Intuitive Control of Deformable Object Simulation using Geometric Constraints

Min Hong[1] Min-Hyung Choi[2] Ravi Yelluripati[2]
Min.Hong@UCHSC.edu *minchoi@acm.org* *yrk@acm.org*

[1] *Bioinformatics, University of Colorado Health Sciences Center, 4200 E. 9th avenue Campus Box C-245, Denver, CO 80262, USA*

[2] *Dept. of Computer Science and Engineering, University of Colorado at Denver, Campus Box 109, PO Box 173364, Denver, CO 80217, USA*

Abstract

This paper describes an effective and intuitive method to control deformable objects using geometric constraints. To achieve a natural and plausible interaction with deformable objects and to setup the desirable initial conditions of simulation, users should be able to define and control the geometric constraints intuitively. Users should be able to utilize the simulation as a problem solving platform by experimenting various simulation situations without major modification of the simulator.

The proposed constraint-based simulation system solves the problem using a non-linear FEM approach to represent deformable objects and constraint forces are generated by defining geometric constraints on the nodes of the object to apply the restriction. It allows users to define and modify geometric constraints and an algorithm converts these geometric constraints into constraint forces which seamlessly integrate controllability to the simulation system. Simulator can handle linear, angular, inequality based geometric constraints on the objects. Our experimental results show that constraints are maintained in the tight error bound and preserve desired shape of deformable object during the entire simulation.

1. Introduction

Due to the increasing demand of visual realism in computer animation and medical visualization, simulation of deformable model is becoming a major issue. To accomplish sophisticated visual realism, the physically based representation of object deformation is essential. In addition, intuitive control of deformation becomes an even more important issue, since deformable objects, unlike rigid objects, change their formation continuously and defining and enforcing geometric constraints are not trivial. Especially, modeling the deformation system of human organs for surgical simulation is extremely complex. For instance, human articulations consist of different layers of tissues, bones, ligaments, and tendons. In addition, organs are surrounded by other organs and the physical properties of these human organs and tissues are not well-known. Under this situation, the intuitive geometric constraints can be a good role to control soft objects for surgical simulation.

Modeling and simulation of deformation under external forces have been well-studied using various techniques such as Mass-spring Model, FEM (Finite Element Method), and BEM (Boundary Element Method). However, to be practically used in various simulation environments, ability to instinctively define and enforce geometric controls with physical accuracy is very important. The penalty method which has been applied to solve the inter-penetration between two colliding elastic bodies uses extra energy terms to force the constraints, but it is not accurate and can be a delicate situation to find proper penalty coefficients [11]. Further penalty method needs tiny time steps to create visually satisfactory animations [12]. This involves significant computations which makes it difficult for real time simulation. In our simulator, geometric constraints are formulated and enforced to maintain the restrictions between nodes. The constraints are seamlessly integrated into the FEM based deformation model.

The rest of this paper is organized as follows: We provide comprehensive related work on geometric constraints and some different modeling approach of soft objects in section 2. The brief overview of nonlinear FEM model is elaborated in section 3. Section 4 describes that the computations involved in defining geometric constraints and some types of constraints are discussed. We demonstrate the solution method how to apply and build node-node constraints in section 5. In section 6, we provide our experimental result with two examples.

2. Related Work

Geometric constraints related works have been well studied in mechanical engineering [13] but the standpoint of geometric constraint is different with computer graphic applications. Our objective is to achieve a reasonable visual appearance and accuracy in deformable object simulation and previous researches for controlling the objects using constraints are mainly investigated in rigid objects or free form deformations. The early effort which used hard springs to adjust the constraints gives similar effects as constraint forces, but it may generate stiff numerical conditions. The drawbacks of stiff condition are that the simulator should use small time steps for integration, which will result in unnecessary or excessive computation. Hence it cannot be applied to real-time interactive simulation. Lagrange multipliers method is widely used in rigid body dynamic simulation presented by Witkin et al [6] and Baraff [7]. Baraff introduced Lagrange Multipliers to avoid intractable problem of parameterizing a system's degrees of freedom [2]. Witkin [1] introduced a fast method to generate physically based animation with geometric and dynamic constraints.

In modeling of deformable objects view, soft object simulation has a long history from free form deformation to FEM. However, the focus of existing research was on the deformation modeling under given external forces. Intuitive control of the behavior of model during simulation has not been addressed to setup and create scenario of a simulation.

FEM approach is used most often to produce deformation in animation and simulation. Terzopoulos et al. [4] introduced deformable elastic models to apply the principles of elasticity. Bro-Nielsen [5] has investigated modeling of the deformable human organs with FEM into surgical simulation. For more fast interactivity, Cotin et al. [8] investigated their effort to speed up FEM computation using hybrid method. They combined pre-computation technique and tensor-mass method into surgical simulation and training to obtain real-time volumetric deformations and cutting operations. Unlike FEM, BEM only discretizes the boundary surface so this method is not very accurate. But this algorithm requires less computation time and more suitable for real-time interactive simulation [9].

3. FEM Model

Simulating deformation is finding the displacement of every node per time step under Newtonian physics law. Mass-spring Model can be one way of modeling because it is a simple physical model, easy to program, and it demonstrates reasonably fast simulation speed. However, it is difficult to find proper spring coefficients and deal incompressibility of objects, so it may not address reasonable accuracy of deformation.

FEM is a well-known numerical continuum approach to compute the displacement of every node in elasticity analysis. The deformable object is discretized into a suitable number of finite elements. Analysis of deformable object deals with the computation of its internal and external forces and displacements of every node, which are initially unknown. These unknown variables can be obtained by solving a set of basic equations which govern the behavior of the object. Unfortunately linear FEM is not accurate for large deformations, but soft deformable objects usually can have large deformations. The linear FEM which does not allow reasonably accurate global behavior but it is another way of elasticity modeling [10]. Thus globally admissible nonlinear FEM is used in our simulator with explicit time integration for fast computation. In nonlinear FEM, the internal forces and deformations are measured by the strain which is nonlinear in displacement and velocity and the stress which is nonlinear in strain. The strain is measured using the right Cauchy-Green deformation tensor, $C = F^T F$, which is invariant under rigid transformation. Using the linear formulation in FEM, the system equation is given as [10]

$$M\ddot{u} + D\dot{u} + Ku = f$$

where the mass matrix M , the damping matrix D and the stiffness matrix K are constant. u is an N-dimensional vector of the discrete displacement field. f is the traction force vector. The above equation for nonlinear approaches takes the form

$$M\ddot{u} + D\dot{u} + R(u) = 0$$

where $R(u)$ is the internal force vectors due to deformation. Soft objects have small stiffness coefficients in all directions due to soft nature which will make explicit time integration schemes an appropriate choice and use of large time steps is viable [12]. Further the Mass matrix M and Damping Matrix D are approximated by diagonal matrices to reduce computational overheads. Thus non-linear system of equations with approximated Mass and Damping matrices, and the use of appropriate time steps enables real time simulation of fairly large meshes.

4. Geometric Constraints

Constraints have to conserve the geometric characteristics such as distance between the nodes, angle between the adjacent surface elements, during simulation. For every simulation, there is a scenario that describes the initial conditions, constant external forces and relations that must be kept throughout the simulation. Users should be able to define and modify the relation of objects to adjust the conditions. These conditions can be described in polynomial equations $C(q)=0$ where q stands for a vector that uniquely defines the status of the entire simulation system. Typical constraints include fixed distance, and angles between nodes or a set of nodes. Geometric constraints are applied to pairs of nodes of the tetrahedral mesh and it is desired that the node pairs maintain their relative positions and do not get affected when external forces are applied on the deformable object, as a whole. For example, the nodes at the point of contact of the wheels inner surface and the outer surface of a metallic rim always remain in contact, even when the external forces are applied on the top and bottom of wheel, as shown in Fig 1. In order to define such spoke-like constraints, pairs of nodes with center as one arbitrary node and one node on the wheel are taken each time and a node-arbitrary node constraint is defined. Note that the node at the center of the wheel, even though is not part of the object can also be included as a constraint node, which we call as arbitrary node. Also one node can be applied as a constraint node with several other nodes. This enables ease of defining constraints to real world objects, which is often the case.

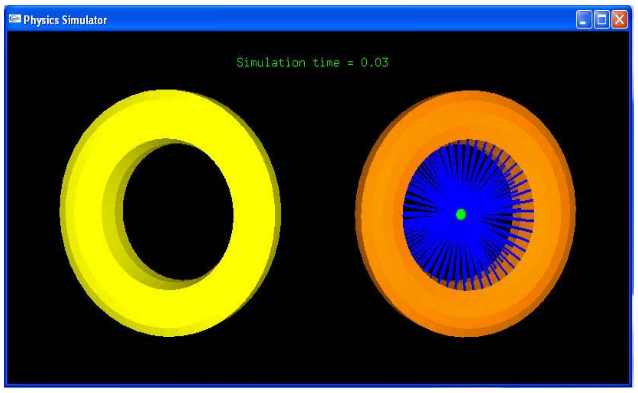


Fig. 1 Defining Geometric Constraints

The simulator can basically be applied to a variety objects and derived results depend on how the geometric constraints have been effectively applied. Based on how the constrained nodes are applied it is possible to introduce a skewed result of simulation. To avoid unreasonable or unnatural simulation, the user should study the constraints on various node pair operations by hit and trial and verify the output (the simulation).

Any relative condition between nodes, as long as it can be represented in algebraic equation and its partial derivatives exist, geometric constraints can be formulated and enforced in the simulator. The following constraint equations have been integrated to simulator.

- Node-node constraints

Node-node constraints preserve the distance between two nodes on the object.

$$\left|q_i - q_j\right|^2 - R^2 = 0$$

here q_i and q_j are the node positions and $q_i \neq q_j$. R is the desired distance between two nodes should be maintained during simulation.

- Node-arbitrary point constraints

Node-arbitrary constraints maintain the distance between a node on the object and a point which is a desirable position to fix on the outside of object.

$$\left|q_i - q_c\right|^2 - R^2 = 0$$

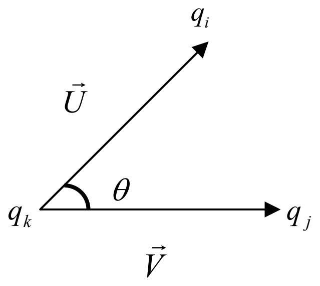
where q_c is arbitrary position which we want to fix specific node to some place.

- Angular constraints

Angular constraints restrict the angle between three nodes.

$$\left|\vec{U} \bullet \vec{V}\right|^2 - \theta^2 = 0$$

$\vec{U}$ is a vector of node position between q_i and q_k , $\vec{V}$ is a vector of node position between q_j and q_k , and θ is the desirable angle to maintain.



- Parallel constraints

Parallel constraints restrict two vectors in the parallel during the entire simulation.

$$\left|\vec{U} \bullet \vec{V}\right|^2 = 1$$

here $\vec{U}$ is a vector of node position between q_i and q_j , $\vec{V}$ is a vector of node position between q_k and q_l .

- Inequality constraints

Inequality constraints set up the restriction for the node positions to be within a radius. Since Linear Solver system cannot solve inequalities, a variable W is added so that it becomes an equality problem, which is solvable.

$$\left|q_i - q_j\right|^2 - R^2 - W^2 = 0$$

All nodes which are constrained should satisfy the above equations.

5. Solution Method

Once the geometric constraints are defined, the constraints forces are computed to maintain the given constraints. The geometric constraints are represented with a set of equations. For example, suppose that the simulation requires restriction between two nodes (node-node constraint). When x_1, y_1, z_1 is position of one node and x_2, y_2, z_2 is position of the other node, the distance r between two nodes should be continuously maintained during the simulation. This node-node constraint can be defined by following equation:

$$C = (x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^2 - r^2 = 0$$

If n is total number of nodes in the object, $3n$ vector q represents a state vector of each node position and W is $3n \times 3n$ inversed mass of node matrix which is diagonal elements contains the node's masses, and F is $3n$ global force vector. Suppose the simulation starts with position and velocity of nodes in legal state, $C = \dot{C} = 0$, then $\ddot{C}$ has to minimize close to 0 with additional constraint force F_C . From given constraint function C , $\dot{C}$ can be computed by taking derivatives. The derivatives of C is $\dot{C} = \frac{\partial C}{\partial q}\dot{q} = 0$ and we can denote $\frac{\partial C}{\partial q}$ by J . J is Jacobian matrix of C which includes connectivity information of constraints. The time derivative of the Jacobian matrix gives $\ddot{C} = \frac{\partial^2 C}{\partial q \partial t}\dot{q} + \frac{\partial C}{\partial q}\ddot{q} = 0$ and we can denote $\frac{\partial^2 C}{\partial q \partial t}$ by $\dot{J}$ as well. Thus second derivative of constraint function can be

$$\ddot{C} = \dot{J}\dot{q} + J\ddot{q} = 0$$

To solve unknown constraint forces F_c , F_c added with previous global force vector F and then acceleration $\ddot{q}$ is replaced to $W(F + F_c)$ by the governing equation of Newtonian physics $f = ma$.

$$\ddot{C} = \dot{J}\dot{q} + JW(F + F_c) = 0$$

F_C can be calculated by $J^T\lambda$ where components of λ are, Lagrange Multipliers, a vector of C dimension. Re-arrange of equation gives

$$JWJ^T\lambda = -\dot{J}\dot{q} - JWQ - k_s C - k_d \dot{C}$$

$-k_s C - k_d \dot{C}$ term is added to reduce the numerical error where k_s is spring constant and k_d is damping constant [2]. From above equation all other values are known instead of λ , so Lagrange multiplier is computed by solving the linear system and then multiplied with Jacobian matrix to obtain, $F_C = J^T\lambda$, constraint forces. J and $\dot{J}$ are the partial derivative of node position and they can be calculated using following matrix:

$$J = \begin{bmatrix} A & B & 0 & \cdots & 0 \\ 0 & C & D & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \end{bmatrix}$$

Node-arbitrary node constraints example in Fig. 1 used following vectors to create matrix J , where x_i, y_i, z_i are current position of constrained nodes and C_x, C_y, C_z are fixed position of node.

$$\begin{aligned} A &= \lfloor 2(x_0 - C_x) \quad 2(y_0 - C_y) \quad 2(z_0 - C_z) \rfloor \\ B &= \lfloor -2(x_0 - C_x) \quad -2(y_0 - C_y) \quad -2(z_0 - C_z) \rfloor \\ C &= \lfloor 2(x_1 - C_x) \quad 2(y_1 - C_y) \quad 2(z_1 - C_z) \rfloor \\ D &= \lfloor -2(x_1 - C_z) \quad -2(y_1 - C_y) \quad -2(z_1 - C_z) \rfloor \end{aligned}$$

The size of J matrix is number of constraints by 3 times of number of nodes. $\dot{J}$ can be calculated same as above with current velocity instead of current position.

Obtained constraint forces added into force computation routine to integrate into the FEM simulation. Then explicit ordinary differential equation predicts next position of each node.

6. Experiments

We created a fine triangulated surface mesh for the desired object. Using NETGEN [3] we generated a 3D tetrahedral mesh, which is given as input to our simulator. In the figure below, we demonstrate the bouncing of a stiff but still deformable metal ball against a soft board – an example of the deformation under constrained conditions. The four corners of the board are constrained through the four rods. The four arrows indicate direction and magnitude of constraint forces to keep the restricted distance against the downward forces which is generated by falling ball. The magnitudes of four contact forces are different since the metal ball is not dropped exactly middle of board. Due to flexibility of board the direction of constraint forces can be upward as well. This arrangement maintains constant distance of the four rods from the wires, as it is deformed by the fall of the ball by gravity.

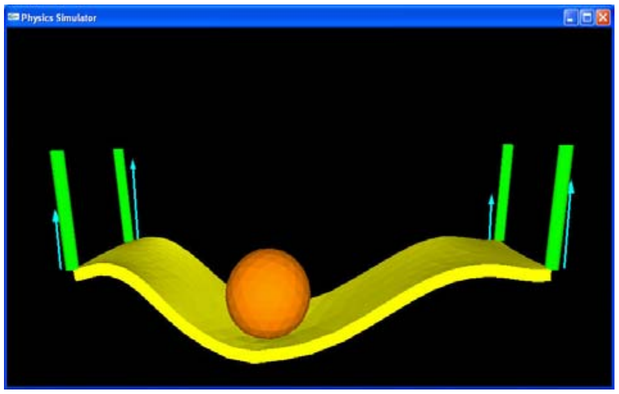


Fig. 2 Ball & board with four corner constraints

Fig. 3 shows the deformation of the wheel without any geometric constraints on the left wheel and on the right, is the one with geometric constraints. The geometry has been maintained at the contact surface of the wheel with the rim on the constrained model while on the unconstrained one, the geometry could not be preserved when subjected to a constant force. The right tire shows that the inner geometry is maintained the same even after the deformation while the top and bottom parts of the wheel show little deformation. Thus the whole wheel unit shrinks, maintaining the geometric constraints and also obeying Newton's laws. The simulator can also be used to simultaneously display two simulations of the same object with different geometric constraints on each one.

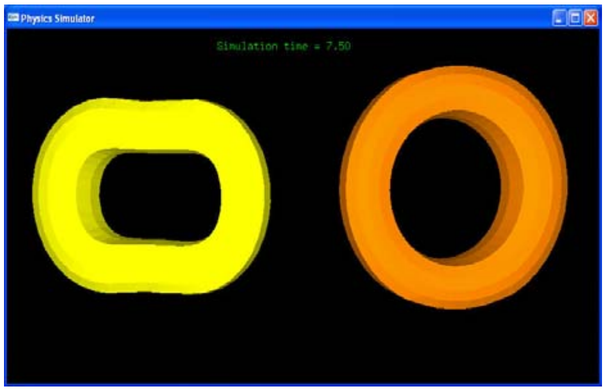


Fig. 3 Unconstrained and constrained wheels

In this study, distance constraints are applied to simulate reasonable behavior of deformable object and these constraints maintain the almost same distance during simulation. To prove the efficiency of constraints forces, Fig. 4 shows a graph for a sample simulation with two deformable tires. We applied 135 constraints to represent stiff metal rims and the error is computed by summing up the difference between original distance and current distance for each constraint at each time step. As shown in the bottom of the graph, accumulated error was kept under 2.5E-0.5 after a brief increase from 5.0E-6.

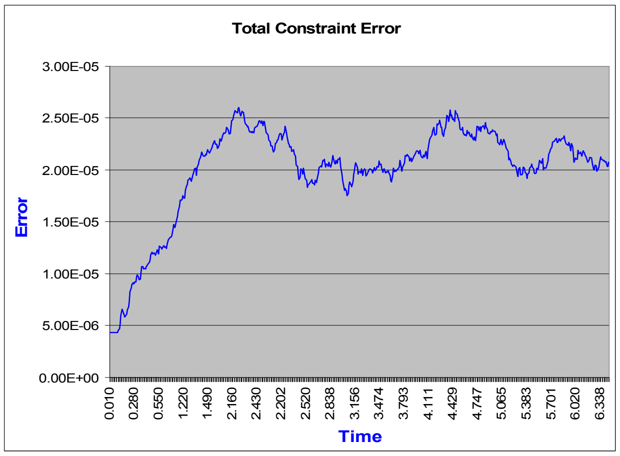


Fig. 4 Graph of total error between constrained node pairs

7. Conclusion

In this paper, it has been demonstrated how geometric constraints can be applied in the deformation of physically based modeling and simulation in an intuitive and effective manner. With the simulator, it has also been shown how easily users can experiment various simulation situations and modifying geometric constraints without modification of the whole simulator code.

This has been achieved by the use of nonlinear FEM to represent the deformable object; wherein the simulator converts the user-defined geometric constraints into the constraint forces. The geometric constraints have been shown to maintain tight error bound conditions while preserving the desired shape of the deformable objects. Our current simulator can handle constraints represented by polynomial equations but robust collision and contact technique is desirable for dynamic simulation of deformable objects.

8. Acknowledgments

This research was partly supported by CASI (Colorado Advanced Software Institute G-0183-7 TTG02-07) and we would like to thank Chris Lee for helpful advice and coding.

9. References

[1] Andrew Witkin, William Welch, Fast Animation and Control of Nonrigid Structures, ACM SIGGRAPH 1990 Conference Proceedings, 24, 4, 243-252, 1990

[2] David Baraff, Linear-Time Dynamics using Lagrange Multipliers, Computer Graphics, 30, 137--146, 1996

[3] NETGEN is an automatic 3d tetrahedral mesh generator.http://www.sfb013.unilinz.ac.at/~joachim/netgen/

[4] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer, Elastically deformable models. ACM SIGGRAPH 1987 Conference Proceedings, 21(4):205–214, 1987.

[5] Morten Bro-Nielsen, Finite Element Modeling in Surgery Simulation. Journal of the IEEE, 86(3) : 490 - 503, 1998

[6] D. Baraff. Issues in computing contact forces for non-penetrating rigid bodies. Algorithmica, 10:292–352, 1993

[7] A. Witkin, M. Gleicher, and W. Welch, Interactive dynamics. In Proceedings 1990 Symposium on Interactive 3d Graphics, 24, 11–21, March 1990.

[8] Stephane Cotin, Herve Delingette, and Nicholas Ayache, A hybrid elastic model for real-time cutting, deformations, and force feedback for surgery training and simulation, The Visual Computer, 16, 7, 437-452, 2000

[9] Doug L. James and Dinesh K. Pai, ArtDefo - Accurate Real Time Deformable Objects, ACM SIGGRAPH 1999 Conference Proceedings, 65-72, 1999

[10] Xunlei Wu, Michael S. Downes, Tolga Goktekin, and Frank Tendick. Adaptive Nonlinear Finite Elements for Deformable Body Simulation Using Dynamic Progressive Meshes. EUROGRAPHICS 2001

[11] Gentaro Hirota, Susan Fisher, Andrei State, Henry Fuchs and Chris Lee. An Implicit Finite Element Method for Elastic Solids in Contact. Proceeding of Computer Animation 2001

[12] Y. Zhuang and J. F. Canny, Real-Time Simulation of Physically Realistic Global Deformations. IEEE Visualization Late Breaking Hot Topics Proc., October 1999.

[13] Herbert Goldstein, Classical Mechanics, Wesley, Reading, MA