^k) \quad (2)$$

In the postorder traversal, once the terminal nodes have been reduced, their parents are the new terminal nodes of the tree. Therefore, the recursive use of the reduction rule (2) reduces all the nodes in the tree.

Consequently, if the reduction-graph can be arranged as a tree or set of trees, for each node it is possible to express every angle constraint as a reduction distance constraint, and the initial model is *d-reducible*. $\square$

Figure 4 shows the reduction-graph of Figure 1. In this case there is no cycle. There is one tree with only one one-edge cycle. In order to achieve the *d-reduced* model, d_6^r and d_7^r can first be computed from the terminal nodes:

$$d_6^r = f(\alpha_1^c, d_1^c, d_2^c)$$

$$d_7^r = f(\alpha_2^c, d_3^c, d_4^c)$$

d_6^r is the reduction distance of α_1^c and d_7^r of α_2^c . From these two distances and α_4^c it is possible to compute d_8^r . It is the reduction distance of α_4^c :

$$d_8^r = f(\alpha_4^c, d_6^r, d_7^r)$$

Finally, d_9^r can be computed from d_8^r , d_5^r and α_3^c :

$$d_9^r = f(\alpha_3^c, d_8^r, d_5^c)$$

Figure 4. Reduction-graph of Figure 1

Corollary 2.1. The postorder traversal of each associated tree in the reduction-graph, from the terminal nodes to the root node, gives the sequence to follow in the d -reduction process.

In Figure 4, the postorder traversal of the tree shows the evaluation order for solving the initial model $\mathcal{M}$. In the tree, the evaluation order of α_1^c and α_2^c is not relevant.

Property 2.2. In a well-constraint 2-dimensional model, the application of simple d -reductions in the d -reduction process does not produce an over-constraint model.

Proof. Given the situation shown in Figure 5, in a reduction triangle there are three points P_i, P_j, P_l , an angle constraint α^c and two known distances d_{ij}, d_{il} involved (d_{ij} and d_{il} can be either constraint distances fixed by the user or reduction distances computed in previous steps). The simple d -reduction will change the angle constraint by the reduction distance d^r .

If a model with n points is well-constraint, the number of constraints for any subgraph with m points ($m \leq n$) verifies:

$$r_m \leq 2m - 3$$

Taking into account the set s_k of the points connected with P_j, P_l through existing constraints (Figure 5), then the set s_k is a subgraph of the constraint graph that contains P_j and P_l . The number of constraints in the set s_k is

$$r_k < 2k - 3$$

Figure 5. Simple d -reduction situation

The proof is based on demonstrating that the case $r_k = 2k - 3$ is not possible.

In fact, when we consider the set s_k , and we add the point P_i , we have one more point and at least three more constraints ($d_{ij}^c, d_{il}^c, \alpha^c$). In this case, if the number of constraints in s_k is $r_k = 2k - 3$, in the set $s_k + \{P_i\}$ there are:

$$r_{k+1} = 2k \text{ constraints}$$

but over $k + 1$ points the number of constraints must be:

$$r_{k+1} \leq 2k - 1$$

Therefore, $r_k = 2k - 3$ is not possible if the model is well-constraint. We conclude that for the set s_k :

$$r_k < 2k - 3$$

Now, the replacement of α_c for d^r in Figure 5 is always possible: the insertion of d^r in s_k will lead to $r_k \leq 2k - 3$, and this does not produce any over-constraint situation in the model. $\square$

3. Examples of d -reduced models

In this section, we show some examples of the reduction-graph associated to well-constraint models with distance and angle constraints.

In Figure 6, there is a model with angle and distance constraints that has the reduction-graph shown in Figure 7. The associated reduction-graph is a disconnected graph. In this case, it is not possible to apply the d -reduction process, because there are subgraphs with more than one one-edge cycle.

Figure 6. A constraint model that has an associated disconnected reduction-graph

Figure 7. An example of a disconnected reduction-graph

The model shown in Figure 8 is an example of a connected reduction-graph with more than one one-edge cycle, which is not possible to d -reduce. Figure 9 shows the associated reduction-graph.

Figure 10 shows a constraint model with a reduction-graph that is disconnected (Figure 11, but it can be d -reduced).

4. Model conversion from $3P$ -angle form to d -reduced form

Let us assume a general 2D $3P$ -angle model with distance constraints between two points, and angle constraints between two vectors with coincident end points.

As already stated, every angle constraint belongs to a reduction triangle. By considering the number of angle and distance constraints involved, reduction triangles can have the configurations shown in Figure 12. The cases in the first

Figure 8. Non-d-reducible constraint model

Figure 9. Example of a connected reduction-graph with more than one one-edge cycle

row (case D) are in fact not reduction triangles (the model is locally *d-reduced*). The cases in the second row (such as case C, with one angle constraint) can be reduced using theorem 2.1 and the algorithm from corollary 2.1. The cases marked “oc” (overconstraint) are impossible, since they mean that the model is over-constraint, according to the condition (1). The cases with three angle constraints are either inconsistent (if the angles do not add up to π) or they are equivalent to the cases with two angle constraints (the third angle can be deduced from the first two angles). Therefore, the general *d-reduction* algorithm must only consider the cases valid for theorem 2.1, plus two extra cases with two angle constraints: cases A and B in Figure 12.

3P-angle models with case B reduction triangles can be reduced by applying a simple pre-process to these triangles. After the pre-process, the model has no reduction triangle of type B, and the model can be reduced using corollary 2.1.

Figure 10. Model with a disconnected reduction-graph that can be d-reduced

Figure 11. Example of d-reducible disconnected reduction-graph

The pre-process is based on the use of the trigonometric rule that computes the remaining two distances in every type *B* reduction triangle:

$$(d_1^r, d_2^r) = f2(\alpha_1^c, \alpha_2^c, d^c)$$

This rule can be applied independently to every reduction triangle of type *B*, as they are independent: given any two reduction triangles of type *B*, they cannot share a common reduction distance (two triangles of this type sharing a common reduction distance are over-constraint, and this does not fulfill the conditions fixed in (1)). On the other hand, the application of this rule cannot produce an over-constraint model, due to the same arguments used in property 2.2.

Although particular models containing type *A* reduction triangles can be successfully reduced, we have no general rule for reducing them. In fact, there are models (see Figure 13) with type *A* reduction triangles that must be solved using a global system of equations.

The configurations shown in Figure 12, characterize the families of 2D models that can be *d-reduced* except general models including some configurations of type *A*.

Number of distance constraints (d^c)				
	0	1	2	3
Number of angle constraints(α^c)	0			D
	1		C 	oc
	2	A 	B 	oc
	3	A	B	oc

Figure 12. Configurations of reduction triangles in 3P-angle models

Figure 13. Example of a model with type A reduction triangles

5. Geometric Relaxation for distance constraint solving

In this section, in order to solve 2D models with distance constraints, iterative relaxation techniques are presented and discussed. The solver that we have implemented, works with well-constraint 2D models without degrees of freedom, with only distance constraints defined between points. Although the solver is presented with 2D models, it can be extended to 3D models.

Definitions

Given a 2D system with n points and m constraints $m = 2n - 3$, we can define the **difference vector** $\vec{\delta}$, as the difference between the state vector $\vec{d}$ of the actual values of the m distances and the equilibrium vector $\vec{d}^c$ of the target values of the m distance constraints.

$$\vec{\delta} = \vec{d} - \vec{d}^c = (\delta_1 \dots \delta_m)^T$$

The **Square error** E_T is defined as the square of the euclidian norm of the difference vector:

$$E_T = \sum_i \delta_i^2 = \|\vec{\delta}\|^2$$

Every constraint defined in a model, has an actual value and an equilibrium value. The euclidean norm of the *difference vector* $\vec{\delta}$ is a measure of the constraint satisfaction of a model.

Given a 2D well-constraint model with n points and $m = 2n - 3$ distance constraints, the constraints are fulfilled iff $\vec{\delta} = 0$.

Geometric relaxation strategies

Next, different constraint solver strategies are proposed for systems with only distance constraints based on iterative geometric relaxation methods. In this way, provided that $\|\vec{\delta}\| \geq \varepsilon$ at each iteration, the constraint solver relaxes points or constraints.

This means that at each step the solver has to decide to select one constraint to relax or to perform a global point relaxation in order to move all the points slightly.

The following cases can be addressed:

- **One-point relaxation:** Given a point P in the model, if the $\text{grad}(\vec{\delta}) \neq 0$ is not null, P can be moved in the opposite direction to the gradient vector. In this way, the maximum decrease of $\|\vec{\delta}\|$ is achieved. The problem arises when there is a local minimum, and the gradient vector is null for all points. This case has not been implemented because very often the relaxation process falls within a local minimum (Witkin et al. 1987). There are even cases in which at the initial state all the points are in a local minimum. In these cases, the gradient at each point is null.
- **One-constraint relaxation:** It is also possible to relax one constraint d_i at every iteration. In this case, local minima can also be reached. When the constraint relaxed d_i is the maximum valued component of $\|\vec{\delta}\|$, convergence conditions can be established (Solano 1999), and a good behavior of the convergence of the solver has been experimentally observed.
- **Global point relaxation:** In this case a space deformation is applied in order to preserve the decreasing of $\|\vec{\delta}\|$. The global deformation (for more

details see (Solano and Brunet 2000, Solano 1999)) works by converting the constraint system to a representation based on the set of relative vectors defined by

$$\vec{R}_i = \frac{P_{i2} - P_{i1}}{d_i^c}$$

P_{i1} and P_{i2} being the endpoints of the distance constraint d_i . Using this representation, it can be shown that it is always possible to compute a displacement of all points P_j of the system in a way such that

$$\|\vec{R}_i\| \rightarrow 1 \quad \forall i$$

The displacement computation requires solving a linear system of equations at each iteration. In the limit, the relaxation algorithm produces $\|\vec{R}_i\| = 1 \quad \forall i$.

- Global constraint relaxation: This case would require a solution for all the constraints at the same time. Therefore, it is unaffordable.

As a consequence, a hybrid iterative constraint solver based on relaxation has been developed. The relaxation process is performed in the following way. While $\|\vec{\delta}\| \geq \varepsilon$, at each iteration, the constraint solver works on points or constraints. This means that the solver in each step has to decide to select one constraint to relax or to perform a space deformation in order to move slightly all the points.

The following algorithm has been developed in order to perform the relaxation process:

algorithm *Hybrid Constraint relaxation*

while $\|\vec{\delta}\| > \varepsilon$

 Choose the constraint i such that $|\delta_i|$ is maximum

 Relax the constraint i

 From the new point locations, update the difference vector

$\vec{\delta}$ and compute $\vec{\delta}$

if $\vec{\delta}$ decreases, then validate the new point locations

else compute a global point relaxation in order to move the

 whole set of points and decrease $\vec{\delta}$

endif

endwhile

endalgorithm

Following there is an example that shows the evolution of $\|\vec{\delta}\|$ in the case of a set of six points that initially form a regular hexagon (Figure 14). In Figure 15, one final solution is shown.

Between the points, the following distance constraints have been defined:

Figure 14. Initial hexagon configuration

Figure 15. Final double triangular configuration

$$d_1^c = \text{distance}(P_1 P_2)$$

$$d_2^c = \text{distance}(P_1 P_4)$$

$$d_3^c = \text{distance}(P_1 P_6)$$

$$d_4^c = \text{distance}(P_2 P_3)$$

$$d_5^c = \text{distance}(P_5 P_6)$$

$$d_6^c = \text{distance}(P_3 P_4)$$

$$d_7^c = \text{distance}(P_4 P_5)$$

$$d_8^c = \text{distance}(P_3 P_6)$$

$$d_9^c = \text{distance}(P_2 P_5)$$

In this example, and due to the constraint values used, the initial state is in a local minimum in every point.

Figures 14 and 15 show the initial and the final configuration of the system. In Figures 16 and 18, the evolution of $||\vec{\delta}||$ at each iteration using the iterative global relaxation solver and the hybrid method proposed are shown. In both cases, the local minima are also successfully avoided using the proposed methods. Figure 17 shows the evolution in a one-constraint relaxation case.

6. Conclusions

The *d-reduction* process method for transforming models with distance and angle constraints into equivalent models with only distance constraints has been presented. The *d-reduction* is a preprocess for obtaining a model with only distance constraints.

The family of models that can be *d-reduced* using the presented scheme has been characterized. It has been proved that the *d-reduction* process does not produce over-constraint situations.

Different geometric relaxation strategies that can be used for solving distance constraints have been presented and discussed.

References

- Bouma, W.; Fudos, I; Hoffman, C.; Cai, J. and Paige, R. (1995) *A Geometric Constraint Solver*. CAD, vol.27, no.6, pp.487-501.
- Emmerick, M.J.G.M. (1990) *Interactive Design of Parameterized 3D Models by Direct Manipulation*. PhD thesis, Delft University Press.
- Laman, G. (1970) *On Graphs and Rigidity of Plane Skeletal Structures*. Journal of Engineering Mathematics, vol.4, no.4.
- Owen, J.C. (1996) *Constraints on Simple Geometry in Two and Three Dimensions*. International Journal of Computational Geometry & Applications, vol.6, no.4, pp.421-434.
- Rossignac, J.R.; Borrel, P. and Nackman, L.R. (1989) *Interactive Design with Sequences of Parameterized Transformations*. Intelligent CAD Systems II. V.Akman, P.J.W.ten Hagen, P.J. Veerkamp (eds), pp.93-125, Springer-Verlag.

- Solano, L. (1999) *Constructive Solid Modeling Based on Constraints*. PhD Thesis. Universitat Politècnica de Catalunya.
- Solano, L. and Brunet, P. (2000) *A Geometric Relaxation Solver for Constraint-Based Models*. In CAD Tools and Algorithms for Product Design. P.Brunet, C.Hoffman and D.Roller (Eds), pp.251-268. Springer-Verlag.
- Witkin, A.; Fleischer, K. and Barr A. (1987) *Energy Constraints on Parameterized Models*. Computer Graphics, vol.21, pp.225-232.

Figure 16. $\|\vec{\delta}\|$ evolution with global point relaxation

Figure 17. $\|\vec{\delta}\|$ evolution with one-constraint relaxation

Figure 18. $\|\vec{\delta}\|$ evolution with hybrid method