Strategies to Accelerate Deformable and Rigid Bodies Simulations

A non-polygonal approach to contact detection

José Serrano
Instituto Superior Técnico, Universidade de Lisboa
Lisboa, Portugal
jose.c.serrano@tecnico.ulisboa.pt

ABSTRACT
Collision detection is a core component in every physics-based simulation. In interactive applications, simple geometry is traditionally used to approximate the physical objects for performance reasons. However, they hardly retain the detail present in the higher resolution meshes, used in 3D environments. On the other hand, polygonal meshes are slow as contact geometries, and only a low-resolution version can satisfy the strict performance constraints of interactive applications. Even more challenging to approximate are deformable objects, like cloth, which can change shape over time. This work, studies the feasibility of using a type of smooth convex surface, namely superellipsoids, as a contact geometry. And presents an iterative method to compute the minimum distance to an arbitrary point in space, and the nearest point on the superellipsoid's surface. Since it can be used for elementary collision tests with vertices, this approach aims to provide a more efficient and less memory-demanding alternative to meshes in physical simulations. To test the performance of the contact detection algorithm, a case-study of a tablecloth falling onto a superellipsoid, and another of a sequence of superellipsoids falling into a bowl, were performed. Benchmarks show the developed algorithm is adequate for real-time applications and provide better efficiency and accuracy than mesh-based methods, with speedups of 5 to 26 times faster.

Author Keywords
Collision detection; superellipsoid; cloth simulation; rigid body simulation.

INTRODUCTION
The movement and interactions of physical bodies with the environment, under the action of forces, are mathematically described by three fundamental principles of Mechanics, known as the Newton's laws of motion. Commonly referred as the action-reaction law, the third law of motion is directly related with one of the major interactions between material bodies, that is, the collision between them (Johnson et al 1987). While in the real world this phenomenon happens naturally, this is not true when the same is mapped to a physical simulation. The contact geometry of the simulated object must be modelled and programmed to detect and respond to an intersection. Without them objects would simply pass through each other undetectable.

Since the geometry used for contact detection can greatly influence the object's behavior, it should be carefully chosen, so that the simulation is convincing when compared to its real counterpart. A common way to represent contact geometries in computer graphics is to approximate objects using polygonal meshes. Since they are traditionally used to visually represent a 3D object, it makes sense to use them as its own contact geometry. Although, simple shapes are frequently used instead, when it is not required to have a precise collision behavior or when a certain performance goal must be met. Despite meshes being the "motto", an object can alternatively be defined by analytical curves and surfaces (Barr 1981). There is clearly a trade-off to consider when choosing contact geometries, between realistic accuracy and performance.

While collision detection for rigid bodies is well investigated, a challenging problem arises when simulating deformable models (Nealen et al. 2006). As these objects can change shape over time, any attempt to represent them with stiff geometries become inadequate. Thus, the solutions for this normally pass by decomposing the object or space into smaller parts and then, after a broad-phase culling, doing elementary tests, i.e., a vertex or triangle of the model, which was deemed close enough to the contact geometry after a broad-phase culling (Teschner et al. 2005).

A well-researched class of deformable bodies, which has many applications, is cloth. The most popular methods to simulate cloth are force based, the mass-spring system (Provot 1995) and the Finite Element Method (FEM) (Etzmuß et al. 2003). A more recent, geometric based approach in dynamics simulation, namely, Position-Based Dynamics (PBD) (Müller et al. 2007) differs from the traditional force and impulse based methods in the way that it acts directly on positions, omitting the velocity and acceleration layers. This approach provides speed and a high level of control compared to other methods. In real world applications of cloth simulation, the modelling of the other objects in the scene, which should interact with the cloth mesh, presents itself with a drawback. Usually, the primitive geometric shapes available are not flexible enough to represent the desired object with detail. And so, developers tend to resort to meshes. However, since this representation can only display straight faces, any curve must be discretized. Thus, to approximate a smooth curvature and improve collision precision, a large number of vertices are

used, which in consequence, slow down the collision detection algorithms and have a high memory cost.

There are, as already mentioned, analytical alternatives to polygonal meshes. One type of Smooth Convex Surface (SCS), namely superellipsoid (SE), which has a flexible closed shape, could be an adequate solution to this problem. Primarily, this paper aims to research and analyze the potential of using superellipsoids for collision detection with vertex-based objects, as an alternative to meshes, and presents a fast collision detection algorithm, suitable for interaction with rigid and deformable models, with a special attention to cloth. For an elementary collision test, the geometric problem of contact detection can be formulated as the following: given an arbitrary superellipsoid and an arbitrary point in non-conformal configurations (arbitrary position, orientation and dimensions), what is the minimum distance between them and what is the nearest point on the surface of the superellipsoid? Based on this distance, it is possible to determine if the two objects are colliding or not. And together with the information of the nearest point, it is certain how to correctly push an intersecting vertex out of the surface. This work is built upon two pillars described above: 1) position-based dynamics is the base for the simulations; 2) solid objects interacting with the vertex-based models in the simulation are modelled using SEs, replacing the traditional mesh representation. This type of surfaces provides better precision in collision detection and efficiency alongside the use of acceleration structures.

LITERATURE REVIEW

The following review of the literature focuses on research work done in the context of position-based dynamics, cloth simulation and usage of the superellipsoid shape.

Position-based dynamics and cloth simulation

Position-Based methods are especially useful in interactive environments, because they are stable and provide great controllability. For a more in-depth explanation of this topic the reading of (Bender et al. 2014) is recommended.

The simulated object can be considered as a system of particles and its equilibrium state defined through geometric constraints. Thus, an object is represented by a set of N particles and M constraints. Each particle i contains three values, namely: mass (m_i), position (x_i) and velocity (v_i). Given a time step Δt, the algorithm for the simulation is as in Figure 1, where p_i is the new position of the particle i.

In line 5-6, permanent exterior forces such as gravity or wind are used to update the velocities and positions through a time integration operation. These new locations pi are only used as predictions for the collision detection step in line 7, from which a set of constraints is generated. Then, the solver (8-10) utilizes the information from the system and collision constraints to iteratively project the predicted positions to new corrected locations. Lastly, the corrected positions are used to update the finals velocities and positions.

```
1:  for all vertices i do
2:      initialize x_i = x_i^0, v_i = v_i^0, w_i = 1/m_i
3:  end for
4:  loop
5:      for all vertices i do v_i ← v_i + Δt w_i f_ext(x_i)
6:      for all vertices i do p_i ← x_i + Δt v_i
7:      for all vertices i do genCollConstraints(x_i → p_i)
8:      loop solverIteration times
9:          projectConstraints(C_1, ..., C_{M+M_Coll}, p_1, ..., p_N)
10:     end loop
11:     for all vertices i do
12:         v_i ← (p_i − x_i)/Δt
13:         x_i ← p_i
14:     end for
15:     velocityUpdate(v_1, ..., v_N)
16: end loop
```

Figure 1. PBD algorithm (Bender et al. 2014).

In (Müller et al. 2010) the authors proposed a way to create convincing high-resolution cloth behavior on top of coarse meshes based on the observation that global dynamic behavior of cloth and wrinkle formation can be separated. Therefore, dynamic simulation and collision handling is performed on a low-resolution mesh while detail generation is handled on a high-resolution mesh attached to the coarse mesh.

Strain limiting is an important topic in of cloth simulation because real-time applications use a low solver iteration count which yields stretchy cloth. What a strain limiting method does is to constrain the overall stretch of the cloth below a certain threshold in a separate pass executed before or after the regular cloth solver. In most cases, this pass moves the positions of vertices directly. Therefore, most strain limiting methods fall under the category of position-based methods (Bender et al. 2014). Provot (Provot 1995) was among the first to use strain limiting in the context of cloth simulation. He performs a single iteration through all cloth edges after a force based solver and project adjacent particles so that the edge length does not exceed the stretch limit. Later, (Desbrun et al. 1999) and (Bridson et al. 2003) extended this strain limiter method, but with multiple iterations through all edges. A surprisingly simple and robust approach called Long Range Attachments presented by (Kim et al. 2012) exploits the fact that over-stretching almost always occurs when cloth is attached. They apply unilateral attachment constraints to attach all vertices to one or more attachment points directly, instead of relying on error propagation of the solver from these attachments to the rest of the cloth object preventing cloth from getting stretched. Position-based techniques are also used in (Somasundaram 2016) as a post simulation effect, to provide artists a simple and fast way to selectively detect and smooth folds in a cloth mesh.

The Graphics Processing Unit (GPU) is also commonly used for this kind of simulations. The first implementation of cloth simulation on the GPU came by the work of Green (Green 2003). Force-based simulations usually use very high-resolution cloth meshes. This kind of simulations could take

20 to 40 minutes per frame for 2M triangles on the CPU (Selle et al. 2009), even with multi-core parallelism. Utilizing the GPU computational power brought down these times, as reported in (Tang et al. 2013), to 2 to 3 minutes per frame for a 2M triangle mesh.

Contact Detection

Contact geometries

The representation of an object, used for collision detection, is often called contact geometry. This geometric description is not necessarily equivalent to the visual representation of the object, which is ruled by polygonal meshes in 3D environments. Meshes hold convenient properties for graphics rendering, however, they can only faithfully describe faceted objects and any smooth curve must be approximated by triangulation. For contact detection this is undesirable, as collision detection algorithms become slower and memory usage higher, the more vertices and triangles are used. Geometric shapes may also be represented implicitly, by an implicit function, which defines the surface, $F(x, y, z) = 0$ (1). The implicit function is also called an inside-outside function, because its evaluation states if a point is either inside, outside or on the surface itself. For points inside the surface the result is less than 0, while points outside the surface evaluates as greater than 0.

Superellipsoids

Smoothness and convexity (Figure 2) are two important geometrical properties for collision detection between superellipsoids. A surface is said to be smooth if it can be differentiated, up to a desired order, over its entire domain. Thus, smooth surfaces do not present sharp edges or peaks, and because they are differentiable at any point, the surface normals and tangents are always defined.

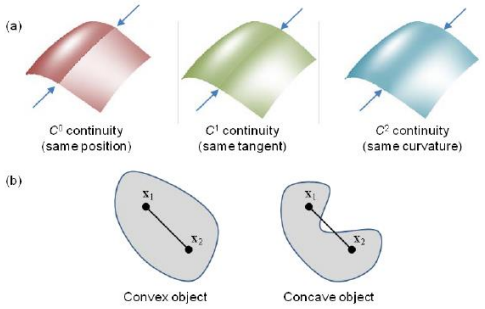

Figure 2. (a) Smoothness and (b) convexity of surfaces (Lopes 2013).

The superellipsoid is a generalization of the ellipsoid proposed in (Barr 1981) and it is defined by its implicit function:

$$F_{SE}(x,y,z) = \left(\left|\frac{x}{a_1}\right|^{\frac{2}{\epsilon_1}} + \left|\frac{y}{a_2}\right|^{\frac{2}{\epsilon_1}}\right)^{\frac{\epsilon_1}{\epsilon_2}} + \left|\frac{z}{a_3}\right|^{\frac{2}{\epsilon_2}} \qquad (2)$$

where $a1$, $a2$, $a3 > 0$ are the radii dimensions of the shape in the x, y and z directions of the local Cartesian coordinates, and $\epsilon 1, \epsilon 2 \in]0,2[$ are the roundness factors for the xOy plane and z axis, respectively. The range between 0 and 2 of the roundness exponents, ensures the resulting shapes are convex. The inside-outside function of the superellipsoid in its canonical form is: $F_{SE}(x, y, z) - 1 = 0$ (1).

Because the superellipsoid's surface is C^2 continuous, the surface normal direction, $\mathbf{n}_{SE} = \nabla F = [\mathrm{n}_x\ \mathrm{n}_y\ \mathrm{n}_z]^{\mathrm{T}}$, can be readily computed by taking the gradient of F_{SE}:

$$\mathbf{n}_{SE}(x,y,z) = \begin{bmatrix} \frac{\mathrm{sign}(x)}{a_1}\left|\frac{x}{a_1}\right|^{\frac{2}{\epsilon_1}-1}\lambda(x,y,z) \\ \frac{\mathrm{sign}(y)}{a_2}\left|\frac{y}{a_2}\right|^{\frac{2}{\epsilon_1}-1}\lambda(x,y,z) \\ \frac{\mathrm{sign}(z)}{a_3}\left|\frac{z}{a_3}\right|^{\frac{2}{\epsilon_2}-1} \end{bmatrix} \qquad (2)$$

with $\lambda(x,y,z) = \left(\left|\frac{x}{a_1}\right|^{\frac{2}{\epsilon_1}} + \left|\frac{y}{a_2}\right|^{\frac{2}{\epsilon_1}}\right)^{\frac{\epsilon_1}{\epsilon_2}-1}$. The radial distance of a given point in space, $\mathbf{x}_{SE} = [\mathrm{x}_x\ \mathrm{x}_y\ \mathrm{x}_z]^{\mathrm{T}}$, to the surface of the superellipsoid, i.e., the Euclidean distance from the surface to the point along its radial direction, can be obtained analytically as in (Jaklič et al. 2000). A slightly modified version to allow negative distance is:

$$d_{Radial}(\mathbf{x}) = \|\mathbf{x}\|\left(1 - F_{SE}(\mathbf{x})^{-\frac{\epsilon_2}{2}}\right) \qquad (3)$$

A superellipsoid can also be represented parametrically using angle-center parameters (Wellmann et al. 2008):

$$\mathbf{S}_{SE}(\varphi_1,\varphi_2) = \begin{bmatrix} \mathrm{sign}(\cos\varphi_1\ \cos\varphi_2)a_1|\cos\varphi_1|^{\epsilon_1}|\cos\varphi_2|^{\epsilon_2} \\ \mathrm{sign}(\sin\varphi_1\ \cos\varphi_2)a_2|\sin\varphi_1|^{\epsilon_1}|\cos\varphi_2|^{\epsilon_2} \\ \mathrm{sign}(\sin\varphi_2)a_3|\sin\varphi_2|^{\epsilon_2} \end{bmatrix} \qquad (4)$$

With $-\pi \le \varphi_1 < \pi$, $-\frac{\pi}{2} \le \varphi_2 \le \frac{\pi}{2}$. This representation is useful for the visualization of the shape, as a polygonal mesh can easily be generated from these parameters (Figure 5 & 6).

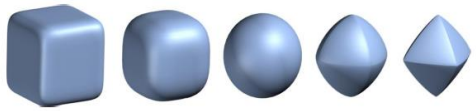

Figure 3 Superellipsoids with varying exponents ($\epsilon_1 = \epsilon_2 = \epsilon$). From left to right ($\epsilon$): 0.3, 0.65, 1.0, 1.35, 1.7 (Gonçalves 2015).

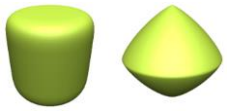

Figure 4 Two examples of superellipsoids with non-uniform exponents. From left to right: (ϵ_1=1.0, ϵ_2=0.3), (ϵ_1=1.0, ϵ_2=1.6).

Broad-phase and narrow-phase

Collision detection methods can be computationally expensive when the simulation involves a high number of bodies. The performance of the application can rapidly degrade, if every possible pair of contacts needs to be tested for collision, even when the objects are not close to each other. A solution to this problem consists in a form of culling, to minimize the total number of tests. A traditional approach called Space Partitioning (Ericson 2004), is to divide the simulated space into hierarchically smaller regions, considering only the collision pairs that share the same region. Another form of collision culling is the use of so-

called bounding volumes, which consists in employing simpler geometric primitives to fully enclose the simulated objects and perform proximity queries on these first. Proximity queries between these primitives are usually very fast, effectively accelerating the overall collision detection. The culling process is called the broad-phase, leading to a set of contact pairs known to be, at least, very close to each other. Thereafter, the more complex collision detection methods are utilized on the designated narrow-phase.

Related work of contact detection and modeling with ellipsoidal surfaces

Most collision detection algorithms involving smooth convex surfaces, including ellipsoids and superellipsoids, rely on the common normal concept. This concept states that given two points, *P* and *Q* on two C^1 continuous surfaces respectively, the normal direction of each surface at those points must be the same and have the same direction of the distance vector between P and Q (Figure 5).

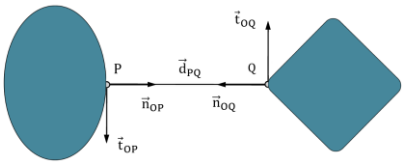

Figure 5. Orthogonal and collinear vector relationships that define the common normal concept.

In (Chakraborty et al. 2008), an interior point approach is used to solve an optimization problem based on the common normal concept and compute the minimum distance between implicit convex surfaces in general, with computation times of around 1 ms for a distance query. (Pazouki et al. 2012) propose a three-level parallelizable algorithm for large scale multibody systems, using ellipsoids. The top level is a space partitioning method, followed by a simple yes or no intersection query between a pair of ellipsoids. On the lowermost level, an optimization problem based on the common normal concept method of (Wellmann et al. 2008), is solved to compute the minimum distance points.

Superellipsoids were used in (Moustakas et al. 2007), to accelerate mesh-mesh collisions. The object's mesh enclosed by a superellipsoid bounding volume. Afterwards, the distance between the SE and the mesh is projected to the superellipsoid's surface and stored, effectively creating a distance field. These are pre-calculated, so that during the simulation, collision detection is performed using the superellipsoid and the distances information, instead of triangle per triangle tests. A mathematical framework for efficient and accurate distance computation between two smooth convex surfaces is presented in (Lopes et al. 2010), which relies on the common normal concept and the Householder formula to compute orthonormal sets of 3D vectors given a surface normal vector (Lopes et al. 2013). Unlike other popular methods, the proposed contact methodology does not resort to optimization problems. The work of Lopes et al. was later extended in (Gonçalves 2015). A superellipsoid-cloth collision detection was proposed. The implicit function of the superellipsoid is evaluated at each vertex of the cloth's mesh. AABBs were used to avoid performing this test for every vertex of the mesh and a force is then applied to intersecting vertices with the direction of the superellipsoid's surface normal. Even though the results are visually convincing, using the surface normal can be very unprecise, unless the vertex is exactly on the superellipsoid's surface. This method also does not provide any information on the minimum distance or nearest contact point. Finally, the author presented a sketch-based application for 3D modeling of articulated bodies, using superellipsoids as contact volumes for the different body parts.

Superellipsoids are investigated and applied with positive results across diverse fields of study, from geomechanics (Zhao et al. 2017) and biomechanics (Lopes et al. 2015; Lopes et al. 2016) to robot grasp planning (Gonçalves et al. 2017). In general, the state-of-the-art focuses on accuracy controlled simulations. However, the applicability of this analytical shape in less accuracy-dependent interactive applications, is scarce among the literature. Applications like the Multibody Sketcher (Gonçalves 2015) could increase the appeal of using superellipsoids in this spectrum of interactive simulations and although the author applied it in cloth simulation, the solutions are still very incomplete.

CONTACT DETECTION ALGORITHM

This work proposes the use of a type of implicit SCS, namely superellipsoid, as a modelling alternative to meshes in physical simulations with deformable or rigid models, having cloth as a case study. Therefore, an algorithm for fast collision detection between a superellipsoid and an arbitrary vertex is presented, based on the implicit surface normals of the analytical shape. This approach takes inspiration on the method of Newton-Raphson (Ypma 1995), used in numerical analysis, for root-finding. It works by taking a first initial guess value which is then utilized to find successfully better approximations to the roots of a real-valued function. In an equivalent way, the proposed algorithm starts by finding an initial guess point on the surface of the superellipsoid, which is then used to iteratively converge to the correct point of minimum distance to the intersecting vertex. Each iteration's point and the corresponding surface normal is used to compute the next potential collision point. With any iterative method, there are control variables to stop the cycle, in this case, either a given error tolerance is achieved or the maximum number of iterations allowed is reached, breaking the loop.

The presented collision detection algorithm supports signed distance for vertices inside the superellipsoid and is intended for use in a narrow-phase, after a broad-phase culling of the colliding object's vertices. The method's inputs are:

- $\mathbf{x}$ – the intersecting point in local coordinates of the SE;
- $\mathbf{p}$ – the collision point in local coordinates of the SE;
- $\mathbf{n}$ – the contact normal from p to x;
- d – the minimum distance;

- ε_{min} – the error tolerance;
- N – the maximum number of iterations.

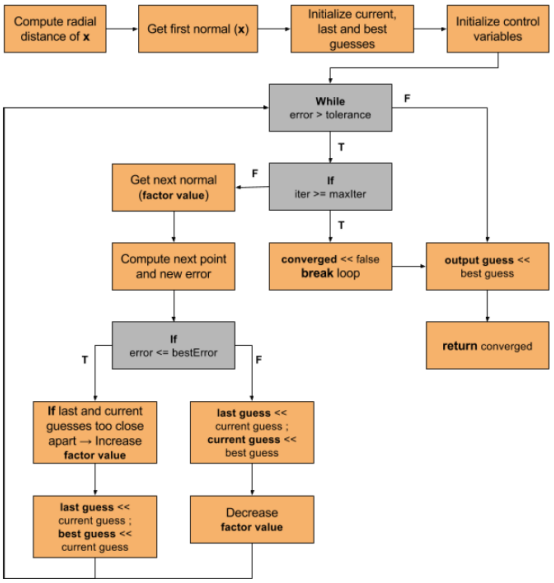

Figure 6. Overview of the contact detection algorithm flow.

Guess Content and Error Calculation

Each iteration's guess is comprised of four fields:

- collision point (**p**) – the nearest point on the surface of the superellipsoid;
- normal ($\mathbf{n}_{\mu}$) – the vector from the collision point to the intersecting point;
- surface normal ($\mathbf{n}_{SE}(\mathbf{p})$) – the implicit normal on this point of the surface;
- distance (d) – the Frobenius* norm of the guess normal.

* The Frobenius norm, also called the Euclidean norm: is a matrix norm of a *mxn* matrix **A** defined as the square root of the sum of the absolute squares of its elements.

At the end of every iteration, the absolute error of the current guess is calculated and compared to the best, i.e., the smallest error achieved so far. The computation of this error is based on a smoothness property of the superellipsoid, the C^2 continuity. Since there are no sharp edges, a normal to the surface can always be derived, therefore, the direction of collision with any point is always a surface normal. That being said, if the correct nearest point on the surface **p**, to an arbitrary vertex **x**, would be projected along its surface normal with length equal to the correct minimum distance *d*, the resultant point would be equal to **x** and the absolute error, i.e., the Euclidean distance between them, zero. Accordingly, the error of each iteration is given by,

$$\varepsilon = \| (\mathbf{p} + d\ \mathbf{n}_{SE}(\mathbf{p})) - \mathbf{x}) \|_F \quad (7)$$

This formulation works in the same way for a vertex inside the superellipsoid, using a signed negative distance *d*. The error threshold for stopping the distance iteration is adjustable and passed as input. Although the proposed algorithm was developed with a precision of 1e-3 in mind, that is, 1 mm in a unit system of meters. Considering that the algorithm is purely geometrically, it is not possible to compute the more commonly used absolute error, relative to the correct nearest point on the surface, as this can only be computed numerically. If the latter is desired, the error from Equation (7) can serve as indication for, at least, the same order of precision.

Collision Point Computation

Actually, every new collision point calculation is in fact a normal computation, i.e., the normal vector which subtracted to the intersecting vertex gives off the new point on the surface. In Figure 7, an intersecting vertex, represented by a small orange sphere is shown. On this configuration, the radial direction (yellow) goes from the center of the superellipsoid to the vertex. The section of this line exterior to the surface is considered the mentioned normal, with length equal to the radial distance of the vertex. Thus, the collision point on the surface is computed as $\mathbf{p} = \mathbf{x} - d\ \mathbf{n}_{\mu}$ (8), with $\mathbf{n}_{\mu}$ previously normalized. To guarantee that the collision point belongs to the surface, every resultant point is afterwards projected to the surface, along its radial direction: $\mathbf{p} = \mathbf{p} - d_{Radial}(\mathbf{p}) \left(\frac{\mathbf{p}}{\|\mathbf{p}\|_F} \right)$ (9). The guess normal and distance must then be corrected, to maintain the algorithm stable and precise. This step is straightforward, the normal is just the vector from **p** to **x,** $\mathbf{n}_{\mu} = \mathbf{x} - \mathbf{p}$, and the distance is the Frobenius norm of the normal, $d = \|\mathbf{n}_{\mu}\|_F$.

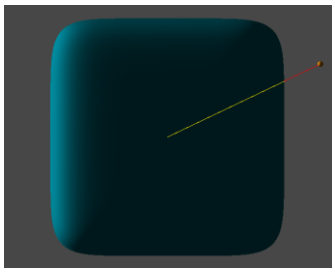

Figure 7. Isometric view of the radial direction (yellow) and the radial distance: length of the exterior section (red).

First Guess Computation

Mirroring the surface normal

The first approach mirrors the surface normal from the radial intersection point (Figure 8), simply subtracting it to the vertex. The corresponding surface normal on the new point(blue) is very close to the correct solution, making this approach adequate for zones with little variance of curvature.

An important step before every new collision point computation is to adjust the distance value, so that the new point is not far from the surface. This problem is reduced to a 2D right triangle trigonometry problem. From the definition of cosine, $\cos\theta = adjacent/hypotenuse$ $(=)\ adjacent = hypotenuse * \cos\theta$, we can derive the adjusted distance, or adjacent triangle side, from the current distance and the dot product between the normal and the surface normal: $d = d\ (\mathbf{n}_{\mu} \cdot \mathbf{n}_{SE}(\mathbf{p}))$ (10). Note that, both the normal and surface normal vectors should be normalized.

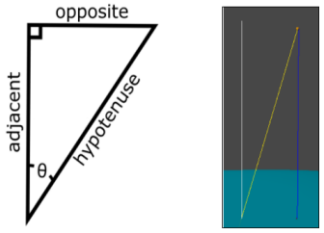

Figure 8. Collision normal (yellow) as the hypotenuse, and surface normal (white) as adjacent side. The adjusted distance (length of blue surface normal).

Weighted average of normals

The second approach is based on an average of the radial direction and the surface normal, and is used when the first method fails to compute a good guess. To estimate this, after calculating the potential point with the first method, the cosine of the angle between the, original and new (white and blue lines in Figure 8, respectively), surface normals is measured. The potential point is considered good enough if the cosine value exceeds 0.97, 1.0 meaning the normals are parallel. When the threshold is not reached, it means the nearest point is in a more curved region of the object, thus a safer first normal is required. For this reason, a weighted average of the radial direction and the surface normal is computed, with 80% and 20% weight, respectively. An example can be depicted in Figure 9, where the averaged normal (cyan) is subtracted to the vertex, concluding in the new collision point.

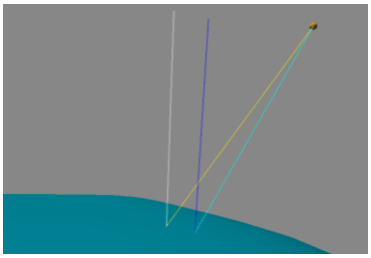

Figure 9. First point computed from the weighted averaged normal. Original and new surface normals, in white and blue, respectively. Original and new normal directions, in yellow and cyan, respectively.

Iterative Guess Computation

Standard normal computation

The standard normal computation is similar to the weighted average in the previous subsection, with the difference that the weights are regulated by a control variable, $\lambda \in [0,1]$. The unit average $\mathbf{v}_{avg}$, i.e. with length equal to 1, of two vectors $\mathbf{v}_0$ and $\mathbf{v}_1$, can be obtained by normalizing their sum, $\mathbf{v}_{avg} = \|\mathbf{v}_0 + \mathbf{v}_1\|$. Since the average is always normalized, the weighted average is defined by the weights proportions relative to each other. In the sum, the weights are both 1.0, implicitly. Based on the previous formulation, the standard normal computation of each iteration is as follows,

$$\mathbf{n}_{\boldsymbol{\mu}} = (1-\lambda)\,\mathbf{n}_{\boldsymbol{\mu}} + \lambda\,\mathbf{n}_{SE}(\mathbf{p}), \quad \mathbf{n}_{\boldsymbol{\mu}} = \frac{\mathbf{n}_{\mu}}{\|\mathbf{n}_{\mu}\|_F} \qquad (5)$$

Note that, while one of the weights decreases, the other one increases, with the sum of the weights always equaling 1. In zones with abrupt changes of curvature where the algorithm starts to diverge, λ is decreased to adapt to the situation.

Stabilizing normal computation

The major difficulty of this algorithm happens at the peak changes of curvature. The problem is that even a minimal change in the collision point can have a drastic change on the direction of the surface normals. The second approach attempts to resolve this problem by taking advantage of two consecutive guesses, when the last iteration's guess is divergent, this is, a guess whose error failed to improve the best guess so far. Both last and best normals are averaged, $\mathbf{n}_{\mu_{avg}} = \|\mathbf{n}_{\mu_{best}} + \mathbf{n}_{\mu_{last}}\|_F$ (6), as well as the surface normals $\mathbf{n}_{SE\,avg} = \|\mathbf{n}_{SE}(\mathbf{p}_{best}) + \mathbf{n}_{SE}(\mathbf{p}_{last})\|_F$ (7), to create a hypothetical point between the two. This is then used to compute the new normal in a similar way to the standard one:

$$\mathbf{n}_{\boldsymbol{\mu}} = (1-\lambda)\,\mathbf{n}_{\mu_{avg}} + \lambda\,\mathbf{n}_{SE\,avg}, \quad \mathbf{n}_{\boldsymbol{\mu}} = \frac{\mathbf{n}_{\mu}}{\|\mathbf{n}_{\mu}\|_F} \qquad (14)$$

Another advantage of this approach is the fact that divergent guesses can be useful and not just wasted computation time.

Factor Value Adjustment

In order to accommodate a bigger variance of curvature, the size of each iteration step should decrease accordingly. The variable, $\lambda \in [0,1]$, controls this size by managing the weights of the averaged normal of Equation (11) and Equation (14). The initial value for this variable is 0.5, which produces an equally weighted average. On the boundaries, 0 and 1, the resultant averaged normal is equal to the normal vector and the surface normal, respectively. The former generates a null step, because the new point would be equal to the last. And the latter is analogous to mirroring the surface normal.

In every step, after the error calculation, the new error is compared to the best error so far. If the first is worse, the guess is considered divergent and reset to the best guess values. In which case, the factor value is decreased. Often, once the algorithm stabilizes, it can be slow to converge because λ is too small. Likewise, when the new guess is better compared to the best so far, the value is increased, although with a few conditions: (1) The new guess was obtained through the standard normal computation only; (2) Both guesses are close in terms of surface normals and collision points.

Algorithm Output

Once the error tolerance or the maximum number of iterations is reached, the iterative loop stops and the output variables are prepared using the best guess found. These are:

- **p** - nearest point in SE local coordinates;
- **n** - contact normal to the vertex in SE local coordinates;
- *d* - minimum distance between the SE and the vertex.

The output provides all the necessary information to correctly react to a collision. The nearest point is guaranteed to be on the surface of the superellipsoid and the signed minimum distance provides penetration depth information.

RESULTS AND DISCUSSION

A benchmark study to compare two contact geometry representations (polygonal meshes and superellipsoids) in terms of computational efficiency and geometric accuracy, was performed in FCL. The FCL's mesh-based method was used for comparisons. Two case studies, performed with the PBD library, showcases an application of the proposed method to collision detection in a cloth simulation and rigid bodies simulation. The contact detection method presented computes de minimum distance between an arbitrary point and the superellipsoid. Therefore, one contact pair consists of one point and one superellipsoid, and the application of the method to this pair is considered a distance query. The point of a contact pair is analogous to a mesh vertex, so the naming of point or vertex can be used interchangeably.

The machine used to run this project's tests is a desktop (Intel® Core™ i5-6600 @ 3.30 to 3.90 GHz, NVIDIA® GeForce® GT 960 and 16 GB of RAM), running on Windows 10 x64.

Accuracy and Efficiency

To evaluate these metrics, various configurations of the superellipsoid (implicit and mesh) and the hypothetical intersecting vertex were considered. For the vertex, two relative positions are considered, one outside of the surface and another slightly inside. For the superellipsoid, the seven pre-defined shapes of Fig. 3 and Fig. 4 were used, providing a good distribution along the domain of the roundness exponents. To faithfully test the different curvatures posed in each superellipsoid, a battery of 10,000 points is generated all along the surface, and the averaged computation times and geometric errors are measured. Also, for the comparison benchmarks, a mesh resolution of 272 vertices was chosen to provide a good balance between speed and precision and is around the maximum resolution advised in game engines, for performance reasons. For the implicit method, an error tolerance of 1e-3 was used as the precision target and the maximum number of iterations was set to 30, as a safeguard.

Mesh vs Implicit Accuracy

Table 1. Geometric error (average and standard deviation) on 7 different shapes (ϵ_1; ϵ_2), taken from 10 000 queries outside the surface (+0.05m). Additionally, the success rate of reaching the precision target of 1e-3, and the error (average) achieved from unsuccessful queries.

Representation	Shape	Average error (mm)	σ of error (mm)	Convergence (%)	Average error when diverge (mm)
Mesh(FCL)	(0.3; 0.3)	6.89	3.99	10.52	7.64
Mesh(FCL)	(0.65; 0.65)	5.91	2.78	5.44	6.23
Mesh(FCL)	(1; 1)	5.30	1.98	5.28	5.58
Mesh(FCL)	(1.35; 1.35)	4.49	2.05	6.28	4.76
Mesh(FCL)	(1.7; 1.7)	3.94	3.53	19.56	4.75
Mesh(FCL)	(1; 0.3)	6.39	3.29	7.72	6.90
Mesh(FCL)	(1; 1.6)	3.83	2.04	6.32	4.06
Implicit	(0.3; 0.3)	0.41	0.27	100	0
Implicit	(0.65; 0.65)	0.59	0.21	100	0
Implicit	(1; 1)	2.11E-14	2.54E-14	100	0
Implicit	(1.35; 1.35)	0.46	0.19	100	0
Implicit	(1.7; 1.7)	0.73	0.55	86	2.50
Implicit	(1; 0.3)	0.40	0.28	100	0
Implicit	(1; 1.6)	0.34	0.24	100	0

Table 2. Geometric error (average and standard deviation) on 7 different shapes (ϵ_1; ϵ_2), taken from 10 000 queries inside the surface (-0.015m). Additionally, the success rate of reaching the precision target of 1e-3, and the error (average) achieved from unsuccessful queries.

Representation	Shape	Average error (mm)	σ of error (mm)	Convergence (%)	Average error when diverge (mm)
Mesh(FCL)	(0.3; 0.3)	1.09	0.79	56.88	2.01
Mesh(FCL)	(0.65; 0.65)	1.48	1.60	67.29	3.82
Mesh(FCL)	(1; 1)	1.88	2.31	66.71	5.00
Mesh(FCL)	(1.35; 1.35)	0.66	0.50	79.48	1.75
Mesh(FCL)	(1.7; 1.7)	0.48	0.34	87.64	1.61
Mesh(FCL)	(1; 0.3)	0.96	0.70	63.52	1.92
Mesh(FCL)	(1; 1.6)	0.59	0.37	85.92	1.66
Implicit	(0.3; 0.3)	0.21	0.11	100	0
Implicit	(0.65; 0.65)	0.06	0.02	100	0
Implicit	(1; 1)	2.10E-14	2.46E-14	100	0
Implicit	(1.35; 1.35)	0.06	0.03	100	0
Implicit	(1.7; 1.7)	0.19	0.21	96.08	1.37
Implicit	(1; 0.3)	0.15	0.09	100	0
Implicit	(1; 1.6)	0.09	0.09	100	0

The average error results clearly show the impact of lower resolutions on the mesh representation. Because every curve has to be discretized, the first three shapes and the 6th, which are the most rounded, reveal the greater loss in precision. While the two most faceted, 5th and 7th, have better results. The implicit representation demonstrates globally better results than meshes. The shapes with more abrupt changes in curvature are the most challenging for the implicit method, with the 5th shape resulting in some unsuccessful queries. However, the average error of these queries shows the results are close to the targeted precision, and still better than the mesh equivalent on the outside version. The average standard deviations directly exhibit the rate of curved vs flat regions of the surface. Shapes with a greater amplitude of curvatures show the greater deviation values, diminishing towards the middle line, the sphere shape, where the curvature is consistent throughout the surface.

Mesh vs Implicit Efficiency

Table 3. Computation times (total and average) on 7 different shapes (ϵ_1; ϵ_2), taken from 10 000 queries outside the surface (+0.05m). Additionally, the speedup of the implicit version against the mesh-based method.

Shape	Mesh total time (ms)	Implicit total time (ms)	Mesh average time (µs)	Implicit average time (µs)	Speedup
(0.3; 0.3)	350.16	26.27	35.02	2.63	13.33
(0.65; 0.65)	290.52	17.63	29.05	1.76	16.48
(1; 1)	325.72	12.34	32.57	1.23	26.39
(1.35; 1.35)	303.72	19.84	30.37	1.98	15.31
(1.7; 1.7)	336.65	58.63	33.66	5.86	5.74
(1; 0.3)	265.10	23.29	26.51	2.33	11.38
(1; 1.6)	392.70	21.25	39.27	2.12	18.48

Table 4. Computation times (total and average) on 7 different shapes (ϵ_1; ϵ_2), taken from 10 000 queries inside the surface (-0.015m). Additionally, the speedup of the implicit version against the mesh-based method.

Shape	Mesh total time (ms)	Implicit total time (ms)	Mesh average time (µs)	Implicit average time (µs)	Speedup
(0.3; 0.3)	323.64	18.01	32.36	1.80	17.97
(0.65; 0.65)	236.48	16.98	23.65	1.70	13.93
(1; 1)	243.48	11.31	24.35	1.13	21.53
(1.35; 1.35)	254.48	16.33	25.45	1.63	15.58
(1.7; 1.7)	264.25	25.17	26.42	2.52	10.50
(1; 0.3)	281.01	22.66	28.10	2.27	12.40
(1; 1.6)	334.71	21.67	33.47	2.17	15.44

In terms of efficiency, this is where the implicit representation really shines, with significant speedups across every shape. The results indicate similar responses from the two methods to the different shapes: shapes with more abrupt changes in curvature are the slowest to compute.

Performance in Cloth Simulation

In order to evaluate the performance of the proposed algorithm in the simulation of cloth or deformable models in general, the superellipsoid shape was implemented within the PBD library, which provides a cloth simulation based on position-based dynamics. A case study of a tablecloth falling onto a superellipsoid (Figure 10) was studied and is presented here.

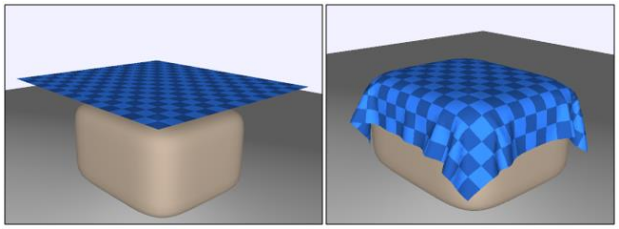

Figure 10. A tablecloth, with 6254 triangles and 3240 vertices, falling onto a superellipsoid ($\epsilon_1 = 0.4$; $\epsilon_2 = 0.3$) with radii dimensions of 3.5, 3.0, 2.0 for the x, y and z directions of the local Cartesian coordinates.

The standard graphical systems performance tests were performed. The metrics of interest to this work are the framerate and the timings of the most important steps of the PBD algorithm in Figure 1. The PBD library implements a slightly modified version, where the solver constraint projections are split into two, position constraints projections and velocity constraint projections. The collision detection step is performed between the two and the resultant constraints are solved on the velocity step. Therefore, the timings will be broken down as follows:

- Time Integration;
- Position Constraint Projections;
- Velocity Constraint Projections;
- Collision Detection;
- Velocity Update;
- Total Simulation Time.

Note that the collision detection step includes a broad-phase performed by the PBD library on the cloth mesh, based on a hierarchy of sphere bounding volumes. Some parameters of note: the superellipsoid mesh used for visualization 702 vertices; the implicit method was set with a precision target of 1e-3 and 10 maximum iterations; the collision distance tolerance was set to 0.025 m. The simulations ran for a total of 10 seconds with 2 different cloth resolutions (3240 and 6460 vertices), while measuring the frames per second (Figure 11) and the average timings mentioned above (Table 5; Table 6).

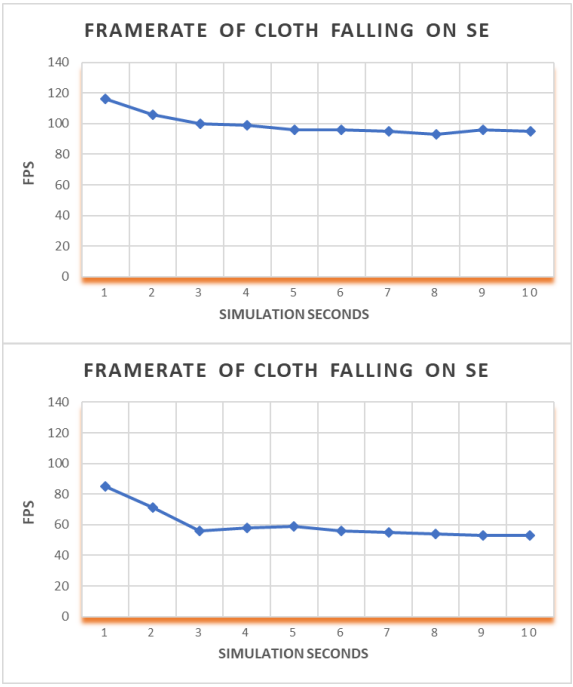

Figure 11. Framerate of tablecloth (3240 on top, 6460 on bottom: vertices) falling onto a superellipsoid (10s of simulation). The cloth starts to collide with the superellipsoid between seconds 2 and 3.

Table 5. Average time of the PBD algorithm most important steps. Cloth mesh with 3240 vertices.

Step	Average Time (ms)
Time Integration	0.025
Position Constraints	2.423
Velocity Constraints	0.914
Collision Detection	4.170
Velocity Update	1.540
Total Timestep	9.170

Table 6. Average time of the PBD algorithm most important steps. Cloth mesh with 6460 vertices.

Step	Average Time (ms)
Time Integration	0.066
Position Constraints	5.324
Velocity Constraints	1.544
Collision Detection	7.509
Velocity Update	1.599
Total Timestep	16.231

For the cloth resolution used in the first test, the simulation has more than enough performance for real-time interactive requirements, even with most of the vertices constantly in contact. Decreasing the number of iterations or the target precision of the implicit method had no visible effect on the framerate, suggesting that the collision algorithm is sufficiently efficient and precise, and the higher computation time is due to the sheer number of vertices in contact. To prove this, another simulation was run, this time with almost twice the cloth resolution, 6460 vertices. It was expected that the timings would be roughly two times the values of the first test. The results are indeed close to what was expected but with some interesting facts. The collision detection timings

were slightly better than expected, while the timings of the position constraints solver step were worse. This confirms the implicit contact method is sufficiently efficient for collision detection with cloth models alongside broad-phase culling.

Performance in Rigid Bodies Simulation

Since the proposed contact detection method works with any vertex-based model, the simulation of rigid bodies is also investigated. A case study of 50 random superellipsoids falling into a rigid bowl (Figure 12), was implemented in the PBD library and is presented here. In this simulation each superellipsoid is represented by a triangular mesh with 506 vertices. Each collision query between two superellipsoids is performed by the implicit contact detection method of the second object, using the polygonal mesh of the first object as input. It starts by a simple AABB test, proceeding to a bounding volume hierarchy broad-phase culling of the mesh vertices, and finally, elementary tests with vertices are performed by the proposed contact detection algorithm of this paper. The random SEs have exponents (ϵ_1; ϵ_2) ranging from 0.1 to 1.9, and radii dimensions (a_1, a_2, a_3) between 0.5 and 0.6.

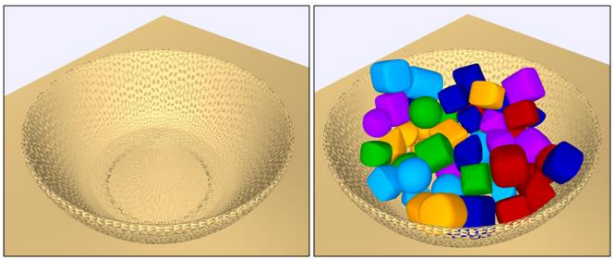

Figure 12. A chain of 50 superellipsoids with random exponents, dimensions and starting positions, fall into a bowl with 5002 vertices.

The graphical system performance was evaluated by measuring the framerate as well as the timings of the most important steps of the PBD algorithm mentioned in the previous section. Some parameters of note: the implicit method was set with a precision target of 1e-3 and 10 maximum iterations and the collision distance tolerance was set to 0.015 m. To completely let the bodies rest on the bowl, the simulations ran for a total of 15 seconds, while measuring the fps (Figure 13) and the average timings (Table 7).

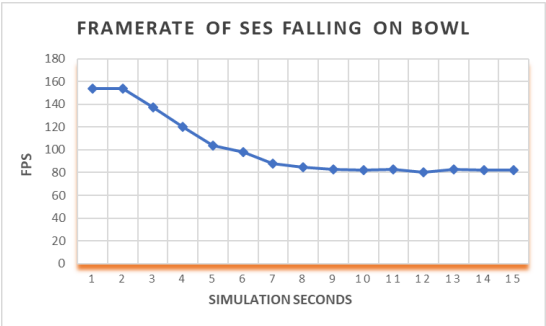

Figure 13. Framerate of 50 superellipsoids falling onto a bowl (15s of simulation). The SEs start to collide with the bowl between seconds 2 and 3, and stabilize around second 11.

Table 7. Average time of the PBD algorithm most important steps. Case study of chain of 50 random superellipsoids falling into a bowl.

Step	Average Time (ms)
Time Integration	0.018
Position Constraints	0.001
Velocity Constraints	0.119
Collision Detection	5.520
Velocity Update	3.257
Total Timestep	8.922

In the case of rigid body simulation, the proposed algorithm proved capable of interactive framerates as well. The load distributions of the PBD algorithm steps are very different compared to the cloth case study, but not surprising. Since the objects are rigid, constraints are solved for each object as a whole and not for each vertex like with cloth, so the solver does not have to do much work. The higher load in the velocity update step derives from the sudden changes in velocity when the superellipsoids hit the bowl and the erratic collisions until they rest. Collision detection has the highest load in the timestep due to the constant collisions, which maintain the superellipsoids packed together.

CONCLUSIONS

Real-time interactive applications imply strict performance constraints, which can often be difficult to meet in physical simulations, where geometric accuracy is of paramount importance. However, polygonal meshes are still standardly used for contact detection, due to their facility in approximating any complex 3D object. In this paper, superellipsoids and their implicit representation were studied and applied to physical simulations with mesh-based models. Since many real-life objects can be modelled with these analytically defined geometries, this representation can become an efficient and less memory-demanding way to model our scenes. A superellipsoid-vertex contact detection algorithm was implemented in FCL, and benchmarked against the mesh-based equivalent method, in terms of geometric accuracy and efficiency. The implicit superellipsoid version performed considerably faster than FCL's mesh minimum distance query, with measured speedups from 5.7 to 26.4 times, and with greater precision across every superellipsoid configuration. Two case-studies were performed in the PBD library, with the objective of studying the performance and applicability of the superellipsoid to deformable and rigid body simulations. The first was a cloth model (3k and 6k vertices) falling onto a superellipsoid, with most of its vertices in contact; and the second was a sequence of 50 random superellipsoids falling into a bowl until packed together. Both simulations ran with framerates around 60fps or higher for a medium-high test machine, showing that the proposed method is adequate for real-time interactive applications. With collision detection being an essential part in a multeity of fields, and materials, such as cloth, having so many applications, this solution can potentially enhance the current interactive applications and promote future research and analysis, not only in physical simulations but other areas as well.

ACKNOWLEDGMENTS

I thank all the people who helped me, in some way or another, in the conclusion of this project. I also acknowledge the financial support supported by national funds through the Portuguese Foundation for Science and Technology with reference IT-MEDEX PTDC/EEI-SII/6038/2014.

REFERENCES

1. Barr, Alan H. "Superquadrics and angle-preserving transformations." *IEEE Computer graphics and Applications* 1.1 (1981): 11-23.
2. Bender, J., Müller, M., Otaduy, M. A., Teschner, M., and Macklin, M. "A Survey on Position-Based Simulation Methods in Computer Graphics." *Computer graphics forum*. Vol. 33. No. 6. 2014.
3. Bridson, Robert, Sebastian Marino, and Ronald Fedkiw. "Simulation of clothing with folds and wrinkles." *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*. Eurographics Association, 2003.
4. Chakraborty, N., Peng, J., Akella, S., & Mitchell, J. E. (2008). Proximity queries between convex objects: An interior point approach for implicit surfaces. IEEE Transactions on Robotics, 24(1), 211-220.
5. Desbrun, Mathieu, Peter Schröder, and Alan Barr. "Interactive animation of structured deformable objects." *Graphics Interface*. Vol. 99. No. 5. 1999.
6. Ericson, C. (2004). Real-time collision detection. CRC Press.
7. Etzmuß, O., Keckeisen, M., & Straßer, W. (2003, October). A fast finite element solution for cloth modelling. In Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference on (pp. 244-251). IEEE.
8. Gonçalves, A. "Efficient Contact Detection for Game Engines and Robotics", Lisbon. Master Thesis, Instituto Superior Técnico (2015).
9. Gonçalves, A. A., Bernardino, A., Jorge, J., & Lopes, D. S. (2017). A benchmark study on accuracy-controlled distance calculation between superellipsoid and superovoid contact geometries. Mechanism and Machine Theory, 115, 77-96.
10. Green, Simon. "Stupid opengl shader tricks." Advanced OpenGL Game Programming Course, Game Developers Conference. 2003.
11. Jaklič, A., Leonardis, A., & Solina, F. (2000). Superquadrics and their geometric properties. In Segmentation and recovery of superquadrics (pp. 13-39). Springer Netherlands.
12. Johnson, Kenneth Langstreth, and Kenneth Langstreth Johnson. Contact mechanics. Cambridge university press, 1987.
13. Kim, Tae-Yong, Nuttapong Chentanez, and Matthias Müller-Fischer. "Long range attachments-a method to simulate inextensible clothing in computer games." *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 2012.
14. Lopes, DS (2013), 'Smooth convex surfaces for modeling and simulating multibody systems with compliant contact elements', PhD Thesis, Instituto Superior Técnico, Universidade de Lisboa, Portugal.
15. Lopes, D. S., Neptune, R. R., Ambrósio, J. A. and Silva, M. T. "A superellipsoid-plane model for simulating foot-ground contact during human gait." *Computer methods in biomechanics and biomedical engineering* 19.9 (2016): 954-963.
16. Lopes, D. S., Neptune, R. R., Gonçalves, A. A., Ambrósio, J. A. and Silva, M. T. "Shape Analysis of the Femoral Head: A Comparative Study Between Spherical, (Super) Ellipsoidal, and (Super) Ovoidal Shapes." *Journal of Biomechanical Engineering* 137.11 (2015): 114504.
17. Lopes, D. S., Silva, M. T., & Ambrósio, J. A. (2013). Tangent vectors to a 3-D surface normal: A geometric tool to find orthogonal vectors based on the Householder transformation. Computer-Aided Design, 45(3), 683-694.
18. Lopes, D. S., Silva, M. T., Ambrósio, J. A., and Flores, P. "A mathematical framework for rigid contact detection between quadric and superquadric surfaces." *Multibody System Dynamics* 24.3 (2010): 255-280.
19. Moustakas, K., Tzovaras, D., & Strintzis, M. G. (2007). SQ-Map: Efficient layered collision detection and haptic rendering. IEEE Transactions on Visualization and Computer Graphics, 13(1).
20. Müller, Matthias, and Nuttapong Chentanez. "Wrinkle meshes." *Proceedings of the 2010 ACM SIGGRAPH/Eurographics symposium on computer animation*. Eurographics Association, 2010.
21. Müller, M., Heidelberger, B., Hennix, M., and Ratcliff, J. "Position based dynamics." *Journal of Visual Communication and Image Representation* 18.2 (2007): 109-118.
22. Nealen, A., Müller, M., Keiser, R., Boxerman, E., & Carlson, M. (2006, December). Physically based deformable models in computer graphics. In Computer graphics forum (Vol. 25, No. 4, pp. 809-836). Blackwell Publishing Ltd.
23. Pazouki, A., Mazhar, H., & Negrut, D. (2012). Parallel collision detection of ellipsoids with applications in large scale multibody dynamics. Mathematics and Computers in Simulation, 82(5), 879-894.
24. Provot, Xavier. "Deformation constraints in a mass-spring model to describe rigid cloth behaviour." *Graphics interface*. Canadian Information Processing Society, 1995.
25. Selle, A., Su, J., Irving, G., and Fedkiw, R. "Robust high-resolution cloth using parallelism, history-based collisions, and accurate friction." *IEEE Transactions on Visualization and Computer Graphics* 15.2 (2009): 339-350.
26. Somasundaram, Arunachalam. "Selective and dynamic cloth fold smoothing with collision resolution." *Proceedings of the 2016 Symposium on Digital Production*. ACM, 2016.
27. Stumpp, T., Spillmann, J., Becker, M., and Teschner, M. "A Geometric Deformation Model for Stable Cloth Simulation." *VRIPHYS* 8 (2008): 39-46.
28. Tang, M., Tong, R., Narain, R., Meng, C., and Manocha, D. "A GPU-based Streaming Algorithm for High-Resolution Cloth Simulation." *Computer Graphics Forum*. Vol. 32. No. 7. 2013.
29. Teschner, M., Kimmerle, S., Heidelberger, B., Zachmann, G., Raghupathi, L., Fuhrmann, A., ... & Volino, P. (2005, March). Collision detection for deformable objects. In Computer graphics forum (Vol. 24, No. 1, pp. 61-81). Blackwell Publishing Ltd.
30. Wellmann, C., Lillie, C., & Wriggers, P. (2008). A contact detection algorithm for superellipsoids based on the common-normal concept. Engineering Computations, 25(5), 432-442.
31. Ypma, T. J. (1995). Historical development of the Newton–Raphson method. SIAM review, 37(4), 531-551.
32. Zhao, S., Zhang, N., Zhou, X., & Zhang, L. (2017). Particle shape effects on fabric of granular random packing. Powder Technology, 310, 175-186.