Haptic Rendering of Dense 3D Point Cloud Data

Sreeni K. G.[*] Subhasis Chaudhuri[†]

Vision and Image Processing Laboratory, Department of Electrical Engineering,
Indian Institute of Technology, Bombay, Powai, Mumbai 400076

ABSTRACT

This work is aimed at rendering an object described by a dense point cloud data without a pre-computed polygonal mesh and surface normal information. We use a proxy based rendering technique with proxy collocated with the HIP in free space. To avoid sinking of proxy into the object during collision a sphere of sufficiently large radius is selected as the proxy so as to avoid it going through the point cloud. Once collision is detected, we minimize a cost function depending on the current HIP and proxy positions and find a new goal position for the proxy corresponding to the local minimum of the cost function. Our rendering algorithm continuously evaluates a tangential vector for the proxy to move over the object surface during collision. The penetration depth is calculated from the proxy to the HIP and is used to calculate the reaction force for the haptic device. We used our technique to render several dense point cloud based models. We also use the surface normal evaluated at the point of contact to shade the object surface when shown visually.

Index Terms: Haptic Rendering, Voxel-based Rendering, Distance field

1 INTRODUCTION

For haptic rendering the simplest way of representing any 3D object is by its bounding surface. Triangular meshes are very common in representing an object's bounding surface in both graphics and haptics as the subsequent operations are very simple once the mesh structure is given. The god object rendering method proposed by Ziles and Salisbury [23] can effectively render triangular meshes. But when it is required to represent a highly detailed environment, the number of triangles required will increase drastically. In Fig. 1, the mesh model of a bowl with 0.3 million points is shown. Under such a highly detailed environment, point cloud itself can be an effective display primitive and hence point cloud models are receiving increasing attention in describing objects in computer graphics. However there is no good method to haptically render a virtual object with point cloud. This is due to the fact that defining a proxy and its associated movement during haptic interaction is very difficult in case of unstructured data like that of a point cloud. In this work we propose a simple and computationally efficient, proxy based technique for rendering dense point cloud without creating a triangular mesh.

The organization of the paper is as follows. An introduction to the relevant literature is given in section 2. We try to solve proxy-based haptic rendering as a minimization problem and is explained in section 3. Section 4 explains the proposed method of rendering. The new method is used with different point cloud based models and the results are explained in section 5. We also used the direction of reaction force at the point of contact for shading the point cloud and we conclude the paper in section 6.

[*]e-mail: sreenikg@ee.iitb.ac.in
[†]e-mail: sc@ee.iitb.ac.in

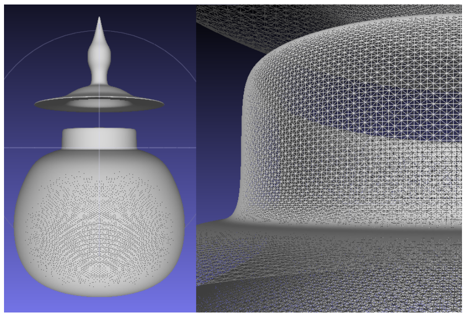
Figure 1: Illustration of a highly dense object model with 0.3 million points and its zoomed mesh model on the right side.

2 LITERATURE REVIEW

The process by which the desired haptic sensory stimuli are imposed on the user to convey information about a virtual object is called haptic rendering. In the haptic rendering literature there are mainly two different approaches: polygon (geometry) based rendering and voxel based rendering. Some methods also use hybrid approaches [7]. A good introduction to the basic haptic rendering technique is given by Salisbury *et al.* [18] [19].

2.1 Polygon based Rendering

The traditional haptic rendering method is based on a geometric surface representation which consists of mainly triangular meshes. With polygon based rendering, each time when the haptic interface point (HIP) penetrates the object, the haptic rendering algorithm calculates the closest surface point on the polygonal mesh, and the corresponding penetration depth. If $\overrightarrow{x}$ is the depth of penetration in the model, the reaction force can be calculated as $\overrightarrow{F} = -k\overrightarrow{x}$, where k is the stiffness constant.

The above method has problems while determining the appropriate direction of the force while rendering thin objects. Ziles and Salsbury, and Ruspini *et al.* independently introduced the concept of god-object [23] and proxy algorithm [17], respectively, which can solve the problems associated with thin objects. In the god-object rendering method, the authors use a second point in addition to the HIP called "god-object", sometimes called the ideal haptic interface point (IHIP). While moving in free space the god-object and the HIP are collocated. But as the HIP penetrates the virtual object, the god-object is constrained to lie on the surface of the virtual object. The position of the god-object can be determined by minimizing the energy of the spring between the god-object and the HIP, taking into account constraints represented by the faces of the virtual object [19]. If (x,y,z) are the coordinates of the virtual object and (x_h,y_h,z_h) represents the coordinates of the HIP, then the stationary point on the Lagrangian $Q(x,y,z)+\sum_{i=1}^{N} l_i(A_ix+B_iy+C_iz-D_i)$ gives the desired IHIP position, where N is the number active constraints at the current proxy location and A_i, B_i, C_i, D_i are the homogeneous coefficients for the constraint plane equations. Here $l_i{}'s$

are the Lagrangian multipliers. The function $Q(x,y,z)$ is given by

$$Q(x,y,z) = \frac{(x-x_h)^2}{2} + \frac{(y-y_h)^2}{2} + \frac{(z-z_h)^2}{2} \qquad (1)$$

The 'force shading' technique (haptic equivalent of Phong shading) introduced by Morgenbesser and Srinivasan refined the above algorithm while rendering smooth objects [11]. One common problem with the mesh based representation is when the object is not fully enclosed by the bounding planes and a small hole may remain, where the IHIP sinks during the rendering process.

2.2 Voxel based Rendering

Voxel based rendering is a different approach to haptic rendering technique where the object is represented on a grid of voxels. The most basic representation for a volume is the classic voxel array in which each discrete spatial location has a one-bit label indicating the presence or absence of material. Avila *et al.* have used additional physical properties like stiffness, color and density during the voxel representation [1]. The voxmap-point shell algorithm uses the voxel map for stationary objects and point shell for dynamic objects [10] [14]. Point shell has been defined as a set of point samples with associated inward facing normals. However, these normals are not available and one needs to compute the normal at every location from the given data.

The external surface ∂O of a solid object O can be described by the implicit equation [6] as

$$\partial O = \{(x,y,z) \in \mathbb{R}^3 \mid \phi(x,y,z) = 0\}$$

where ϕ is the implicit function (also called the potential function) and (x,y,z) is the coordinate of a point in 3D. In other words the set of points for which the potential is zero defines the implicit surface. This has found applications in haptic rendering. A distance field is a function where each point within the field represents the distance from that point to the closest point on the bounding surface ∂O and the sign denotes whether a given point is inside or outside of a solid object O. The usual convention is that the sign is negative for inside of the object [5].

A basic approach to calculate distance to triangular patches has been explained by Payne and Toga [13]. Another hierarchical approach is the mesh sweeper algorithm proposed by Guéziec [4]. If the distance is needed only near the surface of the object, we can use a bounding volume around each triangle and can calculate the distances at grid nodes inside the specified bounding volume [2]. In 2000 Mauch proposed an algorithm in which he converted the features of a triangular mesh to a polyhedron [9], the feature vertex becomes a cone with a polygonal base, edge becomes a wedge and the face becomes a prism. The polyhedron containing points closer to the respective feature are then scan converted to compute the distances. Sigg *et al.* used a graphic hardware to scan convert the characteristics for a faster implementation [21].

Once the distance close to the boundary is generated it may be propagated to the remaining volume. This is the principle of distance transform. In Chamfer distance transform (CDT), the new distance of a voxel is computed from the distance of its neighbours by adding values from a distance template [15, 16]. In CDT the accuracy reduces as the distance from the surface increases. Similarly, a vector distance transform uses boundary conditions of voxels containing the vector to the closest point on the surface, and propagating these vectors according to a given vector template [12]. The fast marching method (FMM) technique was proposed first by Tsitsiklis [22] and then Sethian [20] which computes the arrival time of a front expanding in the normal direction at a set of grid points by solving the Eikonal equation from a given boundary condition.

Collision of the HIP with the objects and the corresponding penetration depth can be easily detected by computing the potential at the HIP and hence an appropriate force can be rendered. For haptic rendering one should be able to render the object within 1ms as any haptic interface system must be temporally sampled at a rate better than 1kHz [18]. This method also have problem while rendering thin objects due to absence of IHIP.

Over the last decade, the scanning technology has improved drastically and a dense point cloud data representing an object can be easily obtained. Researches have been trying to bypass the steps of mesh formation or distance transformation for the dense data and render it directly. Lee *et al.* have proposed a rendering technique with point cloud data which computes the distance from HIP to the closest point on the moving least square (MLS) surface defined by the given point set [8]. Here the same problem occurs as in the distance field based rendering technique, since we do not keep track of HIP penetration and therefore is not good in rendering thin objects. El-Far *et al.* used axis aligned bounding boxes to fill the voids in the point cloud and then rendered with god object rendering technique [3]. Hence we would like to develop a proxy based technique to render the point cloud data so that even thin objects can be rendered well. However we do not want to build a mesh structure as is done in god object rendering to make it computationally efficient.

3 Haptic Rendering as a Minimization Problem

Let us consider an object in 2D for illustration with its boundary shown with the curve in Fig. 2 (a). The outward normal on the curve is shown at point p. The yellow circle in Fig. 2 (a) shows the initial HIP position moving towards the object in the direction shown with the blue arrow. The HIP in Fig. 2 (a) is outside the object and as there is no contact with the object, proxy is collocated with the HIP. So the yellow circle in Fig. 2 (a) also represents the proxy position.

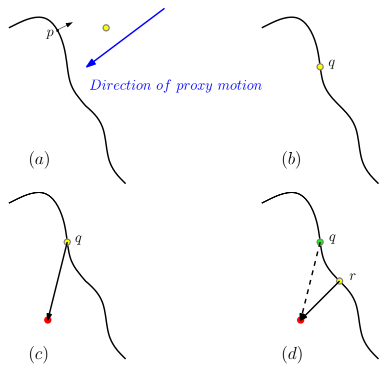

Figure 2: Illustration of movement of proxy while collision with object bounding curve.

In Fig. 2(b), the HIP just touches the bounding curve at q and from there onwards the proxy cannot move with the HIP. In 2(c) HIP has penetrated the object boundary and is shown with a circle in red color. Once the HIP penetrates the boundary the reaction force on the haptic device needs to be computed from the depth of penetration and is always normal to the bounding curve (bounding surface in 3D). The penetration depth has to be calculated from the closest point on the object boundary as shown in Fig. 2(d) and hence the proxy has to be moved to the location r where the penetration vector is oriented opposite in direction to the normal at r. The green circle in Fig (d) represents the old proxy position and yellow circle represents the desired location. Mathematically the new proxy position can be found by minimizing the distance between proxy and HIP such that the proxy stays on the boundary of the object near the current proxy position.

Let (x_h, y_h, z_h) and (x, y, z) are the HIP and the proxy positions, respectively and $f(x, y, z) = 0$ represents the bounding surface of the object. Let us denote $Q'(x, y, z) = Q(x, y, z) + \lambda f(x, y, z)$ with λ being a Lagrange multiplier. The shape of the function Q' on the curve changes with the change in HIP and proxy position. In order to find the local minimum of the function Q' its gradient can be evaluated and we can take steps along the negative gradient direction. HIP is assumed to be stationary, until the minimization process converges. ie. haptic interaction is much slower than the computation time.

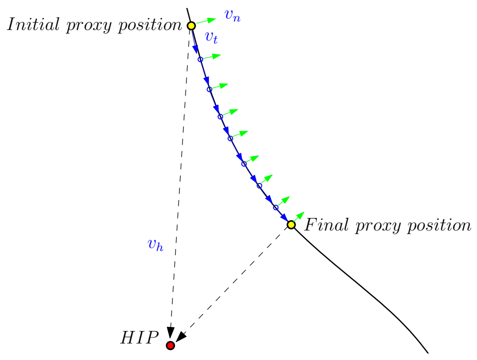

Figure 3: Illustration of proxy movement during energy minimization.

The proxy movement during minimization of the function Q' is shown in Fig. 3. The initial proxy position is shown with an yellow circle at the top. The vector $\overrightarrow{v_n}$ represents the normal at the initial proxy position and $\overrightarrow{v_h}$ is the vector from proxy to the HIP. Instead of evaluating gradient of Q' near the current proxy position and then finding the direction of motion for the proxy, the tangential vector in the plane of $\overrightarrow{v_n}$ and $\overrightarrow{v_h}$ is evaluated which provides a quick and greedy approximation of the negative gradient direction of Q'. At each step if we move the proxy towards the tangential direction on the curve, the proxy will finally come to rest at a point where the angle between $\overrightarrow{v_h}$ and $\overrightarrow{v_n}$ is 180 degree. In Fig. 3 the outward surface normal is shown with green arrows. From the function Q' evaluated at the initial and subsequent proxy positions it can be seen that the function decreases while moving down as shown. Once the proxy moves in the direction of tangential vector, $\overrightarrow{v_t}$ the vector $\overrightarrow{v_h}$ and the movement direction are again computed. This is repeated until the magnitude of tangential component becomes zero. The minimum value for Q' is attained in a finite number of steps if the HIP is stationary. The initial proxy position can be selected as a point outside the object in the case of object being a closed one.

4 Proposed Method

With point cloud data we neither have a well defined bounding surface as in a polygonal mesh model nor have the surface normal at each point. There are methods to render an object defined by polygonal meshes as discussed in section 2.1. If one fits an implicit surface for this point cloud data, such a rendering is possible. However, we do want to avoid surface fitting problem in this paper. The rendering method we propose does not fit the points globally to form the bounding surface. We use a proxy based rendering technique in which a sphere of sufficiently large radius is defined as the proxy, instead of a point so as to avoid it sinking through the point cloud as well as to compute the surface normal. As the HIP collides the object, HIP penetrates the object while the proxy stays on the surface and moves over it so that the distance between proxy and HIP is minimized. Since we do not have an explicit definition of the bounding surface and one must prevent the proxy from sinking into the object, we replace the definition of proxy from being a point to a spherical volume with a certain radius. The penetration of proxy into a point cloud can be easily determined by checking for points lying within the proxy volume. This is one of the key concept in this paper.

4.1 Controlling Proxy Movement

As we move the haptic device in free space the proxy is collocated with the HIP as in god object rendering. The presence of a point inside the proxy volume increases the chance of collision of HIP with the object and hence any further movement of proxy into the object is to be restricted. As we do not have any pre-computed mesh structure it is not possible to form constraints as in god object rendering technique to maintain the proxy on the surface. Hence, dynamically we find out a function which measures the sinking of proxy into the object. Fig. 4 shows a typical situation of proxy sinking inside the object illustrated in 2D. The big red circle with center C_p is the proxy just sinking in the object surface described by the point cloud and the small blue dots represent the points on the object surface. The radius of the proxy selected is dependent on the density of point cloud so as to avoid it going through it and hence we cannot select the proxy as a point. Choice of a larger value of proxy radius will however impede the haptic experience in case of a fine local surface texture. We evaluate the vector sum of the radial overshoot of each point inside the proxy and is used as a measure of sinking of proxy in the object.

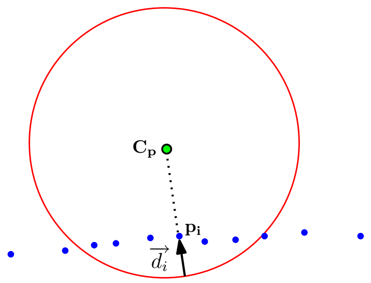

Figure 4: Illustration of the amount of proxy sinking in an arbitrary point cloud.

If $\overrightarrow{d_i}$ denotes the radial overshoot of each point inside the proxy, the sinking measure $\overrightarrow{v_n}$ may be evaluated as

$$\overrightarrow{v_n} = \sum_{i=1}^{N} \overrightarrow{d_i} \tag{2}$$

For each point p_i overshoot d_i can be evaluated using equation 3 where r_p and C_p are the radius and the center of the proxy, respectively.

$$\overrightarrow{d_i} = (r_p - |\overrightarrow{C_p - p_i}|) \frac{(\overrightarrow{C_p - p_i})}{|\overrightarrow{C_p - p_i}|} \tag{3}$$

The peculiarity of the function $\overrightarrow{v_n}$ is that its magnitude is a measure of the amount of sinking of proxy into the object surface approximated by the point cloud and the direction indicates the direction in which the proxy has to be moved so as to reduce the sinking. Although $\overrightarrow{v_n}$ is a function of all the vectors d_i the point near to the proxy center will get more weightage in deciding the direction of $\overrightarrow{v_n}$. If the proxy radius is small compared to the local variation in the curvature, and the points are sufficiently dense, the vector direction $\overrightarrow{v_n}$ approximates the direction of surface normal defined by the points.

Our algorithm checks the proxy sinking by evaluating the function $\overrightarrow{v_n}$ at each step. A non zero value for $|\overrightarrow{v_n}|$ denotes that the proxy has touched the object bounding surface. The movement of the proxy along $\overrightarrow{v_n}$ prevents it from sinking. We may select the constant $0 < k_n < 1$ such that $k_n\overrightarrow{v_n}$ keeps the proxy near the surface. A large k_n moves the proxy far away from the surface, making it to move freely towards the HIP. This causes a large variation in computing the penetration depth causing vibration in the haptic device which is undesirable. The pseudo code to move the proxy from sinking into the object is given below.

$$if \quad (|\overrightarrow{v_n}| > \zeta) \quad then \quad C_p \leftarrow C_p + k\,\overrightarrow{v_n}$$

The value of ζ decides the allowable proxy sinking. It can be selected as 0.05 times the proxy radius or less. The magnitude of the vector $\overrightarrow{v_n}$ is zero in free space when there is no points inside the proxy volume. A small non-zero $\overrightarrow{v_n}$ is desirable for the calculation of the tangent component which is discussed next.

4.2 Tangent Vector Evaluation

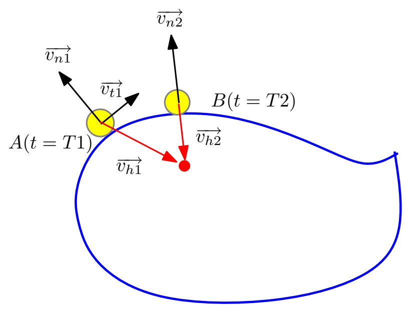
$\overrightarrow{v_{n2}}$
$\overrightarrow{v_{n1}}$
$\overrightarrow{v_{t1}}$
$B(t = T2)$
$A(t = T1)$
$\overrightarrow{v_{h2}}$
$\overrightarrow{v_{h1}}$

Figure 5: Illustration of tangent vector evaluation from the current proxy position and HIP. The proxy should be moved along the tangent direction.

Fig. 5 illustrates the proxy movement and the computed surface normal for an arbitrary object. The object boundary is shown with a blue contour. The HIP shown with a red ball is penetrated inside the object while the proxy is shown with the yellow ball on its surface. Let A be the initial proxy position at time $t = T_1$. At position A, $\overrightarrow{v_{h1}}$ denotes the vector from proxy to the HIP, and $\overrightarrow{v_{n1}}$ denotes the vector approximated to be along the surface normal. There is a tangential component of $\overrightarrow{v_{h1}}$ over the surface unless the surface normal is exactly opposite to $\overrightarrow{v_{h1}}$ when there is no need to move the proxy further. The vector tangent to the surface normal in the plane of $\overrightarrow{v_{n1}}$ and $\overrightarrow{v_{h1}}$ gives the direction of movement of proxy which reduces the distance between proxy and the HIP. In the figure, $\overrightarrow{v_{t1}}$ represents the tangential component of $\overrightarrow{v_{h1}}$ on the surface. In a finite number of steps the proxy comes to B at time $t = T_2$, where $\overrightarrow{v_{n2}}$ and $\overrightarrow{v_{h2}}$ are opposite in direction.

In free space $\overrightarrow{v_n}$ and $\overrightarrow{v_t}$ are zero so proxy moves with the HIP. As the proxy starts sinking into the object the component $\overrightarrow{v_n}$ tries to keep the proxy on the object surface while $\overrightarrow{v_t}$ guides it over the surface. When $(\overrightarrow{v_n}.\overrightarrow{v_h}) > 0$, HIP is outside the object and the proxy should move along with the HIP. The pseudo code to move the proxy towards the HIP is

$$if \quad ((\overrightarrow{v_n}.\overrightarrow{v_h}) \geq 0) \quad then \quad C_p \leftarrow C_p + k_h\,\overrightarrow{v_h}$$

when $k_h = 1$ proxy can move to the HIP in one step, but it can not check for the collision with the object. When k_h is very small proxy cannot move along with the HIP in free space. So we can select a suitable value of k_h depending on the speed at which proxy position is updated. A negative value for $(\overrightarrow{v_n}.\overrightarrow{v_h})$ indicates the collision of HIP with the object and hence the proxy is allowed only to move over the surface. The following pseudo code will ensure the tangential movement of proxy on the surface when $(\overrightarrow{v_n}.\overrightarrow{v_h}) < 0$.

$$if \quad (|\overrightarrow{v_t}| > 0) \quad then \quad C_p \leftarrow C_p + k_t\,\overrightarrow{v_t}$$

The value of k_t also depends on the proxy update speed. When $k_t = 0$ proxy will not move over the surface after collision, but sticks on the surface. As we increase k_t proxy starts rolling over the surface. A large value of k_t may lead to undesirable vibration of the haptic device in presence of large variation in surface curvature in the neighbourhood. The parameters k, k_h and k_t are chosen empirically in our work.

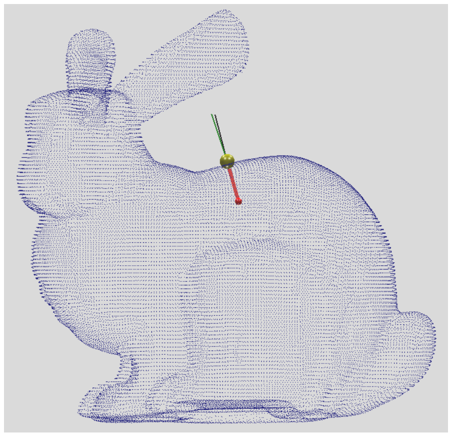

Figure 6: Comparison of the surface normal estimated by local plane fitting of point cloud data with v_n computed using the proposed method (Data courtesy:-www.cc.gatech.edu/projects/large_models).

Fig. 6 shows the point cloud model of a bunny with 35947 points. The yellow ball in the scene is the proxy touching the object surface as the HIP penetrates. The black line above the proxy shows the true surface normal computed locally by fitting the neighbouring points in a plane and the green line shows the direction of v_n as computed in equation (2). We can see that the vectors are oriented almost in the same direction. Note that the magnitude of computed $\overrightarrow{v_n}$ provides a measure of proxy sinking, which cannot be obtained through plane fitting. As the vector $\overrightarrow{v_n}$ is evaluated in each step, only an approximated surface normal is sufficient to find $\overrightarrow{v_t}$.

4.3 Haptic Rendering

Having found $\overrightarrow{v_n}$ and $\overrightarrow{v_t}$, we are now in a position to render the reaction force. The haptic rendering includes two steps

1. Detection of collision of the HIP with the object.
2. Force computation if a collision is detected.

4.3.1 Collision Between the HIP and the Object

Collision of HIP with the object is detected when $(\overrightarrow{v_n}.\overrightarrow{v_h}) < 0$, which indicates that the proxy has touched the object. As we move the haptic device inside the haptic interaction space the neighbourhood of proxy inside the proxy volume is searched. The proxy moves with the HIP when there is no surface point inside the proxy volume and therefore no force is fed back to the device. The presence of a point inside the proxy volume indicates that collision has occurred between the proxy and the object.

4.3.2 Force Computation for Haptic Rendering

When $(\overrightarrow{v_n}.\overrightarrow{v_h}) < 0$, a collision is detected with the object. Now the proxy is allowed to move only tangentially on the surface. The reactive force is proportional to the distance between the HIP and the proxy. The reaction force is calculated as $\overrightarrow{F} = -k\overrightarrow{x}$, where k is the Hooke's constant. The penetration depth $\overrightarrow{x}$ is given by

$$\begin{aligned} \overrightarrow{x} &= (|\overrightarrow{v_h}| - r_p)\frac{\overrightarrow{v_h}}{|\overrightarrow{v_h}|} \quad for \quad |\overrightarrow{v_h}| \geq r_p \\ &= \overrightarrow{0} \quad otherwise \end{aligned} \tag{4}$$

where r_p is the radius of proxy used. Here the sinking of proxy is assumed to be negligible while calculating the penetration depth.

4.4 Neighbourhood Search

In order to search the neighbourhood of the proxy in a computationally efficient way, we partition the haptic space using a regular 3D grid. To understand the data structure, let us look at a 2D grid shown in Fig. 7. The 2D object boundary points are shown with blue dots. The dotted horizontal and vertical lines in the figure show the 2D partitioning with the grid nodes at the center shown with a red circle. We create a data structure with the red circle as the parent node and the blue circles inside the corresponding square box as the corresponding child nodes. For each red circles the blue circles coming inside the dotted square box are the child nodes. When a particular proxy position is queried, the grid nodes close to the proxy are found and hence the nearest points in the point cloud can be checked for collision. The value of h decides how finely we want to divide the space.

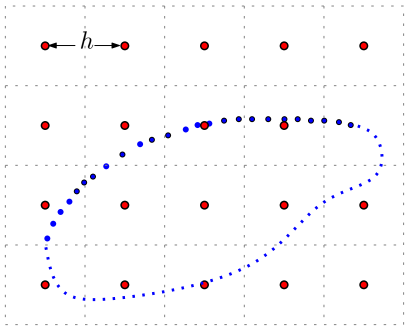

Figure 7: Data structure for fast neighbourhood search.

5 Results

The proposed method has been implemented in visual C++ in a windows XP platform with a CORE 2QUAD CPU @ 2.66GHz and a 2GB RAM. We use a 3-DOF Falcon haptic device from NOVINT for haptic rendering which is having roughly 4 inch cube interaction space. We have experimented our technique with various models. For fast accessing of proxy neighbourhood a data structure is created with the point cloud as explained in section 4.4. The grid we have selected is of size $200X200X200$ with a spacing of 0.06cm (ie, h=0.06cm) between adjacent grid nodes in all the three axes. Haptic rendering algorithm works with a 1 kHz refresh rate on a separate thread.

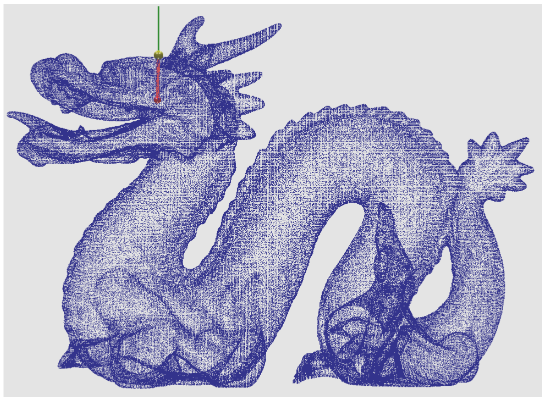

Figure 8: Point cloud model of a dragon with computed normal at a given proxy point (data courtesy:-www.cc.gatech.edu/projects/large_models).

Fig. 8 shows a point cloud model of a dragon with 437645 points. The Proxy is shown with an yellow ball on the dragon surface and the red line is the line joining proxy with the HIP. The computed surface normal at the proxy position is shown with a green line. The radius of the proxy we used with dragon is 0.1cm and the time required to evaluate the tangent vector is 0.0328ms. This means that we could iterate the proxy updation nearly 30 times for a haptic rendering speed of 1 kHz. However, for a dense point cloud, one rarely needs that many iterations. Typically 5-6 times suffice.

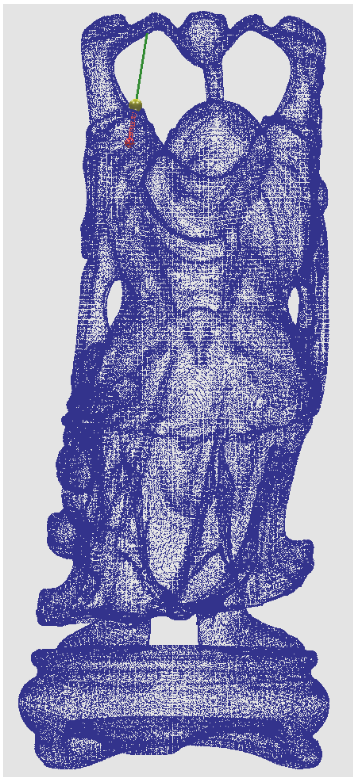

Figure 9: Point cloud model called happy with 543652 points (data courtesy:-www.cc.gatech.edu/projects/large_models)

Fig. 9 (a) shows a dense point cloud model called happy with 543652 points. HIP and proxy positions are shown in the figure. As the points in the model are more dense the proxy size can be reduced. We used a proxy of radius 0.1cm and the evaluation time for the tangent vector is 0.035ms at the shown proxy location.

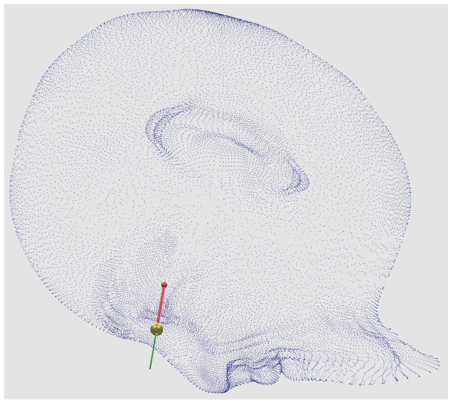

Figure 10: Point cloud model bonzew with 31718 points and the computed surface normal (data courtesy:- www.turbosquid.com)

Point cloud model of a bonzew with 31718 points shown in Fig. 10. Unlike the previous model here the points are highly non-uniform. So we used a proxy of larger radius, 0.17cm with bonzew and the evaluation time for the tangent vector is 0.0235ms at the shown proxy location. Proxy shown with yellow coloured ball is near the eye position of the model.

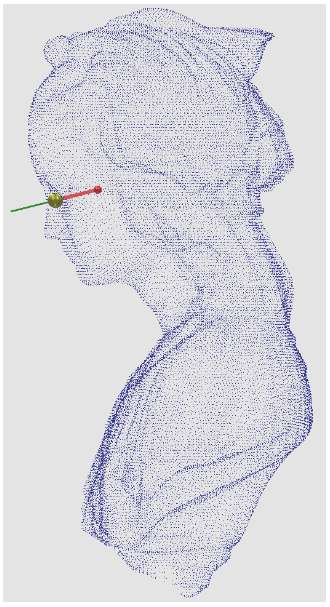

Figure 11: Point cloud model of a bust and the computed surface normal (data courtesy:- www.turbosquid.com).

Fig. 11 shows the point cloud model of a bust with 47516 points. Proxy is touching the point cloud near the eye portion of the model. The direction of surface normal is also shown in the figure. We used a proxy of radius 0.13cm with the bust and the evaluation time for the tangent vector is 0.0285ms at the shown proxy location.

In another experiment a point cloud representation of an angel with 237018 points is rendered and is shown in Fig. 12. The radius and and the time to evaluate the tangent with this model are 0.1cm and 0.0185ms, respectively.

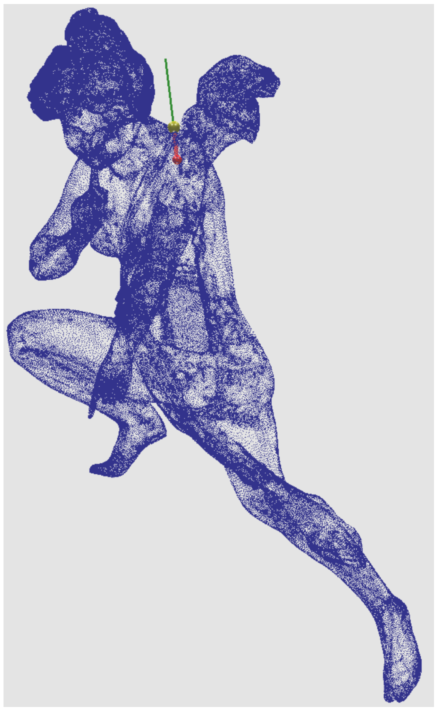

Figure 12: Point cloud model of an angel with 237018 points and the computed surface normal at the proxy location (data courtesy:- www.cc.gatech.edu/projects/large_models)

The proposed method can also be used to find the surface normal at a point for any dense point cloud. As we feel the bounding surface of the object defined by the point cloud, the algorithm calculates the reaction force. Since the direction of reaction force is normal to the surface at the point of contact, it can be used to shade the corresponding pixel. According to any standard shading model and a known light source direction, Fig. 13 shows the model of a bunny and the head portion of the model is shaded using the haptic reaction force. The yellow ball in the scene is the proxy and is of radius 0.17cm and the tangent evaluation time is 0.0141ms.

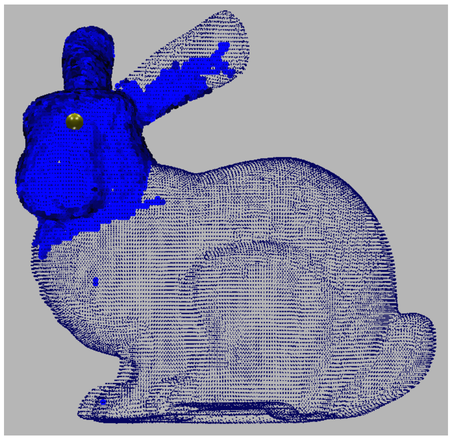

Figure 13: Illustration of shading the pixel using the computed normal (data courtesy:- www.cc.gatech.edu/projects/large_models).

In order to validate our results we use a point cloud model of a known sphere as shown in Fig. 14. The reaction force with our rendering technique is compared with the force computed using the known implicit equation of the sphere. The magnitude of force vs time graph is plotted for both the cases and is shown in Fig. 15.

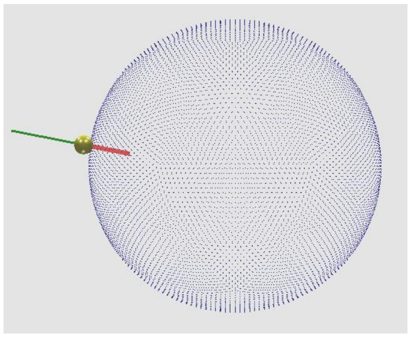

Figure 14: Illustration of rendering a known sphere for comparison purposes.

In Fig. 15 the red line shows the actual force and the blue line shows the rendered force using our technique. It is clear from the figure that the rendered force is quite close to the actual force. The departure from the actual value is only when there is a sudden movement of HIP and this is because of the time taken by the proxy to move to the equilibrium position.

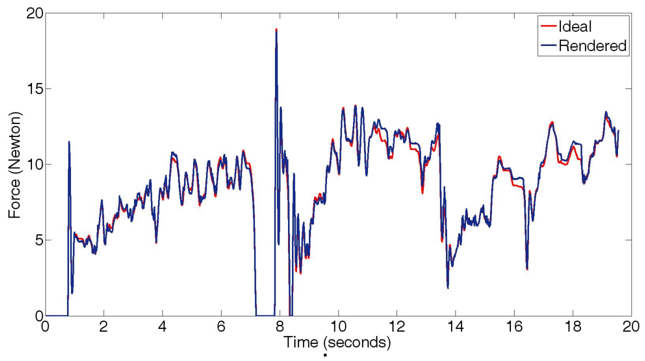

Figure 15: Comparison of rendered force with the ideal one during a particular interaction with a known spherical object.

6 Conclusions

In this work, we propose an algorithm to render objects described by a dense point cloud data without a pre-computed polygonal mesh and normal information. As we use a proxy based rendering technique, we are able to render thin objects also with our method. To avoid sinking of proxy into the object during collision, a sphere of sufficiently large radius is selected as proxy and this also allows us to feel the point cloud object from both outside and inside of its surface, if the surface is not a closed one. Once collision is detected, we minimize a cost function depending on the current HIP and proxy position and find a new goal position for the proxy corresponding to the local minimum of the cost function. Our rendering algorithm continuously evaluate a tangential vector for the proxy to move over the object surface during collision. The penetration depth is calculated from proxy to the HIP and is used to calculate the reaction force for the haptic device.

Our rendering technique uses regular partitioning of 3D space with $200X200X200$ grid nodes. The happy model (Fig. 9) with the highest number of points in our experiment used a proxy of radius 0.1cm. The neighbourhood search and the tangent vector calculation together took only 0.035ms which is far less than the haptic updation time of 1ms for the model.

References

[1] R. S. Avila and L. M. Sobierajski. A haptic interaction method for volume visualization. *Visualization Conference, IEEE*, 0:197, 1996.

[2] F. Dachille and A. E. Kaufman. Incremental triangle voxelization. In *Graphics Interface*, pages 205–212, 2000.

[3] N. R. El-Far, N. D. Georganas, and A. E. Saddik. An algorithm for haptically rendering objects described by point clouds. In *Proceedings of the 21th Canadian Conference on Electrical and Computer Engineering*, Niagara, ON, Canada, 2008.

[4] A. Guéziec. 'meshsweeper': Dynamic point-to-polygonal-mesh distance and applications. *IEEE Transactions on Visualization and Computer Graphics*, 7:47–61, January 2001.

[5] M. W. Jones, J. A. Brentzen, and M. Sramek. 3d distance fields: A survey of techniques and applications. *IEEE Transactions on visualization and Computer Graphics*, 12:581–599, 2006.

[6] L. Kim, A. Kyrikou, M. Desbrun, and G. Sukhatme. An implicit-based haptic rendering technique. In *Proceeedings of the IEEE/RSJ International Conference on Intelligent Robots*, 2002.

[7] L. Kim, G. S. Sukhatme, and M. Desbrun. A haptic rendering technique based on hybrid surface representation. *IEEE Computer Graphics and Applications, Special Issue on Haptic Rendering - Beyond Visual Computing*, 24(2):66–75, March 2004.

[8] J.-K. Lee and Y. J. Kim. Haptic rendering of point set surfaces. *World Haptics Conference*, 0:513–518, 2007.

[9] S. Mauch. A fast algorithm for computing the closest point and distance transform. Technical report, California Institute of Technology, 2000.

[10] W. A. Mcneely, K. D. Puterbaugh, and J. J. Troy. Six degree-of-freedom haptic rendering using voxel sampling. In *Proc. of ACM SIGGRAPH*, pages 401–408, 1999.

[11] S. M. A. Morgenbesser, H.B. Force shading for haptic shade perception. In *Proceedings of the ASME Dynamic Systems and Control Division*, volume 58, pages 407–412, 1996.

[12] J. C. Mullikin. The vector distance transform in two and three dimensions. *CVGIP: Graph. Models and Image Process.*, 54:526–535, November 1992.

[13] B. A. Payne and A. W. Toga. Distance field manipulation of surface models. *IEEE Comput. Graph. Appl.*, 12:65–71, January 1992.

[14] M. Renz, C. Preusche, M. Ptke, H. peter Kriegel, and G. Hirzinger. Stable haptic interaction with virtual environments using an adapted voxmap-pointshell algorithm. In *Proc. Eurohaptics*, pages 149–154, 2001.

[15] F. Rhodes. Discrete euclidean metrics. *Pattern Recogn. Lett.*, 13:623–628, September 1992.

[16] A. Rosenfeld and J. L. Pfaltz. Sequential operations in digital picture processing. *J. ACM*, 13:471–494, October 1966.

[17] D. C. Ruspini, K. Kolarov, and O. Khatib. The haptic display of complex graphical environments. In *Proc. of ACM SIGGRAPH*, pages 345–352, 1997.

[18] K. Salisbury, F. Conti, and F. Barbagli. Haptic rendering: Introductory concepts. *IEEE Computer Graphics and Applications*, 24(2):24–32, 2004.

[19] L. S.D. and D. A.M. A survey of haptic rendering techniques. *Computer Graphics Forum*, 26(1):50–65, March 2007.

[20] J. A. Sethian. A fast marching level set method for monotonically advancing fronts. *Proceedings of the National Academy of Sciences of the United States of America*, 93(4):1591–1595, 1996.

[21] C. Sigg, R. Peikert, and M. Gross. Signed distance transform using graphics hardware. In *Proceedings of the 14th IEEE Visualization 2003 (VIS'03)*, VIS '03, pages 83–90, Washington, DC, USA, 2003. IEEE Computer Society.

[22] J. N. Tsitsiklis. Efficient algorithms for globally optimal trajectories. *IEEE Transactions on Automatic Control*, 40(9):1528–1538, 1995.

[23] C. Zilles and J. Salisbury. A constraint-based god-object method for haptic display. *Intelligent Robots and Systems, IEEE/RSJ International Conference on*, 3:3146, 1995.