Multimed Tools Appl
DOI 10.1007/s11042-015-2490-z

6DoF haptic rendering using distance maps over implicit representations

Konstantinos Moustakas

Received: 16 April 2014 / Revised: 25 December 2014 / Accepted: 1 February 2015

Abstract This paper presents a haptic rendering scheme based on distance maps over implicit surfaces. Using the successful concept of support planes and mappings, a support plane mapping formulation is used so as to generate a convex representation and efficiently perform collision detection. The proposed scheme enables, under specific assumptions, the analytical reconstruction of the rigid 3D object's surface, using the equations of the support planes and their respective distance map. As a direct consequence, the problem of calculating the force feedback can be analytically solved using only information about the 3D object's spatial transformation and position of the haptic interaction point. Moreover, several haptic effects are derived by the proposed mesh-free haptic rendering formulation. Experimental evaluation and computational complexity analysis demonstrates that the proposed approach can reduce significantly the computational cost when compared to existing methods.

Keywords Haptic rendering · Collision detection · Implicit representation · Support plane mapping

1 Introduction

Human perception combines information of various sensors, including visual, aural, haptic, olfactory, in order to perceive the environment. Virtual reality applications aim to immerse the user into a virtual environment by providing artificial input to its interaction sensors (i.e., eyes, ears, hands, etc.). The visual and aural inputs are the most important factors in human-computer interaction (HCI). However, virtual reality applications will remain far from being realistic without providing to the user the sense of touch. The use of haptics augments the standard audiovisual HCI by offering to the user an alternative way of interaction with the virtual environment [3]. However, haptic interaction involves complex and computationally

K. Moustakas (✉)
Electrical and Computer Engineering Department, University of Patras, Rion-Patras, Greece
e-mail: moustakas@upatras.gr

intensive processes, like collision detection or distance calculation, that place significant barriers in the generation of accurate and high fidelity force feedback.

The sense of touch can be given to users interacting with virtual reality environments through a haptic device. Typical haptic devices include, but are not limited to, exoskeletons that can apply forces to the fingertips and pen-like robotic interfaces. Through such devices the users can get the sense of touch while interacting with a virtual environment. However, many computationally intensive procedures are implied in order to get a realistic high fidelity sense of touch.

The haptic devices can be considered as interfaces that can apply the force, they are instructed, at a specific point of the device (e.g., single point of interaction for pen-like devices, fingertips for exoskeletons). The proper estimation of this force, while navigating a virtual environment, is called haptic rendering. Moreover, this point, where force is exerted, is supposed to move within a virtual environment and is called Haptic Interaction Point (HIP).

Haptic devices, apart from being capable to exert force (i.e. serve as output), also enable the user to move the haptic interaction point within the virtual environment (i.e. serve as input). As soon as the HIP penetrates a virtual object of the environment, collision detection algorithms can identify the colliding parts of the objects and then estimate the force that is applied both to the HIP and the virtual object in order to "resolve the collision". The force applied to the virtual object "moves it" within the virtual environment, while the force applied on the HIP is rendered through the haptic device and is "felt" by the user.

1.1 Related work

Seen from a computational perspective, haptic rendering can be decomposed in two different but heavily interrelated processes, namely collision detection and force calculation. Initially, collisions have to be identified and localized and then the resulting force feedback has to be estimated so as to accurately render the force that will be fed back to the user using specific assumptions on the physical model involved.

Concerning collision detection, most approaches presented in the past are based on building a Bounding Volume Hierarchy (BVH) around the object consisting of primitive objects like spheres [11], OBBs [10] or volumes based on complex dynamically transforming geometries k-DOPs [12]. The hierarchy of the processed mesh is built, based on topological criteria. The root of the tree built, contains the entire object, while the leafs just contain single triangles. Different algorithms for building this hierarchy have been proposed in the past [10, 26]. In these methods, if intersection is detected between the BV of the root and an object, the algorithm checks for intersection between the child nodes of the tree and the object and so on, until the leaf nodes are reached and the accurate points of a potential collision are found.

The intersection tests between BVs are based on the Separating Axis Theorem for convex objects [5, 10]. The theorem states that for a pair of disjoint convex objects there exists an axis, such that the projections of the objects on this axis do not overlap. Intersection tests for BVs exploit this theorem by testing the existence of a Separating Axis in a set of candidate axes. The basic difference among different types of BVs is the number of axes needed to be tested. There is a trade-off between the bounding efficiency and the computational cost of the intersection test. BVs with low efficiency, such as spheres, can be tested very fast for intersection while more efficient bounding volumes, such as the OBBs, require much more computation.

Despite the accuracy of these methods, which are extensively used in the literature, the computational cost of performing the intersection tests between the objects is very high, especially when these consist of a large number of triangles or when they participate in multiple simultaneous collisions. Recently, methods for collision detection based on distance fields were introduced [9, 16, 18, 25], which decrease the computational cost dramatically. These methods require, at a preprocessing stage, to generate distance fields for the objects, which are stored in arrays. In particular, a bounding box is assumed for each object. A 3D grid is defined inside each box and a distance value is assigned to every point of the grid, which indicates the distance of the specific point from the mesh. Negative values indicate that the point lies inside the mesh. These distance values are usually obtained using level set [19] and fast marching algorithms [23].

Concerning haptic rendering research can be divided into three main categories [14]: Machine Haptics, Human Haptics and Computer Haptics [24]. Machine Haptics is related to the design of haptic devices and interfaces, while Human Haptics is devoted to the study of the human perceptual abilities related to the sense of touch. Computer Haptics, or alternatively haptic rendering, studies the artificial generation and rendering of haptic stimuli for the human user [17]. It should be mentioned that the proposed framework takes into account recent research on human haptics, while it provides mathematical tools targeting mainly the area of computer haptics.

The simplest haptic rendering approaches focus on the interaction with the virtual environment using a single point. Many approaches have been proposed so far both for polygonal, non-polygonal models, or even for the artificial generation of surface effects like stiffness, texture or friction, [13]. The assumption, however, of a single interaction point limits the realism of haptic interaction since it is contradictory to the rendering of more complex effects like torque. On contrary, multipoint, or object based haptic rendering approaches use a particular virtual object to interact with the environment and therefore, besides the position of the object, its orientation becomes critical for the rendering of torques. Apart from techniques for polygonal and non-polygonal models [13], voxel based approaches for haptic rendering [21] including volumetric haptic rendering schemes [20] have lately emerged. Additionally, research has also tackled with partial success the problem of haptic rendering of dynamic systems like deformable models and fluids [1].

1.2 Motivation and contribution

In general, with the exception of some approaches related to haptic rendering of distance or force fields [2], one of the biggest bottlenecks of current schemes is that haptic rendering depends on the fast and accurate resolution of collision queries. The proposed approach aims to widen this bottleneck by providing a free-form implicit haptic rendering scheme based on support plane mappings. In particular, a 3D object is initially modelled using the associated support plane mappings [27]. Then the distance of the object's surface from the support plane is mapped at discrete samples on the plane and stored at a preprocessing step. During run-time and after collision queries are resolved, the force feedback can be analytically estimated, while several haptic effects, like friction, texture, etc. can be easily derived. This results in constant time haptic rendering based only on the 3D transformation of the associated object and the position of the haptic probe.

The rest of the paper is organized as follows. Section 2 briefly describes the support plane mapping formulation and Section 3 the haptic rendering scheme. In Section 4 several haptic effects are derived using the proposed formulation, while in Section 5 the computational complexity and simulation results of the approach are analyzed. Finally, conclusions are drawn in Section 6.

2 Support plane mappings

Support planes are a well studied subject of computational geometry and have been employed in algorithms for the separation of convex objects [4, 7, 27]. From a geometrical perspective, a support plane E of a 3D convex object O is a plane such that O lies entirely on its negative halfspace H_E^- as illustrated in Fig. 1. Support planes have become useful in previous algorithms based on the concept of support mappings. A support mapping is a function that maps a vector $\mathbf{v}$ to the vertex of $vert(O)$ of object O that is "most" parallel to $\mathbf{v}$ [8, 27]. As a direct consequence, a support plane can be defined as the plane that passes through $s_O(\mathbf{v})$, the support mapping of $\mathbf{v}$, and is parallel to $\mathbf{v}$.

2.1 Collision detection using SPMs

The importance of support planes is intuitively apparent: they provide an explicit way of deciding whether another object could possibly intersect with the one that the support planes refers to. Based on this simple but important feature of support planes, a slightly more generalized formulation can be derived introducing the concept of support plane mappings [28] by the following definitions:

Definition 1 E is a *Support Plane* (SP) of the object O if

1. $\mathbf{x} \in H_E^-, \ \forall \mathbf{x} \in O$
2. E and O have at least one common point.

Definition 2 Let the object O and E_O be a set of Support Planes of O. A Support Plane Mapping (SPM) of O is defined as $M_O(\mathbf{v}) = E \in E_O : \mathbf{v} \cdot \mathbf{n}_E = \max\{\mathbf{v} \cdot \mathbf{n} | \mathbf{n} \in \mathbf{n}_{E_O}\}$ where $\mathbf{n}_E$ denotes the normal of support plane E and $\mathbf{n}_{E_O}$ the set of all normals in E_O.

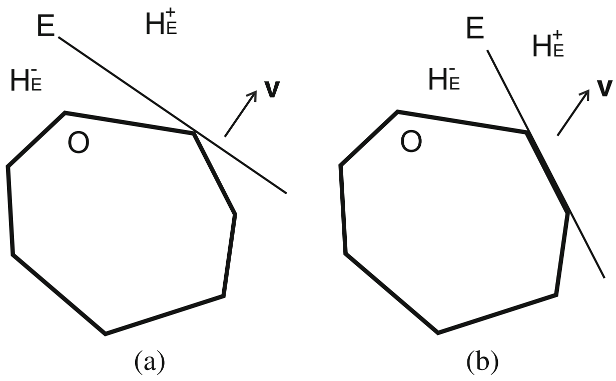

Fig. 1 Support Plane Mappings. In **a** the Support Plane E is generated using a vertex-based mapping, i.e. $E = M_O(\mathbf{v})$, and the normal is parallel to the direction of the input vector $\mathbf{v}$. In **b** E comes from a face-based mapping and it lies on a face of O

The difference between the above definitions and previous work is that the above definitions do not make any assumption about the convexity of O or, concerning only the SPM, about the set of Support Planes E_O. For example, a Support Plane Mapping can be constructed using the infinite set of all support planes of object O and the support mapping s_O. This kind of SPM is referred as the *Vertex-based* SPM (Fig. 1a) of O, since s_O maps to $vert(O)$. The Vertex-based SPM is actually an alternative definition for generating support planes [7]. Since there is no restriction on the used set of Support Planes E_O, another approach to construct a SPM would be to use the set of support planes that lie on at least one face of O. This kind of mapping is referred as the *Face-based* SPM (Fig. 1b) of O. Note that both Vertex-based and Face-based SPMs are uniquely defined for each object.

In practice we are mostly interested for having enough support planes to surround the given object. Therefore we define the *fully bounding* SPM of an object O as a SPM such that the planes of the respective E_O form a finite sub-space $G = \bigcap_i H^-_{E_i}$ for every $E_i \in E_O$, that completely contains O. This sub-space G serves, implicitly, as a convex bounding representation of the object. Note that both Vertex-based and Face-based SPMs are fully bounding SPMs. Based on this formulation of support plane mappings, collision rejection can be performed. In particular, SPMs allow for an extra step of collision culling after the bounding boxes collide. Thus, even if the broad phase of collision detection reports that two bounding volumes collide, the algorithm performs en extra step of collision culling using the SPMs, before entering the narrow phase of collision detection. This extra culling step, that further reduces the number of times an algorithm has to enter the narrow phase of collision detection, has been reported to increase performance of collision detection by about one order of magnitude. More detail on the approach is given in [28].

2.2 Scalar and vectorial haptic support plane maps

After collision is detected, the force feedback provided to the user through the haptic device has to be calculated. In the present framework, force feedback is obtained directly from the model adopted for collision detection, thus handling collision detection and haptic rendering in an integrated way, as described in the sequel.

Let the parametric form of the support plane equation $S_{SP}(\eta, \omega)$ be:

$$\mathbf{S_{SP}}(\eta, \omega) = \mathbf{x_0} + \eta\mathbf{u} + \omega\mathbf{v}, \forall\eta, \omega \in \Re \tag{1}$$

where $\mathbf{u}$ and $\mathbf{v}$ constitute an orthonormal basis of the support plane, η, ω the indices of the parametric definition and $\mathbf{x_0}$ its origin.

Assuming now a dense discretization of the η, ω space, we can define a discrete distance map of the support plane SP and the underlying manifold mesh surface S_{mesh}, by calculating the distance of each point of SP from S_{mesh}:

$$D_{SP}(\eta, \omega) = ICD\,(S_{SP}, S_{mesh}) \tag{2}$$

where ICD calculates the distance of every point sample (η, ω) of the support plane SP, alongside the normal direction at point (η, ω), from the mesh S_{mesh} and assigns the corresponding values to the distance map $D_{SP}(\eta, \omega)$. The distance map is used in the sequel to analytically estimate the force feedback.

It should be mentioned that the above scalar distance maps accurately encode the surface if and only if there is an injective projection of all surface parts with at least one support plane. However, in the existence of large concavities like the case of Fig. 2 such an injective projection does not exist. In that case, vectorial distance maps can be utilized that include information about the distance of all sections of the ray cast in the normal direction of the

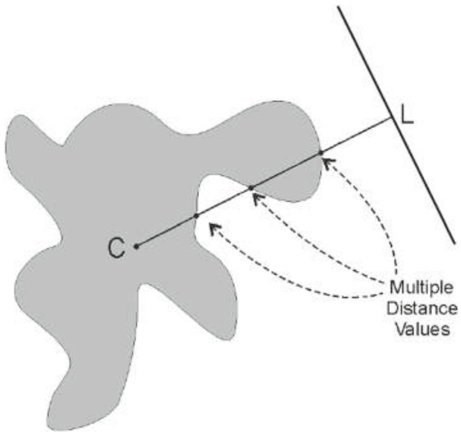

Fig. 2 Vectorial distance maps

support plane to the object mesh as illustrated in Fig. 2. This way, all occluded parts of the objects can be processed by the SPM-based haptic rendering scheme. In the following and without loss of generality scalar distance maps are assumed.

3 Haptic rendering using SPMs

Based on the support planes and their associated distance maps, the force feedback can be analytically estimated both for the three and six degrees of freedom case as described in the following sections.

3.1 3DoF point-based haptic rendering

Referring to Fig. 3, let point $\mathbf{H}_p$ be the haptic interaction point (HIP) and S_{mesh} represent the local surface of the object. $\mathbf{H}_p$ lies inside the surface of the object S_{mesh}.

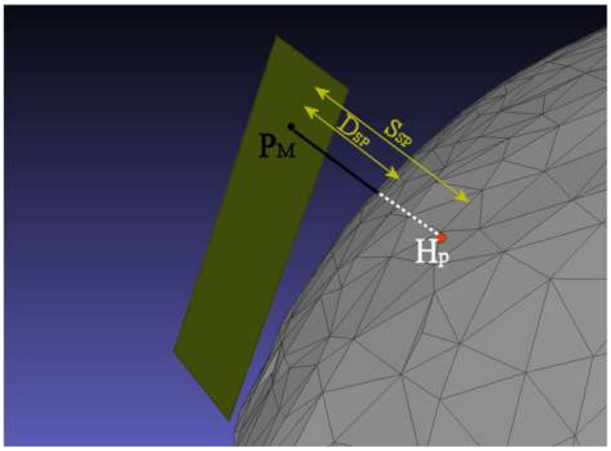

Fig. 3 Distance calculation using distance maps over support planes

Let also S_{SP} represent the distance of point $\mathbf{H}_p$ from the support plane, which corresponds to point $\mathbf{P}_M$ on the SP. If collision is detected, the absolute value of the force fed onto the haptic device is obtained using a penalty based method. In particular:

$$\|\mathbf{F}\| = k \cdot |S_{SP} - D_{SP}(\mathbf{P}_M)| \tag{3}$$

where k is the spring constant. $D_{SP}(\mathbf{P}_M)$ is the distance of point $\mathbf{P}_M$ from the mesh and is stored in the distance map of the support plane. Notice that the term $|S_{SP} - D_{SP}(\mathbf{P}_M)|$ is an approximation of the actual distance of $\mathbf{H}_p$ from the mesh that becomes more accurate if the support plane surface approximates well the mesh.

The direction of the force should in general be perpendicular to the local area, where collision is detected. An obvious solution to the evaluation of the direction of this force would be to detect the surface element (i.e. triangle), where the collision occurred and to provide the feedback perpendicularly to it. This approach is not only computationally intensive, but also results in non-realistic non-continuous forces at the surface element boundaries. In the present framework the analytical approximation of the mesh surface is used utilizing the already obtained SP approximation and the distance map. Based on this approximation the normal to the object's surface can be approximated rapidly with high accuracy. In particular, if $D_{SP}(\eta, \omega)$ is the scalar function of the distance map on the support plane, as previously described, the surface S_{mesh} of the modelled object can be approximated by (4) (Fig. 3):

$$\mathbf{S}_{mesh}(\eta, \omega) = S_{SP}(\eta, \omega) - D_{SP}(\eta, \omega)\mathbf{n}_{SP} \tag{4}$$

where S_{SP} is the surface of the support plane, D_{SP} the associated distance map and $\mathbf{n}_{SP}$ its normal vector that can be easily evaluated through $\mathbf{n}_{SP} = \mathbf{u} \times \mathbf{v}$.

Now the calculation of the force feedback demands the evaluation of the normal vector $\mathbf{n}_S$ on the object's surface that is obtained through (5). In the following the brackets (η, ω) will be omitted for the sake of simplicity.

$$\mathbf{n}_S = \frac{\partial \mathbf{S}_{mesh}}{\partial \eta} \times \frac{\partial \mathbf{S}_{mesh}}{\partial \omega} \tag{5}$$

where

$$\frac{\partial \mathbf{S}_{mesh}}{\partial \eta} = \frac{\partial \mathbf{S}_{SP}}{\partial \eta} - \frac{\partial D_{SP}}{\partial \eta}\mathbf{n}_{SP} - D_{SP}\frac{\partial \mathbf{n}_{SP}}{\partial \eta} \tag{6}$$

Since $\mathbf{n}_{SP}$ is constant over SP, (6) becomes:

$$\frac{\partial \mathbf{S}_{mesh}}{\partial \eta} = \mathbf{u} - \frac{\partial D_{SP}}{\partial \eta}\mathbf{n}_{SP} \tag{7}$$

A similar formula can be extracted for $\frac{\partial \mathbf{S}_{mesh}}{\partial \omega}$:

$$\frac{\partial \mathbf{S}_{mesh}}{\partial \omega} = \mathbf{v} - \frac{\partial D_{SP}}{\partial \omega}\mathbf{n}_{SP} \tag{8}$$

All above terms can be computed analytically, except from $\frac{\partial D_{SP}}{\partial \eta}$ and $\frac{\partial D_{SP}}{\partial \omega}$ that are computed numerically.

Substituting now (4), (6), (7), (8) in (5) the normal direction $\mathbf{n}_S$ can be obtained.

Since, the direction of the normal along the surface of the modelled object is obtained using (5), the resulting force feedback is calculated through:

$$\mathbf{F}_h = k\,|S_{SP} - D_{SP}(\mathbf{P}_M)|\,\frac{\mathbf{n}_S}{\|\mathbf{n}_S\|} \tag{9}$$

3.2 Spherical haptic interaction point

The aforementioned equations can be directly applied for the case of an infinite-small interaction point. However, in practise, for the haptic proxy a rigid body is considered, not only for performing 6DoF haptic rendering, but also for 3DoF force feedback estimation, so as to allow for noise-less (small force discontinuities) interaction by averaging the forces applied by the mesh to the proxy. A typical proxy object is the sphere. An analysis on how to directly use a spherical probe with the proposed framework is described in the sequel. It should be emphasized that the analysis could be potentially generalized for any object that can be represented in an implicit form.

Referring to Fig. 4 that for simplicity depicts the 2D case, let C_S denote the support set of the projection of the sphere on the support plane. Moreover, let S^+ and S^- denote the surface of the sphere that is further and closer to the support plane than the sphere center respectively.

Then the force feedback that is due to S^+ can be estimated from the following formula:

$$\begin{aligned}\mathbf{F}_{S^+} &= \frac{1}{N^+}\int\limits_{S\in C_S} k\cdot\max\left(S^+(\eta,\omega)-S_{SP}(\eta,\omega),0\right)\cdot\mathbf{n}_S dS\\ &= \frac{1}{N^+}\int\int\limits_{\eta,\omega\in C_S} k\cdot\max\left(S^+(\eta,\omega)-S_{SP}(\eta,\omega),0\right)\cdot\mathbf{n}_S d\eta d\omega\\ &= \frac{1}{N^+}\sum\sum_{\eta,\omega\in C_S} k\cdot\max\left(S^+(\eta,\omega)-S_{SP}(\eta,\omega),0\right)\cdot\mathbf{n}_S\end{aligned} \tag{10}$$

where N^+ is the number of the (η,ω) points on the distance map that satisfy the relation $S^+(\eta,\omega) > S_{SP}(\eta,\omega)$, are thus contributing to the estimation of the force feedback.

An identical derivation can be formulated for $\mathbf{F}_{S^-}$ that estimates the force due to the surface S^- of the sphere. The final force can be trivially obtained through:

$$F = \frac{N^+}{N_{total}}F^+ + \frac{N^-}{N_{total}}F^- \tag{11}$$

The careful reader would notice that using the above approach only the surface of the sphere contributes to the estimation of the force feedback. A similar equation, using the sphere volume that is colliding with the object, for the estimation of the force feedback, can be easily derived by using a formulation similar to the one of (10) and a triple integral adding one more dimension along the normal direction of the support plane.

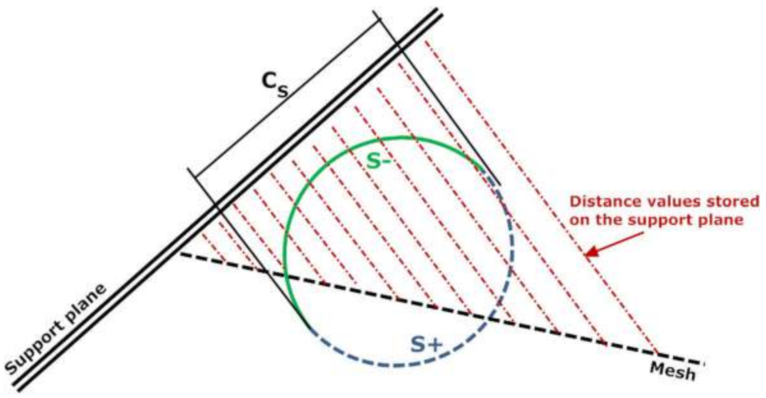

Fig. 4 Distance calculation using distance maps over support planes

3.3 6DoF object-based haptic rendering

Let us now assume that the haptic interaction point is actually a haptic interaction object that is also modelled using support plane mappings and distance maps.

Referring to Fig. 5 that for simplicity depicts the 2D case, let M_1 and M_2 be the mesh areas of two different objects possibly involved in collision and S_1 and S_2 their respective support planes. Based on the approach presented in [28] it can be decided whether M_1 and M_2 are probably involved in collisions using their SPMs S_1 and S_2. Now the question is: Is it possible to identify the collisions and calculate 6DoF force feedback without entering the computationally intensive narrow-phase [28] of the collision detection algorithm?

This question reduces to the problem of calculating the impact volume (3D case) or impact surface (2D case) S as depicted in Fig. 5. Let us consider for sake of simplicity and without loss of generality the 2D case. If **d** is a material density function then the mass corresponding to the collision area S can be described by the following formula.

$$V = \int\limits_S \int \mathbf{d} dS \tag{12}$$

Now referring again to Fig. 5, let W be a sampling point of the support plane S_2 and W_2 the point of mesh M_2 that can be trivially reconstructed by projecting W along side the normal direction n_2 as far as the distance map $D_{SP}(W)$ dictates. Now, W_2 can be also trivially projected on the support plane S_1 and retrieve the corresponding distance value $D_1(M_1)$ of S_1 from the mesh M_1. Now assume function f is defined as follows:

$$f = D_1(W_2) - D_1(M_1) \tag{13}$$

where $D_1(W_2)$ is the distance of point W_2 from S_1 and $D_1(M_1)$ the corresponding distance of S_1 from mesh M_1. Penetration can be reported only for cases where function f is positive.

Let now C_1 and C_2 (Fig. 5) denote the projection of the colliding volume on the support planes S_1 and S_2 respectively. It is obvious that C_1 is the support set of positive values of

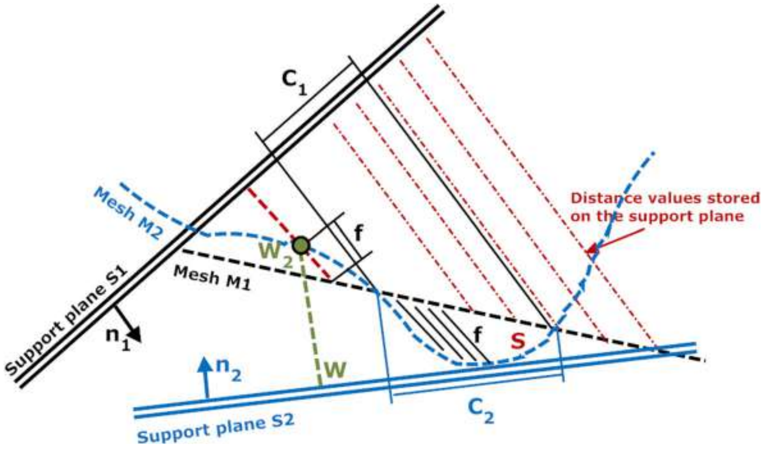

Fig. 5 Distance calculation using distance maps over support planes

function f. Then, (12) can be transformed as follows:

$$V = \int_{t_1 \in C_1} \mathbf{d} \cdot f \, dt_1 = \int_{t_2 \in C_2} \mathbf{d} \cdot f \cdot \cos(\mathbf{n}_1 \mathbf{n}_2) \, dt_2 \tag{14}$$

A similar analysis can be followed for the three dimensions and the corresponding formula is:

$$V = \int \int_{\eta, \omega \in C_2} \mathbf{d} \cdot f \cdot \cos(\mathbf{n}_1 \mathbf{n}_2) d\eta d\omega \tag{15}$$

Now since the volume of the penetrating object part and its centre of inertia is known, 6DoF haptic feedback can be easily estimated. In the context of the proposed framework a single force is estimated for the penetrating part of the haptic interaction object that is applied on the gravity centre of the colliding volume. Other approaches for approximate 6DoF haptic rendering like the one presented in [15] can be also implemented.

4 Haptic effects

The analytical estimation of the force feedback based only on the object 3D transformation, the probe position and the distance maps, provides the opportunity to develop closed form solutions for the rendering of physics-based or symbolic force effects; the following sections indicatively describe some of them.

4.1 Force smoothing

By applying a local smoothing operation to the distance map, the resulting force feedback is smooth in the areas around the edges, without being over-rounded as is the case with the force shading method [22]. A typical example of distance map preprocessing so as to achieve force smoothing using a Gaussian kernel is given by the following equation:

$$D'_{SP}(\eta, \omega) = D_{SP}(\eta, \omega) * G_\sigma(\eta, \omega) \tag{16}$$

where G_σ is a 2D Gaussian kernel and "$*$" denotes convolution. It is evident that different smoothing operators can be easily applied. A very useful operator that can be implemented is the force smoothing only in areas that are not smooth due to the finite tessellation (sampling) and not in object and surface boundaries, following the popular in computer graphics "crease angle" concept. A haptic "crease angle" rendering can be trivially performed by applying anisotropic diffusion or even an edge-preserving smoothing operator on the distance map.

4.2 Friction and damping

The force calculated from (9) is always perpendicular to the object's surface. If no friction component is added, the resulting force feedback will be like touching a very slippery surface. In order to avoid this defect, a friction component is added to the force of (9). In particular:

$$\mathbf{F}_{friction} = -f_C \cdot \left(1 + k_f \left| S_{SP} - D_{SP}(\mathbf{P}_M) \right| \right) \cdot \frac{\mathbf{n}_f}{\|\mathbf{n}_f\|} \tag{17}$$

where f_c is the friction coefficient and $\mathbf{n}_f$ the direction of the motion of the processed point, i.e. $\mathbf{n}_f = \mathbf{P}_t - \mathbf{P}_{t-\Delta t}$, where $\mathbf{P}_t$ is the current position of the processed point and $\mathbf{P}_{t-\Delta t}$ its position at the previous frame. Term $k_f \left| S_{SP} - D_{SP}(\mathbf{P_M}) \right|$ is used in order to increase the magnitude of the friction force when the penetration depth of the processed point increases. The variables S_{SP} and $D_{SP}(\mathbf{P}_M)$ are defined in (3), while factor k_f controls the contribution of the penetration depth to the calculated friction force.

In a similar sense, damping can be considered by including in the force feedback formula the term $F_{damping} = -k_d \cdot \dot{P}_M$.

Finally, the force fed onto the haptic device yields from the addition of the reaction, the friction and the damping force:

$$\mathbf{F}_{haptic} = \mathbf{F}_{reaction} + \mathbf{F}_{friction} + \mathbf{F}_{damping} \tag{18}$$

4.3 Texture

Similarly using the proposed framework for haptic rendering, haptic texture can be also simulated by applying appropriate transformations on the acquired distance map. An example for simulating surface roughness is provided below, where Gaussian noise is added on the distance map. No computational cost is added, since the procedures for calculating the force direction are not altered due to the existence of haptic texture. The only difference lies in the evaluation of the magnitude of $\mathbf{F}_{texture}$, which now yields from:

$$\mathbf{F}_{texture} = k \left| S_{SP} - \left(D_{SP}(\mathbf{P}_M) + n_g \right) \right| \frac{\mathbf{n}_S}{\|\mathbf{n}_S\|} \tag{19}$$

where n_g denotes the gaussian noise.

5 Complexity and experimental results

In the following an analysis of the computational complexity of the proposed scheme in comparison to the typical state-of-the-art mesh-based haptic rendering scheme is discussed.

It should be emphasized that an experimental analysis, in terms of timings for simulation benchmarks, of the proposed support plane mapping based haptic rendering approach would not be fair for the state-of-the-art approaches. This is due to the fact that it would encode the superiority of SPM based collision detection and would not directly highlight the proposed haptic rendering approach and its performance over traditional haptic rendering schemes. However, two experiments are presented where the proposed haptic rendering scheme is compared to the state-of-the-art mesh-based haptic rendering, based on haptic proxy objects, in terms of timings in computationally intensive surface sliding experiments. In these experiments timings related to the cost of performing collision detection are not taken into account.

5.1 Computational complexity

After collision is reported, a typical force feedback calculation scheme would need to identify the colliding triangle of the involved 3D object in $O(n)$ time, where n is the number of triangles, or in $O(logn)$ time if bounding volume hierarchies are used. It should be also mentioned that if more complex data structures are used like the "Doubly Connected Edge List" the aforementioned computational cost becomes even smaller. Such structures are not however common in computer graphics, where the majority of the applications use as a representation structure the "indexed face set". Then the force can be calculated in constant $O(1)$ time. In order to avoid force discontinuities, for example force shading, and if there is no adjacency information then the local neighbourhood of the colliding triangle can be

Table 1 Computational complexity comparison

Process	Mesh-based	Free-form
Force	$O(n)$ or $O(logn)$	$O(1)$
Smoothing	$O(n)$ or $O(logn)$	$O(1)$
Memory	–	$O(m \cdot s)$

found again in $O(n)$ time, where n is the number of triangles, or in $O(logn)$ time if bounding volume hierarchies are used. Finally, the mesh-based haptic rendering scheme has no additional memory requirements per se.

On the other hand, concerning the proposed free-form implicit haptic rendering scheme, after collision is detected, the support plane of the object that is attached to the haptic interaction point (HIP) is provided and the resulting force feedback can be calculated in constant time $O(1)$ using (9). In order to avoid depth discontinuities the distance map can be smoothed, in an image processing sense, at a preprocessing phase. Even if this step is performed during run-time it would take $O(k)$ time, where k is the local smoothing region or the filtering kernel window. On the other hand the proposed scheme has $O(m \cdot s)$ memory requirements, where m is the number of support planes and s the number of samples per support plane. Taking now into account that the more support planes are used the smaller their size and the less samples are necessary for a specific sampling density we can safely assume that the memory requirements are linear to the total number of samples that depends on the sampling density used.

Table 1 summarizes the computational complexity analysis of the proposed free-form haptic rendering scheme, when compared to the mesh-based approach.

5.2 Experimental results

Concerning the quantitative results, interaction with two objects was considered, namely the Galeon and Teeth models of 4698 and 23000 triangles respectively. The objects are illustrated in Figs. 6 and 7 respectively. Moreover, CHAI-3D [6] was used for interfacing with the haptic devices Novint Falcon and Phantom Omni.

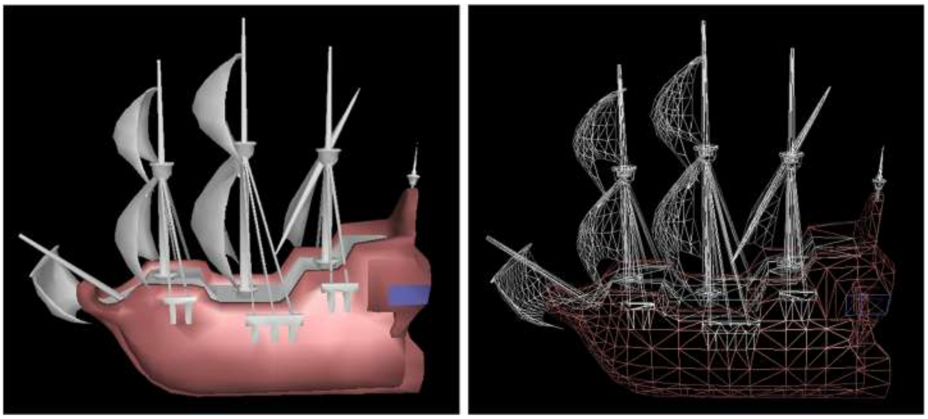

Fig. 6 Galeon model, 4698 triangles

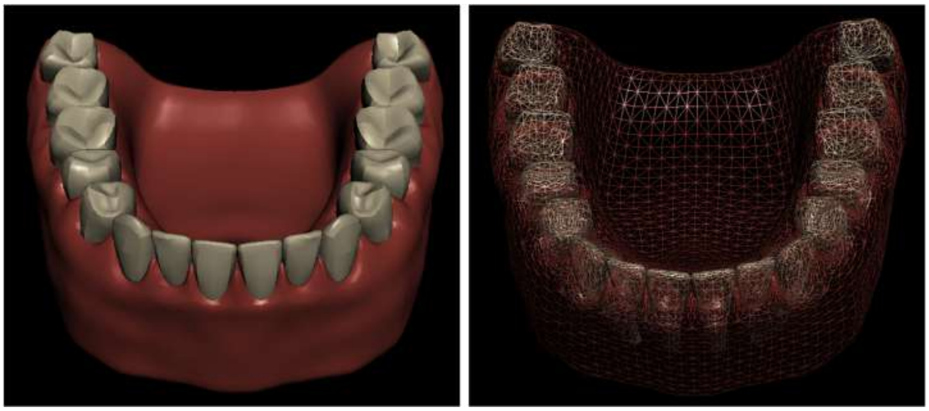

Fig. 7 Teeth model, 23000 triangles

Moreover, the force estimation algorithms were applied on a predefined trajectory of the haptic probe, so as to assure fair comparison. In particular, the trajectory of the haptic probe in the 3D space has been initially recorded while being in sliding motion over the objects' surface. Then this trajectory has been used as input for both algorithms so as to extract the timings mentioned below.

Tables 2 and 3 present the mean timings and their standard deviation of the force estimation throughout the simulation using the mesh-based,and the proposed free-form haptic rendering scheme for the case of the Galeon and the Teeth models respectively.

It should be emphasized that the above timings need to be taken into account under the exact experimental setting. In particular, concerning the proposed approach 1000 support planes were used for the case of the Galeon and 1500 for the case of the Teeth model. Distances are estimated for all support planes and forces are calculated for the closer one. This procedure, could be optimized by partitioning the space in a preprocessing step and knowing beforehand to which support plane, each point in space "belongs to", thus reducing the search from $O(n)$ to $O(logn)$. Moreover, concerning the mesh-based approach force shading has been also used.

It is evident that the proposed scheme reduces significantly the computational cost in the performed simulations. This significant gain comes at an expense of two limitations. Firstly, special care has to be taken at the preprocessing step so that the models are well approximated using the support planes and the distance maps. For example if the objects demonstrate large complex concavities the use of vectorial distance maps is inevitable. Secondly, the proposed scheme cannot be, in its current form, directly applied to deformable models. An extension to piecewise or free-form deformable models, where deformations can be analytically expressed seems possible and remains a direction for future work.

Table 2 Galeon model: Interaction timings

Process	Mean time (ms)	σ
Mesh-based	1.2	0.22
Free-form	0.028	0.006

Table 3 Teeth model: Interaction timings

Process	Mean time (ms)	σ
Mesh-based	4.2	0.82
Free-form	0.036	0.005

6 Conclusions

The proposed approach introduces an implicit free-form haptic rendering scheme of rigid bodies based on distance maps over support plane mappings and therefore exploits the superiority and bounding efficiency of SPMs for collision detection and extends it for direct closed-form haptic rendering. Moreover, the derivation of analytical expressions of widely used haptic effects becomes straightforward. The proposed approach is seen to be highly efficient when compared to the state-of-the-art mesh-based haptic rendering at a cost, however, of increased memory requirements.

References

1. Barbic J, James D (2009) Six-dof haptic rendering of contact between geometrically complex reduced deformable models: haptic demo. In: Proceedings of Eurohaptics, pp 393–394
2. Barlit A, Harders M (2007) Gpu-based distance map calculation for vector field haptic rendering. In: Eurohaptics, pp 589–590
3. Burdea G, Coiffet P (2003) Virtual reality technology. 2nd edn. Wiley-IEEE Press
4. Chung K, Wang W (1996) Quick collision detection of polytopes in virtual environments. In: ACM symposium on virtual reality software and technology 1996, pp 1–4
5. Coming D, Staadt O (2008) Velocity-aligned discrete oriented polytopes for dynamic collision detection. IEEE TVCG 14(1):1–12
6. Conti F, Barbagli K, Morris D, Sewell C (2005) Chai 3d: an open-source library for the rapid development of haptic scenes. In: IEEE World Haptics, Pisa
7. Dobkin DP, Kirkpatrick DG (1985) A linear algorithm for determining the separation of convex polyhedra. J Algorithm 6(3):381–392
8. Ericson C (2005) Real-Time collision detection. The Morgan Kaufmann series in interactive 3D technology. Morgan Kaufmann, San Mateo
9. Fuhrmann A, Sobottka G, Gross C (2003) Distance fields for rapid collision detection in physically based modeling. In: Proceedings of GraphiCon 2003, pp 58–65
10. Gottschalk S, Lin M, Manocha D (1996) OBBTree: a hierarchical structure for rapid interference detection. In: Computer graphics, ACM SIGGRAPH, pp 171–180
11. Hubbard P (1996) Approximating polyhedra with spheres for time-critical collision detection. ACM Trans Graph 15(3):179–210
12. Klosowski J, Held M, Mitchell J, Sowizral H, Zikan K (1998) Efficient collision detection using bounding volume hierarchies of k-DOPs. IEEE Trans Vis Comput Graph 4(1):21–36
13. Laycock S, Day A (2007) A survey of haptic rendering techniques. Comput Graph Forum 26(1):50–65
14. Lin M, Otaduy M (2008) Haptic rendering: foundations, algorithms and applications. p. A.K.Peters Publishing
15. McNeely W, Puterbaugh K, Troy J (1999) Six degree-of-freedom haptic rendering using voxel sampling. In: Proc. of ACM Siggraph, pp 401–408
16. Moustakas K, Nikolakis G, Kostopoulos K, Tzovaras D, Strintzis M (2007) Haptic rendering of visual data for the visually impaired. IEEE Multimedia 14(1):62–72
17. Moustakas K, Tzovaras D, Strintzis M (2007) Sq-map: efficient layered collision detection and haptic rendering. IEEE Trans Vis Comput Graph 13(1):80–93
18. Osher S, Fedkiw R (2002) Level set methods and dynamic implicit surfaces. Springer, Berlin Heidelberg New York
19. Osher S, Sethian JA (1988) Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. J Comput Phys 79(1):12–49

20. Palmerius K, Cooper M, Ynnerman A (2008) Haptic rendering of dynamic volumetric data. IEEE Trans Vis Comput Graph 14(2):263–276
21. Petersik A, Pflesser B, Tiede U, Hohne K (2001) Haptic rendering of volumetric anatomic models at sub-voxel resolution. In: In Proceedings of Eurohaptics, Birmingham, pp 182–184
22. Ruspini D, Kolarov K, Khatib O (1997) The haptic display of complex graphical environments. In: Computer graphics (SIGGRAPH 97 conference proceedings), pp 345–352
23. Sethian J, Ciarlet P, Iserles A, Kohn R, Wright M (1999) Level set methods and fast marching methods : evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science. Cambridge University Press, Cambridge
24. Srinivasan M, Basdogan C (1997) Haptics in virtual environments: taxonomy, research status and challenges. Computers and graphics, pp 393–404
25. Teschner M, Kimmerle S, Heidelberger B, Zachmann G, Raghupathi L, Fuhrmann A, Cani MP, Faure F, Magnenat-Thalmann N, Strasser W, Volino P (2004) Collision detection for deformable objects. In: Proc. of Eurographics 2004, pp 119–135
26. van den Bergen G (1997) Efficient collision detection of complex deformable models using AABB trees. J Graph Tools 2(4):1–13
27. van den Bergen G (2003) Collision detection in interactive 3D environments. The Morgan Kaufmann series in interactive 3D technology. Morgan Kaufmann, San Mateo
28. Vogiannou A, Moustakas K, Tzovaras D, Strintzis M (2010) Enhancing bounding volumes using support plane mappings for collision detection. Comput Graph Forum 29(5):1595–1604

Konstantinos Moustakas received the Diploma degree and the PhD in electrical and computer engineering from the Aristotle University of Thessaloniki, Greece, in 2003 and 2007 respectively.

He has been a visiting lecturer in the same department during 2008. During 2003-2007 he served as a research associate in the Information Processing Laboratory of the Aristotle University of Thessaloniki and the Informatics and Telematics Institute, Centre for Research and Technology Hellas, where he has also served as a post-doctoral research fellow in the period 2007-2011. Since January 2012 he is an Assistant Professor in the Electrical and Computer Engineering Department of the University of Patras and Head of the Visualization and Virtual Reality Group. His main research interests include virtual augmented and mixed reality, haptics, virtual human modeling, information visualization, physics-based simulations, computational geometry, 3D content-based search, computer vision, and stereoscopic image processing. During the latest years, he has been the (co)author of more than 100 papers in refereed journals, edited books, and international conferences. He serves as a regular reviewer for several technical journals and has participated to more than 12 research and development projects funded by the EC and the Greek Secretariat of Research and Technology. He is a member of the IEEE, the IEEE Computer Society and Eurographics.