International Journal of Computer Applications (0975 - 8887)
Volume 61 - No. 7, January 2013

Distance Field based Haptic Rendering of Scattered Oriented Points

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

ABSTRACT

This work is aimed at rendering of an object described by a scattered, oriented point set data without explicitly finding the bounding surface. We propose a proxy based rendering technique on a distance field based representation of the object in a regular 3D grid of voxels. Our method initially finds a 3D indicator function in the grid of voxels from the available point set data through sphere packing. The indicator function is further used to find the implicit potential function (distance field) in the voxel grid by iteratively moving the bounding surface of the object inward till all the voxels are covered. Our algorithm uses a gradient descent based minimization of the distance between HIP and proxy during the penetration of HIP in the object. The rendering algorithm interpolates the distance at the proxy point from the neighboring voxels in order to find the gradient for the purpose of moving the proxy. We experimented on a large set of scattered point set data and could effectively render them using a three degree of freedom haptic device.

Keywords:

Haptic interface point (HIP), voxel-based rendering, distance field, haptic rendering.

1. INTRODUCTION

Visual rendering from point samples is a well studied problem in computer graphics. But a very little work has been done in the area of haptic rendering of a sparse point set data. Reconstruction from sparse data is an ill-posed problem and there is no unique solution. In surface reconstruction we are interested in finding a polygonal (commonly, a triangulated mesh) approximation of a moderately dense point cloud. In this paper we aim at finding a volumetric description of the object from an oriented sparse point set data suitable for haptic rendering. In order to effectively render the object geometry, a 3D regular voxel grid is constructed and the minimum Euclidean distance of each grid node from the boundary of the object is calculated. We present a fast algorithm to compute the distance field specifically for a sparse data set. Once this distance field is calculated the object is haptically rendered using a proposed proxy based rendering method. The accuracy of the reaction force depend upon the accuracy with which the distance field is computed. Fig. 1a shows a scattered oriented point cloud in 2D for illustration. Our algorithm first compute a 3D indicator function (shown in Fig. 1a) from the scattered points and then computes the distance function as shown in Fig. 1c. We propose a proxy based rendering of the computed distance function using a 3-DOF haptic device.
To provide a combined hapto-visual immersion of the subject into the virtual world, the object is rendered visually using the zero iso-surface of the computed distance field. Sparse oriented point cloud data, in practice, may be generated using haptic scanning of a real non-deformable object. For haptic scanning of a given object, a force sensor may be attached to the haptic device to obtain the force along with the position information [10]. In the case of deformable objects there could be certain local deformation of the object based on the Hooke's law depending on the applied force by the haptic device. In that case an appropriate back projection is required to get the surface points.
The organization of the paper is as follows. An introduction to the relevant literature is given in section 2. Section 3 explains the proposed method of fast computation of distance field for sparse points, following which the rendering scheme is presented. The results of various experimentations are presented in section 4, and we conclude the paper in section 5.

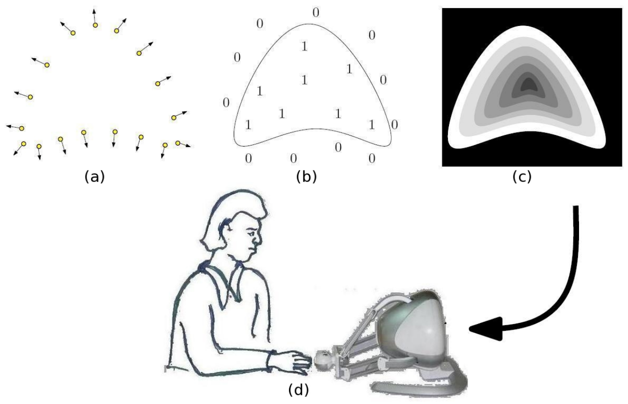


Fig. 1: Illustration of haptic rendering of distance field based description of an object from sparse oriented points.

2. LITERATURE REVIEW

As there is not much work done in the area of sparse point based rendering, we shall first briefly review the work done in the area of surface reconstruction and rendering in computer graphics. In the area of surface reconstruction there are several methods such as Delaunay triangulation [15], alpha shapes [5] or Voronoi diagram [2][1]. These methods typically create a triangular (polygonal, in general) mesh that interpolates most of the points and the associated surface normals help in the rendering process.
There are other methods to directly reconstruct an approximate surface in an implicit form using surface fitting. Global fitting method commonly defines the implicit surface as the zero set of a radial basis function (RBF) fitted to the given surface data[6]. In the case of local fitting methods, a subset of nearby points are considered at a time. A simple scheme is to define the implicit

function as the signed distance from the closest point of the estimated tangent plane[9]. Level set based surface reconstruction method uses distance values accumulated in volumetric grid and the reconstructed surface is represented as the zero-isosurface of the scalar valued distance function[34]. There are also approaches which combines global as well as local fitting scheme such as Poisson approximation by Hoppe *et al.*[12].

In the haptic rendering literature there are mainly two different approaches: Polygon based rendering and voxel based rendering. Some methods also use hybrid approaches[14]. A good introduction to the basic haptic rendering technique is given by Salisbury *et al.* [28] and Laycock *et al.*[16]

2.1 Polygon Based Rendering

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 x is the depth of penetration in the model, the reaction force can be calculated as $F = -kx$, where k is the stiffness constant. Generally we assume the stiffness to be constant throughout the object. But it can be a function of position also, as is common for an object consisting of constituent materials with varying stiffness, ie, $k = K(x, y, z)$ where (x, y, z) are 3D coordinates. It is assumed that $K(x, y, z)$ is known.

The above method has problems while determining the appropriate direction of the force while rendering thin objects. Zilles and Salisbury, and Ruspini *et al.* independently introduced the concept of god-object[35] and proxy algorithm[26], 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. If (x, y, z) are the coordinates of the virtual object and (x_p, y_p, z_p) represents the coordinates of the HIP, the spring energy is given by

$$\begin{aligned} L = \frac{(x-x_p)^2}{2} + \frac{(y-y_p)^2}{2} + \frac{(z-z_p)^2}{2} + \\ l_1(A_1x + B_1y + C_1z - D_1) + \\ l_2(A_2x + B_2y + C_2z - D_2) + \\ l_3(A_3x + B_3y + C_3z - D_3) \end{aligned} \tag{1}$$

where L is the cost function to be minimized, l_1, l_2, l_3 are Lagrange multipliers and A, B, C, D are the homogeneous coefficients for the constraint plane equations. The 'force shading' technique (haptic equivalent of Phong shading) introduced by Morgenbesser and Srinivasan refined the above algorithm while rendering smooth objects[20]. 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

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[3]. The voxmap-point shell algorithm uses the voxel map for stationary objects and point shell for dynamic objects[19][23]. Point shell has been defined as a set of point samples and associated inward facing normals. However, these normals are not available and one needs to compute the normal at every location.

The external surface ∂O of a solid object O can be described by the implicit equation[13] as

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

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 potential is zero defines the implicit surface. This has found applications in haptic rendering. Points with positive potential are outside the object and if $\phi(x, y, z) < 0$ then the point (x, y, z) is inside the surface. A volumetric surface can be defined as a discrete potential function stored on a regular 3D grid. The potential at each point is a signed scalar value which indicates the proximity to the surface. Thus, in effect the potential function is nothing but the distance field of the point set defining an implicit surface. 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[11].

The simplest way to compute the shortest distance to a set of surface points belonging to an object over a 3D discrete grid $\mathbf{V}$ is that, for each voxel $v \in \mathbf{V}$ the distances to all the points are computed and the smallest one is stored. The 3D object geometry is commonly represented using triangular meshes. We can generate distance fields only from polygonal meshes which are closed. If N_v is the number of voxels and N_t is the number of triangles then the brute force method requires $N_v N_t$ steps to compute the distance field. For fast accessing of these triangles, hierarchical data structures can be used. A basic approach to calculate distance to triangular patches has been explained by Payne and Toga[22]. Another hierarchical approach is the mesh sweeper algorithm proposed by Guéziec[8]. 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[7]. Mauch converted the features of a triangular mesh to a polyhedron[18], 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[30].

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 neighbors by adding values from a distance template[25, 24]. 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[21]. The fast marching method (FMM) technique was proposed first by Tsitsiklis[33] and then Sethian[29] 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.

Once the distance field is evaluated and sampled in the 3D grid, the potential at any point inside the grid may be computed using a trilinear interpolation of the eight nearest neighboring grid distances. 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. However this rendering method also suffers from the thin object problem. So we propose a proxy based rendering on the computed distance field. 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 [28]. In the next section we show how we can speed up the computation in case sparse point sets. For haptic rendering from dense point cloud data Leeper *et.al.* have proposed a locally computable implicit function method [17]. Instead of using a point proxy, a spherical volume proxy has been used in [31] and [27]. In the work by [32] rendering was restricted to a point cloud defined by Monge surface. The current work concentrate on how one can render sparse data set.

3. PROPOSED METHOD

The input data for our rendering technique is a set of sparse samples $s(\mathbf{P}, \mathbf{n})$ each consisting of a point $\mathbf{P}$ and the corresponding outward facing unit normal $\mathbf{n}$. We call this type of data as augmented point set. Note that the data $(\mathbf{P}, \mathbf{n})$ resembles more like the concept of a point shell defined in [23]. The proposed algorithm has two steps.

(1) distance field calculation from the augmented point set $(\mathbf{P}, \mathbf{n})$,
(2) haptic rendering with the computed distance field.

In order to find the bounding surface of the object represented by the sparse point set we initially find an indicator function (or characteristic function) from the given point set. The indicator function, in effect, divide the volume into two parts, inside and outside of the object bounding surface. *i.e.*, the indicator function value is one for the region inside the object and zero outside. Once the indicator function is obtained the distance field can be found using technique explained in section sec. 3.2.

3.1 Computation of 3D Indicator Function

The volume reconstruction problem may be defined as: Given a sparse set of augmented points $(\mathbf{P}, \mathbf{n})$, find the bounding surface and hence the distance field over a uniform grid of voxels. We note here that we need to find the distance field for points lying inside the objects only. Since there is no force field outside the object we need to search only in the direction opposite to the available outward surface normal $\mathbf{n}$. For this, the following simple lemma is useful. The proof is quite easy and omitted.

LEMMA 1. *When a single point and the direction of normal at that point is given only one additional point is needed to uniquely define the sphere on which they both lie.*

The bounding surface is obtained as a union of the minimum sized packing spheres for all points in the given data set $s(\mathbf{P}, \mathbf{n})$. This is illustrated for 2D case in Fig. 2. The point $\mathbf{P}_1$ and $\mathbf{n}_1$ are given. In order to mathematically find the embedded circle with the minimum radius, consider three points $\mathbf{P}_2, \mathbf{P}_3$ and $\mathbf{P}_4$ as shown. We want to find the embedded circle of minimum radius at $\mathbf{P}_1$ with at least one of these points $\mathbf{P}_2, \mathbf{P}_3, \mathbf{P}_4$ on the circle. Let $\mathbf{a}_{ij}$ be the vector connecting point i to j. If the dot product $-\mathbf{n}_1.\mathbf{a}_{12}$ is greater than zero (or $\theta_{12} < \pi/2$) it is possible to find a circle with $\mathbf{P}_2$ on its boundary. The radius of the corresponding circle is given by

$$r_{12} = \frac{1}{2}|\mathbf{a}_{12}|\sec(\theta_{12}), \tag{2}$$

from which we can calculate the center as $\mathbf{c}_{12} = \mathbf{P}_1 - r_{12}.\mathbf{n}_1$. The pair $(\mathbf{c}_{12}, r_{12})$ represents a unique circle with $\mathbf{n}_1$ as the outward normal at $\mathbf{P}_1$ and both the points $\mathbf{P}_1, \mathbf{P}_2$ on its circumference. Similarly we can find another circle between the points $\mathbf{P}_1$

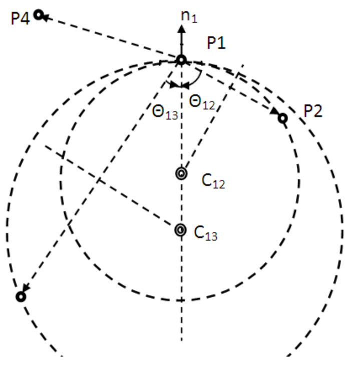


Fig. 2: Illustration of how to calculate the circle of minimum radius at a given sample point for a given point set.

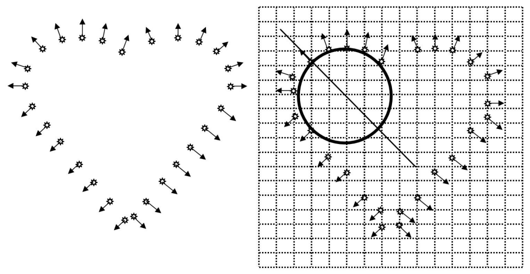


Fig. 3: Illustration in 2D of (a) a sampled point set $(\mathbf{P}, \mathbf{n})$ for an arbitrary object, and (b) a packed circle (sphere in 3D) after mapping these points on a regular grid.

and $\mathbf{P}_3$ of radius $r_{13} = \frac{1}{2}|\mathbf{a}_{13}|\sec(\theta_{13})$, where θ_{13} is the angle between the vectors $\mathbf{a}_{13}$ and $-\mathbf{n}_1$. If $\theta_{ij} > \pi/2$ as with point $\mathbf{P}_4$, it is not possible to have a circle with $\mathbf{P}_4$ on the boundary. If $\theta_{ij} = \pi/2$, it represents a circle with infinite radius. So we search only among the points with $\theta_{1j} < \pi/2$. At point $\mathbf{P}_1$, we find the minimum of all the radii r_{ij} for all the sampled points $j = 2, 3, 4...$ This may be mathematically represented as

$$r_i = \min_{j \neq i} \ r_{ij} \quad \forall\ j \tag{3}$$

where r_i is the packing radius at the sample point i. The minimum of the calculated radius and its corresponding center $\mathbf{c}_i$ represents the embedded circle at $\mathbf{P}_1$. Hence we observe that the computation time for the embedding circle at every sample point is proportional to the number of sample points available. However, one may further reduce the computation if only a smaller subset of points which are in the neighborhood of $\mathbf{P}_1$ is chosen to reduce the search space. As mentioned earlier, the proposed method of computing the distance field is based on the concept of sphere embedding inside an object [4]. We start with forming a regular 3D grid of an appropriate resolution. The resolution could be dependent on how finely the object has been sampled and at what resolution it is required to be rendered in a given haptic platform. The algorithm is explained geometrically in 2D for better understanding.
Consider the augmented point set $s(\mathbf{P}, \mathbf{n})$ as shown in Fig. 3(a). At each point, the position as well as the unit outward surface normal are known. One of the embedded circle with the given point set is shown in Fig. 3(b). We note that the 3D (2D in the figure) volume enclosed by the union of all the packing spheres represents the approximated 3D shape of the sampled object. All the grid nodes inside the union of all these circle are expected to be inside the object so the value of the indicator function is 1 for

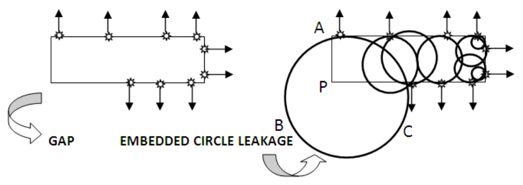


Fig. 4: Illustration of how an improper sampling could introduce a large error in rendering the surface. The embedded circle ABC leaks out of the object as there is no sampled point near the corner P.

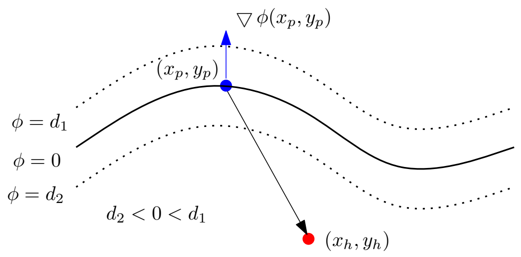


Fig. 5: Part of the distance function corresponding to an arbitrary object in 2D

such nodes and 0 for the remain nodes. If the sampled points are not dense enough, the embedded circle (sphere in the case of 3D) may leak out of the actual boundary of the object from which samples are taken. In Fig. 4 the rectangle is sampled sparsely and the sample points with normals are shown with arrows. In the figure we can see that near the densely sampled region most of the points inside the circles are inside the object. However, the circle ABC leaks out of the object by a large value as there are no points near the corner P.

3.2 Distance Field from Indicator Function

Distance field is computed only for the points inside the object. Using the method described in section 3.1 the 3D indicator function is directly computed from the packed spheres. To find the distance field all the grid nodes with indicator function 0 are set as outside nodes. As we do not want to find the distance function outside the object the outside nodes can be set with distance value 1. Now the grid nodes with at least one outside grid node in the nearest neighborhood is termed as boundary grid node. An iterative procedure is used to move the boundary node inward during the distance calculation. At each level of iteration the distance values are stored in the current boundary nodes are relabeled as outside node. In the initial level (level 0) we store the value zero as distance in the boundary nodes, next level (level 1) of boundary nodes with -1 level 2 with -2 and so on. This process is repeated until the total number of boundary nodes reduces to zero.
Let S be the set of all the inside nodes, δ denotes the boundary nodes. We use a structuring element B of unit radius to erode the set S. Mathematically the distance field is computed as

$$\begin{aligned} \phi(\mathbf{X}) &= 0 \qquad \forall\ \mathbf{X} \notin S \\ &= -n \qquad \forall\ \mathbf{X} \in \delta(S \ominus nB). \end{aligned} \tag{4}$$

3.3 Minimization of the Distance Between Proxy and HIP

Let $\phi(x, y, z)$ denotes the 3D potential function, then $\phi = 0$ represents the zero iso-surface of the object which is actually the bounding surface of the object. In order to render the bounding surface of the object haptically we define a proxy which must not penetrate the bounding surface of the object but can move freely over the surface. Our proxy based haptic rendering technique minimizes the distance between proxy and HIP so that the proxy is constrained to move over the zero iso-surface of the object. To understand the minimization procedure consider an arbitrary 2D distance function shown in Fig. 5. There are three different contours shown in the figure corresponding to three different iso values. The curve shown with thick blue line is the zero iso-contour of the object. The region above the zero iso-contour is outside the object boundary with distances greater than zero while the region below the zero iso-contour represents the inside region of the object with negative distances. The dotted line on either side of the zero iso-contour represents the iso-contour corresponding to distances d_1 and d_2 where $d_1 < 0 < d_2$. In the figure proxy is shown with a blue circle and HIP with a red circle. $\bigtriangledown\phi(x_p, y_p)$ shows the normal to the iso-contour at the proxy point. In free space the proxy position (x_p, y_p) should follow the HIP position (x_h, y_h). But when HIP is penetrated inside the object proxy moves over the object so as to minimize the distance between them. So we can write the objective function to be minimized as $\psi(x, y, z)$ subject to $\phi(x, y, z) = 0$ where

$$\psi = \frac{(x - x_h)^2}{2} + \frac{(y - y_h)^2}{2} + \frac{(z - z_h)^2}{2}. \tag{5}$$

We convert the constrained optimization problem as an unconstrained one using the following objective function

$$\Psi = \frac{1}{2}|\mathbf{X}_p - \mathbf{X}_h|^2 + \frac{\lambda}{2}|\phi(\mathbf{X}_p)|^2 \tag{6}$$

where λ is a regularization parameter. A gradient descent method can be used to minimize the cost function Ψ. Assuming the HIP to be fixed at $\mathbf{X}_h$, (ie, proxy updation is much faster than the haptic interaction which we show to be true in our experimental studies later) Ψ can be written as a function of current proxy position $\mathbf{X}_p$ only. So we can iteratively solve the problem by taking a small positive step size $\gamma > 0$ such that

$$\mathbf{X}_p^{(k+1)} = \mathbf{X}_p^{(k)} - \gamma \bigtriangledown \Psi(\mathbf{X}_p^{(k)}) \tag{7}$$

where $\mathbf{X}_p^{(k)}$ and $\bigtriangledown\Psi(\mathbf{X}_p^{(k)})$ are the proxy location and the gradient of the objective function, respectively, in the current iteration and $\mathbf{X}_p^{(k+1)}$ is the proxy location for the next iteration. The gradient of the objective function can be computed as

$$\bigtriangledown\Psi(\mathbf{X}_p^{(k)}) = (\mathbf{X}_p - \mathbf{X}_h)^{(k)} + \lambda\phi(\mathbf{X}_p^{(k)}) \bigtriangledown \phi(\mathbf{X}_p^{(k)}). \tag{8}$$

Substituting equation (8) in equation (7), the resulting solution becomes

$$\mathbf{X}_p^{(k+1)} = \mathbf{X}_p^{(k)} - \gamma(\mathbf{X}_p - \mathbf{X}_h)^{(k)} - \beta\phi(\mathbf{X}_p^{(k)}) \bigtriangledown \phi(\mathbf{X}_p^{(k)}) \tag{9}$$

where the constant $\beta = \gamma\lambda$, $(\mathbf{X}_p - \mathbf{X}_h)$ is the vector from HIP to the proxy and $\bigtriangledown\phi(\mathbf{X}_p^{(k)})$ is the gradient of the distance function at the current proxy location. In the solution given in equation (9), the term $(\mathbf{X}_p - \mathbf{X}_h)$ tries to minimize the distance between HIP and proxy while the term $\phi(\mathbf{X}_p) \bigtriangledown \phi(\mathbf{X}_p)$ keeps the proxy near the bounding surface of the object during collision. When $\phi(\mathbf{X}_p) < 0$, proxy is outside the object and the resulting solution becomes $\mathbf{X}_p = \mathbf{X}_h$.

3.4 Haptic Rendering

As the distance samples are stored on a regular grid, distance at any point inside the grid can be interpolated from the 8 nearest neighboring grid nodes. The gradient can also be computed at all the nodes directly from the distance field. Haptic rendering process includes two steps.

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

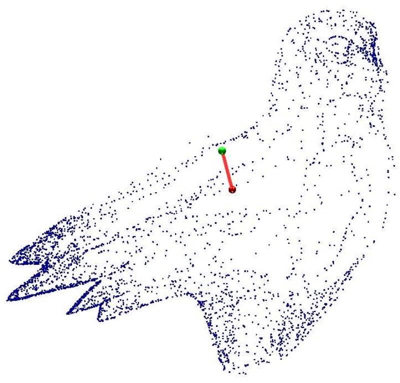
3.4.1 Collision between the HIP and the object:. As we move the haptic device inside the 3D grid the distance is computed at the proxy location, $\mathbf{X}_p$ by trilinear interpolation of the distance stored at the grid location. If the interpolated distance is greater than zero, the proxy lies outside the object, proxy moves with the HIP, no collision is detected and therefore no force is fed back.

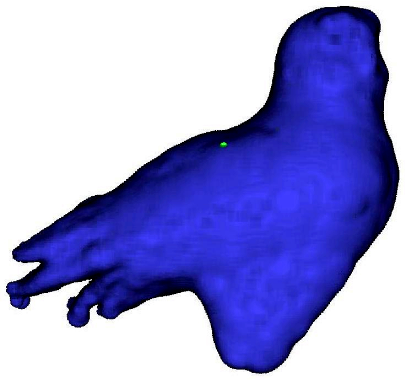
3.4.2 Force Computation for Rendering:. If the interpolated distance at the proxy location is less than zero a collision is detected and an appropriate force has to be fed back. Once collision is detected proxy moves tangentially over the iso-surface so as to minimize the distance between proxy and HIP as discussed in sec. 3.3, while the reaction force is as given in equation (10) where k is the Hooke's constant and $\mathbf{X}_h - \mathbf{X}_p$ is the penetration depth of HIP in the object.

$$\mathbf{f} = -\,k\,(\mathbf{X}_h - \mathbf{X}_p). \tag{10}$$

4. EXPERIMENTAL RESULTS

The proposed method was implemented in visual C++ in a windows XP platform with a CORE 2QUAD CPU @ 2.66GHZ with 2GB RAM. In order to generate the data set for our algorithm, we created virtual objects using polygonal meshes and they are rendered haptically with the standard god-object rendering technique. These virtual objects are scanned manually using a 3-DOF haptic device and the position and force values are sampled at different points. The distance field is calculated in a fixed regular grid of size 300X300X300. The size and spatial resolution of the grid depend on two factors: the interaction space of the haptic device used to render the object and the resolution at which the object should be rendered. We use a 3-DOF haptic device from NOVINT with a 4 inch cube haptic interaction space. Moreover, the object is visually rendered as points on the zero-iso-surface of the computed distance field. This allows the user a joint hapto-visual experience while touching the object. We have selected the above grid size with an appropriate resolution so as to fit the active space of the haptic device suitable with our experimental setup. During the haptic rendering phase, the calculation of force at a point takes only $0.02\mathrm{ms}$ with the above grid size, which is much lower than the required limit of $1\mathrm{ms}$ for a smooth haptic rendering.
Depending on the size and shape of the object the distance field computation may use only a small portion of the grid size. For example, the point cloud data shown in Fig. 6 with 3,725 points uses a grid of size 186X173X87 and it took approximately 38.09 seconds to calculate the precomputable distance field. The algorithm computes the distance values at 8,24,463 inside grid nodes. The zero iso-surface are shown in Fig. 6(b) for the visual rendering purpose. The green sphere in the figure shows the proxy point and the red sphere shows the HIP point during the rendering process. The red line indicate the penetration depth at the shown HIP point. The user experience on handling the object in the haptic space was also found to be very satisfactory.
Another sparse oriented point cloud data shown in Fig. 8a has 5153 points. The distance field is computed in a grid of size 126X187X72. The computation time is 26.25s for 3,04,072 internal nodes. The proxy and HIP positions are also shown in the figure. Point set data with 3751 points are shown in Fig. 8b. It took 110.82 seconds to calculate the distance field at 11,29,209 inside points. Another point set data corresponding to a water hydrant is shown in Fig. 8c with 3756 points. Distance field is computed at 5,86,222 points in 30.86 seconds. Augmented point set data $s(\mathbf{P}, \mathbf{n})$ is generated by haptic scanning of a given polygonal mesh using a 3-DOF haptic device. To perform haptic sampling the polygonal mesh is haptically rendered using the god object rendering technique. The position of the haptic device and the corresponding reaction force are sampled during the interaction with the object. The device positions corresponding to the non zero force sample are then projected back to the surface to


(a)


(b)

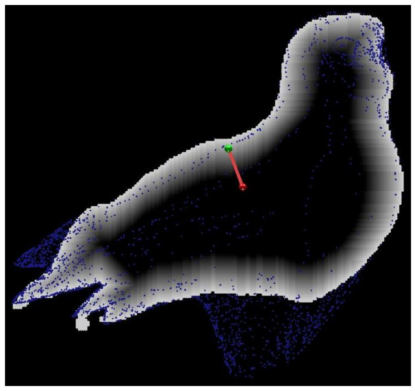
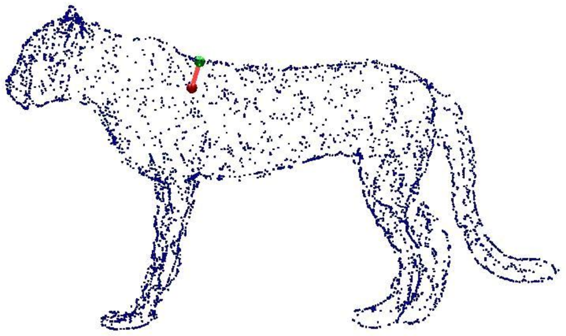
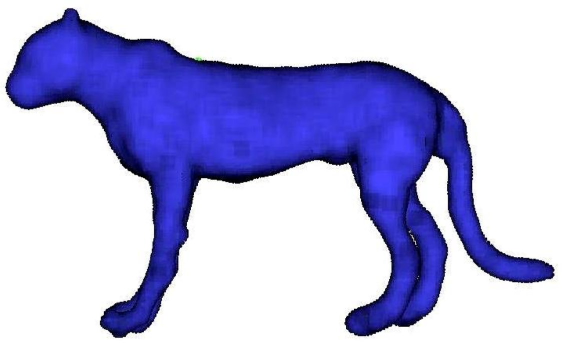

(c)

Fig. 6: (a) Scattered oriented points and (b) zero iso-surface of the distance field visually rendered using a known light source direction (c) distance field for a particular cross section of the object (data courtesy: 'www.top3dmodels.com').

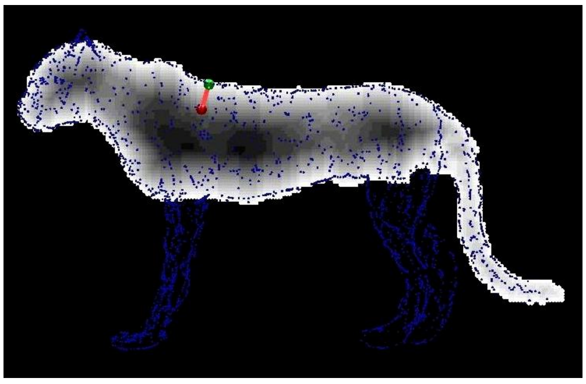
get the point set, the direction of reaction force gives the orientation of each point.
In practice it is not required to find the distance field at all the inside grid nodes, since the proxy rarely sinks more than 3-4 voxel length in the object during the minimization of equation (6). Distance calculation only up to 5 or 6 nodes inside the actual bounding surface nodes is enough for a stable haptic feedback. Table. 1 compares the computational time requirement for different data sets. Note that this is required to be pre-computed only once following while the haptic interaction could be very fast as the proxy updation requires 0.02ms only.
In order to validate the results obtained using the proposed rendering technique we used a point cloud data corresponding to a



(a)



(b)

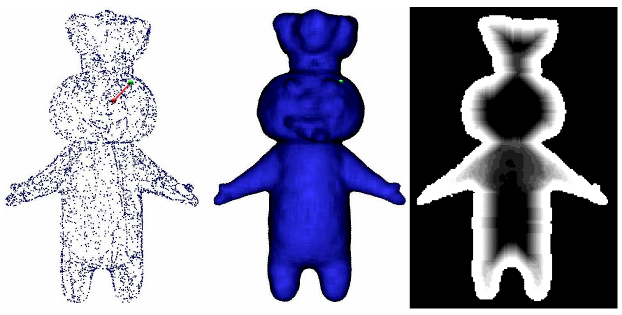


(c)

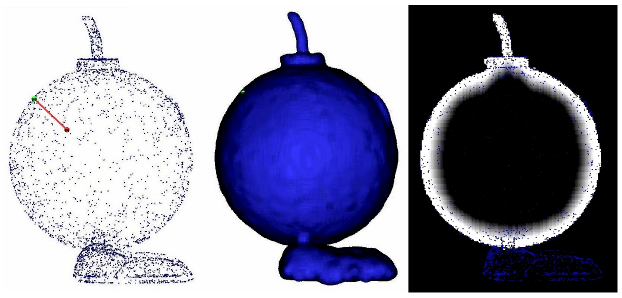
Fig. 7: (a) Scattered oriented points and (b) zero iso-surface of the distance field visually rendered using a known light source direction (c) distance field for a particular cross section of the object (data courtesy: 'www.turbosquid.com').

Table 1. : Summary of computational requirement for various point cloud data used.

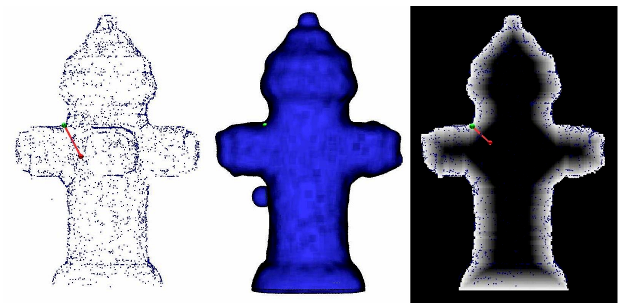
Object name	Number of oriented points	Number of inside nodes	Distance field computation time (s)
Hydrant	3,756	5,86,222	30.86
Owl	3,725	8,24,436	38.09
Boy	5,153	3,04,072	26.26
Bomberman	3,751	11,29,209	110.82
Jaguar	4,433	1,34,987	11.673



(a)



(b)



(c)

Fig. 8: Scattered oriented points, visually rendered zero iso-surface of the distance field for a known light source direction and a cross section of the computed distance field for 3 different objects. (a) boy (b) bomberman (c) water hydrant (data courtesy: 'www.turbosquid.com').

known spherical object such that the bounding surface $\phi(\mathbf{X})$ is precisely known. The rendered reaction force for a particular interaction with the object is compared with the force computed using the known, implicit equation of the sphere. The reason for selecting a sphere for the comparison purpose is that the sphere equation can readily give the penetration depth of the HIP in the object. The surface of the sphere is interacted with a 3-DOF haptic device. The haptic position and the corresponding rendered reaction force are sampled from the device during the interaction. The ideal reaction force is computed using the implicit sphere equation. The magnitude of force is plotted as a function of time for both the cases and is shown in Fig. 9. The red line in the figure shows the force computed using the implicit sphere equation. The blue line shows the rendered force using the proposed technique. The rendered force can be seen to be very close to the actual force substantiating our claim that the proposed method does provide a good haptic experience to the user.

5. CONCLUSIONS

In this work we haptically rendered a sparse oriented point cloud data without explicitly finding the bounding surface. We propose a novel distance field based rendering of an oriented point

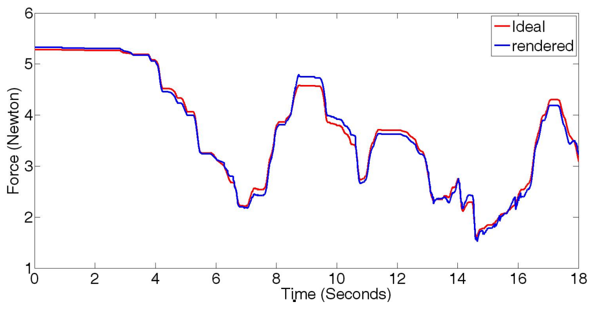


Fig. 9: Comparison of the rendered force with the ideal one for a particular interaction with a known spherical object.

cloud in a regular 3D grid of voxels. Once the points are mapped to the nearest nodes, all the remaining processing is done only in the 3D grid. Distance field is sampled in each grid node by the method of embedded spheres inside the object. The zero-isosurface of the distance field which is the boundary surface of the object is visually shown for a combined hapto-visual experience. In effect we use a distance field based haptic rendering and a simultaneous point based graphic rendering. We validated our results using scattered oriented data corresponding to a spherical object and found that the proposed rendering method works very well with scattered data.

6. REFERENCES

[1] Nina Amenta, Marshall Bern, and Manolis Kamvysselis. A new Voronoi-based surface reconstruction algorithm. In *SIGGRAPH*, pages 415–421, 1998.

[2] Nina Amenta, Sunghee Choi, and Ravi Krishna Kolluri. The power crust, unions of balls, and the medial axis transform. *Computational Geometry: Theory and Applications*, 19:127–153, 2000.

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

[4] Ilya Baran and Jovan Popovic. Automatic rigging and animation of 3d characters. *ACM Trans. Graph.*, 26(3):72, 2007.

[5] Fausto Bernardini, Joshua Mittleman, Holly Rushmeier, Claudio Silva, Gabriel Taubin, and Senior Member. The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions on Visualization and Computer Graphics*, 5:349–359, 1999.

[6] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans. Reconstruction and representation of 3D objects with radial basis functions. In *Computer Graphics (SIGGRAPH 01 Conf. Proc.), pages 6776. ACM SIGGRAPH*, pages 67–76. Springer, 2001.

[7] Frank Dachille and Arie Kaufman. Incremental triangle voxelization. In *Graphics Interface*, pages 205–212, 2000.

[8] André Guéziec. 'Meshsweeper': Dynamic Point-to-Polygonal-Mesh Distance and Applications. *IEEE Transactions on Visualization and Computer Graphics*, 7:47–61, January 2001.

[9] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *COMPUTER GRAPHICS (SIGGRAPH 92 PROCEEDINGS)*, pages 71–78, 1992.

[10] R. Hover, M. Harders, and G. Szekely. Data-driven haptic rendering of visco-elastic effects. In *Proceedings of the 2008 Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 201–208, Washington, DC, USA, 2008. IEEE Computer Society.

[11] Mark W. Jones, J. Andreas Brentzen, and Milos Sramek. 3D Distance Fields: A Survey of Techniques and Applications. *IEEE Transactions on visualization and Computer Graphics*, 12(4):581–599, 2006.

[12] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, SGP '06, pages 61–70, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association.

[13] Laehyun Kim, Anna Kyrikou, Mathieu Desbrun, and Gaurav Sukhatme. An implicit-based haptic rendering technique. In *Proceeedings of the IEEE/RSJ International Conference on Intelligent Robots*, volume 3, pages 2942–2948, 2002.

[14] Laehyun Kim, Gaurav S. Sukhatme, and Mathieu 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.

[15] Ravikrishna Kolluri, Jonathan Richard Shewchuk, and James F. O'Brien. Spectral surface reconstruction from noisy point clouds. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '04, pages 11–21, New York, NY, USA, 2004. ACM.

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

[17] Adam Leeper, Sonny Chan, and Kenneth Salisbury. Constraint based 3-DOF haptic rendering of arbitrary point cloud data. In *RSS Workshop on RGB-D Cameras*, University of Southern California, June 2011.

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

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

[20] Srinivasan 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.

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

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

[23] Matthias Renz, Carsten Preusche, Marco Ptke, Hans peter Kriegel, and Gerd Hirzinger. Stable haptic interaction with virtual environments using an adapted voxmap-pointshell algorithm. In *Proc. Eurohaptics*, pages 149–154, 2001.

[24] Frank Rhodes. Discrete Euclidean metrics. *Pattern Recogn. Lett.*, 13:623–628, September 1992.

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

[26] Diego C. Ruspini, Krasimir Kolarov, and Oussama Khatib. The haptic display of complex graphical environments. In *Proc. of ACM SIGGRAPH*, pages 345–352, 1997.

[27] Fredrik Ryden, Sina Nia Kosari, and Howard Jay Chizeck. Proxy method for fast haptic rendering from time varying point clouds. In *IROS*, pages 2614–2619. IEEE, 2011.

[28] Kenneth Salisbury, Francois Conti, and Federico Barbagli. Haptic rendering: Introductory concepts. *IEEE Computer Graphics and Applications*, 24(2):24–32, 2004.

[29] 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.

[30] Christian Sigg, Ronald Peikert, and Markus 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.

[31] K. G. Sreeni and Subhasis Chaudhuri. Haptic Rendering of Dense 3D Point Cloud Data. In *IEEE Haptic Symposium*, pages 333–339, Vancouver, BC, Canada, March 4-7 2012.

[32] K. G. Sreeni, K. Priyadarshini, A. K. Praseedha, and Subhasis Chaudhuri. Haptic Rendering of Cultural Heritage Objects at Different Scales. In *Eurohaptics*, Tampere, Finland, June 12-15 2012.

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

[34] Hong-Kai Zhao, Stanley Osher, and Ronald Fedkiw. Fast surface reconstruction using the level set method. In *Proceedings of the IEEE Workshop on Variational and Level Set Methods (VLSM'01)*, VLSM '01, pages 194–, Washington, DC, USA, 2001. IEEE Computer Society.

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