ISSN: 0976-9102(ONLINE)
DOI: 10.21917/ijivp.2013.0077

HAPTIC RENDERING OF THIN AND SOFT OBJECTS

Anish Garg[1] and K.G. Sreeni[2]
Vision and Image Processing Laboratory, Department of Electrical Engineering, Indian Institute of Technology, Bombay, India
E-mail: [1]anish_garg@iitb.ac.in and [2]sreenikg@ee.iitb.ac.in

Abstract
In this work, we take up the problem of modeling deformations, both local and global, along with stable translations for radially-thin virtual objects especially like that of a wire or a spaghetti noodle, during haptic manipulation. To achieve this we recommend the use of mass-spring systems rather than geometric models like vertex based and free form deformation as they do not model the physics behind the interactions. Finite Equation Methods (FEMs) are also not chosen as they are computationally expensive for fast haptic interactions and force feedback. We have explored different types of distribution of masses within the volume of the object, in order to come up with a suitable distribution of masses and network of springs and dampers so that the simulations mimic the behavior of a real object. We also model the constraint forces like normal and frictional forces between the object and the plane on which it is kept. Further, we simulate the effect of a varying temperature distribution of the object and discuss how anisotropic deformation of an object may be effected. We demonstrate through experimentations that it is indeed possible to haptically interact with virtual soft objects.

Keywords:
Deformation Modeling, Mass-Spring System, Haptic Manipulation, Soft Object, Anisotrpoic Deformation

1. INTRODUCTION

During haptic interaction with most virtual objects, even though a user can sometimes sense small deformations, the graphics show a rigid geometry. This causes a conflict within the user's mind due to hapto-visual incongruence. Therefore, there is a need to deform the object locally to make the rendering more realistic. Apart from this, when a force is applied to an object in the virtual world, a soft object can also undergo bending, translation and rotation. In order to model this, global deformations of the object also need to be considered. There is a growing demand for such type of models especially in the fields of surgical simulations and force-feedback based gaming softwares.

A lot of work has been done in computer graphics in the field of deformation but adapting it to haptics has always been a challenge. All the calculations pertaining to force feedback have to be done at a haptic interactive rate of about 1000 Hz which has remained as a bottleneck. If the force-feedback rate is low, user can feel jerks or vibrations in the motion of the haptic device. Moreover, most of the work involving the study of global deformations has been done with one face or side of the object fixed to a plane or to the ground and object is put under some constant load. We simulate haptic interactions with a freely movable soft object. We explore different geometries which would enable us to model local and global deformations with stable translations. Modeling both large and small scale deformations in a radially thin object becomes a bit tricky as possible global motion must be accounted for. To achieve a good global deformation and translation, it is imperative to have a lesser number of nodes, while for a good local deformation more number of nodes are required. In our work, we strive to achieve a balance between the two. In this paper we propose a methodology to haptically render soft objects for virtual interactions.

Currently we are building a haptic system for an objective analysis of the level of motor disorder in human subjects (such as those during Parkinson's disease) and it is imperative that the subjects are asked to interact with soft, virtual objects to measure the tremor. This requires that soft objects are simulated with high accuracy in a virtual environment. Unfortunately, none of the available object modeling techniques was found to meet this requirement. This is the motivation of our work.

The organization of rest of the paper is as follows. Section 2 presents the related work in the field of deformation modeling. The possibility of using a geometric based model is explored in section 3. The different mass distributions for a mass-spring system are investigated in section 4. Section 4.2 describes an algorithm for detecting collision between the haptic device and the object. Computation of various types of forces necessary for haptic interaction is discussed in section 4.3. The displacement of vertices and the normal correction are described in section 4.4. The effects of partially heating the object are discussed in section 4.5. Section 5 presents the results obtained and we conclude the paper in section 6. In all our notations of forces, velocities, position and other variables, subscript of a variable denotes the identity of the element or the node and the superscript denotes either the nature of the variable or the time instant when the variable is calculated, depending upon the context.

2. RELATED WORK

Most of the work done in the field of deformation modeling for haptic applications can be divided broadly into two categories: geometric-based deformations and physics based deformations. Geometric modeling involves manipulation of vertices or the volume of an object to achieve the desired result. These methods do not necessarily focus on the physics involved behind the deformations. The aim is to achieve controllable and smooth deformations. These methods typically require less computation time compared to physics-based deformation. Physics-based models of deformation focus on the physical laws involved behind the haptic interactions. They incorporate the internal and external forces which bring about deformation of the object. These methods are computationally expensive and often difficult to implement in real-life applications, but the advantage is that the force-feedback to the haptic device can be estimated from the model itself rather than independent calculations as might be required in geometric based deformation.

Geometric-based models can be further divided into vertex-based and spline-based deformation. In vertex based method the contact vertex is displaced and the surrounding vertices are displaced according to a Gaussian [1] or polynomial function that fits the experimental data. The spline based methods assign control points to a certain volume. These control points are manipulated by the user during deformation to bring about a smooth deformation of the object. These methods are also known as free-form deformations [2], [3], [4].

Physics based models can be distinguished as mass spring models or finite equation models. In mass-spring models, each vertex is considered as a particle and is assigned a mass. Adjoining vertices are attached through a network of springs and dampers [5]. Finite equation methods (FEMs) divide the object into small elements. The properties of these elements and the nature of contact forces with other elements are defined and calculated at every step. FEMs are the most accurate methods for deformation modeling but they are computationally quite expensive. As a result some assumptions or modifications are done to reduce the computation time for these models [6]. For example, in [7] organs are modeled as thin walled structures enclosing a fluid; this reduces the 3D problem to 2D.

All the above models do deformation calculations on an object defined on a mesh structure. A new type of approach, called the method of finite spheres [8] [9], has emerged to reduce this dependency on the mesh of the object. Authors in [10] introduce reduced deformation models in which displacement of every vertex is considered as a combination of different displacement fields. Luo et al. propose a new, Duffing equation based model to effect the local deformation, friction on the surface of the object and force-feedback to the haptic device [11]. For global deformation they suggest a beam-skeletal type of model for the object. A good comparative analysis of vertex-based, spline-based, mass-spring system and finite equation based deformations is provided in [12] along with tips to increase the stability and reduce the computation time of various models. A general survey on haptic rendering techniques along with a brief survey on deformation techniques and dynamics of rigid objects is provided in [13]. Nealen et al. provide an extensive survey on Lagrangian mesh based methods which include FEMs, finite difference method, finite volume method, boundary element method and mass-spring systems, Lagrangian mesh-free methods and reduced deformation models [14]. Authors in [15] focus on interpolating functions in spline based methods, FEMs and finite difference methods.

We focus our attention on spring-mass systems to model deformation. This technique has been a favorite in computer graphics to model the behavior of cloth [16] [17], soft tissues, muscles [18] and facial expressions [19]. In [20] three types of springs, structural, shear and flexion have been introduced to model muscle deformation. The concept of angular springs is introduced in [21] to conserve volume of each element of the muscle. A similar approach to mass-spring model is to calculate forces on vertices of an object by modeling energy functions to depict the distance between neighbors, surface area and volume of each tetrahedral element. These functions are then differentiated to obtain forces on each vertex of the object [22].

While modeling the mass-spring system for an object, the estimation of parameters especially the spring stiffness and spring damping coefficients is very important for the stability of the model. These coefficients also play a vital role to determine the material properties of the object. The most basic technique is that of hit and trial, where the spring parameters are manually changed until the object behaves like a real one under external forces. However, there exist more refined techniques based on Young’s modulus and Poisson ratio of the object to determine the spring parameters. In [23], iterative Voronoi based diagrams are used to get the mass distribution and an annealing algorithm is used to calculate the spring coefficients at every step. In [24], authors propose a new formula for estimation of an uniform spring coefficient for all the springs. To model the elasticity of an object, authors in [25] propose that the elasticity coefficient or the spring constant increases exponentially over each hierarchical neighborhood of the contact point. Based on shear and tensile forces applied on a 2D mesh element, [26] calculates the spring stiffness coefficient in terms of Young’s modulus and Poisson ratio. However, when the same methodology is applied in 3D, the authors prove that no general solution exists for every value of Young’s modulus and Poisson ratio. They do calculate spring stiffness coefficients for the 3D case but with the specific configuration in which there are no elemental cube face diagonals. Maciel et al., suggest that the spring stiffness coefficients in case of rectangular mesh structure should be decreased by a constant in comparison to that calculated for tetrahedral meshes [27]. They calculate this constant by considering the different angles which the spring makes with its adjoining springs. By using a linear elastic isotropic reference model, authors in [28] analytically derive spring stiffness coefficients for triangular, rectangular and tetrahedral meshes. In case of a tetrahedral mesh structure they calculate spring stiffness coefficient for a regular tetrahedral element and in case of irregular tetrahedral element, they use the concept of equivalent length to assign a uniform spring stiffness coefficient to all the springs of the tetrahedral element. In [29], a tensor-mass model is proposed in which the spring stiffness coefficients are calculated based on Lame coefficients. Gelder proposed the most widely accepted technique for calculating the spring stiffness coefficients [30]. In his proposed formula a spring stiffness coefficient is dependent on the Young’s modulus, rest length and the volume of the object associated with the spring. In [31], the spring parameters are calculated at each step and are dependent on the derivatives of the internal forces acting on the point particles. This method is computationally expensive and not suitable for haptics where the calculations have to be performed at a very high rate.

To calculate the spring damping coefficients, authors in [32] propose a formula so that the object showcases a behavioral consistency at different resolutions. Here the spring damping coefficient is inversely proportional to the rest length of the spring and is independent of its spring coefficient. Authors in [33] propose particle damping coefficient to be directly proportional to the mass of the vertex and spring damping coefficient proportional to the spring stiffness coefficient by comparing the behavior of mass-spring system with an FEM model.

To solve the differential equation for the vertices at each time step, a few algorithms exist like a second order Leapfrog integrator [34], a modified midpoint method [20], forward Euler integrator and backward Euler integrator [35]. The backward Euler integrator is the most accurate method. However, at every time step along with the computation of force on each vertex, the derivatives of the force with respect to the position and the velocity of the vertex also have to be computed in this method which makes it computationally unfavorable for haptics.

Our main focus lies in achieving global deformations of a radially thin object. For simplicity, we consider a cylindrical object of very small radius like a wire or a thread or a spaghetti kept on a floor or plane. Most of the work related to modeling deformations has been done in computer graphics and such techniques need to be incorporated in haptics where the computational speed and haptic interactions play a vital role. The work done in the field of haptics related to deformation is limited to that of local deformations of objects. We incorporate global deformations and stable translations of an object along with local deformations under haptic manipulations. We start off with a geometric based Gaussian model where the focus is on smooth and fast deformations rather than on the precise physics involved behind the internal forces of the object. We then move on to a physical based model of mass spring systems for simulations where accuracy is of utmost importance.

3. GEOMETRY BASED MODEL

In this model we divide the radially thin object into small cross-sectional, cylindrical elements of finite width. These finite cylindrical elements are attached to adjacent elements via springs with stiffness coefficient as *k*. It may be noted that for haptic interactions, use of only springs may introduce unnecessary vibrations during the interaction and may lead to instability. Hence all the strings must be attached (in parallel) with a damper with damping coefficient as k^d, as shown in Fig.1.

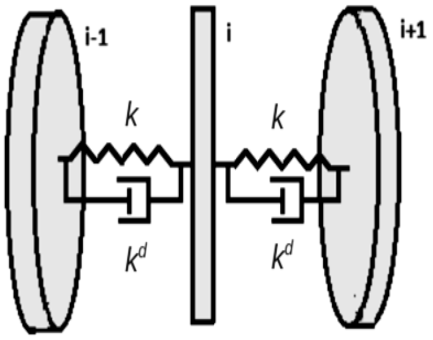


Fig.1. Finite volume cross-sections of an object connected via spring-damper system

The resultant of internal spring forces for an element *i* denoted by $\mathbf{F}_i^S$ is given by,

$$\mathbf{F}_i^s = \sum \left[k(|\mathbf{x}| - \mathbf{x}_0) - k^d \left(\frac{\frac{d\mathbf{x}}{dt}.\mathbf{x}}{|\mathbf{x}|} \right) \right] \frac{\mathbf{x}}{|\mathbf{x}|} .$$

The superscript *s* stands for spring force, *x* represents the vector from the center of mass of neighboring element of *i* to the center of mass of *i*, summation is over the neighboring elements of *i* and $\mathbf{x}_0$ denotes the rest length of the spring connecting *i* and its neighboring element. The haptic interaction force $\mathbf{F}_i^c$ on contact element *c* of the object is governed by the depth of penetration of the haptic interface point within the contact element. As the object is semi-rigid, the object should undergo some bending along with some global motion. Now, to simulate such a behavior an interpolating function like a spline or a Gaussian is used to interpolate the haptic force to the rest of the elements of the object. We use a Gaussian function for interpolation and the resultant force due to haptic interaction on an element *i* is given by,

$$\mathbf{F}_i^h = \mathbf{F}_c^h e^{(-d^2/2\sigma^2)}$$

The superscript *h* denotes haptic interaction force, *d* represents the Euclidean distance of the center of mass of the *i*th element to center of mass of the contact element and *σ* controls the spread of the deformation. *σ* can be manipulated to model different magnitudes of bending.

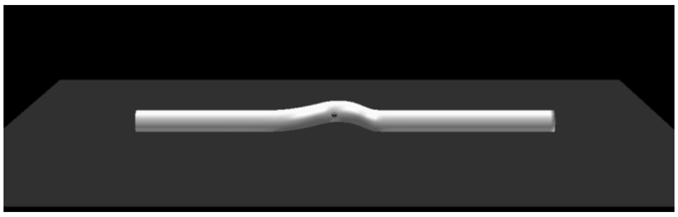


Fig.2. Global deformations of a thin cylindrical object under a haptic force using geometric based deformation model

This method of geometric based model can give graphically smooth and computationally fast results as shown in Fig.2. However, it does not model the physics involved behind the deformation. In addition the volume of the object also does not remain constant during the simulation. Hence, we move onto mass-spring systems of deformation.

4. MASS SPRING MODEL

In mass-spring models, the mass of the object is concentrated at certain points called vertices or nodes. These nodes are connected to each other via a network of springs and dampers. We consider three kinds of distribution of nodes or point particles in order to analyze which model can give large scale global deformations to mimic the real life behavior of a soft object such as a thread, wire or spaghetti noodle. Initially we consider a parallelopiped element based 3D mesh as shown in the Fig.3. The surface nodes are connected to 8 other surface nodes and inside nodes. Thus, the surface nodes are connected to a total of 17 nodes. The inside nodes are connected to 26 other nodes, some of which might be surface nodes depending upon the location of the inside node.

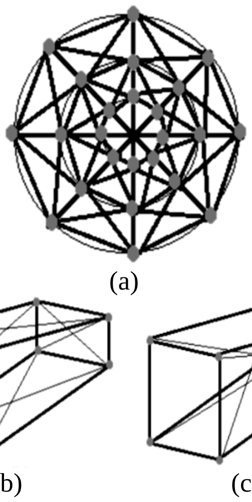


Fig.3(a). Cross-sectional view of the cylindrical object with blue dots indicating the nodes and the straight lines resemble the springs connecting them in the same plane. (b). A finite volume element showing the face diagonal springs except for those already shown in Fig.3(a). (c). A finite volume element showing the other inner diagonal springs

This type of a model though good for local deformations, does not model the global deformation well, as can be seen in Fig.4. The reason behind such a behavior of the object is that the change in displacement of the surface node by the external haptic interaction affects only a small volume of the object across the cross-section in this type of geometry. Also, the density of nodes increases as we move inside towards the central axis of the object which makes the global deformation difficult. For small displacements the force decreases rapidly in a mass-spring system as the neighborhood hierarchy increases from the displaced node independent of the mass of the nodes.

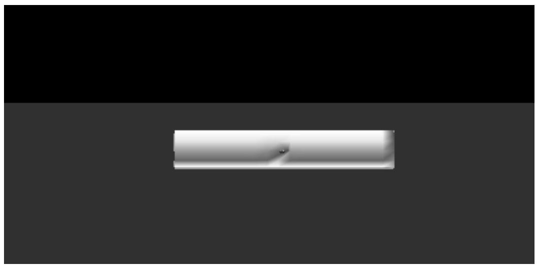


Fig.4. Illustrating local deformation of a cylindrical object under haptic interactions using the mesh structure shown in Fig.3

For the second geometry we consider the case of a cuboid element based 3D mesh as shown in Fig.5. We maintain a uniform grid. Only those grid points are taken as nodes which satisfy the inequality,

$$| \mathbf{x}_i - \mathbf{c}_i | < r_i$$

where, $\mathbf{x}_i$ represents the position of a grid point in the plane of the cross-section i, $\mathbf{c}_i$ represents the position of the center of mass of the cross-section i and r_i represents the radius of the cross-section i. In this geometric model the inside nodes are connected to 26 other nodes whereas the number of nodes to which the surface nodes are connected varies from 11 to 20 depending upon their location.

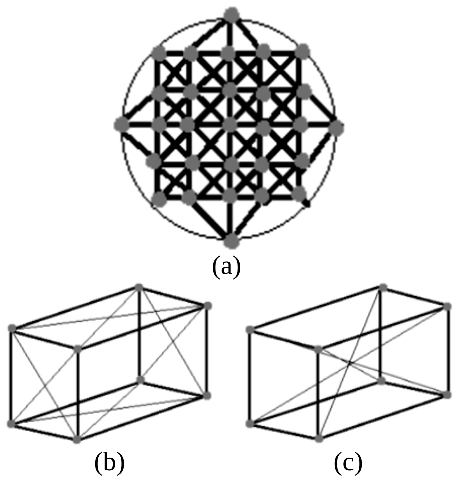
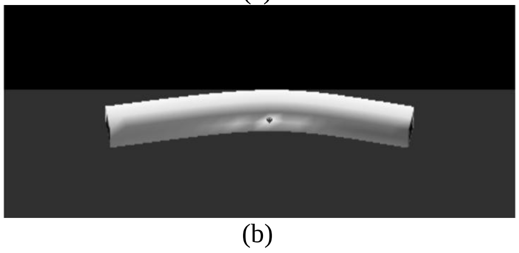


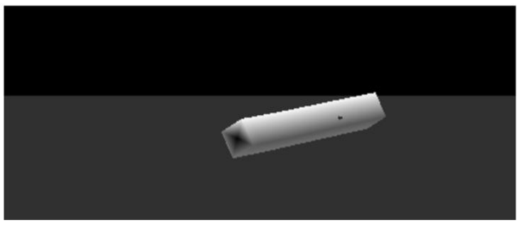
Fig.5. Illustration of cuboid based model. (a). Cross-sectional view of the cylindrical object with blue dots indicating the nodes and the straight lines resemble the springs connecting them. (b). A finite volume element, showing the face diagonal springs except for those already shown in Fig.5(a). (c). A finite volume element, showing the other diagonal springs

Through this model though we achieve a good local deformation and bending of the object, we are unable to obtain large scale translations of the object. In addition to that a more number of nodes makes the deformation loop computationally more and more expensive and the haptic object has to be moved very slowly in order to let the object recover and try to minimize its energy. To counter that we decrease the number of nodes in each cross-section of the object keeping the grid uniform and the radius constant. This is done so that the haptic interaction force propagates across the cross-section to cause global movements. However, the decrease in the number of nodes also decreases the number of surface nodes which results in poor local deformations. This effect can be seen in Fig.6 where the number of nodes in a single cross-section of the object is decreased from 29 nodes to 13 nodes and then to 5 nodes, from (a) to (d). The object also becomes highly unstable when the nodes become as less as 5 nodes per cross-section. Also, the shape is no longer preserved as the circular cross-section changes to a rectangular one.

(a)



(b)


(c)

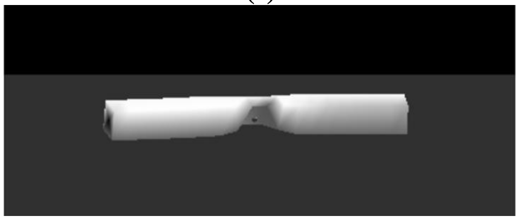

(d)

Fig.6. Behavior of the object with the mesh geometry shown in Fig.5 under haptic interactions. (a). Bending of the object with 29 nodes at each cross-section under haptic interactions. (b).Bending of object with 13 nodes. (c). Good translation of the object with 5 nodes at each cross-section under haptic interactions. With only 5 nodes in each cross-section, the cross-section of the object transforms from circular to square. (d). Poor local deformation of the object with 5 nodes at each cross-section under haptic interactions

From Fig.6, we observe that it is vital to have more surface nodes for a good local deformation but the total number of nodes has to be kept to the minimum to reduce the computation time and for large scale global translations of the object. It is also important that there should be a sufficient distance between the surface node and its first hierarchical neighborhood inside node as we will see in section 4.3. Keeping the above points in mind we design a third type of geometry as shown in Fig.7. In this geometry we have 8 surface nodes and one inside node at the center of mass for each cross-section. Thus, each surface node is connected to 8 surface nodes and 3 inside nodes. Each central or inside node is connected to 24 surface nodes and 2 neighboring inside nodes.

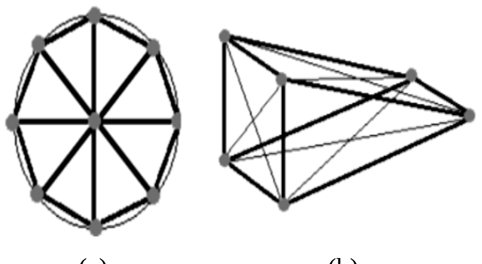

(a) (b)

Fig.7(a). Cross-sectional view of the chosen cylindrical object with blue dots indicating the nodes and the bold lines resemble the springs connecting them. (b). A finite volume element, showing the diagonal springs

The surface nodes ensure a good local deformation and the central node brings about a global deformation and translation of the object. All the further calculations are done on this type of distribution of nodes.

4.1 PARAMETER ESTIMATION

The correct estimation of spring stiffness coefficients is vital for the stability and behavior of the mass-spring model. For the estimation of spring stiffness coefficient many algorithms have been proposed as mentioned in section 2. Some of them update the spring stiffness coefficients at each time step and some update the stiffness constants after a few time steps using iterative algorithms. These methods are useful in graphics display where the computations can be relatively slower compared to that of haptics where the update has to be in the order of 1 kHz. Therefore, we use the widely used formula proposed in [30].

$$k_c = E\sum Vol(T_c)/|l_c|^2 \tag{1}$$

where, k_c represents the stiffness coefficient of the spring c, l_c represents the length of the spring c, the summation is over the volumes of each tetrahedral element T_c adjoining the spring c and E is the Young's Modulus of the material. The advantage of using this formula is that the spring stiffness coefficients are pre-computed and valuable computation time is not wasted over their calculations at each time step.

Velocity damping coefficients and spring damping coefficients are equally important as they prevent the particles of the object from oscillations as mentioned in [36]. The value of spring damping coefficients is taken as proportional to the spring constants [33] as given by,

$$k_c^d = \alpha k_c \tag{2}$$

where, k_c^d represents the damping coefficient of the spring c and α is a small positive constant.

4.2 COLLISION DETECTION

The most widely used haptic rendering algorithm is the God-object method proposed in [37]. It is the most robust algorithm for collision detection in terms of haptics, graphics and force feedback. The only disadvantage of the God-object based algorithm is that it depends upon the hierarchical volume bounding box based data structure which is precomputed. For a dynamically changing object, this approach cannot be taken, as re-computation of the hierarchical data set at each time step is computationally quite expensive. Therefore, we follow the closest surface point approach proposed in [38]. In this method a point is assumed on the surface of the object which is attracted towards the haptic device position but constrained to remain on the surface. A collision occurs when the below mentioned condition is satisfied.

$$(\mathbf{x}_s - \mathbf{x}_h).\mathbf{N}_s \geq 0 \tag{3}$$

where, $\mathbf{x}_s$ is the position of closest point s on the surface of the object, $\mathbf{x}_h$ is the current haptic device position, $\mathbf{N}_s$ is the outward normal at the point $\mathbf{x}_s$ and : denotes the dot product. In case of collision, s is considered as the proxy point which is displayed graphically. In a discrete mesh structure finding the closest point on the surface of an object can be a bit troublesome. For geometries with closely located nodes and when the relative density of nodes is higher than the change in curvature of the object, this problem can be simplified. In such a case the nearest node is found and is used for collision detection in inequality

Eq.(3). However, in the nearest node approach, when the nodes are sparse in comparison to the change in curvature of the object, collision might be detected even when there might not be a collision in case of a convex intersection of surfaces as shown in the Fig.8.

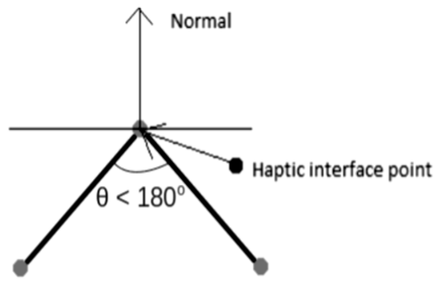


Fig.8. In the case of convex intersection of surfaces (bold lines) of the object for sparsely located nodes, the condition in Eq.(3) might be satisfied even when the haptic interface point is not inside the object

To overcome this problem a novel approach is followed. First, the nearest surface node *sn* of the object to the haptic device position $\mathbf{x}_h$ is found. Then for each adjoining face *j* of this node (Fig.9(a)), we drop a perpendicular point $\mathbf{P}_j$ on the infinite version of the plane *j* from $\mathbf{x}_h$, as shown in Fig.9(b). Now, the point $\mathbf{P}_j$ lies within the finite triangular face *j* if and only if the following conditions are met,

$$\begin{gathered}\big((\mathbf{x}_b-\mathbf{x}_{sn}))\times(\mathbf{P}_j-\mathbf{x}_{sn})\big).\big((\mathbf{x}_b-\mathbf{x}_{sn})\times(\mathbf{x}_c-\mathbf{x}_{sn})\big)>0,\\ \big((\mathbf{x}_c-\mathbf{x}_{sn}))\times(\mathbf{P}_j-\mathbf{x}_{sn})\big).\big((\mathbf{x}_c-\mathbf{x}_{sn})\times(\mathbf{x}_b-\mathbf{x}_{sn})\big)>0,\\ \big((\mathbf{x}_b-\mathbf{x}_b))\times(\mathbf{P}_j-\mathbf{x}_b)\big).\big((\mathbf{x}_c-\mathbf{x}_b)\times(\mathbf{x}_{sn}-\mathbf{x}_b)\big)>0\end{gathered} \tag{4}$$

where, *sn*, *b* and *c* are the vertices of the triangular face *j* and *X* denotes cross-product. If the point $\mathbf{P}_j$ lies within the finite triangular face *j*, then $\mathbf{P}_j$ is taken as the nearest point on the face *j* to x_h and the perpendicular distance from $\mathbf{x}_h$ to the face *j*, as the shortest distance between the face *j* and $\mathbf{x}_h$.

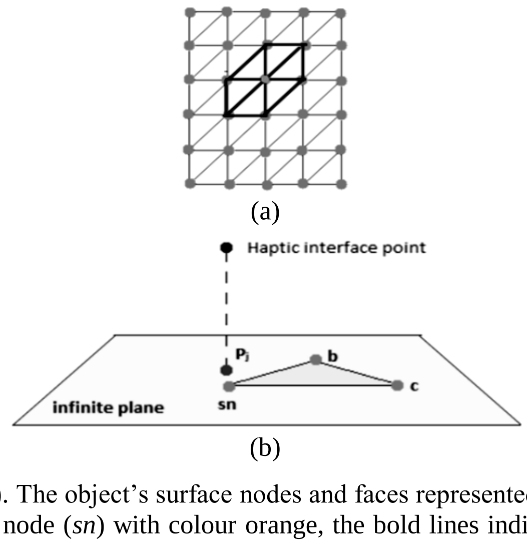


Fig.9(a). The object's surface nodes and faces represented in 2D. For the node (*sn*) with colour orange, the bold lines indicate the boundary of its neighboring faces. (b). Illustration of point $\mathbf{P}_j$ dropped perpendicularly on the infinite version of the triangular face from the haptic interface position

If conditions in Eq.(4) are not met then the closest point on the face *j* to the haptic device position must lie on one of the edges of the triangular face *j*. Now, for each edge *e*, we drop a perpendicular on its infinite version and calculate the perpendicular point $\mathbf{P}_j^e$ as shown in Fig.10. If $\mathbf{P}_j^e$ lies on the finite length of the edge *e*, then $\mathbf{P}_j^e$ is the closest point on the edge *e* to the haptic device. If not, then one of the endpoints would be the closest point on the edge *e* to the haptic device. Thus, the endpoint closest to the haptic device is computed and designated as the closest point on edge *e* to the haptic device. Now, from all the three points from the three edges of face *j*, the point closest to the haptic interface point is computed and stored as the closest point on the face *j* to the haptic interface point.

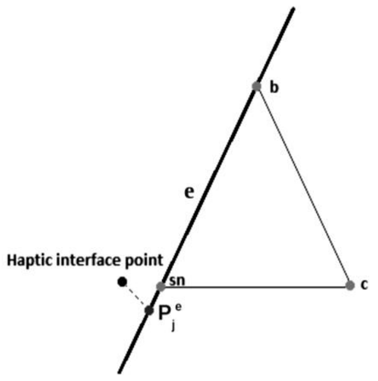


Fig.10. Point $\mathbf{P}_j^e$ dropped perpendicularly on the infinite version of the edge *e* from the haptic interface position when Eq.(4) is not satisfied

In this way, the closest point to $\mathbf{x}_h$ on each face adjoining *sn* is found. Now, the point closest to $\mathbf{x}_h$ from all these points is found and is checked for collision condition in Eq.(3). The closest surface point approach has a slight disadvantage as mentioned in [13]. Consider a cubical object as shown in Fig.11. The user feels a tendency of being thrown out of the object from the perpendicular side of the cube as one moves from point 1 to point 2. However, our model does not suffer from this disadvantage as will be clarified in section 4.3.

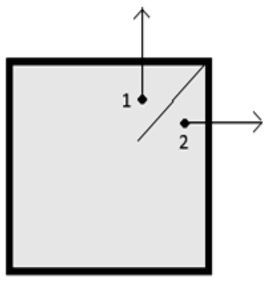


Fig.11. Illustration of when the user feels being pushed out of the surface as (s) he moves from point 1 to point 2

4.3 HAPTIC INTERACTION

Whenever a collision is detected, the nearest surface node to the haptic interface point is deemed as the collision or contact point. There are two methods to model the haptic interaction, one way is to exert a force on the collision point proportional to the amount of penetration of the haptic interface point inside the object and the other way is to displace the collision point along with the haptic device. The first method is widely used in computer graphics for interactive modeling. However, for haptic interactions it is important that the collision point move along with the haptic device during collision. Also, the later method ensures a smoother interaction force and force-feedback than the former.

So, whenever a collision is detected the collision point of the object moves along with the haptic device during that time step. The movement of the collision point can be divided into two components, one along the normal of the collision point and the other perpendicular to it in the tangent plane. Collision point is displaced along with the haptic device in the normal direction only if the direction of the displacement of the haptic device in normal direction makes an angle of 180^0 with the normal of the collision point. This check is necessary to restrict the motion of the collision point to inside the object when the haptic device has penetrated the object. Now, to model the interaction force between the haptic device and the object, we calculate the force $\mathbf{F}_c^h$ that would have to be applied to the collision point c at time t by the haptic device to bring about the displacement $(\mathbf{x}_c^{t+\delta t} - \mathbf{x}_c^t)$ at time $t+\delta t$.

$$\mathbf{F}_c^h = -\mathbf{F}_g - \sum \mathbf{F}_i^{st} + M \frac{\left(\frac{\mathbf{x}_c^{t+\delta t} - \mathbf{x}_c^t}{\delta t} - \mathbf{v}_c^t \right)}{\delta t} \quad (5)$$

$$\mathbf{F}_c^{hn} = (\mathbf{F}_c^h.\mathbf{N}_c^t)\mathbf{N}_c^t$$
$$\mathbf{F}_c^{hp} = \mathbf{F}_c^h - (\mathbf{F}_c^h.\mathbf{N}_c^t)\mathbf{N}_c^t$$

where, M is mass of node c, $\mathbf{v}_c^t$ c is the velocity of the node c at time t, $\mathbf{F}_g$ represents the gravitational force, $\mathbf{F}_c^{hn}$ and $\mathbf{F}_c^{hn}$ represent the normal and the tangential components of haptic interaction force $\mathbf{F}_c^h$ respectively, $\mathbf{F}_i^{st}$ represents the spring-damper force due to the neighboring node i at time t, $\mathbf{N}_c^t$ is the normal at c at time t and δt is the time step. Eq.(5) is derived from backward-forward Euler integrator scheme discussed in section 4.4.1. The displacement of collision point along the tangent plane is governed by the frictional force between the object and the haptic device. The tangential force applied to the collision point cannot exceed the maximum frictional force allowed between the object and the haptic device. If $\mathbf{F}_p^{hn}$ is greater than the maximum allowed frictional force, then the new tangential component $\mathbf{F}_p^{'hn}$ is given by,

$$\mathbf{F}_c^{'hp} = \mu \,|\, \mathbf{F}_c^{hn} \,|\, \frac{\mathbf{F}_c^{hp}}{|\,\mathbf{F}_c^{hp}\,|}$$
$$\mathbf{F}_c^{'h} = \mathbf{F}_c^{hp} + \mathbf{F}_c^{hn}$$

where, μ is the coefficient of friction between the haptic device and the object. Now, the position $\mathbf{x}_c^{t+\delta t}$ of the collision point c is recalculated using Eq.(4). The force feedback $\mathbf{F}_h$ to the haptic device is given by,

$$\mathbf{F}_h = \mathbf{F}_c^{'h}$$

Due to the nature of haptic interaction, while geometric modeling it has to be kept in mind that the distance between the surface nodes and the inside node must be as much, such that the surface collision point does not cover the distance in one single time step. The problem of user being pushed out of the adjacent or opposite face of the object with the closest surface point approach for collision detection, mentioned in section 4.2, does not occur in our haptic model because we move the collision point along with the haptic device. Also, with collision detection loop running at a very high frequency the penetration of the haptic interface point within the object is very small and the problem mentioned becomes negligible.

4.4 DEFORMATION

4.4.1 Vertex Displacement:

The resultant $\mathbf{F}_j$ of all the forces acting on a node j of the object is given by,

$$\mathbf{F}_j = \sum_i (\mathbf{F}_i^s) + \mathbf{F}_g - \beta \mathbf{v}_j$$

where $\mathbf{F}_g$ represents the gravitational force, $\beta\mathbf{v}_j$ is the particle damping term, b is a constant, $\mathbf{v}_j$ is velocity of node j and ($\mathbf{F}_i^s$) is the force due to the spring-damper connecting the neighborhood node i and node j. Each ($\mathbf{F}_i^s$) is given by,

$$\mathbf{F}_i^s = \left[k_{ij}(|\,\mathbf{I}\,| - l_{ij}^o) - k_{ij}^d \frac{\left(\frac{d\mathbf{I}}{dt}.\mathbf{I} \right)}{|\,\mathbf{I}\,|} \right] \frac{\mathbf{I}}{|\,\mathbf{I}\,|}$$
$$\mathbf{I} = \mathbf{x}_i - \mathbf{x}_j$$

where, k_{ij} is the stiffness coefficient, k_{ij}^d is the damping coefficient and l_{ij}^o is the rest length of the spring connecting i and j nodes. Now, we model the frictional forces between the plane and the object. The normal to the plane $\mathbf{N}_p$ points in the upward direction. Frictional force acts on a surface node j if it satisfies two conditions. Node j should touch or penetrate the plane and following inequality must hold true.

$$\mathbf{F}_j\mathbf{N}_p \le 0$$

If both the conditions are satisfied and the magnitude of velocity tangential to the plane is non-zero, then the frictional force $\mathbf{F}_j^r$ on node j is given by,

$$\mathbf{F}_j^r = -\mu_d \,|\, \mathbf{F}_j.\mathbf{N}_p \,|\, \frac{\mathbf{v}_j - (\mathbf{v}_j.\mathbf{N}_p)\mathbf{N}_p}{|\,\mathbf{v}_j - (\mathbf{v}_j.\mathbf{N}_p)\mathbf{N}_p\,|}$$

where, μ_d is the dynamic friction coefficient between the plane and the object. If the magnitude of velocity tangential to the plane is zero and

$$\textit{if} \quad |\,\mathbf{F}_j - (\mathbf{F}_j.\mathbf{N}_p)\mathbf{N}_p\,| < \mu_s \,|\, \mathbf{F}_j.\mathbf{N}_p \,|$$
$$\textit{then} \quad \mathbf{F}_j^r = -\mathbf{F}_j - (\mathbf{F}_j.\mathbf{N}_p)\mathbf{N}_p$$
$$\textit{else} \quad \mathbf{F}_j^r = -\mu_s \,|\, \mathbf{F}_j.\mathbf{N}_p \,|\, \frac{\mathbf{F}_j - (\mathbf{F}_j.\mathbf{N}_p)\mathbf{N}_p}{|\,\mathbf{F}_j(\mathbf{F}_j.\mathbf{N}_p)\mathbf{N}_p\,|}$$

where, μ_s is the static friction coefficient between the plane and the object. In addition to the frictional force, the plane exerts a normal contact force $\mathbf{F}_j^N$ on the active constraint surface node j of the object which is given by,

$$\mathbf{F}_j^N - (\mathbf{F}_j.\mathbf{N}_p)\mathbf{N}_p$$

So the total force $\mathbf{F}_j^{'}$ on node *j* is given by,

$$\mathbf{F}_j^{'} = \mathbf{F}_j + \mathbf{F}_j^N + \mathbf{F}_j^r$$

The problem now boils down to solving the differential equation for each node *j*.

$$M \frac{d^2 x_j}{dt^2} = \mathbf{F}_j$$

To solve this equation at each time step we use a backward forward scheme of Euler integrator. For calculation of velocity we use the forward Euler and for calculation of displacement of nodes we use the backward Euler integrator. This scheme is more stable than forward Euler integrator even at large time steps. When force is independent of velocity this scheme transforms into a more accurate and stable Stoermer - Verlet scheme [14] which is a second order integrator in comparison to pure forward Euler integrator which is only first order. It becomes,

$$\mathbf{a}_j^t = \mathbf{F}_j^t \big/ M$$
$$\mathbf{v}_j^{t+\delta t} = \mathbf{v}_j^{t+\delta t} + \mathbf{a}_j^{t'} \delta t$$

where, $\mathbf{a}_j^t$ represents the acceleration and $\mathbf{v}_j^t$ the velocity of node *j* at time *t*. For those surface nodes of the object with an active plane constraint, an additional check on the velocity of these surface nodes (*j*) has to be placed in order to prevent them from penetrating inside the plane.

$$\mathbf{v}_j\mathbf{N}_p < 0 \tag{6}$$

If Eq.(6) holds true, then assuming an inelastic collision between the plane and the soft object, the new velocity of the surface node is given by,

$$\mathbf{v}^{'}_j = \mathbf{v}_j - (\mathbf{v}_j.\mathbf{N}_j)\mathbf{N}_j$$

Now, the new position $\mathbf{x}_j^{t+\delta t}$ of the node *j* at time *t* + *δt* is given by,

$$\mathbf{x}_j^{t+\delta t} = \mathbf{x}_j^t + \mathbf{v}_j^{t+\delta t} \delta t$$

4.4.2 Normal Correction:

At each time step the normal to each surface node is updated for appropriate light and shadow modeling by OpenGL libraries for the purpose of simultaneous visual rendering. To calculate the surface normal we adopt a method based on the force shading method suggested in [13]. We calculate the surface normal to each surface node at each time step by averaging the surface normals of neighboring triangular faces, weighted by their neighboring angles.

4.5 ANISOTROPIC DEFORMATION

It is often required to render a deformable object when the amount of deformation is also dependent on some external parameters such as whether various parts of the object are dry or wet or when the ambient temperature is changing. It is important to note that such changes could very well be spatially inhomogeneous, for example, when the temperature is different and changing at different spatial locations. One must incorporate such effects also while rendering the haptic interface of a soft object. We now show that such an effect can easily be incorporated into the proposed algorithm during the rendering process. We illustrate this with the example of how to handle spatially inhomogeneous heating effects.

As the object is heated the spring parameters change as the particles are bound by decreasing internal forces. To incorporate the effect of heating the object, the values of the spring stiffness constants are decreased. Terzopoulos et al. suggested the following formula [39] based on Duhamel- Neumann law of thermo-elasticity,

$$k_C^S = x_C^O + \nu(\theta^O - \theta^O)$$

where, k_c^s is the stiffness constant of the spring *c* at temperature θ^s and k_c^0 is at temperature θ^0. *ν* is a positive constant. We decide the value of *n* from the Duhamel- Neumann equation given in [40].

$$\nu = c\alpha_L E \sum Vol(T) / \left| l_c \right|^2 \tag{7}$$

where, α_L is the linear thermal expansion coefficient, *c* is a positive constant and rest of the symbols have the same meaning as in Eq.(1). Eq.(8) gives us a relation on the proportion by which the different spring constants change with respect to each other on heating. As the spring damping coefficients depend upon the spring stiffness constants (Eq.(2)), they decrease proportionally.

5. EXPERIMENTAL RESULTS

All the simulations were implemented using C++ language and OpenGL libraries in Windows XP OS with a CORE i5 processor @ 3.19 GHz with 3.17 GB RAM and a Novint Falcon haptic device.

While calculating the vertex displacement in section 4.4.1, the value of the time step *δt* has to be chosen as close to the performance time of the deformation loop as possible. To achieve this we need to choose the highest attainable time step. We also need to keep in mind that the integrator that is used to calculate the vertex displacement should be accurate and stable at such large time steps. We found that integrators like Leapfrog integrator [34] or commonly used forward integrator are not stable for such large time steps. The backward integrator is stable for large time steps but it increases the computation time in each time step. We found that the backward-forward scheme of Euler integrator as mentioned in section 4.4.1 to be the most stable one for large time steps and sufficiently accurate for our purpose. We also discovered that the stable time step is directly proportional to the density of the object. For an object with density 9049 kg/m^3, time step as high as 1.8 ms can be chosen but for an object with density 1508 kg/m^3 the time step has to be decreased to about 0.7 ms. For less dense objects a larger time step would introduce larger error in each time step of Euler integrator because a lighter object undergoes larger amount of acceleration for a given force, implying a larger local deformation. For friction modeling between the plane and the object, we take the static friction coefficient as that of a wooden floor of 0.5 and dynamic friction as 0.2. For modeling friction between the object and the haptic device, we take the coefficient of friction as 0.3. We take the value of the spring constants as given in Table.1. These spring constants can be changed in the same proportion to model different materials with different Young's modulus. We take the value of α for calculating the damping spring coefficients as 0.000001. The value of β, the velocity damping coefficient, is taken as 0.007. Some researchers model volume conserving forces [22] by differentiating energy functions obtained from each volume element. We do not incorporate these into our model as the maximum volume change in our model was found to be only 2%. Further, these forces increase the much precious computational time for the deformation loop.

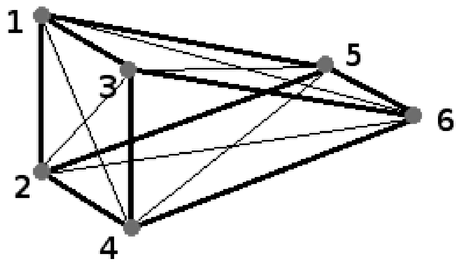


Fig.12. Numbering of nodes of a finite volume element for illustrating the choice of spring constants in Table.1

Table.1. Values of different spring constants for the connecting nodes shown in Fig.12

Spring via nodes	Stiffness value (N/m)
k1-2	1.98
k1-3	5.86
k1-4	0.74
k1-5	2.32
k1-6	0.96
K5-6	23.24

For simulating the deformations, we run four parallel loops. First loop is for collision detection between the object and the haptic device. The displacement of the contact point of the object due to haptic interaction is used to update the position of the contact point along with updating the normal of the contact point and its first hierarchical neighbors. The force to be fed back to the haptic device is also calculated in this loop. The second loop creates a haptic force field for the haptic device and renders the force. In the third loop all the deformation calculations are done. These calculations include the computation of forces due to springs and other constraints on each node of the object. After the force calculation the node positions are updated and normals at all the nodes are recalculated based on new positions of the nodes. The bottleneck in this loop is the calculation of neighborhood spring forces for each node which takes most of the time. Fourth and the final loop is the display loop, which takes care of displaying the graphics part involved in the simulations of deformation of the object with an appropriate shading model. Whenever parameters of the node, i.e. its position and vertices are being updated by any loop, flags are set such that any other loop can access these nodes only after the current loop has finished updating the parameters of all the nodes of the object.

For visual rendering of the environment, we use OpenGL libraries. We have placed five white light sources. Four of the light sources are placed in each of the top four octants of the 3D space and the fifth one is placed on the negative y-axis. We use RGB colour model with depth buffer and culling of faces enabled. For the shading model the smooth function is used which interpolates the colour of each pixel of the object based on the colors of neighborhood vertices and ambient lighting conditions.

The objective of our work has been to achieve local and global deformations with stable translations for a soft virtual object during haptic manipulation. To achieve this we have modeled the distribution of nodes in our object as shown in Fig.7. A significant number of surface nodes helps us to model any complex geometry of the surface of the object (in our case, we have taken a cylindrical surface geometry). As a result we are able to simulate a good local deformation under haptic interactions for the surface of the object as seen in the Fig.13.

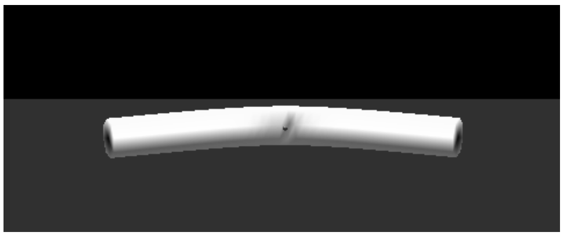


Fig.13. Bending of the object with a good local deformation

We have kept the number of inside nodes per cross-section of the object as minimum as possible such that the haptic interaction force propagates rapidly to the opposite side of the object. The central nodes inside the object are connected to each other via high stiffness coefficient springs (Table.1). These high stiffness value springs ensure a good global deformation (bending) and the translation of the object. The bending of the object can be seen in Fig.13. The translation of the object when haptic device interacts with right part of the object can be seen in Fig.14.

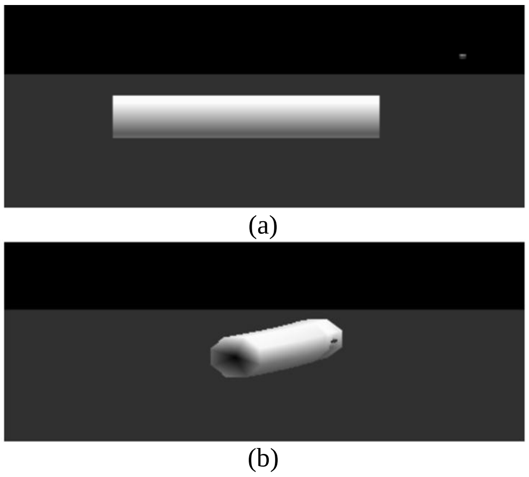


Fig.14(a). Initial position of the object before haptic interactions. (b). Translation of the object after a few haptic interactions

As the length or the number of nodes (n) of the object keeps increasing the deformation loop becomes more and more computationally expensive. As seen in Table.2, deformation loop varies with $O(n)$. There is no significant effect of increasing the number of nodes on computational time of the force rendering loop and the display loop. As the time taken by the deformation loop keeps increasing, the effect of the haptic interaction becomes limited to the neighborhood of the contact point and one is unable to see global deformations and translations of the object. Thus, our work gives excellent results for a limited length of an object. However, we expect to meet the computational demand of any sized object by exporting the computations of the deformation loop to a GPU (such as CUDA).

To model the effect of heating we decrease the spring stiffness constants and spring damping coefficients with temperature increase as described in section 4.5. To simulate this, we have modeled the object in such a way that left half (red colour) of the object is heated while the right half remains at the same temperature. On heating, the object becomes softer and less firmly connected to its neighbors. In Fig.15 we can see the difference in local deformations in the heated and the not-heated part of the object.

Table.2. Average time taken by different computational loops as the length of the object is increased

No. of nodes	**No. of spring node computations**	**Collision loop**	**Deformation loop**
180	3200	0.00019s	0.0047s
360	6400	0.00030s	0.0098s
540	9600	0.00039s	0.0149s
720	12800	0.00050s	0.0194s

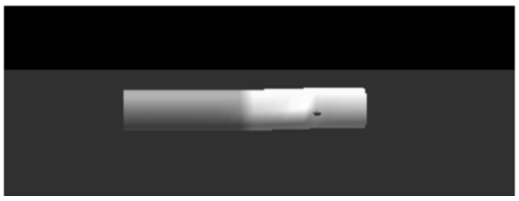
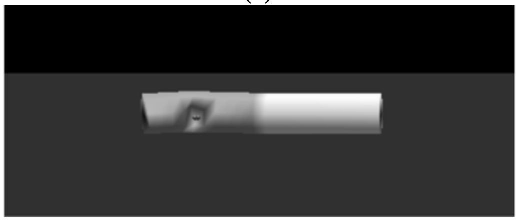
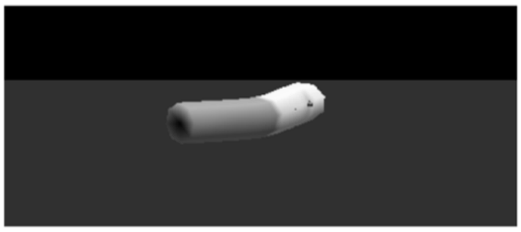

(a)

(b)

Fig.15(a). Minimum local deformation for translational movement in the not-heated part of the object. (b). Local deformation in the heated part of the object


(a)

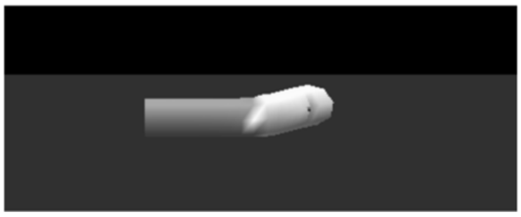

(b)

Fig.16(a). Bending a semi-heated soft object when the not-heated part of the object undergoes haptic interactions. (b). Increased bending of semi-heated object and less translation of heated part, as the temperature of the heated part is increased further

In Fig.16, heated part of the object does not undergo much translation, causing global deformation or bending even when the unheated part is moved under haptic interaction due to the decreased internal forces on account of heating. This is quite as expected due to physics based modeling of the deformation. This result substantiates the use of the proposed deformation model.

6. CONCLUSION

We have been successful in designing a physics based model for haptic interactions along with a fast collision detection algorithm to deal with a soft object. We have achieved very good local and global deformations along with stable translations for thin and soft objects with our proposed nodal distribution geometry. We have also simulated the effects of anisotropic deformation of an object based on its temperature which can easily be extended to incorporate changes in other parameters of the object. We are currently working on designing a neurophysiological tremor analysis system based on the proposed technique. As future work, plasticity of objects can be incorporated. In interactions between the haptic device and the plastic object, these objects store some part of the interaction energy as plastic energy and remain deformed even after collision. Also, after sufficient heating the objects can melt, and then a different type of potential function would define their internal interactive forces. This can be incorporated in haptics as future work to increase the diversity of haptic interactions with soft 3D objects.

REFERENCES

[1] C.J. Luciano, P.P. Banerjee and S.H.R. Rizzi, “Gpu-based elastic object deformation for enhancement of existing haptic applications”, *Proceedings of the 3rd Annual IEEE International Conference on Automation Science and Engineering*, pp. 146-151, 2007.

[2] T.W. Sederberg and S.R. Parry, “Free-form deformation of solid geometric models”, *SIGGRAPH Proceedings of the 13th Annual Conference on Computer Graphics and Interactive Techniques*, Vol. 20, No. 4, pp. 151–160, 1986.

[3] S. Coquillart, “Extended free-form deformation: a sculpturing tool for 3D geometric modeling”, *SIGGRAPH Proceedings of the 17th Annual Conference on Computer Graphics and Interactive Techniques*, Vol. 24, No. 4, pp. 187–196, 1990.

[4] W.M. Hsu, J.F. Hughes and H. Kaufman, "Direct manipulation of free-form deformation", *SIGGRAPH Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques*, Vol. 26, No. 2, pp. 177–184, 1992.

[5] A. Witkin, "An introduction to physically based modeling: Particle system dynamics", *Technical Report*, 1997.

[6] Y. Zhuang and J.F. Canny, "Haptic interaction with global deformations", *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 2428–2433, 2000.

[7] A.H. Gosline, S.E. Salcudean and J. Yan, "Haptic simulation of linear elastic media with fluid inclusions", *Haptics-e: Electronic Journal of Haptics Research*, Vol. 3, No. 5, pp. 266-271, 2005.

[8] S. De, J. Kim and M.A. Srinivasan, "A meshless numerical technique for physically based real time medical simulations", *Proceedings of the Medicine Meets virtual Reality*, 2001.

[9] Y. Lim and S. De, "Real time simulation of nonlinear tissue response in virtual surgery using the point collocation-based method of finite spheres", *Computer Methods in Applied Mechanics and Engineering*, Vol. 196, No. 31-32, pp. 3011–3024, 2007.

[10] D.L. James and D.K. Pai, "Output-sensitive collision detection for reduced deformable models", *Proceedings of the ACM Transactions on Graphics SIGGRAPH*, Vol. 23, No. 3, pp. 393-398, 2004.

[11] Q. Luo and J. Xiao, "Contact and deformation modeling for interactive environments", *IEEE Transactions on Robotics*, Vol. 23, No. 3, pp. 416–430, 2007.

[12] C. Basdogan and C.-H. Ho, "Force reflecting deformable objects for virtual environments", *Laboratory for Human and Machine Haptics, Massachusetts Institute of Technology, Technical Report*, 1999.

[13] C. Basdogan and M.A. Srinivasan, "*Haptic rendering in virtual environments*", Handbook of Virtual Environments, pp. 117-134, 2002.

[14] A. Nealen, M. Mller, R. Keiser, E. Boxerman and M. Carlson, "Physically based deformable models in computer graphics", *Computer Graphics Forum*, Vol. 25, No. 4, pp. 809–836, 2006.

[15] S.F. Gibson and B. Mirtich, "A survey of deformable modeling in computer graphics", *MERL Technical report*, 1997.

[16] H.N. Ng and R.L. Grimsdale, "Computer graphics techniques for modeling cloth", *IEEE Computer Graphics in Textiles and Apparel*, Vol. 16, No. 5, pp. 28–41, 1996.

[17] J. Liu, M. Ko and R. Chang, "A simple self-collision avoidance for cloth animation", *Computers and Graphics*, Vol. 22, No. 1, pp. 117–128, 1998.

[18] D. d'Auliganc, R. Balaniuk and C. Laugier, "A haptic interface for a virtual exam of the human thigh", *Proceedings of the IEEE International Conference on Robotics and Automation*, Vol. 3, pp. 2452–2457, 2000.

[19] D. Terzopoulos and K. Waters, "Physically-based facial modeling analysis, and animation", *Journal of Visualization and Computer Animation*, Vol. 1, No. 2, pp. 73–80, 1990.

[20] Y. Chen, Q. hong Zhu, A. E. Kaufman and S. Muraki, "Physically based animation of volumetric objects," *Proceedings of IEEE Computer Animation*, pp. 154–160, 1998.

[21] L.P. Nedel and D. Thalmann, "Real time muscle deformations using mass-spring systems", *Proceedings of the International Conference in Computer Graphics*, pp. 156–165, 1998.

[22] M. Teschner, B. Heidelberger, M. Mller and M.H. Gross, "A versatile and robust model for geometrically complex deformable solids", *Proceedings of the International Conference in Computer Graphics*, pp. 312–319, 2004.

[23] O. Deussen, L. Kobbelt and P. Tucke, "Using simulated annealing to obtain good nodal approximations of deformable bodies", *Proceedings of Eurographics Workshop on Computer Animation and Simulation*, pp. 30–43, 1995.

[24] K. Ko, H. Kim, S. Bae, E. Kim and Y. Lee, "Determination of spring constant for simulating deformable object under compression", *Key Engineering Materials*, Vol. 417–418, pp. 369–372, 2010.

[25] T. Cui, A. Song and J. Wu, "Simulation of a mass-spring model for global deformation", *Frontiers of Electrical and Electronic Engineering*, Vol. 4, No. 1, pp. 78–82, 2009.

[26] V. Baudet, M. Beuve, F. Jaillet, B. Shariat and F. Zara, "Integrating tensile parameters in mass-spring system for deformable object simulation", *Rapport de recherché, Technical Report*, 2009.

[27] A. Maciel, R. Boulic and D. Thalmann, "Deformable tissue parameterized by properties of real biological tissue", *Proceedings of the International Conference on Surgery Simulation and soft tissue modeling*, pp. 74–87, 2003.

[28] B.A. Lloyd, G. Szekely and M. Harders, "Identification of spring parameters for deformable object simulation", *IEEE Transactions on Visualization and Computer Graphics*, Vol. 13, No. 5, pp. 1081–1094, 2007.

[29] S. Cotin, H. Delingette and N. Ayache, "Real-time elastic deformations of soft tissues for surgery simulation", *IEEE Transactions on Visualization and Computer Graphics*, Vol. 5, No. 1, pp. 62–73, 1999.

[30] A.V. Gelder, "Approximate simulation of elastic membranes by triangulated spring meshes", *Journal of Graphics Tools*, Vol. 3, No. 2, pp. 21–42, 1998.

[31] M. Muller, R. Keiser, A. Nealen, M. Pauly, M. Gross and M. Alexa, "Point based animation of elastic, plastic and melting objects", *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 141–151, 2004.

[32] C. Paloc, F. Bello, R. Kitney and A. Darzi, "Online multiresolution volumetric mass spring model for real time soft tissue deformation", *Medical Image Computing and Computer – Assisted Intervention, Lecture Notes in Computer Science*, Vol. 2489, pp. 219–226, 2002.

[33] B.A. Lloyd, S. Kirac, G. Szkely and M. Harders, "Identification of dynamic mass spring parameters for deformable body simulation", *Eurographics- Short Papers*, pp. 131–134, 2008.

[34] C. Garre and A. Prez, "A simple mass-spring system for character animation", *Computer Animation Final Project, Technical Report*, 2008.

[35] D. Baraff and A. Witkin, “Large steps in cloth simulation”, *Proceedings of the Annual Conference Series in Computer Graphics SIGGRAPH*, pp. 43–54, 1998.

[36] J. Burgin, B. Stephens, F. Vahora, B. Temkin, W. Marcy, P.J. Gorman and T.M. Krummel, “Haptic rendering of volumetric soft bodies objects”, *Proceedings of Third PHANTOM Users Group Workshop*, pp. 9–13, 1998

[37] C. Zilles and J. Salisbury, “A constraint-based god-object method for haptic display”, *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Vol. 3, pp. 146–151, 1995.

[38] R. Shaw, “Nearest neighbor approach to haptic collision detection”, *Proceedings of Third PHANTOM Users Group Workshop*, *Technical Report*, pp. 34–37, 1998.

[39] D. Terzopoulos, J. Platt and K. Fleischer, “Heating and melting deformable models (from goop to glop)”, *Proceedings of the Graphics Interface*, pp. 219–226, 1995.

[40] L. Lin and M. Chiao, “Electro, thermal and elastic characterizations of suspended micro beams”, *Microelectronics Journal*, Vol. 29, No. 4-5, pp. 269–276, 1998.