SQ-Map: Efficient Layered Collision Detection and Haptic Rendering

Konstantinos Moustakas, Dimitrios Tzovaras, and Michael Gerassimos Strintzis, *Fellow*, *IEEE*

Abstract—This paper presents a novel layered and fast framework for real-time collision detection and haptic interaction in virtual environments based on superquadric virtual object modeling. An efficient algorithm is initially proposed for decomposing the complex objects into subobjects suitable for superquadric modeling, based on visual salience and curvature constraints. The distance between the superquadrics and the mesh is then projected onto the superquadric surface, thus generating a distance map (SQ-Map). Approximate collision detection is then performed by computing the analytical equations and distance maps instead of triangle per triangle intersection tests. Collision response is then calculated directly from the superquadric models and realistic smooth force feedback is obtained using analytical formulae and local smoothing on the distance map. Experimental evaluation demonstrates that *SQ-Map* reduces significantly the computational cost when compared to accurate collision detection methods and does not require the huge amounts of memory demanded by distance field-based methods. Finally, force feedback is calculated directly from the distance map and the superquadric formulae.

Index Terms—Collision detection, haptic rendering, force feedback, collision response, superquadrics.

1 Introduction

Human perception combines information of various sensors, including visual, aural, haptic, olfactory, etc., 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 [1].

However, haptic interaction involves complex and computationally intensive processes, like collision detection [2], [3], [4] or distance calculation [5], in order to provide realistic and feasible results. A significant amount of processing is required for the accurate detection of collisions during the simulations. Handling collisions, i.e., object intersections in a scene, is an essential process in realistic simulations. Most approaches presented in the past are based on building a Bounding Volume Hierarchy (BVH) around the object consisting of primitive objects like spheres [2], OBBs [3] or volumes based on complex dynamically transforming geometries k-DOPs [4]. 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 [3], [6]. 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.

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 [7], [8], [9], 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 [10] and fast marching algorithms [11]. Despite their efficiency, these methods are not used extensively for collision detection. The major reason is their huge memory requirements. Adaptively sampled distance fields [12] and methods that trade speed for memory [13] have been presented in the past in order to decrease the cost in memory. However, the memory requirements remain extremely high, especially when the virtual scene consists of many complex objects.

In this paper, a novel layered framework for collision detection is presented, which reduces dramatically its

- *K. Moustakas and M.G. Strintzis are with the Electrical and Computer Engineering Department—Aristotle University of Thessaloniki, University Campus, 54124, Thessaloniki, Greece and the Centre for Research and Technology Hellas—Informatics and Telematics Institute, 1st km Thermi-Panorama Road, 57001, PO Box 361, Thermi—Thessaloniki, Greece. E-mail: moustak@olympus.ee.auth.gr, strintzi@eng.auth.gr.*
- *D. Tzovaras is with the Centre for Research and Technology Hellas—Informatics and Telematics Institute, 1st km Thermi-Panorama Road, 57001, P.O. Box 361, Thermi—Thessaloniki, Greece. E-mail: tzovaras@iti.gr.*

Manuscript received 4 July 2005; revised 8 Feb. 2006; accepted 20 Mar. 2006; published online 8 Nov. 2006.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org, and reference IEEECS Log Number TVCG-0082-0705.

computational cost without any significant loss of accuracy or requirement of high memory use. It can handle efficiently collisions between two rigid and between a rigid and a deformable object. The virtual objects are initially decomposed to simpler ones. Each subobject is then modeled using a superquadric (SQ) that bounds its geometry and collision detection is performed on the subobjects' level. The union of superquadrics generated by the proposed algorithm is a nonconvex surface that bounds entirely the geometry of the modeled object.

The most important feature of *SQ-Map*, when compared to the distance field-based methods, is that its memory requirements are significantly lower. As will be described in the sequel, the proposed method reduces the need for the calculation of the distance field only onto the 2D surface of a superquadric. Thus, only a 2D distance map is generated instead of the 3D distance grid required by methods based on distance fields.

Finally, a robust method is presented for calculating the force feedback for haptic interaction, which generates smooth and realistic force fields and is based on the analytical formula of the superquadric and the estimated distance map. The proposed methods were tested in a nonvisual object recognition application for blind people training and a costume designer application using the PHANToM and the CyberGrasp haptic devices.

The paper is organized as follows: In Section 2 the fundamentals of superquadric modeling are presented. Section 3 presents the object decomposition method. Section 4 describes the backbone of *SQ-Map*, which is the collision detection algorithm based on the superquadric modeling of the 3D objects, while Section 5 analyzes the accuracy of the scheme. In Section 6, the proposed haptic rendering approach is presented. Finally, experimental results are exhibited in Section 7 and conclusions are drawn in Section 8.

2 Implicit Surface Approximation

Superquadrics have been used in the past to model objects using as input, range images, and depth maps [14]. In general, superquadrics is a family of analytical surfaces consisting of superellipsoids, superparaboloids, superhyperboloids, supertoroids, etc. In use for the modeling of 3D objects, which is our concern, surfaces like superellipsoids, which are defined by the implicit (1), are of practical interest:

$$F(x,y,z)=\left(\left(\left(\frac{x}{a_1}\right)^{\frac{2}{\varepsilon 2}}+\left(\frac{y}{a_2}\right)^{\frac{2}{\varepsilon 2}}\right)^{\frac{\varepsilon 2}{\varepsilon 1}}+\left(\frac{z}{a_3}\right)^{\frac{2}{\varepsilon 1}}\right)^{\varepsilon 1}=1. \quad (1)$$

Function (1) is commonly called inside-outside function, because for a 3D point (x,y,z):

If $F(x,y,z)>1$, (x,y,z) lies outside the surface.
If $F(x,y,z)\leq 1$, (x,y,z) lies inside or on the surface.

Deformation parameters, which correspond to tapering, bending, etc. [14], can be added to the implicit equation so as to produce a more flexible model.

After the selection of the appropriate superquadric equation to model the 3D data, the problem of modeling the 3D object using a superquadric reduces to the least squares minimization of the nonlinear inside-outside function $F(x,y,z)$ with respect to several shape parameters. In particular,

$$\begin{aligned}&F(x,y,z)=\\&F(x,y,z;a_1,a_2,a_3,\varepsilon_1,\varepsilon_2,\phi,\theta,\chi,t_x,t_y,t_z,K_x,K_y,k,a),\end{aligned} \quad (2)$$

where (x,y,z) is a point in the 3D space, a_1, a_2, a_3, ε_1, and ε_2 are the superquadric shape parameters, ϕ, θ, and χ, and t_x, t_y, and t_z are the Euler angles and translation vector coefficients respectively, K_x and K_y are tapering deformation parameters and k and a are the bending deformation parameters. The above parameters are determined so as to minimize the following mean square error.

$$MSE=\sum_{i=1}^{N}\sqrt{a_1a_2a_3}(F(x_i,y_i,z_i)-1)^2, \quad (3)$$

where N is the number of points of the 3D object.

The well-known [14] Levenberg Marquardt method for nonlinear least squares minimization is used in the present paper in order to evaluate the shape parameters. Notice that for complex objects, proper division into rigid subobjects is necessary for efficient superquadric approximation. Moreover, in the context of *SQ-Map* the superquadrics are modeled so as to bound the entire geometry of the modeled subobject. Therefore, constrained minimization is applied so as to produce bounding superquadrics.

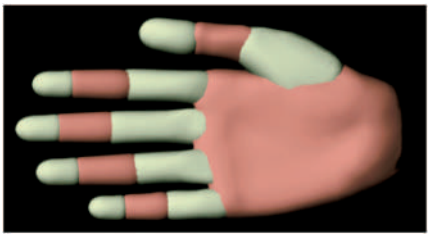


Fig. 1. Segmented virtual hand.

As an example, Fig. 1 illustrates the segmentation of the virtual hand into its rigid components. Assuming that SQ_i represents the superquadric approximation of the ith element of the virtual hand, the superquadric representation of the whole virtual hand (*VH*) can be mathematically described as the union of all superquadrics, i.e., $VH=\bigcup_{i=1}^{16}SQ_i$. Technically, regions that lie inside the SQ segment "A" but correspond to SQ segment "B" are pruned away from SQ segment "A."

3 Object Decomposition

The superquadric approximation cannot produce accurate results if the object to be modeled is very complex. Thus, the virtual objects have to be decomposed into simpler subobjects, which can be accurately described using superquadrics. An automated algorithm for decomposing complex objects into quasiconvex subobjects and modeling each part using superquadrics is introduced in the sequel. In order to decompose an object, an algorithm is proposed based on salient feature extraction and curvature estimation. The method utilizes topological information and, thus,

makes use of the polygonal representation of the object and not only of its vertices. Each of these processes is analyzed in the following subsections.

3.1 Salient Feature Extraction

The developed method for salient feature extraction is based on Hoffman and Singh's theory of salience [15]. In order to make this paper self-contained a brief description of the method follows: The method uses salient features and geodesic distances to define a measure for decomposition in the areas where protruding parts meet the rest of the body of an object. Initially, the dual graph $G = (V, E)$ of the given mesh is generated [16], where V and E are the dual vertices and edges. A dual vertex is the center of mass of a triangle and a dual edge links two adjacent triangles. The degree of protrusion for each dual vertex results from the following equation:

$$p(\mathbf{u}) = \sum_{i=1}^{N} g(\mathbf{u}, \mathbf{v}_i) \cdot area(\mathbf{v}_i), \tag{4}$$

where N is the number of dual vertices in the entire surface, $p(\mathbf{u})$ is the protrusion degree for the dual vertex $\mathbf{u}$, $g(\mathbf{u}, \mathbf{v}_i)$ is the geodesic distance of $\mathbf{u}$ from dual vertex $\mathbf{v}_i$ and $area(\mathbf{v}_i)$ is the area of the triangle $\mathbf{v}_i$.

Using simple gradient-based methods (i.e., steepest descent) all local maxima of the protrusion map $p(u)$ are obtained. In order to avoid fragmentation, geodesic windows are applied and only the global maxima inside the window are considered as salient. A geodesic window, GW, centered at the dual vertex u is defined as follows:

$$GW_{\mathbf{u}} = \{\mathbf{v} \mid \forall \mathbf{v} \in V, g(\mathbf{u}, \mathbf{v}) < \varepsilon\}, \tag{5}$$

where ϵ defines the window size.

In [16], in order to perform segmentation, parts L_x named "locales" are used, which correspond to equidistant areas from the salient feature, in terms of geodesic distance. The difference of the areas of consecutive locales is defined as "*Boundary Strength*." The object is "cut" at the intersection of the locales that exhibit a significant increase of the value of "*Boundary Strength*," which is the case, e.g., for the area where the tail of an animal reaches its body.

3.2 Curvature Estimation and Superquadric Modeling

The approach described in Section 3.1 seems to produce good results for a large variety of cases. However, it does not consider curvature information. Thus, it would not split a torus, where curvature is high but the predescribed criteria are not met. For 3D segmentation this is not a problem, but for the case of superquadric modeling this is not acceptable since the subobjects to be modelled have to be approximately convex. The present framework extends the approach of Section 3.1 [16] and proposes a novel method which incorporates curvature information into the segmentation criteria, thus being able to decompose torus-like objects.

Curvature estimation from 3D triangular meshes has been extensively addressed in the past [17], [18]. In the present framework, the Gaussian curvature is used because it can directly provide information on whether the local surface, for which the curvature is defined, is synclastic or anticlastic [19], i.e., whether the point is elliptic or hyperbolic. At this point it should be emphasized that Gaussian curvature is only used as supplementary information to clarify the above point and, thus, to overcome a specific weakness of the method presented in Section 3.1.

The Gaussian curvature for a 3D point $\mathbf{p}$ is found by:

$$k(\mathbf{p}) = \det(S(\mathbf{p})), \tag{6}$$

where $S(\mathbf{p})$ is the second fundamental tensor [19] and represents the negative derivative of the unit normal vector $\mathbf{N}$ of the surface at point $\mathbf{p}$:

$$S(\mathbf{x}) = -\frac{\partial \mathbf{N}}{\partial \mathbf{x}}. \tag{7}$$

The method described in [18] is adopted for the evaluation of the Gaussian curvature (6), because it does not demand surface fitting operations or the estimation of local spatial derivatives. The local area of a dual vertex $\mathbf{v}$ is not defined to consist just of its adjacent faces, but of a larger area consisting of all dual vertices inside the geodesic window (5) with size $\epsilon = 2\sqrt{area(\mathbf{v})}$, in order to make the estimation robust to noise and to small local variations of curvature. After calculating the Gaussian curvature for all dual vertices of the mesh, the set L_x^{hyp} is defined, which consists of the N_x^{hyp} hyperbolic points of L_x, for each locale.

$$L_x^{hyp} = \{\mathbf{v} | \mathbf{v} \in L_x, k(\mathbf{v}) < 0\}.$$

The criterion for decomposition, which is used as a complement to the one described in Section 3.1, consists of the following two conditions:

- The percentage of the hyperbolic dual vertices of a locale should be over k_h, i.e.,

$$\sum_{\mathbf{v} \in L_x^{hyp}} area\,(\mathbf{v}) > k_h \cdot \sum_{\mathbf{v} \in L_x} area\,(\mathbf{v}).$$

- The absolute mean Gaussian curvature of the hyperbolic points should be at least t_h, i.e.,

$$\left| \frac{1}{N_x^{hyp}} \cdot \sum_{\mathbf{v} \in L_x^{hyp}} k(\mathbf{v}) \right| > t_h,$$

where k_h and t_h are experimentally estimated thresholds. Specifically, k_h is a qualitative threshold that defines the size of the area of the locale that should exhibit hyperbolic geometry so as to decompose the object at the specific locale. On the contrary, t_h is a quantitative threshold on the degree in which the hyperbolic points exhibit hyperbolic geometry. k_h corresponds to the percentage of hyperbolic points at the locale L and t_h to the mean value of the Gaussian curvature of the hyperbolic points of L. If for a specific locale L_x the percentage of hyperbolic points and their mean Gaussian curvature are higher than k_h and t_h, respectively, the object is cut at L_x.

It should also be mentioned that, in order to avoid fragmentation, a minimum size of a subobject is enforced.

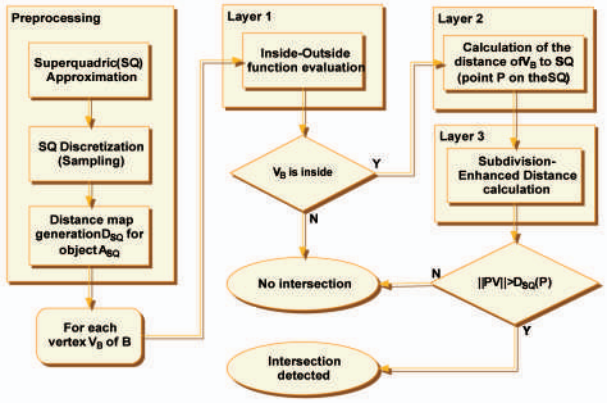


Fig. 2. Superquadric-based collision detection—Block diagram.

Summarizing, the major steps of the object decomposition and superquadric modeling procedure are:

1. Extract salient features, locale surfaces and calculate the Boundary_Strength for each locale as described in Section 3.1.
2. Estimate the Gaussian curvature for each dual vertex (6).
3. Perform Gaussian curvature and "*Boundary Strength*" tests for decomposition incrementally for each locale. If at least one of them is positive perform decomposition according to the specific locale.
4. Model each subobject using a superquadric, so as to bound its geometry.

An example of the decomposition using curvature information can be observed at the elephants trunk in Fig. 10a. In that case the use of "*Boundary Strength*" information only does not lead to decomposition.

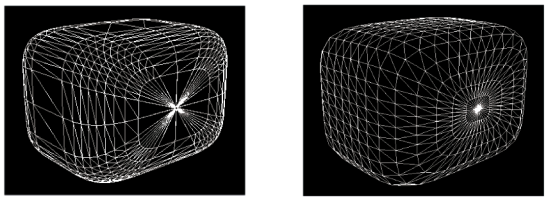
4 Collision Detection Framework

A flow chart of the proposed layered collision detection method, which is described in the sequel, is illustrated in Fig. 2.

In the following terms $\mathbf{A}_{SQ}$ and $\mathbf{B}$ will used to refer to the object that is modeled using superquadrics and a second object possibly colliding with $\mathbf{A}_{SQ}$, respectively.

During the preprocessing phase each part of the decomposed object $\mathbf{A}_{SQ}$ is modeled using a superquadric as described in Section 2. Next, the resulting superquadric is uniformly sampled. The distance of each sample $\mathbf{P}$ from the modeled object part alongside the normal direction of the superquadric surface at point $\mathbf{P}$ is calculated and stored in an array (i.e., distance map).

The proposed superquadric collision detection procedure performs tests between the points of an object $\mathbf{B}$ and the superquadrics that comprise another object $\mathbf{A}_{SQ}$. In order to refrain from these tests, when the objects are not close enough, bounding spheres are defined for each object, as well as for each superquadric. The smaller the number of SQ surfaces an object is decomposed to, the faster the algorithm. In cases of fragmentation, which, however, are rare in practice, the algorithm can be easily combined with an hierarchical procedure for exploiting spatial coherency and minimizing the redundant operations. In the context of this work no more than two levels of bounding sphere hierarchies need to be used to provide satisfactory results.



(a) (b)

Fig. 3. Superquadric rendering using a parameterization of (a) equal spaced intervals and (b) nonequally spaced (transformed) intervals.

During runtime, for each vertex $\mathbf{V}_B$ of $\mathbf{B}$, the following procedures are carried out. Initially, in the first layer, the superquadric implicit equation modeling $\mathbf{A}_{SQ}$ is evaluated. If $\mathbf{V}_B$ lies inside the surface the algorithm proceeds to Layer 2. At this step, the distance of $\mathbf{V}_B$ to the SQ surface is calculated. This distance corresponds to a point $\mathbf{P}_V$ on the superquadric surface and is compared with the value of the distance map at point $\mathbf{P}_V$. If $\mathbf{V}_B$ lies deeper inside the superquadric than allowed, i.e., the distance of $\mathbf{V}_B$ to the SQ surface is larger than the value of the distance map at point $\mathbf{P}_V$, collision is detected. Layer 3 is an enhanced version of Layer 2. In particular, it maps additional information to the distance map, corresponding to the distance of each vertex of $\mathbf{A}_{SQ}$ to the superquadric, in order to avoid possible errors caused by local concavities, etc. Moreover, it utilizes a multiresolutional procedure to refine the distance map at the desired level.

Notice that the collision detection procedure can be restricted to the preferable layer according to the quality of the superquadric approximation and the desired accuracy. In the following, the complete preprocessing and the runtime algorithm are described in detail.

4.1 Preprocessing

4.1.1 Step 1

The first step of the preprocessing phase is to decompose the object, to model each segment using superquadrics as described in Sections 2 and 3 and to define the bounding spheres for each object and subobject.

4.1.2 Step 2

In the second step, a dense mesh T_{SQ} is generated for the superquadric using its parametric equations.

$$\mathbf{r}_{SQ}(\eta,\omega) = \begin{bmatrix} a_1 \cos^{\varepsilon_1}(\eta)\cos^{\varepsilon_2}(\omega) \\ a_2 \cos^{\varepsilon_1}(\eta)\sin^{\varepsilon_2}(\omega) \\ a_3 \sin^{\varepsilon_1}(\eta) \end{bmatrix}, \quad -\frac{\pi}{2} \le \eta \le \frac{\pi}{2} - \pi \le \omega < \pi, \tag{8}$$

where η and ω are uniformly distributed in the specified intervals. More precisely, η_d points are extracted for the η-*space* and ω_d for the ω-*space*.

When discretizing a superquadric using a uniformly distributed parameterization, the generated mesh consists of points that are not uniformly distributed in the superquadric surface, as illustrated in Fig. 3a. This may result to the absence of distance information for large surface

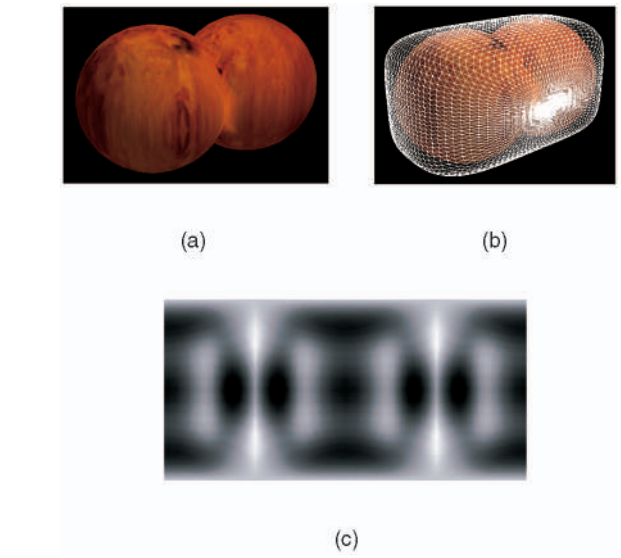


Fig. 4. (a) An object composed of two spheres, (b) its superquadric approximation, and (c) the resulting distance map, $\eta_d = 100, \omega_d = 200$. White corresponds to larger distance values.

elements of the superquadric. Therefore, a parametric transformation [20] is used, in order to generate a mesh with almost equally spaced point samples by projecting the unit superquadric onto a unit sphere and projecting the nodes on the superquadric to the nodes of the sphere. The new parameterization results from the following equation:

$$\begin{bmatrix} \eta' \\ \omega' \end{bmatrix} = \begin{bmatrix} \tan^{-1}\left(\mathrm{sgn}(\tan(\eta))|\tan(\eta)|^{\frac{1}{\varepsilon_1}}\right) \\ \tan^{-1}\left(\mathrm{sgn}(\tan(\omega))|\tan(\omega)|^{\frac{1}{\varepsilon_2}}\right) \end{bmatrix}, \quad \forall \eta, \omega. \tag{9}$$

Notice that the above transform projects the input variables to the interval $-\pi/2 \leq \eta \leq \pi/2$. This is acceptable for η, but not for ω, which, by its definition, should be distributed in the interval $-\pi \leq \omega < \pi$. To overcome this problem, the value of the initial ω is checked and the value of π is added or subtracted from ω', when necessary.

$$\omega'_{new} = \begin{cases} \omega' + \pi, & if\ \omega > \pi/2 \\ \omega' - \pi, & if\ \omega < -\pi/2 \\ \omega', & else. \end{cases} \tag{10}$$

Using the above parameterization, the generated mesh T_{SQ} consists of point samples almost equally spaced along the surface of the superquadric as illustrated in Fig. 3b.

After defining the superquadric mesh, the distance of each point sample from the original mesh is calculated. The discrete distance map $D_{SQ}(\eta, \omega)$ is computed using (11)

$$D_{SQ}(\eta, \omega) = ICD(SQ, S_{mesh}), \tag{11}$$

where ICD calculates the distance of every point sample (η, ω) of the superquadric SQ, alongside the normal direction at point (η, ω), from the mesh S_{mesh} and assigns the corresponding values to the distance map $D_{SQ}(\eta, \omega)$. The distance map is used in the sequel to allow all vertices $\mathbf{V}_B$ of object $\mathbf{B}$ to move freely inside the superquadric only within a specific distance from its surface, defined from $D_{SQ}(\eta, \omega)$.

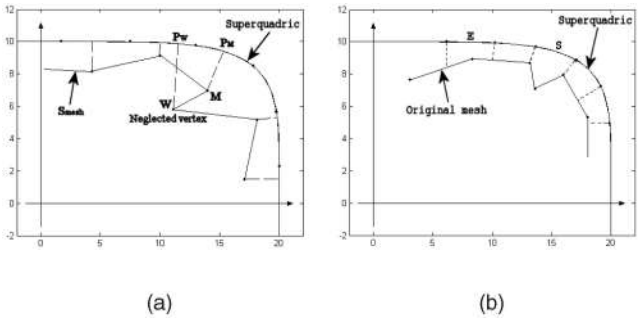


Fig. 5. Superellipses (a) Layer 3: Distance mapping of the mesh vertices and (b) representation of S_{mesh} using Layer 2.

To avoid intersection between a triangle of $\mathbf{B}$ and $\mathbf{A}_{SQ}$, which can happen if the face of the triangle collides with the mesh of $\mathbf{A}_{SQ}$, but its vertices are still outside it, a small offsetting value δ is subtracted from the distance map. The value of δ is chosen adaptively depending on local geometric features. For the choice of δ only the distance map on the superquadric is used. In particular, the value of δ is chosen to be nonzero only for the superquadric samples $\mathbf{x} \in Z_{min}$, where Z_{min} includes the areas on the superquadric where D_{SQ} exhibits local minima, since these are areas that may protrude inside the face of a triangle.

$$\delta(\mathbf{x}) = \begin{cases} a_N \cdot d_\omega \cdot \|\nabla D_{SQ}(\mathbf{x})\|, & if\ \mathbf{x} \in Z_{min} \\ 0 & otherwise, \end{cases} \tag{12}$$

where d_ω is the mean distance between consecutive samples on the superquadric and a_N is a normalization constant experimentally selected to be "1.5".

Using (12), higher values of δ are set in the areas with high positive gaussian curvature. A simple case of an object, the obtained superquadric and the calculated distance map, is illustrated in Fig. 4.

4.1.3 Step 3

In the final step, a distance value is assigned to every superquadric point $\mathbf{P}_M$ that corresponds to the projection of each vertex $\mathbf{M}$ of $\mathbf{A}_{SQ}$ to the superquadric surface, as illustrated in Fig. 5a. If the line connecting the vertex $\mathbf{W}$ of the original mesh S_{mesh} and the minimum distance point $\mathbf{P}_W$ intersects with S_{mesh}, the vertex is neglected and no distance value is assigned to the corresponding point $\mathbf{P}_W$ (Fig. 5a).

If the vertices of $\mathbf{A}_{SQ}$ are not projected onto the SQ surface, as illustrated in Fig. 5b, the SQ approximation with the distance map can be either too loose, the case of arc E, or too strict, the case of arc S. This means that an element of an object that is possibly involved in the collision might penetrate the original mesh in the area of arc E, while collision would be reported even before it enters the original mesh in the area of arc S. The aforementioned issues are the main reason for introducing Layer 3 that handles them properly and is presented in the sequel.

4.2 Runtime Layered Algorithm

The runtime algorithm proceeds as follows:

1. Test for overlapping between the bounding spheres. If positive, proceed to:

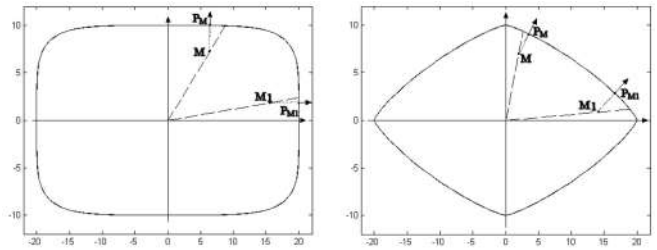


Fig. 6. Superellipses (a) ε = 0.4 and (b) ε = 1.6.

2. For each overlapping pair execute Layer 1 superquadric collision detection. If positive, proceed to:
3. For each overlapping pair execute Layer 2 superquadric collision detection.
4. If $\eta_d\omega_d < k_L \cdot N_m$, proceed to Layer 3 superquadric collision detection. N_m is the number of vertices of the modeled subobject, η_d, ω_d correspond to the superquadric-sampling density values and k_L is an experimentally selected threshold that defines how dense the superquadric sampling has to be so as to proceed to Layer 3. The value of k_L has experimentally been selected to be "10."

The novel three-layered approach for fast resolution of collision detection queries is presented in the sequel. Each layer represents a different collision detection scheme that is based on the previous and overcomes specific weaknesses of them. Furthermore, the calculations in each layer are used in the next layer so as to produce a more accurate result. This incremental nature of the proposed method allows the user to dynamically select the layers that should be used, according to criteria concerning, e.g., real time execution.

4.2.1 Layer 1

The first layer consists of a simple check of the superquadric inside-outside function as described in Section 2. If the result is positive, the algorithm proceeds to Layer 2 or intersection is detected if only Layer 1 is used.

The accuracy of the one-layer collision detection depends entirely on the accuracy of the superquadric approximation. If the 3D mesh was perfectly modeled using a superquadric, collision detection based on this simple check would be perfectly accurate. However, such perfect modeling is rare in practical applications and, therefore, the following Layers are used so as to obtain more accurate results.

4.2.2 Layer 2

In the context of the second Layer, the following procedure is executed at every time update of the simulation. It should be mentioned that, in order to avoid collisions, each vertex $\mathbf{V}_B$ of object $\mathbf{B}$, has to be left to move freely without entering inside $\mathbf{A}_{SQ}$ at each time instance, even if it lies inside the superquadric modeling it. When checking for collision, not only the distance of $\mathbf{V}_B$ from the superquadric at the present frame has to be evaluated, but also the point of the superquadric, which corresponds to the minimum distance. An analytical estimation of the coordinates of this point is possible, but requires the solution of a (4×4) nonlinear system of equations for each vertex [14]. This

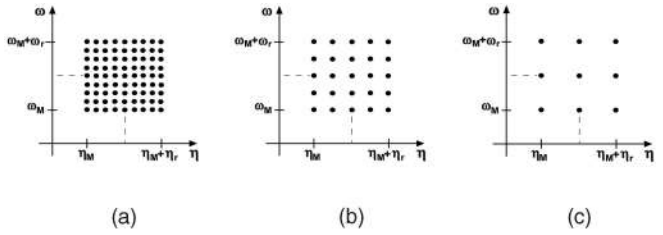


Fig. 7. Search pyramid for $N_L = 3$, $\eta_r, \omega_r > 0$. (a) Level 0. (b) Level 1. (c) Level 2.

procedure is computationally complicated and unacceptable for real time applications. Another approach could be the minimization of the distance equation using a multidimensional Newton iteration, which is, however, seen to suffer from instabilities.

In the context of *SQ-Map*, a fast multiresolution search method is developed to find this minimum distance point. The main aspects of this method are described in the following for the 2D case of a superellipse, without loss of generality, in order to present more effectively important features in the figures.

Consider the superellipse of Fig. 6a. Point $\mathbf{M}$ lies inside the curve and its minimum distance from the superellipse, as well as the minimum distance point $\mathbf{P}_M$ have to be evaluated. As illustrated in Fig. 6a, for the minimum distance point $\mathbf{P}_M$, the normal vector to the surface passes through point $\mathbf{M}$. The equation of the superellipse is:

$$\left(\frac{x}{\alpha_1}\right)^{\frac{2}{\varepsilon}} + \left(\frac{y}{\alpha_2}\right)^{\frac{2}{\varepsilon}} = 1. \tag{13}$$

If $\varepsilon < 1$ (Fig. 6a), point $\mathbf{P}_M$ corresponds to a larger value of angle ϑ ($\vartheta_{\mathbf{P}_M} > \vartheta_{\mathbf{M}}$), where ϑ is the variable of the parametric equation

$$\mathbf{r}(\vartheta) = \begin{bmatrix} a_1 \cos^{\varepsilon}(\vartheta) \\ a_2 \sin^{\varepsilon}(\vartheta) \end{bmatrix}, \quad -\pi \le \vartheta < \pi. \tag{14}$$

Notice that $\frac{\pi}{4} < \vartheta_{\mathbf{P}_M} < \frac{\pi}{2}$. If $0 < \vartheta_{\mathbf{P}_M} < \frac{\pi}{4}$, which is the case for point $\mathbf{P}_{M_1}$ then $\vartheta_{\mathbf{P}_{M_1}} < \vartheta_{\mathbf{M}_1}$. If $\varepsilon > 1$, the above inequalities become reversed as illustrated in Fig. 6b. Assuming sets A and B, where

$$\begin{aligned} A &= \left\{k\frac{\pi}{2} < \vartheta < k\frac{\pi}{2} + \frac{\pi}{4}\right\} \\ B &= \left\{k\frac{\pi}{2} + \frac{\pi}{4} < \vartheta < (k+1)\frac{\pi}{2}\right\}, \end{aligned} \quad k \in Z, \tag{15}$$

then

$$\begin{aligned} &\vartheta_{\mathbf{M}} > \vartheta_{\mathbf{P}_M},\ \textit{if}\ (A \cap \{\varepsilon < 1\}) \cup (B \cap \{\varepsilon > 1\}) \\ &\vartheta_{\mathbf{M}} < \vartheta_{\mathbf{P}_M},\ \textit{if}\ (B \cap \{\varepsilon < 1\}) \cup (A \cap \{\varepsilon > 1\}). \end{aligned} \tag{16}$$

The above are simply extended for the case of the general 3D superquadric by simply replacing ϑ by η and ω. As a result, search directions are obtained for the two coordinate variables (η, ω) of the superquadric. Notice that despite the fact that the superquadric is a 3D surface in the 3D Euclidean space, it is only a 2D area in the non-Euclidean (curved) superquadric space. Thus, 2D search is performed in the area defined from points $(\eta_{\mathbf{M}}, \omega_{\mathbf{M}})$ and $(\eta_{\mathbf{M}} + \eta_r, \omega_{\mathbf{M}} + \omega_r)$, where η_r and ω_r are the search range variables as illustrated in Fig. 7a, which can be positive or negative according to (16).

Notice that the coordinate variables do not correspond to angle values of η and ω, but to indices to their already performed quantization. The search range is defined from the superquadric quantization density (η_d, ω_d) in the following way:

$$\eta_r = \omega_r = 2^{N_L} + 1, \tag{17}$$

where N_L is the integer part of

$$\mathrm{N}_L = \max\Big(\log_2 \frac{\eta_d}{16}, \log_2 \frac{\omega_d}{32}\Big). \tag{18}$$

Fig. 7 illustrates the levels of the created pyramid, which will be used in the 2D distance search method. The function I_S, to be maximized over the search region, is equal to the absolute internal product of the normalized normal vector to the superquadric surface at every point (η, ω) and the normalized line direction connecting points $\mathbf{M}$ and $\tilde{\mathbf{P}}_M$, which lies inside the search region and is a candidate minimum distance point.

$$I_S = \frac{1}{\|\mathbf{V}_{norm}\| \cdot \|\mathbf{L}_{dir}\|} |\mathbf{V}_{norm} \cdot \mathbf{L}_{dir}|, \tag{19}$$

where $\mathbf{V}_{norm}$ is the spatial derivative vector of function F, which is defined in (1), $\mathbf{V}_{norm} = \left[\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}, \frac{\partial F}{\partial z}\right]^T$ and $\mathbf{L}_{dir} = [x_{\tilde{\mathbf{P}}_M} - x_{\mathbf{M}}, y_{\tilde{\mathbf{P}}_M} - y_{\mathbf{M}}, z_{\tilde{\mathbf{P}}_M} - z_{\mathbf{M}}]^T$. I_S is maximized for point $\mathbf{P}_M$ (as shown in Fig. 6a) since, in this case, the vectors $\mathbf{V}_{norm}$ and $\mathbf{L}_{dir}$ are approximately collinear.

As noted earlier, the hierarchical search procedure is applied to find the point $\mathbf{P}_M$ that maximizes I_s: Initially, the function I_S is evaluated for the points of the top level N_L. Next, the function I_S is evaluated at level $N_L - 1$ for the neighbors of the point, which produce the higher I_S value at level N_L. The procedure is repeated until the bottom level is reached and point $\mathbf{P}_M$ is found. The hierarchical search method requires $8N_L + 1$ evaluations of function I_S instead of $(2_L^N + 1) \times (2_L^N + 1)$ of the exhaustive search.

After locating point $\mathbf{P}_M$, the distance D_{SQ} of this point from the original mesh is available from the distance map. If the distance of point $\mathbf{M}$ from point $\mathbf{P}_M$ is higher than D_{SQ}, collision is detected.

4.2.3 Layer 3 (Improvement on Layer 2)

The third layer of the *SQ-Map* collision detection algorithm builds on the results from Layer 2.

In every time update, the same hierarchical minimum distance search method is used to find the minimum distance point on the superquadric mesh. The difference lies on the subdivision procedure applied in this layer and in the use of the more accurate distance map obtained at Step 3 of the preprocessing, in order to achieve more accurate results. In particular, subdivision is performed to the base level of the hierarchy, simply by doubling the sampling density in both dimensions (η, ω) of the superquadric surface, thus quadrupling the samples. The accuracy factor A_S indicates the number of the needed consecutive subdivisions. The number of samples is therefore increased by a factor of 4^{A_S}. In order to speed up calculations, the 3D data of these levels can be evaluated in the preprocessing stage using (9), (10), and (11) instead of calculating them dynamically during runtime. After the minimum distance point $\mathbf{P}_M$ is found, a distance value is assigned to it, using linear interpolation of the preprocessing data obtained at Steps 2 and 3.

5 Accuracy Analysis

Both the collision detection and haptic rendering methods of *SQ-Map* are actually based on an approximate representation of the objects' surface using superquadrics and distance maps. In the following, the conditions that should be satisfied in order to efficiently use *SQ-Map* are analyzed.

Consider the continuous version C_M, C_{SQ} of the mesh surface S_{mesh} and superquadric mesh T_{SQ}, respectively, i.e., C_M consists not only of the mesh vertices, but also of all the points on its triangles (including the edges of course).

Let us also define the following three types of regions for the mesh S_{mesh}:

1. Convex region.
2. Concave region of type Q_1: The concave regions that are comprised of points that can be projected onto the superquadric surface without self-intersecting with S_{mesh} (area of arc S in Fig. 5b).
3. Concave region of type Q_2: The concave regions that are comprised of points that cannot be projected onto the superquadric surface without self-intersecting with S_{mesh} (point $\mathbf{W}$ in Fig. 5a).

Assuming that:

1. C_M is bounded by the superquadric surface C_{SQ}.
2. $\forall \mathbf{x} \in C_M, \quad \exists \mathbf{y} \in C_{SQ}, D_{SQ}(\mathbf{y}) \quad so\ that$

$$\mathbf{x} = \mathbf{y} - D_{SQ}(\mathbf{y}) \cdot \mathbf{n}_{SQ}(\mathbf{y}), \tag{20}$$

   where $\mathbf{n}_{SQ}(\mathbf{y})$ is the normal direction of C_{SQ} at point $\mathbf{y}$ and
3. $D_{SQ}(\mathbf{y})$ is a scalar function with domain C_{SQ} and represents the distance of point $\mathbf{y}$ from C_M alongside the normal direction $\mathbf{n}_{SQ}(\mathbf{y})$.

Lemma. *Using (20) with the scalar function D_{SQ}, an exact representation of C_M is obtained if and only if the mapping f_C of all points of C_M on the superquadric is injective (one-to-one), where f_C projects the C_M onto the superquadric surface.*

Proof. Step 1: Forward proof with contradiction.

If the representation of C_M is exact, let us assume that the mapping is not injective, i.e., there exist at least two points $\mathbf{x}, \mathbf{x}' \in C_M$, $\mathbf{x} \neq \mathbf{x}'$ that are projected onto the same point $\mathbf{y} \in C_{SQ}$. Then:

$$\mathbf{x} = \mathbf{y} - D_{SQ}(\mathbf{y}) \cdot \mathbf{n}_{SQ}(\mathbf{y}),$$

$$\mathbf{x}' = \mathbf{y} - D'_{SQ}(\mathbf{y}) \cdot \mathbf{n}_{SQ}(\mathbf{y}).$$

Representing both equations with respect to $\mathbf{y}$and equalizing:

$$\mathbf{x}' - \mathbf{x} = \Big(D_{SQ}(\mathbf{y}) - D'_{SQ}(\mathbf{y})\Big) \cdot \mathbf{n}_{SQ}(\mathbf{y})$$

But, function $D_{SQ}(\mathbf{y})$is scalar, thus $D_{SQ}(\mathbf{y}) = D'_{SQ}(\mathbf{y})$. As a result $\mathbf{x}' = \mathbf{x}$, thus reaching in a contradiction.

Step 2: Inverse proof.

If function f_C is injective, then for each $\mathbf{x} \in C_M$, there exists a point $\mathbf{y} \in C_{SQ}$ that is uniquely associated with $\mathbf{x}$. Therefore, using the associated distance value $D_{SQ}(\mathbf{y})$the position of every $\mathbf{x} \in C_M$ is accurately and uniquely described from (20). □

Corollary 1. *For the case of the sampled T_{SQ} surface, the lemma can be expressed as follows:*

Using (20) with the scalar discrete function D_{SQ} the error of the representation of C_M can be asymptotically reduced to zero while increasing the sampling density of T_{SQ} if and only if the mapping f_C of all points of C_M on the continuous superquadric surface is injective (one-to-one), where f_C projects the C_M onto the superquadric surface.

Corollary 2. *Using (20) with the scalar discrete function D_{SQ} the error of the representation of S_{mesh} can be asymptotically reduced to zero while increasing the sampling density of T_{SQ} if and only if the surface of S_{mesh} does not include concave regions of type Q_2.*

Corollary 2 obviously stems from the fact that if S_{mesh} includes concave regions of type Q_2 then the mapping function f_C cannot be injective. It is emphasized that, as the above Corollary 2 implies, asymptotically zero error in the representation of S_{mesh} is guaranteed whenever the surface of S_{mesh} is convex or even concave of type Q_1.

The conditions of the lemma and its corollaries are considered during the decomposition procedure of Section 3. In particular, the proposed decomposition scheme is set up so as to cut the initial object in the areas that exhibit concavities of either type Q_1 or Q_2, since in these areas the "Boundary Strength" variable (Section 3) exhibits high values. In this way, quasiconvex subobjects are created that contain as few concave areas as possible.

In the ideal case, the algorithm partitions the object into subobjects with convex or with concave regions of type Q_1 thus satisfying the condition of "Corollary 2." However, in practice, the algorithm may encounter small local concavities of type Q_2. In this case, the approximation will tend asymptotically to a nonzero error value. It is therefore obvious that accuracy is not only related to the sampling density of the superquadric, but also on how well the superquadric approximates the mesh.

Regarding the accuracy of each Layer it should be mentioned that Layer 3 by definition includes all features of Layer 2 and additionally maps the distance information of all vertices of S_{mesh} onto the superquadric. Thus, for the same superquadric sampling density, it provides always a more accurate approximation since it handles directly the errors of too loose or too strict approximation (Fig. 5b) that are inherent in Layer 2.

6 Haptic Rendering

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.

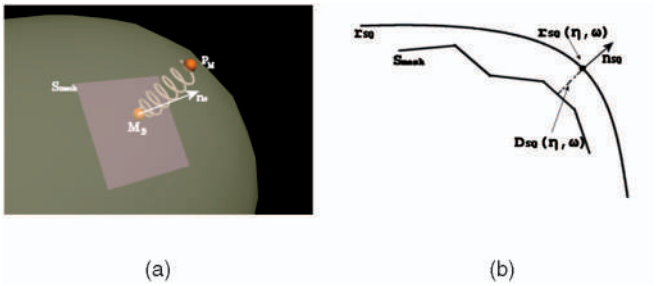


Fig. 8. Haptic rendering.

6.1 Calculating the Force Feedback

Referring to Fig. 8a, let point $\mathbf{M}_B$ be a vertex of object $\mathbf{B}$ and S_{mesh} represent the local surface of $\mathbf{A}_{SQ}$. Also let $S_{SQ}^{\mathbf{M}}$ represent the distance of point $\mathbf{M}_B$ from the superquadric, which corresponds to point $\mathbf{P}_M$ on the superquadric surface. If collision is detected, the absolute value of the force fed onto the haptic device is obtained using a spring model as illustrated in Fig. 8a. In particular:

$$\|\mathbf{F}\| = k \cdot \left| S_{SQ}^{\mathbf{M}} - D_{SQ}(\mathbf{P}_M) \right|, \tag{21}$$

where k is the spring constant. $D_{SQ}(\mathbf{P_M})$ is the distance of point $\mathbf{P_M}$ from the mesh and is stored in the distance map of the superquadric. Notice that the term $|S_{SQ}^{\mathbf{M}} - D_{SQ}(\mathbf{P_M})|$ is an approximation of the actual distance of $\mathbf{M}_B$ from the mesh that becomes more accurate if the superquadric 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 nonrealistic noncontinuous forces at the surface element boundaries. In the present framework the analytical approximation of the mesh surface is used utilizing the already obtained superquadric 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 $D_{SQ}(\eta, \omega)$ is the scalar function of the distance map on the superquadric, as described in Section 4, the surface S_{mesh} of $\mathbf{A}_{SQ}$ can be approximated by (22) (Fig. 8b):

$$\mathbf{S_{mesh}}(\eta, \omega) = \mathbf{r}_{SQ}(\eta, \omega) - D_{SQ}(\eta, \omega)\mathbf{n}_{SQ}(\eta, \omega), \tag{22}$$

where

$$r_{SQ}(\eta, \omega) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_1 \cos^{\varepsilon_1}\eta \cdot \cos^{\varepsilon_2}\omega \\ a_2 \cos^{\varepsilon_1}\eta \cdot \sin^{\varepsilon_2}\omega \\ a_3 \sin^{\varepsilon_1}\eta \end{bmatrix}, \quad \forall \eta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] \omega \in [-\pi, \pi) \tag{23}$$

is the parametric definition of the superquadric and $\mathbf{n}_{SQ}(\eta, \omega)$ the normal vector for each point $\mathbf{r}_{SQ}(\eta, \omega)$. The vector $\mathbf{n}_{SQ}(\eta, \omega)$ is defined as the cross product of the tangent vectors along the coordinate curves [14],

$$\mathbf{n}_{SQ}(\eta,\omega)=\mathbf{t}_{\eta}(\eta,\omega)\times\mathbf{t}_{\omega}(\eta,\omega)=s(\eta,\omega)\mathbf{n}_{d}= \\ =s(\eta,\omega)\begin{bmatrix}\frac{1}{a_1}\cos^{2-\varepsilon_1}\eta\cdot\cos^{2-\varepsilon_2}\omega\\ \frac{1}{a_2}\cos^{2-\varepsilon_1}\eta\cdot\sin^{2-\varepsilon_2}\omega\\ \frac{1}{a_3}\sin^{2-\varepsilon_1}\eta\end{bmatrix}, \tag{24}$$

where

$$s(\eta,\omega)=-a_1a_2a_3\varepsilon_1\varepsilon_2\sin^{\varepsilon_1-1}\eta\cdot\cos^{2\varepsilon_1-1}\eta\cdot \\ \sin^{\varepsilon_2-1}\omega\cdot\cos^{\varepsilon_2-1}\omega. \tag{25}$$

The calculation of the force feedback demands the evaluation of the normal vector $\mathbf{n}_S$ on the object's surface. Similarly to (24), it is obtained through (26). In the following the brackets (η,ω) will be omitted for the sake of simplicity.

$$\mathbf{n}_S=\frac{\partial\mathbf{S}_{mesh}}{\partial\eta}\times\frac{\partial\mathbf{S}_{mesh}}{\partial\omega}, \tag{26}$$

where

$$\frac{\partial\mathbf{S}_{mesh}}{\partial\eta}=\frac{\partial\mathbf{r}_{SQ}}{\partial\eta}-\frac{\partial D_{SQ}}{\partial\eta}\mathbf{n}_{SQ}-D_{SQ}\frac{\partial\mathbf{n}_{SQ}}{\partial\eta}, \\ \frac{\partial\mathbf{S}_{mesh}}{\partial\omega}=\frac{\partial\mathbf{r}_{SQ}}{\partial\omega}-\frac{\partial D_{SQ}}{\partial\omega}\mathbf{n}_{SQ}-D_{SQ}\frac{\partial\mathbf{n}_{SQ}}{\partial\omega}. \tag{27}$$

The term $\frac{\partial\mathbf{r}_{SQ}}{\partial\eta}$ can be computed directly:

$$\frac{\partial\mathbf{r}_{SQ}}{\partial\eta}=\frac{\varepsilon_1}{2}\sin 2\eta\begin{bmatrix}-\alpha_1\cos^{\varepsilon_1-2}\eta\cos^{\varepsilon_2}\omega\\ -\alpha_2\cos^{\varepsilon_1-2}\eta\sin^{\varepsilon_2}\omega\\ \alpha_3\sin^{\varepsilon_1-2}\eta\end{bmatrix}, \tag{28}$$

while the term $\frac{\partial D_{SQ}}{\partial\eta}$ is computed numerically. Finally,

$$\frac{\partial\mathbf{n}_{SQ}}{\partial\eta}=\frac{\partial s}{\partial\eta}\mathbf{n}_d+s\frac{\partial\mathbf{n}_d}{\partial\eta}. \tag{29}$$

Terms $\frac{\partial s}{\partial\eta}$ and $\frac{\partial\mathbf{n}_d}{\partial\eta}$ are easily computed through (24) and (25),

$$\frac{\partial s}{\partial\eta}=A\cos^{2\varepsilon_1}\eta\cdot\sin^{\varepsilon_1-2}\eta\left[1-\frac{2\varepsilon_1-1}{\varepsilon_1-1}\tan^2\eta\right], \tag{30}$$

$$\frac{\partial\mathbf{n}_d}{\partial\eta}=\left(1-\frac{\varepsilon_1}{2}\right)\sin 2\eta\begin{bmatrix}-\frac{1}{a_1}\cos^{-\varepsilon_1}\eta\cdot\cos^{2-\varepsilon_2}\omega\\ -\frac{1}{a_2}\cos^{-\varepsilon_1}\eta\cdot\sin^{2-\varepsilon_2}\omega\\ \frac{1}{a_3}\sin^{-\varepsilon_1}\eta\end{bmatrix}, \tag{31}$$

where $A=-a_1a_2a_3\varepsilon_1\varepsilon_2\sin^{\varepsilon_2-1}\omega\cdot\cos^{\varepsilon_2-1}\omega$.

Using (26)-(31), the direction of the normal $\mathbf{n}_S$ is obtained. Equivalent analysis is performed for the derivatives with respect to the coordinate curve ω. Notice that the object surface normal direction can be precalculated. This provides much faster force feedback evaluation than dynamically calculating the normal from the object surface or the gradient of a distance field, which is a common practice in the haptics literature [21].

Finally, the direction of the normal along the surface of the modeled object is obtained using (26), thus resulting to the reaction force, which is:

$$\mathbf{F}_{reaction}=k\left|S_{SQ}^{\mathrm{M}}-D_{SQ}(\mathbf{P}_M)\right|\frac{\mathbf{n}_S}{\|\mathbf{n}_S\|}. \tag{32}$$

Using (32) the force feedback corresponding to collision between elementary parts (e.g., vertices and triangles) is calculated. When an area of an object collides with a part of a second object, several triangles are colliding with each other, thus resulting in numerous components of the force feedback of (32). When providing force feedback by accumulating these force components, the final force depends on the number of contacts, thus also on the sampling density of the object (high number of contacts for a fine triangulation of an object), and could become very high and lead to instabilities. In the present framework, all force elements inside an impact zone are averaged and then fed onto the device for interaction. The impact zone is defined as the area, where forces should be transferred to the haptic device (e.g., fingers for the CyberGrasp and a small sphere around the pointer for the Phantom). With the use of impact zones, independency from the sampling density of the objects is achieved and no instabilities have been observed.

If smoothing of the force field is required, simple smoothing operations can be applied only on the 2D distance map, thus resulting in smooth force feedback.

It has also to be noted that the force cannot be guaranteed to be continuous. If point $\mathbf{M}_B$ crosses the medial axis of the superquadric a discontinuity will arise on the minimum distance point $\mathbf{P}_M$ on the superquadric and a force discontinuity may be encountered. However, experiments have shown that, in practice, this is not very likely and the possibility of such a discontinuity becomes even lower if the superquadric approximates well the mesh. Moreover, the defect of the discontinuity is lowered through the previously described force averaging in the impact zones.

6.2 Modeling Friction

The force calculated from (32) 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 [22]. In order to avoid this undesirable defect, a friction component is added to the force of (32). In particular,

$$\mathbf{F}_{friction}=-f_C\cdot\left(1+k_f\left|S_{SQ}^{\mathrm{M}}-D_{SQ}(\mathbf{P}_M)\right|\right)\cdot\frac{\mathbf{n}_f}{\|\mathbf{n}_f\|}, \tag{33}$$

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 $k_f|S_{SQ}^{\mathrm{M}}-D_{SQ}(\mathbf{P}_{\mathrm{M}})|$ is used in order to increase the magnitude of the friction force when the penetration depth of the processed point increases. The variables S_{SQ}^{M} and $D_{SQ}(\mathbf{P}_{\mathrm{M}})$ are defined in (21), while factor k_f controls the contribution of the penetration depth to the calculated friction force. Finally, the force fed onto the haptic device yields from the addition of the reaction and the friction force, $\mathbf{F}_{haptic}=\mathbf{F}_{reaction}+\mathbf{F}_{friction}$.

6.3 Modeling Haptic Texture

Using *SQ-Map* for haptic rendering, haptic texture [22] can also be simulated easily by applying appropriate transformations on the acquired distance map. In the present framework, in order to simulate surface roughness, Gaussian noise [22] is added to the distance map. No computational cost is added, since the procedures for calculating the

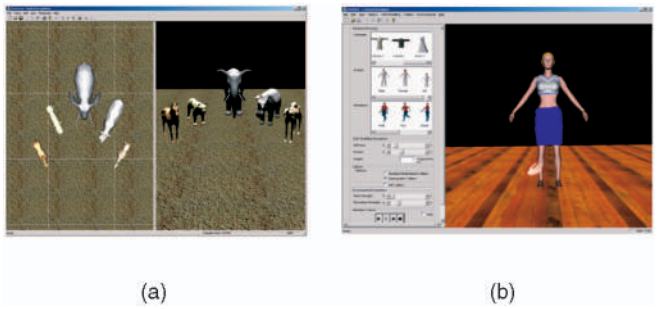


Fig. 9. (a) Object recognition application. (b) Costume designer application.

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:

$$\mathbf{F}_{texture} = k\left|S_{SQ}^{\mathbf{M}} - (D_{SQ}(\mathbf{P}_M) + n_g)\right| \frac{\mathbf{n}_S}{\|\mathbf{n}_S\|}, \qquad (34)$$

where n_g denotes th Gaussian noise.

7 Experimental Results

The proposed methods were evaluated using two off-the-shelf haptic devices, namely, Phantom and CyberGrasp. The experimental evaluation of the *SQ-Map* collision detection and haptic rendering is presented in Sections 7.1 and 7.2, respectively.

SQ-Map was integrated in two different haptic interaction applications, which are the nonvisual object recognition for blind people training and the costume designer application.

The first of the developed applications aims at the recognition of virtual objects (Fig. 9a) using only the sense of touch, which is intended to be used by blind or visually impaired users [23]. In the second, (Fig. 9b), the user is able to interactively simulate clothing [24] as well as to perform fitting and editing operations of cloth over avatars using haptic devices. All results were obtained with an Intel Pentium 4, 3.0 GHz with 512MB RAM and an ATI Radeon 9800XT graphics card.

7.1 Collision Detection Evaluation

The proposed superquadric based collision detection algorithm is compared to standard mesh-based approaches [3], [4] as well as to distance field-based methods [8], [12]. The experiments consist of touching and handling a virtual object using the haptic device, for the two applications (Fig. 9). It should be noted that in both experiments the scene objects are almost at every time step in collision with each other. The virtual hand that is queried against the scene objects is composed of 4,790 vertices and 9,344 triangles. For the performed experiments the precomputation time was found to lie between 4 seconds for the simplest model (Camel) to 7 seconds for the most complex (Elephant). These times include processing for decomposition, superquadric modeling, distance map generation that is performed only once during the modeling of the objects.

Fig. 10 illustrates the decomposition of three of the processed objects, using the algorithms of Section 3.

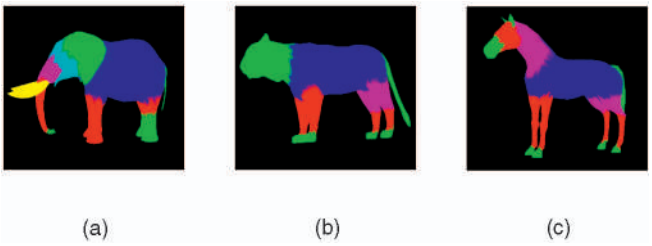


Fig. 10. Decomposition of three of the the processed models. (a) Elephant. (b) Tiger. (c) Horse.

The three different curves of the diagrams in Figs. 11a and 11b represent the percentage of the computational cost of *SQ-Map*, with respect to [3], using the three different layers of the superquadric collision detection procedure (the method in [3] corresponds to 100 percent). Note that they correspond to simulations where the involved objects are continuously colliding. If the objects were not colliding at all, the execution times would be almost identical. The difference in efficiency is obvious, while it becomes more noticeable when handling more complex virtual objects. Moreover, *SQ-Map* can also detect collisions between a rigid and a deformable object. It should also be mentioned that *SQ-Map* (approximate method) does not report the same type of information with the approach in [3] (accurate method), since it approximates the object's surface with an implicit equation and a distance map. The error, as illustrated in Fig. 14c, is calculated for all points $\mathbf{M}_i$ of object $\mathbf{B}$ that lie inside the superquadric modeling $\mathbf{A}_{SQ}$ through:

$$e = \sum_{\forall \mathbf{M}_i \in SQ} \frac{|\hat{D}_M(\mathbf{M}_i, \mathbf{n}_{SQ}(\mathbf{P}_{Mi})) - D_M(\mathbf{M}_i, \mathbf{n}_{SQ}(\mathbf{P}_{Mi}))|}{R_{\max}}, \qquad (35)$$

where $\mathbf{P}_{Mi}$ is the projection of $\mathbf{M}_i$ onto the superquadric $\hat{D}_M$ the estimated distance from S_{mesh} along side the normal direction $\mathbf{n}_{SQ}$ and $\hat{D}_M$ the accurate one. R_{max} is the maximum distance between the vertices of the object.

In both simulations, the error varies around 0.4 percent to 0.5 percent and is also illustrated in Fig. 14c. Finally, experiments have shown that there is no need to use Layer 3 collision detection if the discretization of the superquadric

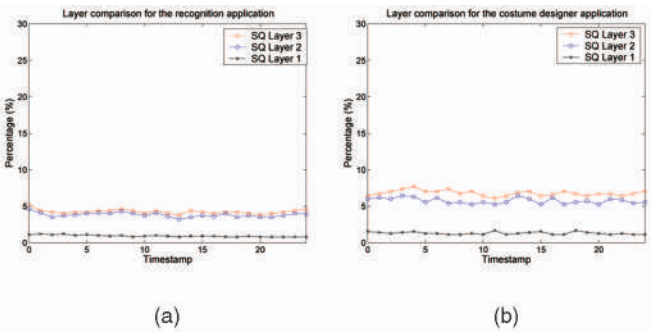


Fig. 11. Percentage of the computational time needed for the *SQ-Map* collision detection method compared to the mesh-based approach [3] for the (a) recognition application and (b) costume designer application.

TABLE 1
Memory Requirements

Model	Polygons	Memory in MBytes			SQ
		DF	ADF	SQ	segments
Virtual hand	9344	344	55	4.6	16
Tiger	8108	312	51	4.3	13
Horse	5946	231	37	2.9	17
Elephant	10272	417	68	5.1	16
Camel	4512	196	31	2.4	14
Rhino	7154	295	52	3.8	13
Scene	45336+	1795+	294+	23.1+	89

surface is fine. In general, there is a trade-off between runtime efficiency and memory allocated in order to store the distance map of the superquadric surface.

Distance field methods [7], [8] are slightly faster than *SQ-Map*, at the cost, however, of huge memory requirements in order to produce a dense 3D grid. Table 1 illustrates the memory requirements of each object individually as well as for the whole scene of Fig. 9a using a fine discretization of the distance field and distance maps. Figs. 12a and 12b illustrate the memory requirements of the distance field method [8], the adaptive distance field method [12] and *SQ-Map* collision detection for the "tiger" model and for the whole scene (Fig. 9a), respectively, using floating point arithmetics for the distance map and grid, with respect to the sampling density.

From Fig. 12, it can be deduced that, despite its efficiency, the distance field method is inappropriate when the virtual environment consists of many detailed objects. Using the adaptively sampled distance fields [12], the memory requirements decrease, but they are still high for the case of the simulation of complex virtual environments, which is the case in Fig. 9a. As a consequence, despite the theoretically unbeatable performance of distance fields, their extremely high memory requirements could lead in practical applications to a significant slow down of performance due to the use of virtual memory.

On the contrary, the memory requirements of the OBB are less than 5MBs for the processed models. This value seems initially comparable to the values of *SQ-Map* as presented in Table 1. However, the memory requirements of the OBB should be considered to be low compared to the *SQ-Map*, since they correspond to an accurate representation of the object, while for *SQ-Map* to an approximate. If the sampling of the superquadric is increased so as to reduce the approximation error, the memory requirements will increase far beyond the ones of the OBB that are constant for a specific model.

Another remarkable method was presented in [13], which trades memory for speed by partitioning the space using 2^{3N}-trees, in particular, 512-trees, instead of octrees and by reducing the tree-depth to three levels. As a result, the time to traverse the tree is held low at the cost, however, of higher memory requirements than the adaptive distance field. Thus, its memory requirements are still much higher when compared to the ADF's and especially to those of the presented approach.

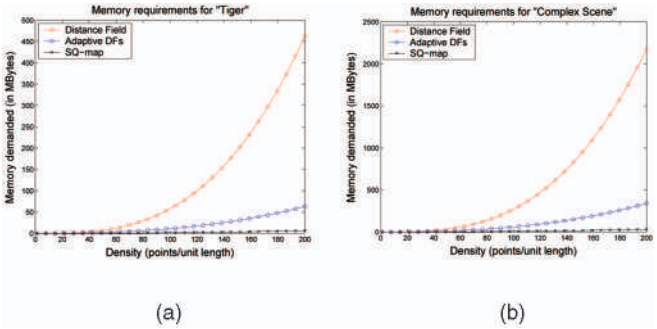


Fig. 12. Memory requirements for the distance field [8], the adaptive distance field [12] and the proposed *SQ-Map* method (a) for the "Tiger" and (b) for the whole scene (Fig. 9a).

The superquadric-based method diminishes the need for memory allocation, while retaining all benefits of the distance field-based collision detection. This is caused by the fact that it uses a 2D distance array mapped onto the surface of the superquadrics instead of the 3D distance grid. On the other hand, there is the need to perform object decomposition when handling complex virtual objects, which is efficiently performed in the context of *SQ-Map*, as described in Section 3.

In general, the approximation of the object's surface using a distance map (height field) [25] is very efficient and simplifies processes like collision detection. However, this provides an accurate approximation only if the objects do not have large concavities on their surface. In the present work, problems of concave features are handled using the decomposition method, which tends to decompose an object into quasiconvex regions. Moreover, the accuracy of the representation is directly related to the sampling density of the superquadric. The higher the sampling density the more accurate becomes the representation. On the other hand, this would result in an increment of the memory requirements of the approach, needed so as to store the more detailed distance maps. This would also be the case for the distance field-based methods, when trying to refine their accuracy.

7.2 Haptic Rendering

Regarding the accuracy of the force feedback, it should be noticed that it depends initially on the 3D object decomposition. If the decomposition and the superquadric modeling is not accurate and large concavities of type Q_2 exist inside a segment, which is approximated with a single superquadric, then it is not possible to represent these concave areas using the distance map information only. In such a case, the resulting force field would not be accurate in the areas where the approximation of the surface is also not accurate. However, this case was not encountered in practice, since the 3D decomposition scheme tends to produce segments without such concavities.

Moreover, the accuracy of the force feedback depends also on the accuracy of the obtained distance map, i.e., the discretization of the superquadric surface. If it is fine enough and the decomposition-superquadric approximation scheme produces segments that satisfy the lemma presented in Section 5, the force will not inherit the force

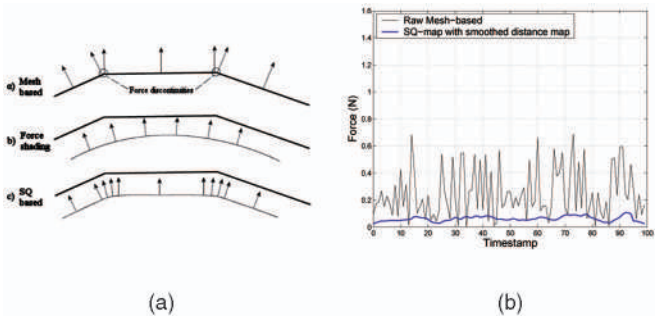


Fig. 13. (a) Force feedback using only the mesh, force shading, and smoothing on the SQ distance map and (b) force feedback directly evaluated from the mesh and force feedback obtained from *SQ-Map* with distance map smoothing.

discontinuity, from which all methods that use the 3D mesh to obtain haptic feedback suffer [21], when moving through mesh polygons. In particular, the force feedback evaluated using the mesh of the object will be perpendicular to its surface, thus resulting to a relatively strong discontinuity when crossing an edge (Fig. 13a). When a simple local smoothing operation is applied 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 [26] or the distance field method when the pointer of the haptic device is not very close to the real surface [22]. Fig. 13b illustrates the magnitude of the difference of the force feedback for subsequent frames ($F_{diff} = \|\mathbf{F}_i - \mathbf{F}_{i-1}\|$) under sliding motion, when force feedback is obtained directly from the mesh and when using the presented method with a smoothed distance map. It is obvious that *SQ-Map* produces smoother force feedback since F_{diff} does not contain spikes and has relatively small values.

Figs. 14a, 14b, and 14c illustrate absolute timings of *SQ-Map* and the "performance index versus error" graph. The upper plot of Fig. 14a illustrates timings for collision detection and force feedback evaluation separately as well as the combined frame update time and corresponds to sliding motion in the recognition application environment using all three layers of the *SQ-Map* framework so as to obtain a very accurate result. Fig. 14b depicts total timings for collision detection and haptic rendering for a relatively high number of contacts. The figure illustrates that the algorithm performs almost equally fast no matter, which is the number of contacts. This does not mean that the algorithm is totally independent of the number of contacts. However, during the collision between two objects, it achieves almost constant performance if the objects are either slightly colliding or under heavy collision or even not colliding at all, but very close with each other, since the algorithm will perform almost the same tests for the points that lie inside the bounding superquadric of the object.

The "performance index versus error" graph of Fig. 14c plots the performance of *SQ-Map* with respect to the error, which corresponds to the desired accuracy, for different sampling densities of the superquadric. The values of "performance" are the average timings when the respective error threshold is forced.

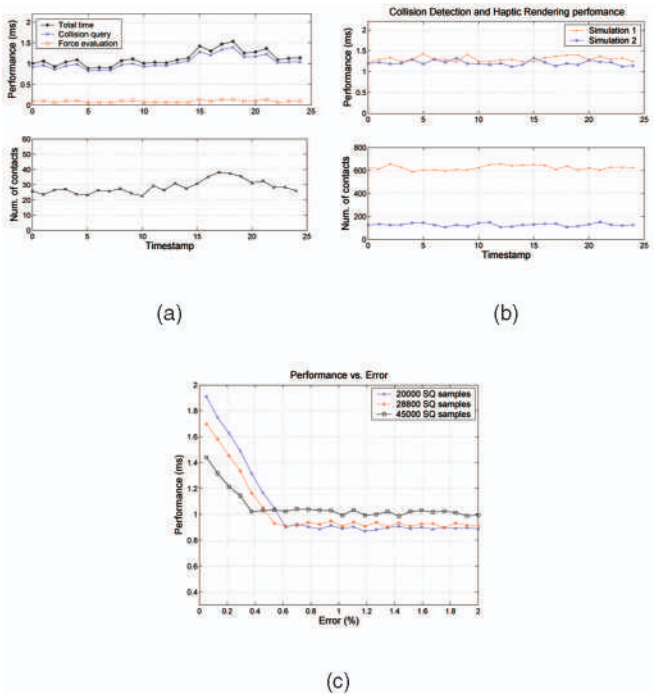


Fig. 14. (a) Absolute timings for *SQ-Map* under sliding motion. (b) Absolute timings for high number of contacts. (c) Performance index versus Error graph. The error is calculated from (35).

The "performance index versus error" graph highlights several interesting properties of the *SQ-Map*. The part of each plot that exhibits almost constant performance corresponds to Layer 2 collision detection. Notice that in Layer 2, the performance decreases slightly when the superquadric discretization is increased. The sloping parts of the graph correspond to Layer 3 collision detection. The performance in this part decreases because more dense subdivision is performed. When using a more dense sampling on the superquadric the error of the Layer 2 collision detection decreases and in the "performance versus error" graph, the transition from Layer 2 to Layer 3 occurs for lower values of the error threshold, i.e., for higher values of accuracy. It is emphasized that the error of Fig. 14c never reaches zero, however, many more subdivisions are performed in Layer 3.

Finally, it should be mentioned that the present work presents an alternative core algorithm for computing the force feedback for haptic display. It is capable to perform 6DOF haptic rendering, which is degraded in the presented experiments only to force feedback and not torque. Moreover, popular state-of-the-art techniques like virtual coupling, precontact braking forces [13], haptic texturing, etc., which are very common in the haptics literature [11], [22], can be easily and efficiently integrated in *SQ-Map* so as to provide more realistic and stable haptic rendering.

8 Conclusions

In this paper, a novel framework for real-time collision detection and haptic interaction with complex virtual environments is presented. The efficient collision detection algorithm models each virtual object using superquadrics. For complex objects, decomposition is performed and each resulting subobject is approximated using a superquadric. The distance of the mesh and the superquadric is mapped

onto its surface and is used in combination with its analytical formula in order to rapidly perform collision detection without the need to execute triangle per triangle intersection tests. The force feedback provided to the user is calculated efficiently, in an integrated way, using the analytical description of the superquadric and the distance map.

In general, the *SQ-Map* has the following contributions and limitations.

Contributions: The superquadrics-based method has certain advantages over the more common OBB or k-DOP. It provides a simple, parameterized implicit representation of the underlying shape. This leads to the improvements reported. This method can be viewed as a trade-off between OBB, which is space-efficient, but slower, and the original distance field method, which is faster, but consumes huge amount of memory. Moreover, it can handle collisions between a rigid and a deformable object. Force feedback is calculated explicitly from the superquadric representation of the shape, while haptic effects can be easily implemented by processing the obtained distance map.

Limitations: Superquadrics representation comes at a cost. Not all objects can be easily decomposed into superquadrics and the precomputation time is increased. SQ-Map computes an approximated distance between colliding objects that becomes more accurate for more accurate superquadric approximations. Moreover, small local concavities may be missed by the decomposition algorithm and special care must be taken to detect edge-edge contacts using an offsetting value on the distance map, since the relative position of the edges of the objects possibly involved in collision cannot be calculated explicitly. Finally, there is no guarantee that the force feedback will be continuous. However, a smooth force field can be easily obtained by applying smoothing operations on the distance map.

Experimental results demonstrate that significant computational gain is expected using *SQ-Map* in haptic virtual reality applications, without a huge increment in memory requirements.

ACKNOWLEDGMENTS

This work has been supported by the EU funded projects SIMILAR (network of excellence) and CATER (STREP) and the Greek Secretariat of Researcg and Technology funded projects VR@Theater and APEIRO.

REFERENCES

[1] G.C. Burdea and P. Coiffet, *Virtual Reality Technology,* second ed. Wiley-IEEE Press, 2003.
[2] P.M. Hubbard, "Approximating Polyhedra with Spheres for Time-Critical Collision Detection," *ACM Trans. Graphics,* vol. 15, no. 3, pp. 179-210, July 1996.
[3] S. Gottschalk, M.C. Lin, and D. Manocha, "OBBTree: A Hierarchical Structure for Rapid Interference Detection," *Computer Graphics, Proc. ACM SIGGRAPH,* pp. 171-180, 1996.
[4] J.T. Klosowski, M. Held, J.S.B. Mitchell, H. Sowizral, and K. Zikan, "Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs," *IEEE Trans. Visualization and Computer Graphics,* vol. 4, no. 1, pp. 21-36, Jan.-Mar. 1998.
[5] M.C. Lin and J.F. Canny, "A Fast Algorithm for Incremental Distance Calculation," *Proc. IEEE Int'l Conf. Robotics and Automation,* vol. 2, pp. 1008-1014, 1991.
[6] G. van den Bergen, "Efficient Collision Detection of Complex Deformable Models Using AABB Trees," *J. Graphics Tools,* vol. 2, no. 4, pp. 1-13, Apr. 1997.
[7] S.J. Osher and R.P. Fedkiw, *Level Set Methods and Dynamic Implicit Surfaces.* Springer-Verlag, 2002.
[8] A. Fuhrmann, G. Sobottka, and C. Gross, "Distance Fields for Rapid Collision Detection in Physically Based Modeling," *Proc. GraphiCon '03,* pp. 58-65, Sept. 2003.
[9] M. Teschner, S. Kimmerle, B. Heidelberger, G. Zachmann, L. Raghupathi, A. Fuhrmann, M.-P. Cani, F. Faure, N. Magnenat-Thalmann, W. Strasser, and P. Volino, "Collision Detection for Deformable Objects," *Proc. Eurographics,* 2004.
[10] S. Osher and J. A. Sethian, "Fronts Propagating with Curvature-Dependent Speed: Algorithms Based on Hamilton-Jacobi Formulations," *J. Computational Physics,* vol. 79, no. 1, pp. 12-49, 1988.
[11] J.A. Sethian, P.G. Ciarlet, A. Iserles, R.V. Kohn, and M.H. Wright, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science.* Cambridge Univ. Press, 1999.
[12] S.F. Frisken, R.N. Perry, A.P. Rockwood, and T.R. Jones, "Adaptively Sampled Distance Fields: A General Representation of Shape for Computer Graphics," *Computer Graphics and Interactive Techniques,* pp. 249-254, 2000.
[13] W.A. McNeely, K.D. Puterbaugh, and J.J. Troy, "Six Degree-of-Freedom Haptic Rendering Using Voxel Sampling," *Computer Graphics and Interactive Techniques,"* pp. 401-408, 1999.
[14] F. Solina and R. Bajcsy, "Recovery of Parametric Models from Range Images: The Case for Superquadrics with Global Deformations," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 12, no. 2, pp. 131-147, Feb. 1990.
[15] D.D. Hoffman and M. Singh, "Salience of Visual Parts," *Cognition,* vol. 63, pp. 29-78, 1997.
[16] H.S. Lin, H.M. Liao, and J. Lin, "Visual Salience-Guided Mesh Decomposition," *Proc. IEEE Int'l Workshop Multimedia Signal Processing (MMSP),* 2004.
[17] G. Taubin, "Estimating the Tensor of Curvature of a Surface from a Polyhedral Approximation," *Proc. Int'l Conf. Computer Vision,* pp. 902-907, 1995.
[18] J. Peng, Q. Li, C.C Jay Kuo, and M. Zhou, "Estimating Gaussian Curvatures from 3D Meshes," *Human Vision and Electronic Imaging, Proc. SPIE,* pp. 270-280, 2003.
[19] M. Do Carmo, *Differential Geometry of Curves and Surfaces.* Prentice Hall, 1976.
[20] D. DeCarlo and D. Metaxas, "Blended Deformable Models," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 18, no. 4, pp. 443-448, Apr. 1996.
[21] G.C. Burdea, *Force and Touch Feedback for Virtual Reality.* Wiley-Interscience Publication, 1996.
[22] L. Kim, A. Kyrikou, G. Sukhatme, and M. Desbrun, "An Implicit-Based Haptic Rendering Technique," *Proc. IEEE/RSJ Int'l Conf. Intelligent Robots and Systems (IROS '02),* 2002.
[23] D. Tzovaras, G. Nikolakis, G. Fergadis, S. Malasiotis, and M. Stavrakis, "Design and Implementation of Haptic Virtual Environments for the Training of Visually Impaired," *IEEE Trans. Neural Systems and Rehabilitation Eng.,* vol. 12, no. 2, pp. 266-278, June 2004.
[24] K. Moustakas, D. Tzovaras, and M.G. Strintzis, "Fast Hierarchical Simulation of Cloth and Deformable Objects Using an Optimal Pyramidal Representation," *Proc. Computer Animation and Social Agents Conf. (CASA '04),* pp. 163-170, July 2004.
[25] M.A. Otaduy, N. Jain, A. Sud, and M.C. Lin, "Haptic Display of Interaction between Textured Models," *Proc. Conf. Visualization (VIS '04),* pp. 297-304, 2004.
[26] D.C. Ruspini, K. Kolarov, and O. Khatib, "The Haptic Display of Complex Graphical Environments," *Computer Graphics (SIGGRAPH '97 Conf. Proc.),* pp. 345-352, 1997.

Konstantinos Moustakas received the Diploma degree in electrical and computer engineering from Aristotle University of Thessaloniki, Thessaloniki, Greece, in 2003. Currently, he is a PhD candidate in the Aristotle University of Thessaloniki, where he also serves as a teaching assistant. He is a research fellow with the Informatics and Telematics Institute, Centre for Research and Technology Hellas, Thessaloniki. His main research interests include virtual reality, collision detection, haptics, deformable object modeling and simulation, 3D content-based search, computer vision, and stereoscopic image processing. During the last three years, he has been the (co)author of more than 20 papers in refereed journals, edited books, and international conferences. He has also been involved in four projects funded by the EC and the Greek secretariat of Research and Technology. He is a member of the Technical Chamber of Greece.

Dimitrios Tzovaras received the Diploma degree in electrical engineering and the PhD degree in 2D and 3D image compression from Aristotle University of Thessaloniki, Thessaloniki, Greece, in 1992 and 1997, respectively. He is a senior researcher in the Informatics and Telematics Institute of Thessaloniki. Prior to his current position, he was a senior researcher on 3D imaging at the Aristotle University of Thessaloniki. His main research interests include virtual reality, assistive technologies, 3D data processing, medical image communication, 3D motion estimation, and stereo and multiview image sequence coding. His involvement with those research areas has led to the coauthoring of more than 35 papers in refereed journals and more than 80 papers in international conferences. He has served as a regular reviewer for a number of international journals and conferences. Since 1992, he has been involved in more than 40 projects in Greece, funded by the EC and the Greek Secretariat of Research and Technology. Dr. Tzovaras is an associate editor of the *EURASIP Journal of Applied Signal Processing* and a member of the Technical Chamber of Greece.

Michael Gerassimos Strintzis (M'70-SM'80-F'04) received the Diploma degree in electrical engineering from the National Technical University of Athens, Athens, Greece, in 1967, and the MA and PhD degrees in electrical engineering from Princeton University, Princeton, New Jersey, in 1969 and 1970, respectively. He then joined the Electrical Engineering Department at the University of Pittsburgh, Pittsburgh, Pennsylvania, where he served as assistant professor (1970-1976) and associate professor (1976-1980). Since 1980, he has been a professor of electrical and computer engineering at the University of Thessaloniki, Thessaloniki, Greece, and, since 1999, director of the Informatics and Telematics Research Institute, Thessaloniki. His current research interests include 2D and 3D image coding, image processing, biomedical signal and image processing, and DVD and Internet data authentication and copy protection. Dr. Strintzis has served as associate editor for the *IEEE Transactions on Circuits and Systems for Video Technology* since 1999. In 1984, he was awarded one of the Centennial Medals of the IEEE. He is a fellow of the IEEE and the IEEE Computer Society.