Proximity Queries between Convex Objects: An Interior Point Approach for Implicit Surfaces

Nilanjan Chakraborty
Computer Science Dept.
Rensselaer Polytechnic Inst.
Troy, New York 12180
chakrn2@cs.rpi.edu

Jufeng Peng
Mathematical Sciences Dept.
Rensselaer Polytechnic Inst.
Troy, New York 12180
pengj@cs.rpi.edu

Srinivas Akella
Computer Science Dept.
Rensselaer Polytechnic Inst.
Troy, New York 12180
sakella@cs.rpi.edu

John Mitchell
Mathematical Sciences Dept.
Rensselaer Polytechnic Inst.
Troy, New York 12180
mitchj@rpi.edu

Abstract— **In this paper, we present an interior point approach to exact distance computation between convex objects represented as intersections of implicit surfaces. The implicit surfaces considered include planes (polyhedra), quadrics, and generalizations of quadrics including superquadrics and hyperquadrics, as well as intersections of these surfaces. Exact distance computation algorithms are particularly important for applications involving objects that make contact, such as in dynamic simulations and in contact point prediction for dextrous manipulation. They can also be used in the narrow phase of hierarchical collision detection. In contrast to geometric approaches developed for polyhedral objects, we formulate the distance computation problem as a convex optimization problem; this optimization formulation has been previously described for polyhedral objects. We demonstrate that for general convex objects represented as implicit surfaces, interior point approaches are reasonably fast and in some cases, owing to their global convergence properties, are the only provably good choice for solving proximity query problems. We use an interior point algorithm that solves the KKT conditions obtained from the convex programming formulation. We present implementation results for example implicit surface objects and demonstrate that distance computation rates of about 1 kHz can be achieved.**

I. Introduction

This paper studies the problem of closest distance computation between two or more convex objects, when each object is described as an intersection of implicit surfaces. Knowledge of the closest distance and the closest points is useful in a variety of applications including dynamic simulations ([1], [27], [38]), dextrous manipulation ([28],[7]), haptics ([10]), and spacecraft safe volume computations ([11]). Distance computations between objects are especially useful in dynamic simulations since the simulators need knowledge of potential collision points ([1], [38], [37], [20]). Furthermore, dynamic simulations of contacting objects can be sensitive to the shapes of the objects ([34], [13]). In fact, a polygonal approximation of a surface may lead to incorrect predictions when surface curvature influences contact dynamics. For example, a simulated circular disc rolling on a table loses energy, when represented as a polygon, due to repeated impacts of the vertices with the table [39]. More generally, polygonalized representations of smooth objects can lead to intermittent loss of contact and bouncing in the simulations. Similarly, exact representation of shape is important when modeling robot fingers in contact with smooth objects during multi-finger dextrous manipulation (Figure 1). Determining the first contact point correctly when fingers reposition during finger gaiting is useful since the manipulation operations are sensitive to the contact point and the normal and curvature at that point ([35],[7],[31]). The proximity query algorithms presented can additionally be used in the narrow phase of a hierarchical collision detection algorithm, which also has applications in robot path planning [32].

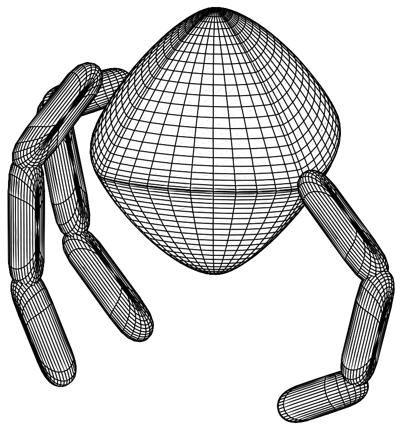
Fig. 1. A dextrous manipulation task that requires closest distance computations to predict the contact points of fingers with an object. The fingers and object are represented as superquadrics.

The general problem of distance computation between two objects X and Y can be written as

$$\begin{aligned} \text{Minimize} \quad & \|\mathbf{x}_g - \mathbf{y}_g\|_2 \\ \text{subject to:} \quad & \mathbf{x}_g \in X, \quad \mathbf{y}_g \in Y \end{aligned} \tag{1}$$

where the two objects X and Y are represented as compact (closed and bounded) sets in $\mathcal{R}^2$ or $\mathcal{R}^3$ and the points $\mathbf{x}_g$ and $\mathbf{y}_g$ are points in the two objects. We focus on representing the sets X and Y as intersections of implicit surfaces, including planes, quadrics, superquadrics, and hyperquadrics.

The general problem of distance computation between two objects has been extensively studied ([15], [18], [25]). However most work on distance computation assumes polyhedral object representations ([12], [24], [15], [26]). The literature on distance computation between general implicit surfaces is relatively sparse because, with a few notable exceptions ([14],[41]), methods for polyhedral representations do not easily generalize to implicit surfaces. There has also been some work on proximity queries and collision detection between spline surfaces [40],[19].

Contributions of the paper: This paper focuses on the minimum distance computation problem between two convex objects, where each object is described as an intersection of implicit surfaces. This class of convex objects includes convex polyhedra, quadrics, superquadrics, and hyperquadrics. While the distance computation problem for convex objects has been known to be a convex optimization problem ([5],[6]), to the best of our knowledge, interior point algorithms have not been previously applied to this problem. Interior point methods are well suited for this class of optimization problems since they are guaranteed to converge to the global optimum for convex problems. Further, they exhibit polynomial convergence for special classes of functions called self-concordant functions. We apply a recently developed interior point algorithm [8], [46] to compute the distance between convex implicit surface objects and demonstrate that it is particularly effective for this class of problems. For quadric surfaces, the algorithm has a polynomial running time and is guaranteed to terminate in a finite number of steps. We also extend the approach to surfaces such as superquadrics and hyperquadrics. To the best of our knowledge, this is the first approach with this demonstrated capability (without discretization, of course). Another important advantage of this method is that it provides a uniform framework for proximity queries between objects described as intersections of convex polyhedra, quadrics, or any arbitrary convex implicit surface. Further, these proximity queries can be used in the narrow phase of hierarchical collision detection for implicit surfaces.

The paper is organized as follows. After a discussion of related work in Section II, we review the mathematical background for our work in Section III. We present the formulation of the closest distance problem in Section IV and describe how it can be solved using interior point algorithms in Section V. We present our implementation results in Section VI and conclude with a discussion of future work in Section VII.

II. Related Work

Proximity queries for polyhedra: Proximity queries and collision detection algorithms have an extensive literature in computational geometry [12], robotics [15], [24], and computer graphics [41]; we provide a sampling of the related work in these areas. Lin and Manocha [25] provide an overview of collision detection and proximity queries. When collision detection algorithms estimate the distance between two objects, they typically use a geometric approach. Popular algorithms for convex polyhedra include GJK [15], Lin-Canny [24], and V-Clip [26]. GJK [15] is an iterative algorithm for distance computation between two convex polyhedra. It uses a support function description of the polyhedra and takes time linear in the number of vertices. Lin-Canny [24] efficiently computes the distance between two convex polyhedra and tracks the closest points using adjacency of features. Its running time is linear in the number of features (faces, edges, and vertices). Both algorithms can track the closest points in (almost) constant time when there is temporal coherence [9]. Bobrow [5] proposed an optimization based approach for computing the distance between two polyhedra. He formulated the problem as a quadratic programming problem and used a gradient projection algorithm to solve the problem. However this approach can suffer from convergence issues [48].

Proximity queries for quadrics and NURBS: Distance estimation between non-polyhedral shapes has focused primarily on quadrics and NURBS surfaces. Only GJK has been extended directly for smooth convex objects [14]; van den Bergen [41] discusses in detail a GJK implementation for convex quadric objects. However this GJK algorithm does not guarantee convergence in a finite number of steps. Further, computing the support mapping is difficult for superquadrics with fractional (non-integer) exponents due to the difficulty of solving polynomials with fractional exponents. Turnbull and Cameron [40] extended GJK to convex NURBS surfaces. They describe a procedure to calculate the support mapping for the NURBS surfaces which reduces to solving two nonlinear polynomial equations in two parameters. They present results for 2D and describe the theory for 3D. Baraff [1] described collision detection algorithms for implicit and parametric curved convex surfaces. He uses the collinearity property of the surface normals of the closest points to numerically compute closest points at the initial configuration. He exploits geometric coherence to compute closest points at subsequent configurations. Lin and Manocha [23] consider curved models described as NURBS surfaces and piecewise algebraic surfaces. Using the collinearity property of the surface normals, they describe the closest points using a set of polynomial equations. However, the number of roots can be prohibitively large as it depends on the degree of the polynomials describing the surfaces; the roots must be examined to identify the closest points. Note also that it is not possible to obtain bounds on the number of roots for systems of equations with fractional indices (as would arise with superquadrics). Johnson and Cohen [19] give a lower-upper bound tree framework for distance computation between any two object representations for which the following set of operations is available: bounding volume generation, lower and upper bound on distance, bounding volume refinement, and determination of computation termination. They have demonstrated their method on polyhedra as well as NURBS surfaces. Patoglu and Gillespie [30] perform real-time tracking of the closest points between two objects modeled with parametric surfaces by formulating it as a control problem and exploiting spatial and temporal coherence.

The literature on distance computation between general implicit surfaces is relatively sparse because, with the exception of GJK, methods for polyhedral representations do not easily generalize to implicit surfaces. Most closely related is recent work on computing the distance between two ellipsoids and other conic sections ([22], [11], [36]). Sohn et al. [36] exploit the fact that the closest points on two surfaces are where their common normals intersect the surfaces. They apply their line geometry approach to ellipsoids, for which the minimum distance computation is reduced to finding the common roots of two polynomial equations of degree 8 and 16. Coppola and Woodburn [11] formulate the problem as an optimization problem. They iteratively solve the problem of closest distance from a point to an ellipsoid to arrive at the optimal solution. Rimon and Boyd [33] use convex optimization techniques to find the minimum volume enclosing ellipsoids to model objects, and then compute a conservative distance estimate between ellipsoids by treating it as an eigenvalue problem.

Proximity queries for superquadrics: The superquadric object representation was introduced by Barr [2] as a generalization of quadrics, and has been well studied in graphics and computer vision [17]. Superquadrics and extensions such as hyperquadrics [16] and deformable superquadrics [3] are a convenient representation for a large class of both convex and non-convex objects. Despite their ability to represent a wide range of objects with a small number of parameters, superquadrics have not been widely used as an object representation for contact tasks such as dynamic simulation or dextrous manipulation. This is in part due to the lack of distance computation and collision detection algorithms. In general, there is no closed form solution for the distance between two superquadrics. In fact, no closed form solution exists even for the distance between a point and an ellipsoid. Although superquadrics are a generalization of quadrics, the problem in generalizing the methods in [1],[15],[11], [36] to superquadrics is that they all lead to polynomial equations with fractional exponents, which are very difficult to solve. In general, we do not know the total number of roots, and even when it is possible to simplify the polynomials, they may have large integer exponents.

III. Mathematical Preliminaries

We now review the mathematical terminology that will be used in the rest of the paper.

Convex Set: A set $U \subseteq \mathcal{R}^n$ is called a convex set if for any two points $\mathbf{u}_1$, $\mathbf{u}_2 \in U$ and any λ with $0 \le \lambda \le 1$, we have

$$\lambda \mathbf{u}_1 + (1-\lambda)\mathbf{u}_2 \in U.$$

Convex Function: A function $f : \mathcal{R}^n \to \mathcal{R}$ is convex if the domain of f (*dom* f) is a convex set and for all $\mathbf{u}_1$, $\mathbf{u}_2 \in$ *dom* f and any λ with $0 \le \lambda \le 1$, we have

$$f(\lambda \mathbf{u}_1 + (1-\lambda)\mathbf{u}_2) \le \lambda f(\mathbf{u}_1) + (1-\lambda) f(\mathbf{u}_2).$$

Convex Programming Problem: Consider the general nonlinear programming problem given by:

$$\begin{aligned} \text{Minimize} \quad & f_0(\mathbf{x}) \\ \text{subject to:} \quad & \mathbf{x} \in U \end{aligned} \tag{2}$$

This nonlinear programming problem is called a convex programming problem if the objective function f_0 is a convex function and the feasible set U is a convex set [4]. Usually the set U is defined by a set of inequality and/or equality constraints. If the inequality constraints defining U are convex functions and the equality constraints are linear, then U is a convex set [6].

Superquadric: A superquadric [16] is defined by the equation

$$\begin{aligned} & \left|\frac{x_1}{a_1}\right|^{n_1} + \left|\frac{x_2}{a_2}\right|^{n_2} + \left|\frac{x_3}{a_3}\right|^{n_3} = 1 \\ & n_i = l_i/m_i, \quad l_i, m_i \in \mathcal{Z}^+, \quad i \in \{1,2,3\} \\ & \text{convex if} \quad 1 \le n_i < \infty \\ & \text{nonconvex if} \quad 0 < n_i < 1 \end{aligned} \tag{3}$$

Although the definition here differs slightly from that in [2], the two definitions are equivalent [16]. Convex superquadrics are a broad class of shapes that include cuboids, rounded cuboids, ellipsoids, spheres, and (rounded) octahedra.

Hyperquadric: A hyperquadric [16] is defined by the equation

$$\begin{aligned} & \sum_{i=1}^{N} |H_i(x)|^{n_i} = 1 \text{ where } N \ge 3 \text{ and} \\ & H_i(x) = (a_i x_1 + b_i x_2 + c_i x_3 + d_i) \\ & n_i = l_i/m_i, \quad l_i, m_i \in \mathcal{Z}^+ \end{aligned} \tag{4}$$

Hyperquadrics are a more general class of shapes than superquadrics. In particular, they include asymmetric shapes.

IV. Problem Formulation

We now formulate the problem of minimum distance computation between two objects as a convex optimization problem. When the two objects are expressed as intersections of implicit surfaces, the minimum distance computation problem of equation (1) can be written as:

$$\begin{aligned} \text{Minimize} \quad & \|\mathbf{x}_g - \mathbf{y}_g\|_2^2 \\ \text{subject to:} \quad & \mathbf{x}_g = \mathbf{R}_{xi}\mathbf{x}_{li} + \mathbf{p}_{xi} \\ & \mathbf{y}_g = \mathbf{R}_{yj}\mathbf{y}_{lj} + \mathbf{p}_{yj} \\ & f_i(\mathbf{x}_{li}) \le 0 \quad i = 1,2,...,m \\ & f_j(\mathbf{y}_{lj}) \le 0 \quad j = m+1, m+2, ..., n \end{aligned} \tag{5}$$

where $\mathbf{x}_g$, $\mathbf{y}_g \in \mathcal{R}^3$ are the global coordinates of points in the two objects X and Y respectively; $\mathbf{R}_{ki}, \mathbf{p}_{ki}$, $k = x, y$, are the rotation matrix and position of a reference frame in each of the intersecting regions, $\mathbf{x}_{li}$, $\mathbf{y}_{lj} \in \mathcal{R}^3$ are the coordinates of the points in the local reference frames of the surfaces, and f_i are the functions representing the implicit surfaces. The above system has $3n$ linear equality constraints and n inequality constraints. Note that for polyhedra and quadrics, it may be more convenient to represent the surfaces in the global reference frame and eliminate the affine equations describing the rigid body transformation in equation (5). The objective function in equation (5) is convex, and if the inequalities represent a convex set (i.e., the objects are convex), the minimum distance computation problem is a convex programming problem. For convex superquadrics and other general convex surfaces, the closest distance problem is a nonlinear program (NLP). If the objects are convex polyhedra (intersection of planes), the closest distance problem becomes a quadratic programming (QP) problem, and for objects described as convex quadric surfaces, the problem reduces to a quadratically constrained quadratic program (QCQP).

The solution to the minimum distance problem of equation (5) gives two closest points that lie on the surfaces of the two objects (i.e., boundaries of the two sets). We use an interior point algorithm [47] for solving this problem. Interior point methods [47] are a class of optimization algorithms for nonlinear programming problems. In contrast to algorithms for finding the closest points that generate iterates that lie on the surface of the objects (gradient projection [5], for example), feasible interior point methods generate iterates that are guaranteed to lie inside the objects and converge towards the closest points on the boundaries of the objects. This is the main conceptual difference between interior point methods and other methods.

V. Interior Point Algorithm

The Karush-Kuhn-Tucker (KKT) conditions give necessary and sufficient conditions for solving the minimum distance problem in equation (5), since it is a convex optimization problem. The KKT conditions for equation (5) are a system of nonlinear equations. Newton's method is a popular algorithm for solving systems of nonlinear equations; it converges to the correct solution if the initial guess is *near enough* [29]. However, in general it is very difficult to supply a good initial guess and there is then no guarantee that Newton's method will converge. Interior point methods approximately solve a sequence of systems of nonlinear equations that are formed by perturbing the complementarity equations in the KKT conditions. Different interior point methods have been proposed along with their convergence analysis and some have been implemented ([8], [46],[43], [44],[45]). Following [6], we present the interior point method by reformulating equation (5) as a *barrier* problem and obtaining the KKT conditions for it.

First, we rewrite the inequality constraints in equation (5) as

$$\begin{aligned} &f_i(\mathbf{x}_{li}) + s_i = 0 \quad i = 1, 2, ..., m \\ &f_j(\mathbf{y}_{lj}) + s_j = 0 \quad j = m+1, m+2, ..., n \\ &s_i \geq 0, \quad s_j \geq 0 \end{aligned} \tag{6}$$

where the slack variables s_i, s_j define the *algebraic distance* of a point to the surface. The problem can then be reformulated so that the inequalities on s_i and s_j are implicit.

$$\begin{aligned} \text{Minimize} \quad & f_0(\mathbf{x}) - \mu \sum_{k=1}^{n} \ln(s_k) \\ \text{subject to:} \quad & \mathbf{h}(\mathbf{x}) = \mathbf{0} \\ & \mathbf{f}(\mathbf{x}) + \mathbf{s} = \mathbf{0} \end{aligned} \tag{7}$$

where n is the total number of constraints describing the two objects, $\mathbf{x} \in \mathcal{R}^{3n+6}$ is the vector of global and local coordinates, $\mathbf{s} \in \mathcal{R}^n$, $\mathbf{h} : \mathcal{R}^{3n+6} \to \mathcal{R}^{3n}$ is the vector of linear equality constraints, and $\mathbf{f} : \mathcal{R}^{3n+6} \to \mathcal{R}^n$ is the vector of inequality constraints. The formulation in equation (7) is the *barrier* formulation [6] of the minimum distance problem of equation (5) and μ is called the barrier parameter. Here, for notational convenience, we have written all the linear coordinate transformation equations and nonlinear (or linear, in the case of polyhedra) object primitive equations as vector equations. The Lagrangian for the above constrained optimization problem can be written as

$$f_0(\mathbf{x}) - \mu \sum_{k=1}^{n} \ln(s_k) + \lambda^{\mathbf{T}}(\mathbf{f}(\mathbf{x}) + \mathbf{s}) + \nu^{\mathbf{T}}\mathbf{h}(\mathbf{x}) \tag{8}$$

where λ and ν are the Lagrange multipliers. Thus, the KKT conditions can be written as

$$\begin{aligned} \nabla f_0(\mathbf{x}) + (\nabla \mathbf{f}(\mathbf{x}))^T \lambda + (\nabla \mathbf{h}(\mathbf{x}))^T \nu &= \mathbf{0} \\ \mathbf{f}(\mathbf{x}) + \mathbf{s} &= \mathbf{0} \\ \mathbf{h}(\mathbf{x}) &= \mathbf{0} \\ \mathbf{LSe} - \mu\mathbf{e} &= \mathbf{0} \end{aligned} \tag{9}$$

Here $\mathbf{L}$ is a diagonal matrix of the λ variables, $\mathbf{S}$ is a diagonal matrix of the slack variables $\mathbf{s}$, and $\mathbf{e}$ is a n-vector of ones. The above represents a system of $8n+6$ nonlinear equations in $8n+6$ variables and can be approximately solved for a given μ. Note that the KKT conditions for the original problem of equation (5) have the complementarity constraints $\mathbf{LSe} = \mathbf{0}$. Thus, as the barrier parameter μ approaches 0, the KKT conditions for the barrier problem of equation (9) approach that of the original problem. The general structure of interior point methods is described in Algorithm 1:

Algorithm 1 Interior point algorithm

```
Input: initial strictly feasible $\mathbf{x}_{initial}$, $\mu_{initial}$, specified tolerance $\epsilon$, and KKT equations
Output: Closest points solution $\mathbf{x}$
repeat
    Approximately solve the KKT conditions for current $\mu$ and obtain current solution $\mathbf{x}$
    Decrease $\mu$
until termination error criterion is satisfied
return $\mathbf{x}$
```

The number of iterations taken for this algorithm to converge to the optimal solution is guaranteed to be polynomial in the number of constraints for a special class of functions called self-concordant functions (see

[6] for details). Thus, problems with linear and quadratic constraints are guaranteed to converge in $O(\sqrt{n})$ iterations (where n is the number of constraints). The implemented interior point methods (LOQO [43], [44], IPOPT [45], KNITRO [8], [46]) vary in the way they approximately solve the system of nonlinear equations for each value of μ, the termination criterion used, and the way in which they update μ.

Newton's method is used for approximately solving the KKT conditions for each value of μ. During each iteration of Newton's method, we need to compute the Hessian $\mathbf{H}$ of the Lagrangian evaluated at the current value of $\mathbf{x}$. This is the coefficient matrix of the system of linear equations to be solved to compute the Newton step. Its general structure for any type of implicit primitive is

$$\mathbf{H} = \begin{pmatrix} \mathbf{A}_1 & \mathbf{A}_2^T & \mathbf{A}_3^T & \mathbf{0}_{(3n+6)\times n} \\ \mathbf{A}_2 & \mathbf{0}_{n\times n} & \mathbf{0}_{n\times 3n} & I_{n\times n} \\ \mathbf{A}_3 & \mathbf{0}_{3n\times n} & \mathbf{0}_{3n\times 3n} & \mathbf{0}_{3n\times n} \\ \mathbf{0}_{n\times(3n+6)} & \mathbf{S} & \mathbf{0}_{n\times 3n} & \mathbf{L} \end{pmatrix} \quad (10)$$

where $\mathbf{A}_1 = \nabla^2 f_0(\mathbf{x}) + \sum_{i=1}^{n} \lambda_i \nabla^2 f_i(\mathbf{x})$ is a $(3n+6) \times (3n+6)$ matrix, $\mathbf{A}_2 = \nabla \mathbf{f}(\mathbf{x})$ is a $n \times (3n+6)$ matrix and $\mathbf{A}_3 = \nabla \mathbf{h}(\mathbf{x})$ is a $3n \times (3n+6)$ matrix. We have shown, using the structure of the matrices $\mathbf{A}_1$, $\mathbf{A}_2$, $\mathbf{A}_3$, that the system of linear equations can be solved in linear time. (We omit the proof here due to space constraints.)

VI. Results

We now present results illustrating our approach. To solve the distance computation problem, we used KNITRO 4.0, a commercially available software ([8], [46]). We use the primal-dual feasible interior point method in KNITRO, where all the iterates are feasible. The initial barrier parameter is set to $\mu = 0.1$ and reduced by an adaptive factor at each iteration based on the complementarity gap. For each value of μ the system of nonlinear equations is approximately solved by Newton's method with the step size determined by a trust region method.

We created six example objects (Figure 2), three of which are superquadrics. The indices and radii of the three superquadrics are $(\frac{4}{3}, \frac{7}{5}, \frac{15}{13})$ and $(1, 0.7, 1.5)$ for Object I (a diamond), $(\frac{23}{11}, \frac{11}{5}, \frac{179}{13})$ and $(1, 2, 1.7)$ for Object II (a soda can), and $(\frac{76}{9}, \frac{71}{5}, \frac{179}{13})$ and $(1, 1, 1.7)$ for Object III (a rounded cuboid). Object IV models a computer mouse and is represented as an intersection of a superquadric and 4 half spaces. The indices and radii of the superquadric are $(\frac{23}{11}, \frac{11}{5}, \frac{17}{7})$ and $(2, 1, 1.7)$. The half spaces are $x_1 \geq \frac{-1}{2}$, $x_1 \leq \frac{1}{2}$, $x_2 \geq -0.75$, and $x_3 \geq 0.4$ where x_1, x_2, x_3 are the local coordinates of the object. Object V is (the convex hull of) a rounded hexagonal nut modeled as the hyperquadric

$$|x_2|^{16} + |x_1 + 0.5x_2|^{16} + |x_1 - 0.5x_2|^{16} + |2.5x_3|^2 \leq 1.$$

Object VI is a pyramid modeled as the hyperquadric

$$|x_1+x_3|^{16}+|x_2+x_3|^{16}+|x_3|^{16}+|x_1-x_3|^{16}+|x_2-x_3|^{16} \leq 1.$$

The run time performance of the algorithm on the example objects is shown in Table I. The running times demonstrate that the distance computation rate is about 1 kHz, which is comparable to the rates required for real time applications like haptic simulations. For comparison with PQP [21], we also generated triangulations of these objects with approximately 9000 triangles each. PQP's distance queries (not collision detection) averaged about 1.4 milliseconds per query over 1000 random configurations. SOLID [41], [42], which supports proximity queries for quadrics without discretization, runs about 80 times faster than our approach for the case of ellipsoids. However SOLID cannot deal with superquadrics or hyperquadrics without discretization.

	Objects	Number of constraints	Proximity query time (millisecs)
1	I, II	2	1.00
2	I , III	2	0.98
3	II , III	2	0.87
4	III , IV	6	0.97
5	II , IV	6	0.80
6	III, V	2	0.97
7	V , VI	2	0.95

TABLE I
Sample run times, in milliseconds, for proximity queries between pairs of objects using KNITRO 4.0. The run times were computed for each pair by averaging the run times over 100,000 random configurations. All data was obtained on a 2.8 GHz Pentium IV machine with 512 MB of RAM.

VII. Conclusion

This paper demonstrates that recently developed interior point algorithms are particularly effective for computing the distance between two or more convex objects, where each object is described as an intersection of implicit surfaces. This broad class of convex objects includes convex polyhedra, quadrics, superquadrics, hyperquadrics, and their intersections. The global convergence properties of the interior point algorithms make them robust even in the absence of any initial information about the closest points. For the class of convex quadric surfaces, the algorithm has a polynomial running time and is guaranteed to terminate in a finite number of steps. To the best of our knowledge, this is the first approach with the demonstrated capability of performing distance queries for surfaces such as superquadrics and hyperquadrics. Another important advantage of this method is that it provides a uniform framework for distance computation between convex objects described as arbitrary intersections of polyhedra, quadrics, or any convex implicit surface. Further, the speed at which distance computation can be performed enables real-time dynamic simulations and haptic interactions.

There are several directions for future work. We would like to extend this approach to nonconvex objects, modeled as unions of convex shapes. We plan to explore alternative interior point algorithms (LOQO [43] and IPOPT [45], for example) to test their performance on the minimum distance problem. Performing warm starts, where a good

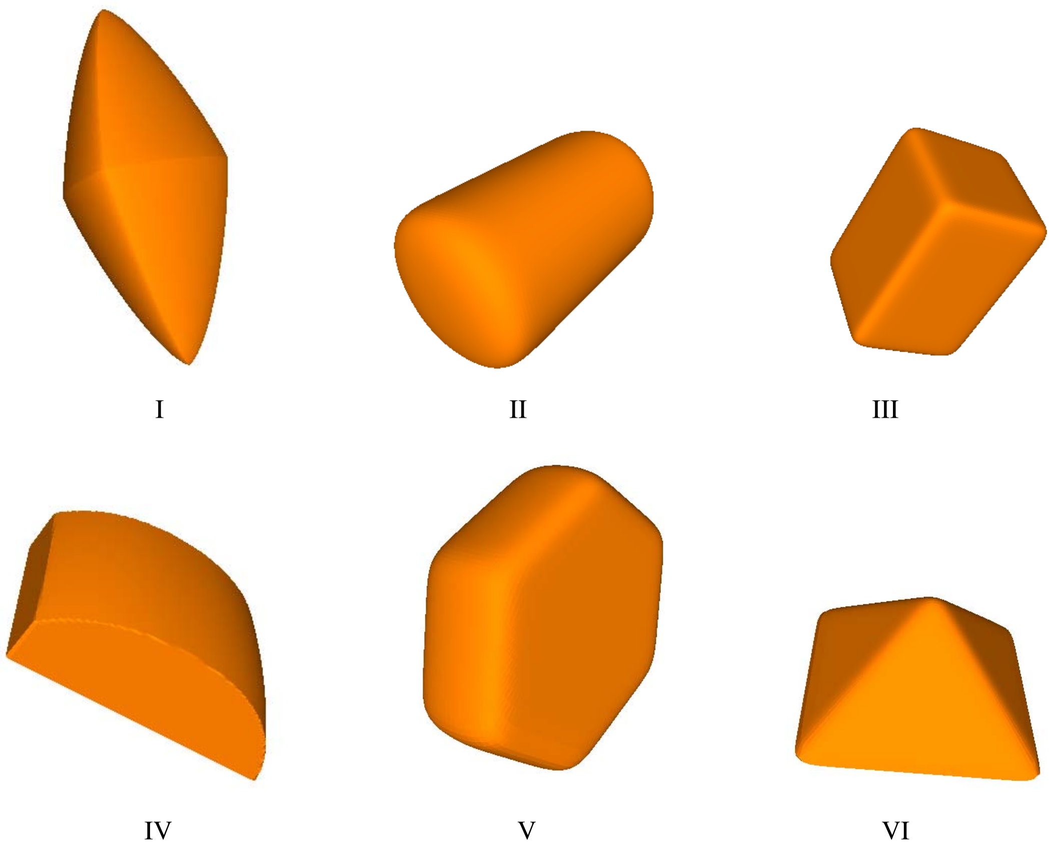

Fig. 2. Example objects. Objects I–III are superquadrics, IV is an intersection of superquadrics and halfspaces, and V–VI are hyperquadrics.

initial estimate for the solution is available, can potentially improve the running time when there is coherence. Longer term directions for future research include tracking closest points continuously, and extending this approach to performing continuous collision detection. Finally, it would be useful to integrate the proximity query algorithm with a dynamic simulation algorithm.

VIII. Acknowledgments

This work was supported in part by NSF under CAREER Award No. IIS-0093233. Thanks to Richard Waltz for help with KNITRO, Buck Clay for graphics software, and Jeff Trinkle, Frank Luk, and Steve Berard for useful discussions.

References

[1] D. Baraff. Curved surfaces and coherence for non-penetrating rigid body simulation. *Computer Graphics*, 24(4):19–28, August 1990.

[2] A. H. Barr. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, Jan. 1981.

[3] A. H. Barr. Local and global deformations of solid primitives. *Computer Graphics*, 18, 1984.

[4] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley, New York, second edition, 1993.

[5] J. E. Bobrow. A direct minimization approach for obtaining the distance between convex polyhedra. *International Journal of Robotics Research*, 8(3):65–76, June 1989.

[6] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2004.

[7] M. Buss and T. Schlegl. A discrete-continuous control approach to dextrous manipulation. In *IEEE International Conference on Robotics and Automation*, pages 276–281, 2000.

[8] R. Byrd, M. E. Hribar, and J. Nocedal. An interior point method for large scale nonlinear programming. *SIAM Journal on Optimization*, 9(4):877–900, 1999.

[9] S. Cameron. A comparison of two fast algorithms for computing the distance between convex polyhedra. *IEEE Transactions on Robotics and Automation*, 13(6):915–920, Dec. 1997.

[10] W. Chou and J. Xiao. An interactive approach to determining complex contact states. In *International Symposium on Assembly and Task Planning*, Montreal, Canada, 2005.

[11] V. Copolla and J. Woodburn. Determination of close approaches based on ellipsoidal threat volumes. In *Advances in the Astronomical Sciences: Spaceflight Mechanics*, volume 102, pages 1013–1024, 1999.

[12] D. P. Dobkin and D. G. Kirkpatrick. A linear algorithm for determining the separation of convex polyhedra. *Journal of Algorithms*, 6:381–392, 1985.

[13] A. Garcia and M. Hubbard. Spin reversal of the rattleback: Theory and experiment. *Proceedings of the Royal Society of London. Series A, Mathematical and Physical Sciences*, 418(1854):165–197, July 1988.

[14] E. G. Gilbert and C.-P. Foo. Computing the distance between general convex objects in three-dimensional space. *IEEE Transactions on Robotics and Automation*, 6(1):53–61, Feb. 1990.

[15] 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 Transactions on Robotics and Automation*, 4(2):193–203, Apr. 1988.

[16] A. J. Hanson. Hyperquadrics: smoothly deformable shapes with convex polyhedral bounds. *Computer Vision, Graphics, and Image Processing*, 44(2):191–210, Nov. 1988.

[17] A. Jaklic, A. Leonardis, and F. Solina. *Segmentation and Recovery of Superquadrics*. Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000.

[18] P. Jimenez, F. Thomas, and C. Torras. 3D collision detection: A survey. *Computers and Graphics*, 25(2):269–285, 2001.

[19] D. E. Johnson and E. Cohen. A framework for efficient minimum distance computations. In *IEEE International Conference on Robotics and Automation*, pages 3678–3684, Leuven, Belgium, May 1998.

[20] D. M. Kaufman, T. Edmunds, and D. K. Pai. Fast frictional dynamics for rigid bodies. In *ACM SIGGRAPH*, Aug. 2005.

[21] E. Larsen, S. Gottschalk, M. Lin, and D. Manocha. Fast distance queries using rectangular swept sphere volumes. In *IEEE International Conference on Robotics and Automation*, pages 3719–3726, San Francisco, CA, Apr. 2000.

[22] A. Lin and S.-P. Han. On the distance between two ellipsoids. *SIAM Journal of Optimization*, 13(1):298–308, 2003.

[23] M. Lin and D. Manocha. Efficient contact determination in dynamic environments. *International Journal of Computational Geometry and Applications*, 7(1 and 2):123–151, 1997.

[24] M. C. Lin and J. F. Canny. A fast algorithm for incremental distance calculation. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1008–1014, Sacramento, CA, Apr. 1991.

[25] M. C. Lin and D. Manocha. Collision and proximity queries. In J. E. Goodman and J. O'Rourke, editors, *Handbook of Discrete and Computational Geometry*, pages 787–808. Chapman and Hall/CRC Press, Boca Raton, FL, second edition, 2004.

[26] B. Mirtich. V-Clip: Fast and robust polyhedral collision detection. *ACM Transactions on Graphics*, 17(3):177–208, July 1998.

[27] B. Mirtich and J. Canny. Impulse-based dynamic simulation. In K. Y. Goldberg, D. Halperin, J.-C. Latombe, and R. H. Wilson, editors, *Algorithmic Foundations of Robotics*. A. K. Peters, Wellesley, Massachusetts, 1995.

[28] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Boca Raton, FL, 1994.

[29] J. Nocedal and S. J. Wright. *Numerical Optimization*. Springer-Verlag, New York, 1999.

[30] V. Patoglu and R. B. Gillespie. Extremal distance maintenance for parametric curves and surfaces. In *IEEE International Conference on Robotics and Automation*, Washington, DC, May 2002.

[31] R. Pelossof, A. Miller, P. Allen, and T. Jebara. An SVM learning approach to robotic grasping. In *IEEE International Conference on Robotics and Automation*, pages 3512–3518, New Orleans, LA, Apr. 2004.

[32] S. Quinlan. Efficient distance computation between non-convex objects. In *IEEE International Conference on Robotics and Automation*, pages 3324–3329, San Diego, CA, May 1994.

[33] E. Rimon and S. Boyd. Obstacle collision detection using best ellipsoid fit. *Journal of Intelligent and Robotic Systems*, 18(2):105–125, Feb. 1997.

[34] J. Sauer, E. Schomer, and C. Lennerz. Real-time rigid body simulations of some classical mechanics toys. In *10th European Simulation Symposium and Exhibition, ESS'98*, pages 93–98, 1998.

[35] T. Schlegl, M. Buss, T. Omata, and G. Schmidt. Fast dextrous regrasping with optimal contact forces and contact sensor-based impedance control. In *IEEE International Conference on Robotics and Automation*, pages 103–108, 2001.

[36] K.-A. Sohn, B. Juttler, M.-S. Kim, and W. Wang. Computing the distance between two surfaces via line geometry. In *Proceedings of the Tenth Pacific Conference on Computer Graphics and Applications*, pages 236–245, 2002.

[37] P. Song, J.-S. Pang, and V. Kumar. A semi-implicit time-stepping model for frictional compliant contact problems. *International Journal for Numerical Methods in Engineering*, 60:2231–2261, 2004.

[38] D. E. Stewart and J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and Coulomb friction. *International Journal of Numerical Methods in Engineering*, 39:2673–2691, 1996.

[39] J. C. Trinkle. Personal communication, 2005.

[40] C. Turnbull and S. Cameron. Computing distances between NURBS-defined convex objects. In *IEEE International Conference on Robotics and Automation*, pages 3685–3690, Leuven, Belgium, May 1998.

[41] G. van den Bergen. A fast and robust GJK implementation for collision detection of convex objects. *Journal of Graphics Tools*, 4(2):7–25, 1999.

[42] G. van den Bergen. Proximity queries and penetration depth computation on 3d game objects. In *Game Developers Conference*, 2001.

[43] R. J. Vanderbei. LOQO user's manual – version 3.10. Technical Report SOR 97-08, Statistics and Operations Research, Princeton University, Dec. 1997.

[44] R. J. Vanderbei and D. Shanno. An interior-point method for nonconvex nonlinear programming. Technical Report SOR 97-21, Statistics and Operations Research, Princeton University, 1997.

[45] A. Wachter and L. T. Biegler. On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming. Technical report, IBM T. J. Watson Research Center, Yorktown, USA, Mar. 2004. accepted for publication in Mathematical Programming.

[46] R. A. Waltz. *KNITRO 4.0 User's Manual*. Ziena Optimization, Inc., Evanston, IL, Oct. 2004.

[47] S. J. Wright. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics, Philadelphia, 1997.

[48] S. Zeghloul, P. Rambeaud, and J. Lallemand. A fast distance calculation between convex objects by optimization approach. In *IEEE International Conference on Robotics and Automation*, pages 2520–2525, Nice, France, May 1992.