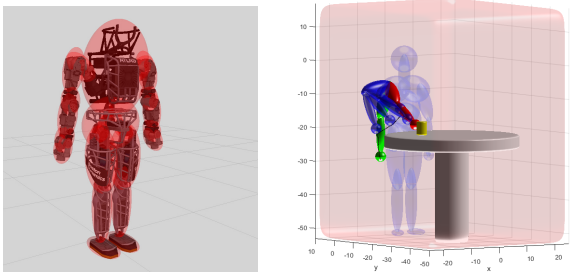
Efficient Exact Collision Detection between Ellipsoids and Superquadrics via Closed-form Minkowski Sums

Sipu Ruan[1], Karen L. Poblete[1], Yingke Li[2], Qian Lin[2], Qianli Ma[3] and Gregory S. Chirikjian*[4]

***Abstract*— Collision detection has attracted attention of researchers for decades in the field of computer graphics, robot motion planning, computer aided design, etc. A large number of successful algorithms have been proposed and applied, which make use of convex polytopes and bounding volumes as primitives. However, algorithms for those shapes rely significantly on the complexity of the meshes. This paper deals with collision detection for shapes with simple and exact mathematical descriptions, such as ellipsoids and superquadrics. These primitives have a wide range of applications in representing complex objects and have much fewer parameters than meshes. The foundation of the proposed collision detection scheme relies on the closed-form Minkowski sums between ellipsoids and superquadrics in n-dimensional Euclidean space. The basic idea here is to shrink the ellipsoid into a point and expand each superquadric into a new offset surface with closed-form parametric expression. The solutions for detecting relative positions between a point and a general convex differentiable parametric surface in both 2D and 3D are derived, leading to an algorithm for exact collision detection. To compare between exact and inexact algorithms, an accuracy metric is introduced based on the Principal Kinematic Formula (PKF). The proposed algorithm is then compared with existing well-known algorithms: Gilbert-Johnson-Keerthi (GJK) and Algebraic Separation Conditions (ASC). The results show that the proposed algorithm performs competitively with these efficient checkers.**

I. Introduction

Collision detection plays an important role in many areas such as computer aided-design (CAD), robot motion planning, computer vision, etc. Many algorithms have been proposed to make collision detection more efficient when using polyhedral objects. However these algorithms rely significantly on the complexity of the meshes representing the objects, which results in a trade-off between accuracy and efficiency. Superquadrics, with ellipsoids being a simplified version within this family of shapes, have become popular recently since they require fewer representation parameters. A real-life scenario using superquadrics is a humanoid robot trying to pick up a cup on a table while avoiding hitting objects in its trajectory. The rigid parts of the robot are encapsulated by a union of ellipsoids and the objects in the environment are enclosed by superquadrics, as shown in Fig. 1. In [1], superquadrics are used for representing different objects in an environment where a PR2 has the task of grasping objects. The major advantage of superquadrics lies in the simple mathematical expressions and varieties of shapes it can describe without meshes. This paper provides a new collision detection paradigm between an ellipsoidal and a superquadric object based on the idea of closed-form Minkowski sums [2].



(a) An Atlas humanoid robot with rigid parts being encapsulated by ellipsoids.

(b) A poly-ellipsoidal humanoid robot picking up a cup while avoiding hitting the table.

Fig. 1. An Examples of the scenarios where ellipsoids and superquadrics come to play a role in robot motion planning tasks.

A. Related Work

There are hundreds of well known algorithms for collision detection. Methods based on Bounding Volume Hierarchy (BVH) [3] use primitive shapes such as spheres, axis-aligned bounding boxes (AABB) or oriented bounding boxes (OBB) [4] to encapsulate polytopes. These methods have been proven to accelerate the collision detection by doing penetration test based on these less complex shapes. Nevertheless, when there are multiple objects in the space, it is better to use hierarchical representations like octrees [5]. There are also methods based on Euclidean distance, which use more memory in order to accelerate the search. These algorithms calculate, in advance, a map of distances between points of bounding boxes belonging to the polytopes. These maps tend to use large quantities of memory, which is the reason why they are not commonly used. In [6] and [7], some approaches based on distance with superquadrics are defined. These methods present less computational time than the regular OBB, but in the case of [6], it takes longer time to preprocess the geometry of the objects.

A shortest distance computation algorithm between two convex or non-convex polyhedra is the Lin-Canny closest feature tracking algorithm described in [8], which is the basis of several frameworks for collision checking including [9]

* Address all correspondence to this author.

[1] Sipu Ruan and Karen L. Poblete are with the Laboratory for Computational Sensing and Robotics, Johns Hopkins University, Baltimore, MD {ruansp,kpoblet1}@jhu.edu

[2] Yingke Li and Qian Lin are with Tsinghua University, Beijing, China

[3] Qianli Ma is with Aptiv Automotive, Pittsburgh, PA

[4] Gregory S. Chirikjian is with Department of Mechanical Engineering, National University of Singapore, Singapore and the Laboratory for Computational Sensing and Robotics, Johns Hopkins University, Baltimore, MD mpegre@nus.edu.sg, gchirik1@jhu.edu

and [10]. The algorithm compares edges, vertices and faces in order to find the closest points. The best application for this algorithm is checking collision while the objects move at a constant speed. The downside of this algorithm is the quantity of information stored about the surface.

The Gilbert-Johnson-Keerthi (GJK) [11], based on Minkowski formulations for convex polytopes, is an algorithm that does not use extensive quantities of memory and is comparably fast to the previous methods. GJK does not calculate the whole Minkowski Sum, but iteratively generates "simplex", a subsection of the Minkowski Sum aimed to find a subset that indicates collision between the shapes. This method is guaranteed to converge after several iterations, but is still subject to the complexity of the shapes. BVH, OBB, AABB with GJK are implemented in the Flexible Collision Library (FCL) [12] and are widely used in robotics, computer graphics, computer games, and other fields.

One method that uses the Interior Point approach is proposed in [13]. The proximity queries are calculated as intersections of implicit surfaces, and the closest distance between their points is calculated as an optimization problem. The algorithm is guaranteed to converge in polynomial time with respect to the constraints.

The algorithms discussed above have shown to perform fair enough. Nevertheless, when the objects are enclosed by bounding volumes that do not fit closely, false positives could occur. Moreover, when using hierarchical methods, the complexity of the object representation and the quantity of collision queries increase. An alternative is to use algebraic methods, like the ones presented in [14]–[16], which give exact conditions of collision detection when the objects are encapsulated by ellipsoids. Their Algebraic Separation Conditions (ASC) are characteristic polynomial equations based on the geometric parameters of the shapes, i.e. shape matrices and configurations (orientations and locations). According to the sign and real values of the roots, it is possible to determine whether two ellipsoids are separated, touching in a point or in collision. This algorithm has shown a significant success in performing collision detection between two ellipsoids. The limitation of this method is that it can only be applied between two ellipses or ellipsoids.

In [17], a method similar to ASC is presented, where a set of nonlinear equations is numerically solved using Newton-Raphson method with Jacobian matrices analytically calculated for superellipsoids. This method does not rely on polyhedron-based geometries and can be extended to other shapes, but it has the inconvenience of having some Jacobian singularities. In [18], a method based on normal vectors is proposed. It formulates collision checking as a 2-dimensional unconstrained optimization problem. The advantage of this method is that it requires less iterations in order to achieve higher accuracy. However, it is only applicable when the superellipsoids are expressed as a collection of smooth convex particles and an explicit relation between the surface points and the surface normals must be provided.

Apart from geometrically enclosing objects, superquadrics are also applied in artificial potential field methods for collision avoidance [19], where the repulsive isopotential contours around the obstacles are modeled as superquadrics.

Another method applicable to superquadrics is described in [20], based on the implicit equation of the evaluated surfaces. The contact query is expressed as a convex nonlinear constrained optimization problem to solve for the distance between the surfaces. To accelerate the process, spheres are used to enclose the objects followed by OBB. When a collision is detected, the accurate contact points are calculated.

B. Contributions

This paper presents a collision detection scheme between ellipsoidal and superquadric objects based on the closed-form Minkowski sums. The major contributions are:
(1) An exact algorithm for collision detection based on a parametric closed-form Minkowski sum expression is proposed;
(2) A collision detection accuracy metric is proposed for inexact algorithms using meshes, based on the Principal Kinematic Formula;
(3) The proposed algorithm is compared with the existing state-of-the-art algorithms, and shows competitive performance in discrete collision detection problems.

The advantages of the proposed method can be summarized as:
(1) The mathematical derivations start with only the parameters of the collision objects (i.e. semi-axes length, exponents and configurations), and performs exact detection and avoids generating meshes;
(2) The algorithm can check collisions between an ellipsoid and any convex differentiable surface embedded in Euclidean space, which gives a wide range of ways to describe objects.

The rest of the paper is organized as follows. In Section II, we first review the closed-form Minkowski Sums between an ellipsoid and any surface, and provide a concrete explicit expression for superquadrics. In Section III, we propose an exact algorithm for collision detection between ellipsoids and superquadrics, both in 2D and 3D. And we introduce an accuracy metric of evaluations for inexact algorithms in Section IV. We then provide benchmarks details with the existing algorithms in Section V, with discussions of the results and analysis of the proposed algorithm. We conclude in Section VII.

II. The Closed-form Minkowski Sum between an Ellipsoid and Any Convex Differentiable Surface in N-Dimensional Euclidean Space

This section reviews the derivations of the closed-form Minkowski sum between an ellipsoid and any convex differentiable surface embedded in n-dimensional Euclidean space ($\mathbb{R}^n$). Then, concrete explicit expressions for superquadrics in both 2D and 3D cases are provided.

A. Review of closed-form Minkowski sums

Assume that S_a is a surface embedded in $\mathbb{R}^n$, with implicit and parametric forms being

$$\Phi(\mathbf{x}_a) = 1 \quad \text{and} \quad \mathbf{x}_a = \mathbf{f}(\psi), \tag{1}$$

where both $\mathbf{x}_a$ and $\mathbf{f}$ are vector-valued functions of ψ. Let E_b be an arbitrary ellipsoid in $\mathbb{R}^n$, with semi-axis lengths given by $\mathbf{b} = [b_1, b_2, ..., b_n]^\top$. Then, the implicit and explicit equations are of the form

$$\mathbf{x}_b^\top B^{-2} \mathbf{x}_b = 1 \text{ and } \quad \mathbf{x}_b = B\mathbf{u}(\psi), \tag{2}$$

where $B = R_b \Lambda(\mathbf{b}) R_b^\top$ is the shape matrix of E_b where $R_b \in$ SO(n) denotes the orientation of the ellipsoid, and $\Lambda(\cdot)$ the diagonal matrix. Here $\mathbf{u}(\psi)$ is the standard parameterization of the n-dimensional hyper-sphere with angle parameters $\psi = [\psi_1, \psi_2, ..., \psi_{n-1}]^\top$.

Then the affine transformations that shrink the ellipsoid to a sphere with radius $r = \min\{b_1, b_2, ..., b_n\}$ on the surface S_a can be expressed as

$$\mathbf{x}'_a = R_b \Lambda(r/\mathbf{b}) R_b^\top \mathbf{x}_a \doteq T\mathbf{x}_a, \tag{3}$$

where $T = R_b \Lambda(r/\mathbf{b}) R_b^\top$ denotes the "shrinking" affine transformation, and is symmetric and positive definite since $\Lambda(r/\mathbf{b})$ is diagonal and positive definite.

The implicit expression for the "shrunk" S_a, denoted as S'_a, is

$$\Phi(T^{-1}\mathbf{x}'_a) = 1. \tag{4}$$

Then the Minkowski sum between S'_a and E'_b, which now is a sphere, is obtained by computing the boundary of the offset surface with offset radius r as

$$\mathbf{x}_{ofs} = \mathbf{x}'_a + r\mathbf{n}', \tag{5}$$

where $\mathbf{n}' = \frac{\nabla\Phi(T^{-1}\mathbf{x}'_a)}{\|\nabla\Phi(T^{-1}\mathbf{x}'_a)\|}$ is the outward normal of the surface and $\nabla\Phi(T^{-1}\mathbf{x}'_a) = T^{-\top}\nabla\Phi(\mathbf{x}_a)$ with $T^{-\top} = (T^{-1})^\top = (T^\top)^{-1} = T^{-1}$.

The Minkowski sum between the original surface S_1 and ellipsoid E_2 can be given by "stretching" the transformed space back, using inverse affine transformation, as

$$\begin{aligned} \mathbf{x}_{eb} = T^{-1}\mathbf{x}_{ofs} \quad &= T^{-1}\left(T\mathbf{x}_a + r\frac{T^{-\top}\nabla\Phi(\mathbf{x}_a)}{\|T^{-\top}\nabla\Phi(\mathbf{x}_a)\|}\right) \\ &= \mathbf{x}_a + r\frac{T^{-2}\nabla\Phi(\mathbf{x}_a)}{\|T^{-1}\nabla\Phi(\mathbf{x}_a)\|} \end{aligned} \tag{6}$$

B. Explicit expressions in canonical form

Eq. (6) can be divided into three unknown parts, i.e. $\mathbf{x}$, $\nabla\Phi(\mathbf{x})$ and T. The explicit expressions for $\mathbf{x}$, $\nabla\Phi(\mathbf{x})$, in both 2D and 3D cases, are derived as follows.

1) 2D case: The explicit expression of a superellipse is

$$\mathbf{x}_a(\theta) = \begin{pmatrix} a_1 \cos^\varepsilon \theta \\ a_2 \sin^\varepsilon \theta \end{pmatrix}, \quad -\pi \le \theta < \pi. \tag{7}$$

The implicit equation is

$$\Phi(\mathbf{x}_a) = \left(\frac{x}{a_1}\right)^{\frac{2}{\varepsilon}} + \left(\frac{y}{a_2}\right)^{\frac{2}{\varepsilon}} = 1. \tag{8}$$

By direct calculations, the gradient in parametric form can be obtained as

$$\nabla\Phi(\theta) = \frac{2}{\varepsilon}\begin{pmatrix} \cos^{2-\varepsilon}\theta/a_1 \\ \sin^{2-\varepsilon}\theta/a_2 \end{pmatrix}. \tag{9}$$

Note that the superellipse becomes an ellipse when $\varepsilon = 1$.

2) 3D case: For a 3D Superquadrics surface, the corresponding explicit expressions can be obtained as

$$\mathbf{x}_a(\eta, \omega) = \begin{pmatrix} a_1 \cos^{\varepsilon_1}\eta \cos^{\varepsilon_2}\omega \\ a_2 \cos^{\varepsilon_1}\eta \sin^{\varepsilon_2}\omega \\ a_3 \sin^{\varepsilon_1}\eta \end{pmatrix}, \quad \begin{matrix} -\pi/2 \le \eta < \pi/2 \\ -\pi \le \omega < \pi \end{matrix} \tag{10}$$

The implicit equation is

$$\Phi(\mathbf{x}_a) = \left(\left(\frac{x}{a_1}\right)^{\frac{2}{\varepsilon_2}} + \left(\frac{y}{a_2}\right)^{\frac{2}{\varepsilon_2}}\right)^{\frac{\varepsilon_2}{\varepsilon_1}} + \left(\frac{z}{a_3}\right)^{\frac{2}{\varepsilon_1}} = 1, \tag{11}$$

the gradient of which in parametric form can be obtained as

$$\nabla\Phi(\eta, \omega) = \begin{pmatrix} \cos^{2-\varepsilon_1}\eta \cos^{2-\varepsilon_2}\omega/a_1 \\ \cos^{2-\varepsilon_1}\eta \sin^{2-\varepsilon_2}\omega/a_2 \\ \sin^{2-\varepsilon_2}\eta/a_3 \end{pmatrix}. \tag{12}$$

Similar to the 2D case, the superquadrics shrinks into an ellipsoid when $\varepsilon_1 = \varepsilon_2 = 1$.

In both 2D and 3D cases, the exponents are within the range of $0 \le \varepsilon \le 2$, so that the shapes remain convex.

III. Collision Detection Based on Closed-form Minkowski Sums

The resulting parametric closed-form Minkowski sums shrink the moving ellipsoid to a point in $\mathbb{R}^n$, so the collision detection problem can be viewed as checking whether the point is outside of a parametric surface. Applying this idea, we first assume the point to be checked is denoted as $\mathbf{p}_0 = [p_1, p_2, ..., p_n]^\top \in \mathbb{R}^n$, and the Minkowski sum boundary is obtained by Eq. (6).

A. Relative position between a point and a parametric surface

The problem is simplified as checking the relative position between a point $\mathbf{p}_0$ and a parametric surface S in $\mathbb{R}^n$. The idea is to firstly find a point $\mathbf{p}_{eb}$ on the parametric Minkowski sum boundary surface that falls on the line defined by origin O and the point $\mathbf{p}_0$. Then if $\mathbf{p}_0$ is farther from the origin than $\mathbf{p}_{eb}$, the point is outside of the boundary surface, therefore, the moving ellipsoid is separated from the fixed superquadrics, i.e.

$$\text{Status} = \begin{cases} \text{In collision} : \|\mathbf{p}_0\| \le \|\mathbf{p}_{eb}(\psi)\| \\ \text{No collision} : \|\mathbf{p}_0\| > \|\mathbf{p}_{eb}(\psi)\|. \end{cases} \tag{13}$$

The key computational step is to search for the point $\mathbf{p}_{eb}$ based on the parametric surface and the point $\mathbf{p}_0$. Concretely, we seek to find $\mathbf{p}_{eb}$ such that its distance to the line $l_{O\mathbf{p}_0}$ is zero. This subproblem is addressed in both 2D and 3D cases as follows.

1) 2D case: Let θ parameterize the closed-form Minkowski sum boundary curve in $\mathbb{R}^2$, then the elements of the points $\mathbf{p}_{eb}$ and $\mathbf{p}_0$ are defined as $\mathbf{p}_{eb}(\theta) = [p_{ebx}(\theta), p_{eby}(\theta)]^\top$ and $\mathbf{p}_0 = [p_{0x}, p_{0y}]^\top$ respectively. The distance between $\mathbf{p}_{eb}$ and $l_{O\mathbf{p}_0}$ can be written as

$$d(\mathbf{p}_{eb}(\theta), l_{O\mathbf{p}_0}) = \frac{|p_{0y}p_{ebx} - p_{0x}p_{eby}|}{\sqrt{p_{0x}^2 + p_{0y}^2}} \quad (\|\mathbf{p}_0\| \ne 0). \tag{14}$$

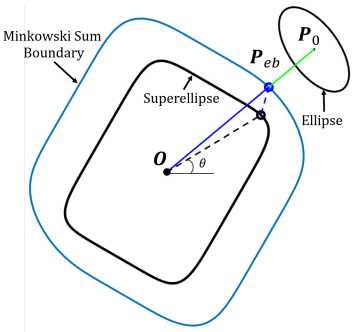


Fig. 2. A demonstration of the collision detection scheme in 2D. $\mathbf{P}_{eb}$ is parameterized by θ, which can be obtained by solving for Eq. (15). In this situation, the ellipse is separated from the superellipse since $\|\mathbf{P}_0\| > \|\mathbf{P}_{eb}\|$.

Setting the distance to be zero gives the objective function

$$F(\theta) = p_{0y}p_{ebx} - p_{0x}p_{eby} = 0. \tag{15}$$

Solving for θ gives the parameter that defines the point $\mathbf{p}_{eb}(\theta)$ on the Minkowski sum boundary. Moreover, if $\theta \in [0,\pi)$, the solution is unique, and the distance expression is valid as long as the point $\mathbf{p}_0$ does not coincide with the origin. Figure 2 demonstrates the collision detection scheme in 2D.

2) 3D case: Two parameters (i.e. η and ω) are used to define the explicit expression of the Minkowski sum boundary surface. Therefore, the distance between $\mathbf{p}_{eb}(\eta,\omega) = [p_{ebx}(\eta,\omega), p_{eby}(\eta,\omega), p_{ebz}(\eta,\omega)]^\top$ and the line defined by $\mathbf{p}_0 = [p_{0x}, p_{0y}, p_{0z}]^\top$ and the origin can be expressed as

$$d(\mathbf{p}_{eb}(\eta,\omega), l_{O\mathbf{p}_0}) = \frac{\|\mathbf{p}_{eb} \times \mathbf{p}_0\|}{\|\mathbf{p}_0\|} \quad (\|\mathbf{p}_0\| \neq 0). \tag{16}$$

Setting the distance to be zero gives the objective function

$$\mathbf{F}(\eta,\omega) = \mathbf{p}_{eb}(\eta,\omega) \times \mathbf{p}_0 = \begin{pmatrix} p_{eby}p_{0z} - p_{ebz}p_{0y} \\ p_{ebz}p_{0x} - p_{ebx}p_{0z} \\ p_{ebx}p_{0y} - p_{eby}p_{0x} \end{pmatrix} = \mathbf{0}. \tag{17}$$

Note that this is not an over-constrained system of equations since each equation in $\mathbf{F}$ can be derived from the other two equations. Hence, in the 3D case, there are two equations and two unknowns, and the solution is unique up to a reflection with respect to the origin when $\eta \in [0,\pi/2)$ and $\omega \in [0,\pi)$.

B. Algorithm for collision detection between ellipsoids and superquadrics

Based on the derivations of separation checking between a point and the parametric closed-form Minkowski sum boundary, we propose an algorithm for collision detection between ellipsoids and superquadrics.

Algorithm 1 solves the collision detection problem in general. In practice, the most computational intensive step is finding the root of the nonlinear equation $\mathbf{F}(\psi) = \mathbf{0}$ in Step 4. Since there is no simple closed-form solution for this equation, numerical root finding needs to be done.

Algorithm 1: Collision Checking Procedure for Ellipsoidal and Superquadric objects

Input: SQ_1 (Semi-axes lengths $\mathbf{a}_1$, Epsilons ε_1, Orientation R_1, Position of center $\mathbf{t}_1$); E_2 (Semi-axes lengths $\mathbf{a}_2$, Orientation R_2, Position of center $\mathbf{t}_2$).

Output: Status (0 for separated, 1 for in collision).

1 Compute the affine transformation $T = R_2\Lambda(r/\mathbf{a}_2)R_2^\top$;
2 Transform SQ_1 and E_2 by T ;
3 Construct the point on the Minkowski Sum boundary $\mathbf{p}_{eb}(\psi)$ via Eq. (6) ;
4 Solve the objective function $\mathbf{F}(\psi) = \mathbf{0}$ for $\tilde{\psi}$ via Eq. (15) or Eq. (17) ;
5 Compute the point $\mathbf{p}_{eb}(\tilde{\psi})$;
6 Compare the magnitudes $\|\mathbf{p}_{eb}(\tilde{\psi})\|$ and $\|\mathbf{p}_0\|$, and determine the Status via Eq. (13).

IV. Algorithm Evaluation Metric

The performance of any collision detection algorithm is evaluated by both efficiency and accuracy, the former of which can be compared by the running time. The accuracy is an equally important judgment for a good collision checking algorithm, but is more difficult to define. Here, we introduce an accuracy evaluation metric based on the volume of all configurations where collision occurs. Such volume can be computed via the Principal Kinematic Formula [21] as

$$I(S_a, E_b) = \int_{\mathrm{SE(n)}} i(S_a \cap gE_b)dg, \tag{18}$$

where S_a and E_b are superquadrics and ellipsoids in $\mathbb{R}^n$ respectively, $g = (R,\mathbf{t}) \in \mathrm{SE}(n)$ describes the pose of E_b (i.e. $gE_b \doteq RE_b + \mathbf{t}$), dg is the natural bi-invariant integration measure for $\mathrm{SE}(n)$ [22] and $i(S_a \cap gE_b)$ is an indicator function defined as

$$i(S_a \cap gE_b) = \begin{cases} 1, & S_a \cap gE_b \neq \varnothing \\ 0, & S_a \cap gE_b = \varnothing \end{cases} \tag{19}$$

Since inscribed meshes or bounding volumes make approximations to the actual objects, there is the possibility that the inexact algorithms return "no collision" when there is actually a collision, and vice versa. Therefore, computing the relative volume of all the possible collision configurations gives a metric to evaluate the accuracy of performing collision detection. The relative volume can be computed as

$$\gamma = \frac{I(Mesh_a, Mesh_b)}{I(S_a, E_b)} \times 100\%, \tag{20}$$

where $Mesh_a$ and $Mesh_b$ are the two objects represented by meshes. Note that for an exact algorithm, such as our proposed method or ASC, $\gamma = 100\%$.

V. Benchmark with Existing Methods

In this section, we compare the computational time and accuracy of the proposed algorithm for discrete collision checking with some state-of-the-art ones. For the ellipsoid-ellipsoid case, both ASC and GJK methods are compared;

TABLE I

A LIST OF BENCHMARKS FOR THE ALGORITHM 1

Dimension	Ellipsoid-Ellipsoid	Ellipsoid-Superquadrics
2D	ASC, GJK (E), GJK (Mesh)	GJK (Mesh)
3D	ASC, GJK (E), GJK (Mesh)	GJK (Mesh)

TABLE II

PARAMETERS FOR ELLIPSOID AND SUPERQUADRICS MESHES.

Dimension/Object	Notation	# Vertices	# Facets
2D/Ellipse (E)	Mesh	50	48
2D/Superellipse (S)	Mesh	50	48
3D/Ellipsoid (E)	Mesh1	25	268
3D/Superquadrics (S)	Mesh1	25	260
3D/Ellipsoid (E)	Mesh2	100	540
3D/Superquadrics (S)	Mesh2	100	534

while for the ellipsoid-superquadrics case, only GJK is compared. All the algorithms are implemented in C++, and the benchmarks run in an Intel Core i7 CPU at 3.60GHz.

A. Benchmark Parameters and Notations

To make a fair comparison, we input the same parameters that defines the geometry (i.e. semi-axes lengths and exponents) and configuration of the objects (i.e. orientation and location). For ASC and our method, those parameters are directly used in the algorithms. For GJK, from the Flexible Collision Library (FCL), they need to be converted to specific representation objects. For the shape representation, we use the built-in "Ellipsoid (E)" and "Mesh" objects for ellipsoids and "Mesh" for superquadrics (S). Table I lists the algorithms and object shapes we use for benchmarks.

We also compare the effects of using different mesh densities by varying the number of vertices and facets that constructs the convex bodies of the objects. To generate those meshes, we use a popular computational geometry library in C++, "CGAL" [23]. Detailed vertices and surface information for the generated meshes are provided in Table II with the notation of each mesh used in the rest of the content.

B. Running Time Results

Using two ellipsoids/superquadrics, we fix one and randomly generate 1000 poses of the other. Then we record the running time of collision checking for each configuration. Figure 3 shows the running time comparisons, with the line segment being standard deviation and its center being the mean.

C. Accuracy Evaluation of Collision Detection

Based on the metric described in Section IV, We evaluate the accuracy γ for different object representations in both 2D and 3D. For the cases of SE(2), Eq.(18) can be calculated as [21]

$$I_{\mathrm{SE}(2)}(S_a,E_b) = 2\pi[A(S_a)+A(E_b)] + L(S_a)L(E_b), \qquad (21)$$

where, $A(\cdot)$ and $L(\cdot)$ denote the area and perimeter of the objects respectively. And for SE(3), the corresponding simple expression for Eq.(18) is

$$I_{\mathrm{SE}(3)}(S_a,E_b) = 8\pi^2[V(S_a)+V(E_b)] + 2\pi F(\partial E_b)M(\partial S_a) + 2\pi F(\partial S_a)M(\partial E_b), \qquad (22)$$

where $V(\cdot)$ is the volume of a body in $\mathbb{R}^n$, $F(\cdot)$ and $M(\cdot)$ are the surface area and the integral of mean curvature of the bounding surface enclosing a spatial body, respectively.

Table III compares the collision detection accuracy for each object representation pair for the corresponding experimental trial in both 2D and 3D.

TABLE III

COLLISION DETECTION ACCURACY FOR EACH CASE IN 2D AND 3D.

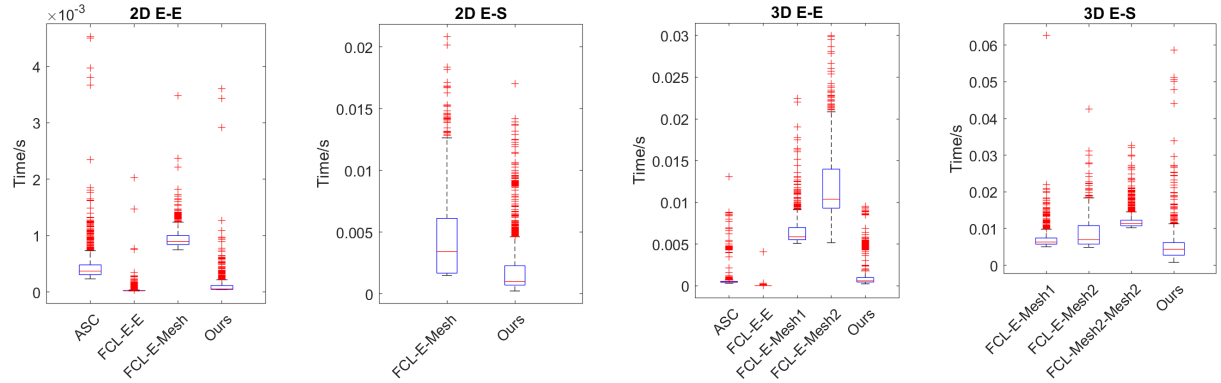
Case	Algorithm/Objects pair	Accuracy (γ)
2D/E-E	ASC	100%
2D/E-E	GJK/E-E	79.53%
2D/E-E	GJK/E-Mesh	82.25%
2D/E-E	Ours	100%
2D/E-S	GJK/E-Mesh	88.65%
2D/E-S	Ours	100%
3D/E-E	ASC	100%
3D/E-E	GJK/E-E	38.18%
3D/E-E	GJK/E-Mesh1	42.08%
3D/E-E	GJK/E-Mesh2	60.98%
3D/E-E	Ours	100%
3D/E-S	GJK/E-Mesh1	28.62%
3D/E-S	GJK/E-Mesh2	72.33%
3D/E-S	GJK/Mesh1-Mesh2	74.77%
3D/E-S	Ours	100%

VI. DISCUSSION

With the current implementation, the Minkowski-based collision checker is competitive with some of the state-of-the-art methods. The significant advantages of our proposed algorithm are the direct use of the shape and configuration parameters without the need of meshes or bounding volumes, and the ability to extend to more complex shapes embedded in the Euclidean space.

In the Ellipsoid-Ellipsoid (E-E) case, both Algebraic Separation Condition and Minkowski-based methods provide exact collision detection, with the difference being the former only solves a cubic or quartic polynomial, which can be efficient. The Minkowski-based method, however, requires solving for the roots of a nonlinear equation. It is essential to apply an efficient nonlinear optimization algorithm and solver (currently we apply a trust-region algorithm, and use the nonlinear root finding method from the well-known "Eigen" library in C++). The results show that, in terms of running time, ours is competitive with ASC, and even outperforms in the 2D case.

Another remarkable advantage of our method is that it can be extended to more complex shapes. Although this paper only derives concrete expressions and conducts experiments for superquadrics, the closed-form Minkowski sum can be applied to any convex and differentiable surface embedded in $\mathbb{R}^n$, as described in Eq. (1). Consequently, the proposed general algorithm can deal with the collision detection problem between an ellipsoid and any object shape as long as it has implicit and parametric expressions.



(a) Ellipse-Ellipse (b) Ellipse-Superellipse (c) Ellipsoid-Ellipsoid (d) Ellipsoid-Superquadrics

Fig. 3. Running time comparisons with existing methods and different object representations. For ASC and our proposed method, the shape and configuration parameters are directly used; For GJK, we make comparisons between different object representations provided in FCL, labeled as "FCL-Object1-Object2".

For objects with complex shapes, generating meshes or bounding with volumes are common methods, with GJK being one of the most rigorous and efficient collision checkers for those primitives. We compare our method with GJK in both the E-E and E-S collision checking scenarios using different mesh densities. It turns out that by using the "Ellipsoid" object in FCL, GJK runs remarkably faster, the reason of which being the bounding volumes have fewer vertices. For the experiments, we use this "Ellipsoid" object for one agent and generate different meshes by varying the number of vertices and facets for the other agent. We also use meshes for both objects to show the changes in the performance of the GJK detection algorithm with different mesh densities. It is obvious that as the number of vertices and facets increase, GJK takes longer time to execute. This gives limitations to the GJK algorithm, which is an inexact checker and depends significantly on the quality of the meshes or bounding volumes. Our algorithm, on the other hand, is an exact checker, with the accuracy $\gamma = 100\%$, therefore outperforms GJK when the number of vertices becomes larger.

Moreover, as the accuracy comparisons show, the volume of all the possible collision configurations computed from the mesh is always smaller than the one of the exact representation of the object, since the former always gives a lower bound for the object if the vertices are generated on the boundary. As a result, even when the two objects collide, inexact algorithms might sometimes return false negative results, and the probability of returning the true results is reflected by the accuracy.

VII. Conclusion

This paper studies the collision detection problem between ellipsoids and superquadrics. The geometric formulations are based on the closed-form Minkowski sums, a parametric expression that enlarges the superquadrics surface boundary and shrinks the ellipsoid into a point in $\mathbb{R}^n$. An algorithm is proposed, which involves computing the closed-form Minkowski sum boundary and finding the relative position between a point and a parametric surface. Furthermore, to evaluate the probability of returning the true collision results, a collision detection accuracy based on the relative volume of all collision configurations of the objects is introduced. The major advantages of the propose Minkowski-based algorithm are:

1) It is an exact collision checker that uses less parameters as input, i.e. the semi-axes lengths and exponents, without depending on the quality of the meshes or bounding volumes;
2) It can work for collision detection between an ellipsoid and any surface, with implicit and parametric expressions, that is embedded in the Euclidean space.

Benchmark experiments are performed in C++ and with some existing popular collision detection algorithms: Algebraic Separation Conditions (ASC) and Gilbert-Johnson-Keerthi (GJK) for ellipsoid-ellipsoid checking, and GJK for ellipsoid-superquadrics checking. The majority of the computational time for the proposed method is spent on solving the roots of a nonlinear equation, and a numerical solver is used in practice. The GJK method depends on the complexity of the meshes belonging to the objects, and different numbers of vertices and facets are used and compared. The benchmark results show that the proposed method is competitive with ASC and GJK, and outperforms as the mesh density increases.

Acknowledgement

The authors thank Dr. Christian Wuelker, Dr. Shengnan Lu and Ms. Mengdi Xu for useful discussions. This work was performed under National Science Foundation grant IIS-1619050 and Office of Naval Research Award N00014-17-1-2142. The ideas expressed in this paper are solely those of the authors.

REFERENCES

[1] A. Makhal, F. Thomas, and A. P. Gracia, "Grasping unknown objects in clutter by superquadric representation," in *2018 Second IEEE International Conference on Robotic Computing (IRC)*. IEEE, 2018, pp. 292–299.

[2] Y. Yan and G. S. Chirikjian, "Closed-form characterization of the minkowski sum and difference of two ellipsoids," *Geometriae Dedicata*, vol. 177, no. 1, pp. 103–128, 2015.

[3] Y. Gu, Y. He, K. Fatahalian, and G. Blelloch, "Efficient bvh construction via approximate agglomerative clustering," in *Proceedings of the 5th High-Performance Graphics Conference*. ACM, 2013, pp. 81–88.

[4] S. Gottschalk, M. C. Lin, and D. Manocha, "Obbtree: A hierarchical structure for rapid interference detection," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. ACM, 1996, pp. 171–180.

[5] D. Jung and K. K. Gupta, "Octree-based hierarchical distance maps for collision detection," *Journal of Robotic Systems*, vol. 14, no. 11, pp. 789–806, 1997.

[6] K. Moustakas, D. Tzovaras, and M. G. Strintzis, "Sq-map: Efficient layered collision detection and haptic rendering," *IEEE Transactions on Visualization & Computer Graphics*, no. 1, pp. 80–93, 2007.

[7] R. J. Portal, L. A. Sousa, and J. M. Dias, "Multibody dynamics method with superquadric contact detection model applied to biomechanics."

[8] M. C. Lin and J. F. Canny, "A fast algorithm for incremental distance calculation," in *Robotics and Automation, 1991. Proceedings., 1991 IEEE International Conference on*. IEEE, 1991, pp. 1008–1014.

[9] J. D. Cohen, M. C. Lin, D. Manocha, and M. Ponamgi, "I-collide: An interactive and exact collision detection system for large-scale environments," in *Proceedings of the 1995 symposium on Interactive 3D graphics*. ACM, 1995, pp. 189–ff.

[10] T. C. Hudson, M. C. Lin, J. Cohen, S. Gottschalk, and D. Manocha, "V-collide: accelerated collision detection for vrml," in *Proceedings of the second symposium on Virtual reality modeling language*. ACM, 1997, pp. 117–ff.

[11] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, vol. 4, no. 2, pp. 193–203, 1988.

[12] J. Pan, S. Chitta, and D. Manocha, "Fcl: A general purpose library for collision and proximity queries," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 3859–3866.

[13] N. Chakraborty, J. Peng, S. Akella, and J. E. Mitchell, "Proximity queries between convex objects: An interior point approach for implicit surfaces," *IEEE Transactions on Robotics*, vol. 24, no. 1, pp. 211–220, 2008.

[14] W. Wang, J. Wang, and M.-S. Kim, "An algebraic condition for the separation of two ellipsoids," *Computer aided geometric design*, vol. 18, no. 6, pp. 531–539, 2001.

[15] Y.-K. Choi, W. Wang, and M.-S. Kim, "Exact collision detection of two moving ellipsoids under rational motions," in *ICRA*, 2003, pp. 349–354.

[16] Y.-K. Choi, J.-W. Chang, W. Wang, M.-S. Kim, and G. Elber, "Continuous collision detection for ellipsoids," *IEEE Transactions on visualization and Computer Graphics*, vol. 15, no. 2, pp. 311–325, 2009.

[17] D. S. Lopes, M. T. Silva, J. A. Ambrósio, and P. Flores, "A mathematical framework for rigid contact detection between quadric and superquadric surfaces," *Multibody System Dynamics*, vol. 24, no. 3, pp. 255–280, 2010.

[18] C. Wellmann, C. Lillie, and P. Wriggers, "A contact detection algorithm for superellipsoids based on the common-normal concept," *Engineering Computations*, vol. 25, no. 5, pp. 432–442, 2008.

[19] P. Khosla and R. Volpe, "Superquadric artificial potentials for obstacle avoidance and approach," in *Robotics and Automation, 1988. Proceedings., 1988 IEEE International Conference on*. IEEE, 1988, pp. 1778–1784.

[20] R. Portal, J. Dias, and L. de Sousa, "Contact detection between convex superquadric surfaces," *Archive of Mechanical Engineering*, vol. 57, no. 2, pp. 165–186, 2010.

[21] G. S. Chirikjian, "Parts entropy and the principal kinematic formula," in *Stochastic Models, Information Theory, and Lie Groups, Volume 2*. Springer, 2012, pp. 187–228.

[22] ——, *Stochastic Models, Information Theory, and Lie Groups, Volume 2: Analytic Methods and Modern Applications*. Springer Science & Business Media, 2011, vol. 2.

[23] The CGAL Project, *CGAL User and Reference Manual*, 4.12.1 ed. CGAL Editorial Board, 2018. [Online]. Available: https://doc.cgal.org/4.12.1/Manual/packages.html