Efficient Contact Detection for Game Engines and Robotics
A non-polygonal approach with smooth convex objects

Artur A. Gonçalves\textsuperscript{abc}

Supervisors: Daniel S. Lopes\textsuperscript{a}, Joaquim A. Jorge\textsuperscript{ae}, and Alexandre Bernardino\textsuperscript{bc}

Abstract—Meshes are considered the gold standard regarding contact geometries of video game engines and robot models. However, meshes are not the best formulations when controlled precision and execution time become paramount. In this paper, we propose Parametric and Implicit formulations for precise contact distance estimations. Indeed, parametric and implicit descriptions provide more compact descriptions than meshes, while making it possible to approximate man-made and organic objects with great precision, namely robotic fingers. These precise geometric representations can then support fast calculation of distances with arbitrary precision without paying a storage penalty, as would be the case for meshes. Our results show that either superellipsoids or superovoids can provide more accurate distance estimations than meshes in practical settings where precise contact points, surface normals and clearance estimations are required. In addition, in order to model smooth convex contact geometries and associated mechanical systems, an interactive sketch-based application was designed to expeditiously model articulated bodies, or multibody systems, using various superellipsoids attached to links of rigid bodies. Furthermore, the potential use of superellipsoids to model deformable contact geometries, such as terrain scenes or cloths, was also explored.

Index Terms—collision detection, ovoidal shape, multibody modeling, iCub robot, grasp modeling

I. INTRODUCTION

For robots to adequately interact with their environment, their actions should be planned with a sufficient degree of accuracy. The simulations for this planning involve, among other things, queries for the distance between objects and parts of the robot, and the penetration depth between intersecting objects and robot parts. Furthermore, these computations must run in real-time. In particular, the case study of this article focuses on the collision detection and minimum distance computations for grasp planning with the iCub robot hand [1]. Grasp planning strategies very often rely on the computation of grasp quality criteria, such as force closure [2], that require the position of the contact points and corresponding surface normals. Hence, the accurate computation of these quantities is highly important for the success of grasp planning algorithms.

One of the most noteworthy interactions between bodies in a mechanical system is collision, i.e., when two bodies physically intersect at one point or region, exerting a force in one another [3][4]. Because collisions greatly influence the behavior of a mechanical system, they must be accurately modeled and accounted for. In computational simulations, collision detection is the operation used to compute whether two objects are intersecting, and if so, in which specific region of space that intersection occurs.

Collision detection depends on the geometry of the considered objects. Therefore, the representation of object’s geometry should be chosen wisely, so that the simulated system is accurate enough when compared to its real counterpart. A common approach is to approximate the objects by polyhedrons, also called meshes [5]. Since meshes can only represent straight edges and faces, any smooth curves must be discretized, thus, approximated using a large number of vertices to achieve a faithful representation of the surface geometry. These high-resolution meshes lead to slow collision detection algorithms and high memory usage, which hurts their usability for real-time applications. In addition, mesh normals can be very noisy and change abruptly throughout the surface, due to their faceted nature.

Alternatively, objects can be defined using analytical curves and surfaces, such as spheres and boxes. In particular, superellipsoids (SE) [6] have been proposed, for which time- and memory-efficient collision detection algorithms exist [7][8]. Superellipsoids have been used as collision detection surfaces for robotic haptic feedback devices [9], and there are examples of usage of these shapes to represent organic structures, such as human femoral heads [10]. Superovoids are marginally more complex than superellipsoids but have a higher accuracy in approximating shapes such as the parts of multi-fingered robotic hands (see Fig. 6), organic structures [10] and everyday life objects.

Contributions: We propose the usage of novel analytical smooth convex surfaces (SCS), in particular the superovoid (SO), as contact geometries to be used in physical simulations for robotic grasp motion planning. We present algorithms to compute the minimum distance between superovoidal surfaces, for both implicit and parametric representations. We provide models of the iCub robot’s hand with superovoids and superellipsoids and compare them to an existing mesh-based implementation in terms of accuracy and computation time. The proposed algorithms are available at a fork of the Flexible Collision Library [11] (FCL) repository, as free and open source. Finally, we present two interactive sketch-based applications: an articulated body modeling application, which was used to create a draft of the iCub’s hand, with attached SCS contact geometries, and a terrain modeling app which showcases the versatility of SCS for 3D modeling and the collision detection between superellipsoids and a deformable cloth.

Note: This paper acts as the extended abstract of its accompanying thesis, of the same name. It is an adaptation of the article ’Accurate Contact Geometries for the iCub Hand’ submitted to ICRA 2016 in the scope of said thesis. While this article focuses on the robotics application of the studied

\textsuperscript{a}INESC-ID Lisboa, Portugal
\textsuperscript{b}ISR Lisboa, Portugal
\textsuperscript{c}Instituto Superior Técnico Lisboa, Portugal

collision detection algorithms, it also summarily presents the thesis’s studies on expeditious 3D modeling with smooth convex surfaces.

II. LITERATURE REVIEW

A. Collision detection

As mentioned earlier, the adequate geometric representation of robotic parts is crucial for efficient collision detection. In [12], oriented bounding boxes and superquadrics are employed to represent a robotic hand and objects to be grasped. For a broad-phase planning, in [13] the volume in which the fingers can move is coarsely represented by cylinders. To exploit concave parts of objects, which provide a secure grip for grasping, [14] use (super)ellipsoids, elliptic cylinders and one-sheet hyperboloids. The problem of representing humanoid hands appears in other areas as well: for hand pose estimation and tracking, and to decrease the collision detection processing time, in [15] one hand is represented with overlapping spheres. Other more complex configurations exist: [16] employs spheres, cones and cylinders.

Collision detection also plays an important role in the simulation of complex biomechanical systems. The Simbody engine [17] was developed to accurately simulate the dynamics of biological structures, and is adjustable to real-time or high-fidelity simulations as needed. It supports rigid, deformable and elastic contact detection and handling, and is used in biomedical and prosthetic research, and in robotics, namely, in the Gazebo robot simulator [18].

Many specialized libraries exist to compute a series of collision-related queries, of which RAPID, SOLID, PQP and KCCD are examples, and focus on specific geometries such as Axis-Aligned Bounding Boxes (AABBs), Oriented Bounding Boxes (OBBs) and Continuous Collision Detection (CCD). On the other hand, FCL [11] performs a vast array of proximity queries, using rigid, deformable or articulated models, represented as primitive geometries, meshes, point clouds, and bounding volume hierarchies (BVH). FCL was designed to allow the addition of new collision detection methods, and different representations of objects. For collision detection purposes, meshes are represented with BVH. FCL is used in MoveIt! [19], a robotic motion planning, navigation and control library.

Collision detection can also be performed based on distance fields: these are pre-calculated 3D arrays overlapped onto the considered mesh geometry, and store the distance from each cell to the mesh’s surface. A more efficient variation proposed in [20] computes a bounding superellipsoid around the object’s mesh and stores the distances to points in the surface of this superellipsoid instead. This reduces the distance field into a 2D array, saving considerable amounts of memory.

Superellipsoids were extensively studied in [20] in the context of collision detection. In [21] the authors proposed a new approach towards the reformulation of contact detection between convex implicit surfaces. Central to this reformulation are the implicit surface representation and the elegant way on how the Householder formula computes orthonormal sets given a normal surface vector [21]. Although the contact detection algorithm was formulated for any pair of smooth convex implicit surfaces, it has been specifically implemented for the (super)ellipsoid-(super)ellipsoid contact pair and the numerical results show that it has second-order convergence. However, this study lacks a comparative analysis between other contact geometry representations such as parametric surfaces and meshes.

B. 3D modeling

As the 3D representation of contact geometries is crucial for the efficiency and accuracy of subsequent simulations, the methods to create them are worth studying. Models are commonly assembled and rigid with contact geometries in professional software suites such as Maya [22], with feature complex WIMP (window, icon, menu, pointer) interfaces, and present a steep learning curve. Alternatively, there has been research in using sketch-based interfaces to create 3D models from 2D rough drawings [23][24][25], and even employing automatic skeleton generation to existing 3D models [26][27]. To generate contact geometries, geometry simplification algorithms such as the one presented in [28] for general volumes, or in [29] for sphere hierarchies, are useful, as simpler contact volumes are usually associated with faster collision detection.

III. SUPEROVOIDS

The superovoid is a smooth convex surface based on the Barr tapered superellipsoid [6]. It is defined by its inside-outside function

\[ F(x, y, z) = \left( \frac{x}{T_x \frac{a_3}{a_1} + 1} \right)^{\frac{1}{T_2}} + \left( \frac{y}{T_y \frac{a_3}{a_2} + 1} \right)^{\frac{1}{T_2}} + \left( \frac{z}{a_3} \right)^{\frac{1}{T_2}} \]  \hspace{1cm} (1)

where \(a_1, a_2, a_3 > 0\) are the lengths of the shape in the X, Y and Z axes, \(\epsilon_1, \epsilon_2 \in [0, 2]\) are the squareness factors for the XY plane and Z axis, and \(T_x, T_y \in [-0.5, 0.5]\) are the tapering (egg-shape) factors in X and Y, respectively.

Points satisfying \(F(x, y, z) = 1\) are on the surface of the superovoid. The points for which the result of the function is lower than 1 are inside the surface, while those that yield a result greater than 1 are outside. Thus, the implicit function for the superovoid in the canonical form is

\[ F(x, y, z) - 1 = 0 \]  \hspace{1cm} (2)

When \(T_x = T_y = 0\), the superovoid is symmetrical in relation to its equatorial plane (it does not have an ovoidal shape anymore), becoming a superellipsoid [6].
Superovoids can also be described using an angle-center parametrization:

\[
P(\varphi_1, \varphi_2) = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} a_1 \left(T_x \frac{x}{n_x} + 1\right) \operatorname{absgn}(\cos \varphi_1, \epsilon_1) \operatorname{absgn}(\cos \varphi_2, \epsilon_2) \\ a_2 \left(T_y \frac{y}{n_y} + 1\right) \operatorname{absgn}(\sin \varphi_1, \epsilon_1) \operatorname{absgn}(\cos \varphi_2, \epsilon_2) \\ a_3 \operatorname{absgn}(\sin \varphi_2, \epsilon_2) \end{bmatrix},
\]

\[
-\pi \leq \varphi_1 \leq \pi \\
-\frac{\pi}{2} \leq \varphi_2 \leq \frac{\pi}{2}
\]

where \( \operatorname{absgn}(x, p) = \text{sign}(x) |x|^p \) (4)

Note that in the computational implementation, the \( z \) coordinate in (3) must be calculated first as the \( x \) and \( y \) coordinates depend on its value.

IV. ALGORITHMS

The proposed algorithms are based on [8] and answer the following problem: given two smooth convex surfaces in 3D space, what is the minimum distance between them? Specifically, what is the pair of points that defines that minimum distance. Using the distance between these points, it is possible to assert whether the surfaces are in collision or not. The original algorithm on [8] is formulated for implicitly-represented surfaces, while we also propose an adaptation using parametric representations. Furthermore, two novel methods of obtaining initial guesses for the iterative algorithm are presented.

The common-normal concept [4] states an important condition for the solution of this problem. Let \( P \) and \( Q \) be two points on two \( C^1 \) continuous surfaces \( (i) \) and \( (j) \), respectively, which are in contact on those points, or have a minimum distance expressed by those points. The common-normal concept states that the normal direction to each surface on the respective points, \( \mathbf{n}_{OP} \) and \( \mathbf{n}_{OQ} \), must be the same, and equal to the direction of the distance vector \( \mathbf{d}_{PQ} \). As a consequence, the tangent and binormal vector directions on both surface points define two planes parallel to each other. A 2D case is illustrated in Figure 1.

The expression for the normal direction to the superovoid surface, at a given point, can be obtained from the gradient of its implicit function [2]. For the tangent and binormal directions, the Householder transformation is employed, as it reveals to be an efficient vector orthogonalization technique whose analytical nature provides a well defined formula for the tangent and binormal vector spaces of implicit surfaces, as explained in [21]. This transformation is expressed as

\[
H = I - 2 \frac{\mathbf{n}_T \mathbf{n}_T^T}{\mathbf{n}_T^T \mathbf{n}_T} = \begin{bmatrix} 1 - 2 \frac{h_1^2}{h_2^2} & -2 \frac{h_1 h_2}{h_2^2} & -2 \frac{h_1 h_3}{h_2^2} \\ -2 \frac{h_1 h_2}{h_2^2} & 1 - 2 \frac{h_2^2}{h_2^2} & -2 \frac{h_2 h_3}{h_2^2} \\ -2 \frac{h_1 h_3}{h_2^2} & -2 \frac{h_2 h_3}{h_2^2} & 1 - 2 \frac{h_3^2}{h_2^2} \end{bmatrix}
\]

(5)

where \( \mathbf{n} \) is the given normal and

\[
\mathbf{h} \equiv \mathbf{n}(\mathbf{n}) = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}^T = \begin{bmatrix} \max\{n_x - \|\mathbf{n}\|_2, n_x + \|\mathbf{n}\|_2\} \\ n_y \\ n_z \end{bmatrix}^T
\]

(6)

with

\[
\mathbf{h} \equiv \|\mathbf{n}\|_2
\]

(7)

Matrix \( \mathbf{H} \) is symmetric and orthogonal, and its rows (or columns) form an orthogonal vector basis. Its first column is collinear to \( \mathbf{n} \). Its second and third columns are perpendicular to \( \mathbf{n} \), and can thus be considered as tangent \( \mathbf{t}(\mathbf{n}) \) and binormal \( \mathbf{b}(\mathbf{n}) \) directions associated with the normal at a given surface point.

For implicit surfaces, the common normal concept can be mathematically expressed as a system of non-linear equations applied to the problem of finding the minimum distance between two superovoids:

\[
\Phi(\mathbf{q}) = 0 \iff \begin{bmatrix} \mathbf{n}_{OP}^T \cdot \mathbf{t}_{OQ} \\ \mathbf{n}_{OP}^T \cdot \mathbf{b}_{OQ} \\ \mathbf{d}_{PQ}^T \cdot \mathbf{t}_{OQ} \\ \mathbf{d}_{PQ}^T \cdot \mathbf{b}_{OQ} \\ F_i \\ F_j \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}
\]

(8)

where

\[
\mathbf{q} \equiv \mathbf{q}_{\text{implicit}} = \begin{bmatrix} x_i \\ y_i \\ z_i \\ x_j \\ y_j \\ z_j \end{bmatrix}^T
\]

(9)

is the vector of unknowns, containing the coordinates of the potential minimum distance points, in the local coordinate system of each implicit surface. The last two geometric constraints in (8) are geometric loci constraints, and ensure the points are on the surface of the superovoids.

For parametric surfaces, the two geometric loci conditions are not considered, as the first four equations are sufficient to find the four unknowns, and the parametric representation guarantees that the considered points lie on the surface. Thus, for the parametric variant of the proposed algorithm, the vector of unknowns becomes

\[
\mathbf{q} \equiv \mathbf{q}_{\text{parametric}} = \begin{bmatrix} \varphi_{1i} \\ \varphi_{2i} \\ \varphi_{1j} \\ \varphi_{2j} \end{bmatrix}^T
\]

(10)

where \( \varphi_{1,2i,j} \) are the parametric coordinates of two potential solution points for the minimum distance problem, on surfaces \( i \) and \( j \).

For the case of superovoids, the vector of geometric constraints \( \Phi \) is composed of \( C^2 \) continuous functions, which makes the function twice-differentiable. As such, (8) can be solved using the iterative Newton-Raphson method. An
improved approximation to the solution \(q_k\), \(q_{k+1}\) is computed based on a potential candidate \(q_k\):

\[
q_{k+1} = q_k - \left(\Phi_q(q_k)\right)^{-1}\Phi(q_k)
\]

where \(\Phi_p\) is the Jacobian matrix of the geometric constraints function \(\Phi\) and \(k\) is the Newton-Raphson iteration index. This Jacobian matrix is computed numerically using finite differences. The iterative procedure stops once the difference between \(q_{k+1}\) and \(q_k\) is less than a certain tolerance, which can be adjusted easily to compute more accurate results, or accelerate the algorithm at the cost of precision.

\[\text{Algorithm} \]

\begin{enumerate}
  \item **A. Initial iteration**
  
  The presented algorithm is of numerical nature, and as such, an initial guess is needed to iterate over and reach an approximation of the solution. Due to the non-linearity of the superovoid expressions and the locality of the Newton-Raphson convergence, the proximity of the initial candidate points to the solution is critical; a poor guess will make the algorithm diverge and not reach a solution, or perform many iterations and run slowly.

  Two methods were explored to find acceptable initial guesses. The first one consists of generating an octree approximation of the superovoids around their surfaces, and compute the minimum distance between them. The resulting pair of points are used as an initial guess for the Newton-Raphson algorithm. An example is visible in Figure 2. FCL already implements octree algorithms through Octomap [30]. This method is used when implicit surfaces are considered.

  The second method is based on the parametric representation of superovoids. The two-dimensional parametric space defined in (3) is split into a quadtree for each superovoid, and the center of each cell of the quadtree is considered. For each level of the quadtree, every pair of center points of each superovoid is evaluated (16 pairs): the equivalent 3D points are computed using (3) and the euclidean distance between them is calculated. The algorithm selects the two cells for which the computed distance is smallest, and repeats the process inside the two selected cells for a given number of iterations.

  The two parametric coordinates found in the final iteration are used as the initial estimation for the parametric Newton-Raphson procedure. Experimentation suggests that 6 iterations yield acceptable estimations and allow the numerical procedure to converge. An example of execution of this algorithm is shown in Figure 3.

  Both these methods converge globally to the solution, albeit at a slower rate than the Newton-Raphson algorithm. This means that, should the Newton-Raphson procedure not converge to the solution, the initial guess can always be recomputed with increased precision, either by considering a more finely divided octrees in the implicit method, or further subdividing the quadtrees in the parametric method. The improved guesses can then be used to restart the Newton-Raphson algorithm.

  Collision detection algorithms can greatly benefit from exploiting temporal coherence, as it is likely that the relative position of two simulated objects will not change significantly during short time-steps. Thus, for the proposed algorithm, the minimum distance points computed in one collision query are used as the initial guess of the following query using the same surfaces.

  \[\text{B. Implementation}\]

  The superovoid shape and the described algorithm were implemented in FCL, in C++. The matrix operations necessary for the Newton-Raphson method use the LAPACK library [31]. The Newton-Raphson procedure stops once the tolerance of \(10^{-6}\) is reached, or when 30 iterations are executed, as a safeguard. Slight modifications were introduced in ROS [32] and MoveIt! [19], in particular to the URDF parsing modules, to support this new shape.

  The code has been published in a fork of FCL’s repository (https://github.com/artur-ag/fcl), free and open source.

V. 3D MODELING WITH SCS

A. Multibody Sketcher

Traditional programs for 3D body modeling are complex and not very user-friendly, making them inappropriate for expeditiously creating sketches of articulated bodies. Multibody Sketcher was developed to tackle these problems, allowing the
design of articulated structures associated with SCS contact geometries. The tool was designed to be used on touch-screen devices such as tablets and table-top screens. The output of this program is a skeleton rigged with superellipsoid-shaped hitboxes, which can be used in physical simulations. A working prototype of Multibody Sketcher was developed using Unity3D, and its usage is shown in Figure 4.

Multibody Sketcher’s workflow is divided into 3 modes. In ‘Skeleton’ mode, the user creates the skeletal structure of the body, by drawing strokes which are segmented into bone chains connected by joints. New bones are snapped into other chains if they are created near an existing joint. In ‘Joint’ mode, the degrees of freedom and ranges of rotation of each joint are configured, via a panel with sliders for each degree of freedom. Pre-defined joints can also be selected. In ‘Hitbox’ mode, the user creates SCS contact geometries and attaches them to each bone. Sketched ellipses are recognized with the CALI library [33] and converted to 3D ellipsoids, which can then be translated, rotated and scaled using 2-finger touch gestures. The superellipsoid exponents are tweaked using a widget invoked with a touch-and-hold action over a superellipsoid.

The squareness of the superellipsoid shape is controlled by the $\epsilon_1$ and $\epsilon_2$ exponents. A touch-based widget was designed to allow users to easily modify these parameters. This widget is composed of a grid, with values of $\epsilon_1$ and $\epsilon_2$ [0, 2] mapped to the horizontal and vertical axes, respectively. Overlaid on the grid’s edges are small icons that allow the user to preview what a superellipsoid looks like when its parameters have the values mapped to the icons positions. The user invokes the widget by touching and holding over a superellipsoid, and without releasing the finger, sliding to the desired shape. Screenshots can be seen in Figure 5.

B. iCub hand representation

The iCub robot’s hands were manually approximated with superovoids and superellipsoids, using the original CAD drawings as reference (http://sourceforge.net/projects/robotcub/). An approximation using polygonal meshes, available at [34] as a URDF file, was used for comparison, in which each finger is divided in 4 segments, or links. The meshes are composed of 300, 440, 910 and 770 vertices, in order from the base of the finger to the fingertip links, which means each finger link is represented by 605 vertices on average.

C. Terrain modeling with SCS

As a secondary goal, the usage of SCS for 3D modeling was also studied in a real-time interactive terrain modeling application, developed in Unity3D for tablets and table-top touch-screen platforms. In this application, called Terrain Cloth, a simulated cloth falls on top of superellipsoid objects, which are creatively laid out such that the cloth resembles...
mountains and hills when it falls and drapes over the objects (see Figure 7).

The collision detection between the superellipsoids and the mesh-based cloth is performed similarly to [35]. The implicit function of the superellipsoids is evaluated on each cloth vertex; if the evaluated result is negative, then the vertex is inside the superellipsoid, and must be pushed out via a contact force. To avoid the computational cost of testing every vertex, these are first grouped into AABBs, which are tested against the superellipsoid’s own AABB first.

VI. RESULTS AND DISCUSSION

The proposed implicit and parametric algorithms were timed and compared with FCL’s mesh minimum distance query. The simulation consisted in moving the iCub’s right hand towards its left, palms facing each other, until the tips of both thumbs touched, and then rotating the right hand inward, causing the fingers of the right hand to intersect and pass through the fingers of the left. This ensured plenty of collisions between the different links of the fingers. The movements were manually performed in the RViz interface. The queries were timed only when the considered pair of surfaces were intersecting.

The hand representations proposed in section V were considered, with mesh, implicit and parametric versions tested separately using the following hand representations: the simplified meshes of the iCub hand from [34], with about 605 vertices per finger link (i); superellipsoids (Figure 6 (b)) using the superellipsoid-specific implementation, which is mathematically simpler (ii); the same representation but with the general superovoid implementation (iii); and the mixed representation (Figure 6 (c)) using the superovoid implementation for the fingertips and the superellipsoid implementation for the other links (iv).

A total of 8 tests were performed in MoveIt! with ROS Indigo, both compiled from source, on a Lubuntu 32 bit system with an Intel® Core™ i7-4700MQ @ 2.40 GHz and 4 GB of RAM. The time results of these tests are shown in Table I and Figure 8. An isolated test of the geometric accuracy of the algorithms was run, using 2 superovoids and comparing them to the 2 corresponding meshes generated from discretizations with varying resolution. Its results can be seen in Figure 9.

The results in Table I indicate that the proposed algorithms run 1.5 to 4 times faster than their mesh equivalent. As for FCL’s mesh collision detection query, despite being significantly faster than the minimum distance queries, it uses an OBB hierarchy to accelerate the computation at the expense of geometric accuracy, hence it cannot be considered a minimum distance technique but rather a proximity query method.

The algorithm for implicit surfaces is considerably slower than the parametric version, due to the octree-based initial estimation being quite expensive (500 µs per estimation, on average), and a new estimation must be calculated if the Newton-Raphson algorithm does not converge using the result from the previous time-step as the initial iteration. The parametric quadtree-based estimation method is simpler, as it is defined in a 2D domain, and the parametric geometric constraints, while requiring more non-linear functions evaluations due to the cosines and sines, guarantee that the potential points are on the surfaces and the numerical method diverges less often. Still, these spurious needs for new estimations increase the time standard deviation of the proposed algorithms, especially for the implicit versions.

The simpler superellipsoid versions were not significantly different than the general superovoid ones in terms of computation time, even though the mathematical expressions for computing normals and values of the implicit function were slightly faster when tested in isolation.

Regarding the geometric precision of the superovoid and superellipsoid queries, these can be easily adjusted, while for mesh-based algorithms, this would involve regenerating the mesh with a different number of vertices and triangles, which might not be feasible in real-time. The reported contact normal for mesh-based algorithms can also be noisy due to the faceted nature of polygonal meshes and strictly depends on its tessellation, which does not happen for the proposed algorithms. As shown in Figure 9, the mesh-based algorithm only achieves a precision of $10^{-4}$ m using more than 786 vertices (see Figure 10); the superovoid version’s precision is $10^{-6}$ m and takes an order of magnitude less time to compute. Furthermore, the superovoid query is faster than the mesh one.
Figure 8. Computation time distributions of the minimum distance queries, comparing FCL’s mesh implementation with the proposed methods. In all cases the average and median times are lower compared to the mesh algorithm. In some rare cases the proposed methods can have high computation times ($\approx 1$ ms) if the Newton-Raphson procedure diverges and has to start over from a new initial estimation (more noticeable in the implicit representations). SE: Superellipsoid, SO: Superovoid, M: mixed.

Figure 9. Comparison of computation times and geometric accuracy between the proposed parametric superovoid algorithm and the FCL mesh minimum distance points query. The meshes are in the shapes of superovoids, of length parameters $a_1, a_2, a_3 \in [0.96, 1.4]$ m, with different resolutions.

Figure 10. Meshes of superovoids used for the accuracy benchmark of Figure 9 in their relative positions. On the left, meshes with 18 vertices, 32 triangles. On the right, meshes with 786 vertices, 1568 triangles.

Table II
QUALITATIVE COMPARISON OF PROS AND CONS OF THE PROPOSED AND STUDIED GEOMETRIES AND ALGORITHMS, IN TERMS OF COMPUTATION TIME (AVERAGE AND STANDARD Deviation), Required MEMORY, AND GEOMETRIC ACCURACY.

<table>
<thead>
<tr>
<th>Geometry Type</th>
<th>Time (avg)</th>
<th>Time ($\sigma$)</th>
<th>Memory</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Meshes (BVH collision)</td>
<td>fastest</td>
<td>low</td>
<td>high</td>
<td>low</td>
</tr>
<tr>
<td>Meshes (min. distance)</td>
<td>slow</td>
<td>low</td>
<td>high</td>
<td>low</td>
</tr>
<tr>
<td>SE/ SO implicit</td>
<td>fast</td>
<td>high</td>
<td>low</td>
<td>high</td>
</tr>
<tr>
<td>SE/ SO parametric</td>
<td>faster</td>
<td>high</td>
<td>low</td>
<td>high</td>
</tr>
</tbody>
</table>

except for meshes with 18 vertices or less, and these meshes lead to very low geometric precision.

Finally, it is worth noting that, using either implicit or parametric representations, it is possible to fully describe a superovoid storing only 7 (shape) + 6 (rigid body) parameters stored as floating point numbers, while polygonal meshes occupy considerably more memory to store vertices, triangles and eventual BVH geometries. Thus, the pros and cons of using implicit or parametric superellipsoids or superovoids for minimum distance computations, compared to meshes, are qualitatively summarized in Table II.

VII. CONCLUSION

Simulating grasping motion in robotics requires evermore geometric precision without trading off computational efficiency. In recent years, meshes have been employed as collision detection geometries, since bounding volume hierarchy algorithms are very time-efficient for that end. However, they fall short on geometric accuracy, and accurate mesh-based minimum distance algorithms require very detailed meshes and perform slowly. In this paper, we presented a precision controlled numerical algorithm to compute the minimum distance between superovoidal contact surfaces that is more...
computationally-efficient than standard mesh approaches. We demonstrated that it can be applied to robotic action planning, by implementing the analytical contact geometries in FCL and using it together with MoveIt! to simulate the iCub’s hand, represented with superovoids and superellipsoids. Both implicit and parametric representations are compared, and the algorithms both perform faster than FCL’s mesh-based minimum distance implementation, with the added advantages of representing the contact geometry with less memory compared to meshes, while achieving arbitrarily accurate contact points and normals. To the authors’ knowledge, the (super)ovoid shape has never been used in robotic grasping simulation. A prototype of an interactive sketch-based tool was also developed, allowing to expeditiously create articulated bodies and rig them with SCS contact geometries. This prototype served as a starting point for modeling the iCub hand with superellipsoids.

There are several improvements and future avenues worth looking into. An accurate starting estimate is critical to make the Newton-Raphson algorithm converge to the correct solution. Methods to obtain this starting iteration should be further compared, and new, faster and more precise methods should be investigated. The implicit and parametric versions of the algorithm should be further compared, in purely geometric benchmarks, where the starting conditions for both algorithms are the same. The Jacobian matrix used in the numerical procedure can be analytically computed, which should accelerate the execution of the algorithm. Smooth convex surfaces, in particular superovoids, may also be adequate representations of other parts of robots, such as the iCub’s arms and torso. The proposed algorithms should be employed and tested in physics simulations, where their higher accuracy would prove even more useful.

The Multibody Sketcher prototype, presented here only as a means to preliminarily sketch the iCub’s hand, should be polished and compared with other modeling applications via user-tests. The Terrain Cloth application was also shown as a demonstration of the versatility of SCS for modeling, but has potential as a sketch-based terrain modeling software, which should be asserted and studied with user-tests.

REFERENCES


