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

Nilanjan Chakraborty, *Student Member, IEEE,* Jufeng Peng, Srinivas Akella, *Member, IEEE,* and John Mitchell

***Abstract*— This paper presents an interior point approach to exact distance computation between convex objects represented as intersections of implicit 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 sufficiently fast, and owing to their global convergence properties, are the only provably good choice for solving proximity query problems for some object classes. We use a primal-dual interior point algorithm that solves the KKT conditions obtained from the convex programming formulation. For the case of polyhedra and quadrics, we establish a theoretical time complexity of $O(n^{1.5})$, where n is the number of constraints. We present implementation results for example implicit surface objects, including polyhedra, quadrics, and generalizations of quadrics such as superquadrics and hyperquadrics, as well as intersections of these surfaces. We demonstrate that in practice, the algorithm takes time linear in the number of constraints, and that distance computation rates of about 1 kHz can be achieved. We also extend the approach to proximity queries between deforming convex objects. Finally, we show that continuous collision detection for linearly translating objects can be performed by solving two related convex optimization problems. For polyhedra and quadrics, we establish that the computational complexity of this problem is $O(n^{1.5})$.**

***Index Terms*— Proximity query, closest points, smooth objects, interior point algorithms, collision detection, dynamic simulation.**

I. Introduction

This paper studies the problem of computing the closest points on two convex objects, when each object is described as an intersection of implicit surfaces. Exact distance computation algorithms are usually used in the narrow phase of a collision detection algorithm in applications where knowledge of the closest points is required rather than just a yes/no answer for collision. Such applications are characterized by existence of intermittent contact, i.e., phases of contact and no contact between the objects, with a concomitant need to predict potential contact points. Some example applications are dynamic simulation ([1],[35],[51]), computer animation ([11]), dextrous manipulation ([37],[7]), and haptics ([33],[15]). Other applications where collision avoidance is the primary goal may also make use of the knowledge of the closest distance information. Examples of such applications include robot path planning [43] and spacecraft safe volume computations [16].

It is well known that the evolution of contact points for continuous contact states depends on the relative curvature of the two contacting bodies ([36], [27]). Polygonalization may lead to poor approximation of the object curvature and consequently affect the accuracy of the dynamic simulation. For applications in engineering analysis of objects in intermittent contact [50], such discretization may not be desirable. To the best of our knowledge, there is no published study on the effects of polygonal approximations even for simple dynamic systems. Therefore, to illustrate the effects of polygonalization, we simulated the motion of a disc rolling on a table. Figure 1 shows that the simulated circular disc rolling on a flat plane loses energy, when represented as a polygon, due to repeated impacts of the vertices with the table. More generally, polygonalized representations of smooth objects can lead to intermittent loss of contact and bouncing in the simulations. Another aspect of dynamic systems with intermittent contact that we are interested in is the effect of shape on the ensuing dynamics. The effect of shape and surface curvature on the dynamic motion of contacting objects is dramatically illustrated by a spinning boiled egg and toys such as the rattleback (or Celtic wobblestone) [47], [19]. Similarly, exact representation of shape is important when modeling robot fingers in contact with smooth objects during multi-finger dextrous manipulation (Figure 2). 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 ([7],[40]).

The general problem of distance computation between two

Nilanjan Chakraborty and Srinivas Akella are with the Department of Computer Science at Rensselaer Polytechnic Institute (Email: chakrn2@cs.rpi.edu; sakella@cs.rpi.edu). Jufeng Peng was with the Department of Mathematical Sciences at Rensselaer Polytechnic Institute and is currently with the Progressive Insurance Company (Email: jamespjf@gmail.com). John Mitchell is with the Department of Mathematical Sciences at Rensselaer Polytechnic Institute (Email: mitchj@rpi.edu). This work was supported in part by NSF under CAREER Award No. IIS-0093233.

The corresponding author is Srinivas Akella, Department of Computer Science, Rensselaer Polytechnic Institute, 110 Eighth Street, Troy, New York 12180, USA. Tel: (518) 276-8770, Fax: (518) 276-4033, Email: sakella@cs.rpi.edu.

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 $\mathbb{R}^2$ or $\mathbb{R}^3$ and the points $\mathbf{x}_g$ and $\mathbf{y}_g$ are points in the two objects. This problem has been extensively studied ([25], [32]), mainly for polyhedral object representations ([18], [21], [31], [34]). In this paper, we focus on representing the sets X and Y as intersections of implicit surfaces, including planes, quadrics, superquadrics, and hyperquadrics. We assume that an implicit surface model of each object is given to us. Our choice of object representation is motivated by the goal of simulating systems with smooth objects, where polygonal discretizations may not be desirable. (While parametric representations can also represent smooth objects, they provide a surface description of objects with nonlinear equations and thus the resulting problem is not a convex optimization problem even for convex objects.) The literature on distance computation between general implicit surfaces is relatively sparse because, with a few notable exceptions ([20],[53]), methods for polyhedral representations do not easily generalize to implicit surfaces. Having a smooth representation of objects and an algorithm to perform distance computation between such representations will enable the study of the effects of shape and polygonalization on dynamic simulation of systems with intermittent contact.

Contributions of the paper: This paper focuses on the problem of computing the minimum distance between two convex objects, where each object is described as an intersection of implicit surfaces. This class of convex objects includes for example, convex polyhedra, quadrics, superquadrics, and hyperquadrics. While the distance computation problem for convex objects represented by convex inequalities 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], [59] to compute the distance between convex implicit surface objects and demonstrate that it is particularly effective for this class of problems. For polyhedral and quadric surfaces, the algorithm takes $O(n^{1.5})$ time, where n is the number of constraints. We also illustrate the approach on surfaces such as superquadrics and hyperquadrics. To the best of our knowledge, this is the first approach with this demonstrated capability (without discretization). 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. We present implementation results for example implicit surface objects that show that the algorithm exhibits linear time performance in practice, and demonstrate that distance computation rates of about 1 kHz can be achieved. We also extend the approach to proximity queries between deforming convex objects. Finally, we show that continuous collision detection for linearly translating objects can be performed by solving two related convex optimization problems. For polyhedra and quadrics, we establish that the time complexity of this continuous collision detection problem is $O(n^{1.5})$.

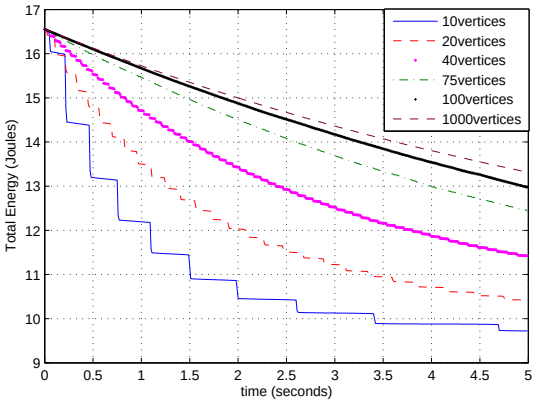


Fig. 1. Energy plots of a circle rolling on a table, with the circle approximated as a polygon with n vertices. As n increases, the polygon better approximates the circle and the energy loss decreases. The dynamic simulation uses the Stewart-Trinkle [51] time stepping formulation and assumes Coulomb friction and inelastic impacts [4].

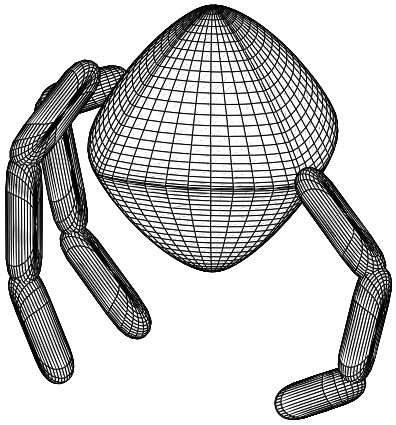


Fig. 2. A dextrous manipulation task that requires closest distance computations to predict the contact points of fingers with an object. The fingers and object are represented as superquadrics.

The 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. Section VI provides theoretical and practical results on the complexity of the closest point algorithm. Section VII extends the approach to continuous proximity queries for linearly translating objects. We present our implementation results in Section VIII and conclude with a discussion of future work in Section IX. A preliminary version of this work appeared in [13].

II. Related Work

Proximity queries for polyhedra: Proximity queries and collision detection algorithms have an extensive literature in computational geometry [18], robotics [21], [31], and computer graphics [53]. We provide a sampling of the related work in these areas; see [32] and [25] for 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 [21], Lin-Canny [31], and V-Clip [34]. GJK [21] 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 [31] 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 [10]. Bobrow [5] proposed an optimization based approach for computing the distance between two convex 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 [62].

Proximity queries for quadrics and NURBS: Distance estimation between non-polyhedral shapes has focused primarily on quadrics and NURBS surfaces. Among the algorithms for polyhedra, only GJK has been extended directly for smooth convex objects [20]; van den Bergen [53] 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 polynomial equations with fractional exponents. Turnbull and Cameron [52] 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 [30] 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). Schomer et al. [48] describe a collision detection algorithm for curved objects bounded by quadric surface patches by finding roots of univariate polynomials of degrees 4 and 8. Johnson and Cohen [26] 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 [39] 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. In fact, no closed form solution exists even for the distance between a point and an ellipsoid. Most closely related is recent work on computing the distance between two ellipsoids and other conic sections ([29], [16], [49]). Sohn et. al. [49] 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 [16] 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 [46] 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. Choi et al. [14] present a continuous collision detection algorithm for two elliptical disks moving in the plane. Collisions are identified by checking for the real roots of the univariate discriminant of the characteristic

equation of the two moving ellipses. Although superquadrics are a generalization of quadrics, the problem in generalizing the methods in [1],[21],[16],[49] 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 $U \subseteq \mathbb{R}^n$ is called a convex set if for any two points $\mathbf{u}_1$, $\mathbf{u}_2 \in U$ and any λ with $0 \leq \lambda \leq 1$, we have

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

Convex Function: A function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ 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 $0 \leq \lambda \leq 1$, we have

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

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 [3]. 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 [24] is defined by the equation

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

Although the definition here differs slightly from that in [2], the two definitions are equivalent [24]. Convex superquadrics are a broad class of shapes that include cuboids, rounded cuboids, ellipsoids, spheres, and (rounded) octahedra. The planes $\left|\frac{x_i}{a_i}\right| \leq 1$, $i = 1,2,3$ define a bounding cube for the superquadric and the indices control the roundedness of the shape. Different shapes can be obtained by varying n_i. The shape is a rhomboid when $n_i = 1$, and the shape becomes a cube as n_i tends to infinity.

Hyperquadric: A hyperquadric [24] is defined by the equation

$$\begin{aligned} & f(\mathbf{x}) = \sum_{i=1}^{N} |H_i(\mathbf{x})|^{n_i} - 1 = 0 \text{ where } N \geq 3 \text{ and} \\ & H_i(\mathbf{x}) = (a_i x_1 + b_i x_2 + c_i x_3 + d_i) \\ & n_i = l_i/m_i, \quad l_i, m_i \in \mathbb{Z}^+ \\ & f(\mathbf{x}) \text{ convex if } \quad 1 \leq n_i < \infty \\ & f(\mathbf{x}) \text{ nonconvex if } \quad 0 < n_i < 1 \end{aligned} \tag{4}$$

Hyperquadrics are a more general class of shapes than superquadrics. In particular, they include asymmetric shapes. In this case also, the intersection of the planes $H_i(\mathbf{x}) \leq 1$ form a bounding polytope for the hyperquadric and the indices control the roundedness of the shape.

Self-concordant functions: A convex function $f : \mathbb{R} \rightarrow \mathbb{R}$ is self-concordant if $|f'''(x)| \leq 2f''(x)^{3/2}$ for all $x \in dom f$. A convex function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is self-concordant if it is self-concordant along every line in its domain (see [6] for details).

IV. Problem Formulation

In this section we present the formulation of the minimum distance computation problem. We first outline a geometric approach to the minimum distance problem that has been popular in prior work on non-polyhedral objects ([1],[30],[49]) and connect it to an optimization formulation. Let f_X be an implicit function representing object X and f_Y be an implicit function representing object Y, and let $\mathbf{x}_g$, $\mathbf{y}_g$ be the global coordinates of points in X and Y respectively. To compute the closest distance between X and Y, the approach uses the geometric condition that the normals on the two surfaces at the closest points are aligned with each other. Using this and the condition that the closest points should lie on the surfaces of the two objects, we can obtain the closest points by solving the following system of nonlinear equations

$$\begin{aligned} \mathbf{x}_g - \mathbf{y}_g &= -\lambda_X \nabla f_X(\mathbf{x}_g) \\ \mathbf{x}_g - \mathbf{y}_g &= \lambda_Y \nabla f_Y(\mathbf{y}_g) \\ f_X(\mathbf{x}_g) &= 0 \\ f_Y(\mathbf{y}_g) &= 0 \end{aligned} \tag{5}$$

where λ_X and λ_Y are scalars. The conditions given by Equation 5 are precisely the Karush-Kuhn-Tucker (KKT) conditions for solving the following optimization problem.

$$\begin{aligned} \text{Minimize} \quad & \|\mathbf{x}_g - \mathbf{y}_g\|_2 \\ \text{subject to:} \quad & f_X(\mathbf{x}_g) = 0 \\ & f_Y(\mathbf{y}_g) = 0 \end{aligned} \tag{6}$$

However note that when f_X and f_Y are nonlinear, the above problem is nonconvex even when the objects are convex and the solution can therefore get stuck in a local minimum.

We now formulate the problem of minimum distance computation between two convex objects as a convex optimization problem. Each object is assumed to be described as an intersection of a finite number of implicit primitives (i.e., a finite number of algebraic inequalities). In general, each of the intersecting surfaces may be specified in its own reference frame. The distance computation problem of Equation 1 can then be written as

$$\begin{aligned} \text{Minimize} \quad & \|\mathbf{x}_g - \mathbf{y}_g\|_2^2 \\ \text{subject to:} \quad & \mathbf{x}_g = \mathbf{R}_{xi}\mathbf{x}_{li} + \mathbf{p}_{xi} \quad i = 1, \ldots, m \\ & \mathbf{y}_g = \mathbf{R}_{yj}\mathbf{y}_{lj} + \mathbf{p}_{yj} \quad j = m+1, \ldots, n \\ & f_i(\mathbf{x}_{li}) \leq 0 \quad i = 1, \ldots, m \\ & f_j(\mathbf{y}_{lj}) \leq 0 \quad j = m+1, \ldots, n \end{aligned} \tag{7}$$

where $\mathbf{x}_g$, $\mathbf{y}_g \in \mathbb{R}^3$ 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 the reference frame of each of the intersecting surfaces with respect to the global frame, $\mathbf{x}_{li}$, $\mathbf{y}_{lj} \in \mathbb{R}^3$ are the coordinates of the points in the local reference frames of the surfaces, and f_k ($k = i, j$) are the functions representing the implicit surfaces in the local reference frames. The above system has $3n$ linear equality constraints and n inequality constraints. Note that the linear constraints in Equation 7 can be any affine transformation (not necessarily a rigid body transformation). In particular, we can handle global deformations like nonuniform scaling by post multiplying the rotation matrix with a scaling matrix.

From the linear constraints in Equation 7 we can easily evaluate $\mathbf{x}_{li}$ and $\mathbf{y}_{lj}$ and so the inequality constraints can be expressed in terms of $\mathbf{x}_g$ and $\mathbf{y}_g$. Therefore, without loss of generality, we can assume that the implicit surface describing the objects is described in a global reference frame. The distance computation problem of Equation 1 is then given by

$$\begin{aligned} \text{Minimize} \quad & \|\mathbf{x}_g - \mathbf{y}_g\|_2^2 \\ \text{subject to:} \quad & f_i(\mathbf{x}_g) \leq 0 \quad i = 1, \ldots, m \\ & f_j(\mathbf{y}_g) \leq 0 \quad j = m+1, \ldots, n \end{aligned} \tag{8}$$

where f_k ($k = i, j$) are the functions representing the implicit surfaces in the global reference frame. The above system has n inequality constraints. The objective function in Equation 8 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 general convex surfaces, the distance computation problem is a nonlinear program (NLP). For objects described as convex quadric surfaces, the problem reduces to a quadratically constrained quadratic program (QCQP), and if the objects are convex polyhedra (intersections of planes), the closest distance problem becomes a quadratic programming (QP) problem.

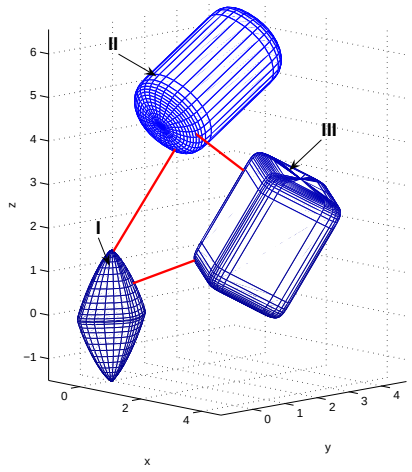


Fig. 3. Three example objects. The closest points of each pair of objects are shown connected by a line segment.

The solution to the minimum distance problem of Equation 8 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 [8] for solving this problem. See Figure 3 for an example solution generated using an interior point algorithm. Interior point methods [60] 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. Sequential quadratic programming (SQP) is another method for solving general nonlinear programming problems [22]. In contrast to SQP, interior point methods have polynomial time convergence guarantees for certain convex problems, as we describe in Section VI. Moreover, an informal comparison of SQP implementations with interior point algorithm implementations on the NEOS server [17] shows the interior point methods to be slightly faster. Therefore, we choose to solve the optimization problem with an interior point algorithm.

V. Interior Point Algorithm

In this section, we present the *primal-dual* interior point algorithm for solving the optimization problem described in Equation 8. The Karush-Kuhn-Tucker (KKT) conditions give necessary and sufficient conditions for solving the minimum distance problem in Equation 8, since it is a convex optimization problem and satisfies Slater constraint qualification. For ease of presentation, we rewrite Equation 8 in a general

nonlinear program format as

$$\begin{aligned} \text{Minimize} \quad & f_0(\mathbf{x}) \\ \text{subject to:} \quad & \mathbf{f}(\mathbf{x}) + \mathbf{s} = \mathbf{0} \\ & \mathbf{s} \geq \mathbf{0} \end{aligned} \tag{9}$$

where $f_0(\mathbf{x}) = \|\mathbf{x}_g - \mathbf{y}_g\|_2^2$, $\mathbf{x} = [\mathbf{x}_g^T, \mathbf{y}_g^T]^T$ is a 6×1 column vector, $\mathbf{s}$ is an $n \times 1$ column vector of slack variables, and $\mathbf{f} : \mathbb{R}^6 \rightarrow \mathbb{R}^n$ is the vector of inequality constraints. The Lagrangian for the above constrained optimization problem can be written as

$$f_0(\mathbf{x}) + \lambda^{\mathbf{T}}(\mathbf{f}(\mathbf{x}) + \mathbf{s}) \tag{10}$$

where λ is an $n \times 1$ vector of Lagrange multipliers. The KKT conditions for Equation 9 are the system of nonlinear equations below:

$$\begin{aligned} \nabla f_0(\mathbf{x}) + (\nabla \mathbf{f}(\mathbf{x}))^T \lambda &= \mathbf{0} \\ \mathbf{f}(\mathbf{x}) + \mathbf{s} &= \mathbf{0} \\ \mathbf{LSe} &= \mathbf{0} \end{aligned} \tag{11}$$

Here $\mathbf{L}$ is an $n \times n$ diagonal matrix of the λ variables, $\mathbf{S}$ is an $n \times n$ diagonal matrix of the slack variables $\mathbf{s}$, and $\mathbf{e}$ is an n-vector of ones. The above is a system of $2n + 6$ nonlinear equations in the $2n + 6$ variables $\mathbf{x}$, λ, $\mathbf{s}$. Equation 11 can be solved by Newton's method for solving systems of nonlinear equations. It converges to the correct solution if the initial guess is *near enough* [38]. 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. The main difficulty in using Newton's method is that we have to ensure $\mathbf{s} \geq 0$, which may lead to very small step lengths that result in convergence problems.

Interior point methods are a general class of algorithms for solving nonlinear programming problems. In essence, these methods approximately solve a sequence of systems of nonlinear equations that are formed by perturbing the complementarity equations ($\mathbf{LSe} = 0$) in the KKT conditions. Following [6], we present the interior point method by reformulating Equation 9 as a *barrier* problem.

$$\begin{aligned} \text{Minimize} \quad & f_0(\mathbf{x}) - \mu \sum_{i=1}^{n} \ln(s_i) \\ \text{subject to:} \quad & \mathbf{f}(\mathbf{x}) + \mathbf{s} = \mathbf{0} \end{aligned} \tag{12}$$

The formulation in Equation 12 is the *barrier* formulation [6] of the minimum distance problem of Equation 9 and μ is called the barrier parameter, with $\mu > 0$. Equation 12 differs from Equation 9 in that the nonnegativity constraints on $\mathbf{s}$ are not present explicitly, but are implicit in the objective. The Lagrangian for the above constrained optimization problem can be written as

$$f_0(\mathbf{x}) - \mu \sum_{i=1}^{n} \ln(s_i) + \lambda^{\mathbf{T}}(\mathbf{f}(\mathbf{x}) + \mathbf{s}) \tag{13}$$

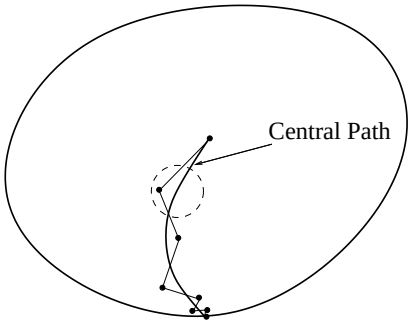


Fig. 4. Schematic illustration of the interior point method for a path following algorithm. The convex region represents the feasible set. The central path is an arc of strictly feasible points that solve Equation 14 as the parameter μ approaches 0. The progress of the iterates generated by the interior point solver is indicated by the polygonal line connecting them. The iterates are guaranteed to lie within a neighborhood, represented by the circular ball, of the central path.

where λ is the vector of Lagrange multipliers. Thus, the KKT conditions can be written as

$$\begin{aligned} \nabla f_0(\mathbf{x}) + (\nabla \mathbf{f}(\mathbf{x}))^T \lambda &= \mathbf{0} \\ \mathbf{f}(\mathbf{x}) + \mathbf{s} &= \mathbf{0} \\ \mathbf{LSe} - \mu\mathbf{e} &= \mathbf{0} \end{aligned} \tag{14}$$

The above equation represents a system of $2n + 6$ nonlinear equations in $2n + 6$ variables and can be approximately solved for a given μ. Note that Equation 11 and Equation 14 differ in the complementarity conditions. As the barrier parameter μ approaches 0, the KKT conditions for the barrier problem (Equation 14) approach the KKT conditions of the original problem (Equation 11). See Figure 4 for a schematic illustration of the interior point method.

For our proximity query problem, feasible interior point methods generate iterates that yield points guaranteed to lie inside the objects and converge towards the closest points on the boundaries of the objects. See the example proximity query in Figure 5.

The general structure of interior point methods is indicated in Algorithm 1, where termination criterion 1 is the ending condition for the whole problem (Equation 11) and termination criterion 2 is the ending condition for approximately solving Equation 14 for the current value of μ. The outer while loop determines the number of times μ has to be updated, i.e., the number of times Equation 14 has to be approximately solved for the sequence of μ values. The inner while loop is a variant of Newton's method used for approximately solving Equation 14 for a fixed value of μ. The different interior point implementations (KNITRO [8], [59], LOQO [56], [57], IPOPT [58]) vary in the way they calculate the step lengths for a particular value of μ, the termination criteria they use, and the way in which they update μ.

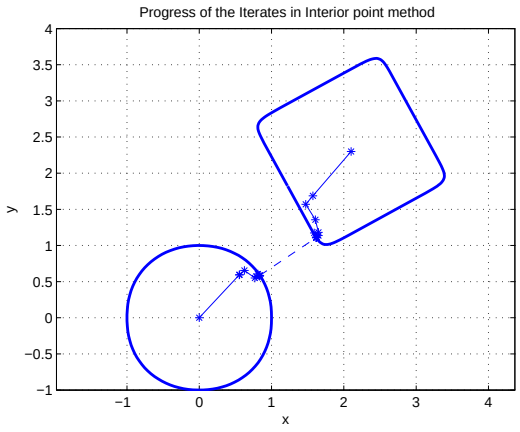


Fig. 5. Example illustrating the sequence of closest point estimates generated by the interior point method for two 2D superquadric objects, with indices $(\frac{23}{11}, \frac{11}{5})$, $(\frac{76}{7}, \frac{71}{5})$ and semiaxes 1. The iterates of the interior point method are mapped to corresponding points in the objects.

Algorithm 1 Interior point algorithm

Input: initial strictly feasible $\mathbf{x}_0$*, initial barrier parameter* μ_0*, specified tolerance* ϵ*, and KKT equations*
Output: Closest points solution $\mathbf{x}$

$k \leftarrow 0$
while termination criterion 1 not satisfied **do**
  while termination criterion 2 not satisfied **do**
    Solve the system of linear equations // *determine Newton direction*
    Determine step length α_k by line search
    $\mathbf{x}_{k+1} \leftarrow \mathbf{x}_k + \alpha_k \Delta \mathbf{x}_k$
    $\mathbf{s}_{k+1} \leftarrow \mathbf{s}_k + \alpha_k \Delta \mathbf{s}_k$
    $\lambda_{k+1} \leftarrow \lambda_k + \alpha_k \Delta \lambda_k$
    $k \leftarrow k + 1$
  end while
  $\mu \leftarrow c\mu$ // $c < 1$, *may be constant or adaptive*
end while
return $\mathbf{x}_k$

VI. Computational Complexity

The total cost of solving a problem using Algorithm 1 is the product of the total number of iterations (considering both loops) and the computational effort in solving the system of linear equations to determine the Newton direction in each iteration. The system of linear equations to be solved to determine the Newton direction is

$$\begin{pmatrix} \mathbf{A_1} & \mathbf{A_2}^T & \mathbf{0}_{6\times n} \\ \mathbf{A_2} & \mathbf{0}_{n\times n} & \mathbf{I}_{n\times n} \\ \mathbf{0}_{6\times n} & \mathbf{S} & \mathbf{L} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\lambda \\ \Delta\mathbf{s} \end{pmatrix} = \begin{pmatrix} -\mathbf{F}_1 \\ -\mathbf{F}_2 \\ -\mathbf{F}_3 \end{pmatrix} \tag{15}$$

where $\mathbf{A}_1 = \nabla^2 f_0(\mathbf{x}) + \sum_{i=1}^{n} \lambda_i \nabla^2 f_i(\mathbf{x})$ is a 6×6 matrix, $\mathbf{A}_2 = \nabla \mathbf{f}(\mathbf{x})$ is a $n \times 6$ matrix, the definitions of $\mathbf{S}$ and $\mathbf{L}$ are the same as before, and $\mathbf{F}_1 = \nabla f_0(\mathbf{x}) + (\nabla \mathbf{f}(\mathbf{x}))^T \lambda$, $\mathbf{F}_2 = \mathbf{f}(\mathbf{x}) + \mathbf{s}$, $\mathbf{F}_3 = \mathbf{LSe} - \mu\mathbf{e}$.

In general, the computational cost of solving a system of n linear equations in n unknowns is $O(n^3)$. However, we now establish that the system of linear equations can be solved in $O(n)$ time. By simple algebraic manipulation of the above equations, we obtain the following formulas for $\Delta\mathbf{x}$, $\Delta\lambda$, and $\Delta\mathbf{s}$:

$$\begin{aligned} \Delta\mathbf{x} &= \mathbf{G}^{-1}(-\mathbf{F}_1 - \mathbf{A}_2^T(\mathbf{S}^{-1}\mathbf{L})(\mathbf{F}_2 + \mathbf{L}^{-1}\mathbf{F}_3)) \\ \Delta\lambda &= (\mathbf{S}^{-1}\mathbf{L})(\mathbf{A}_2\Delta\mathbf{x} - \mathbf{F}_2 + \mathbf{L}^{-1}\mathbf{F}_3) \\ \Delta\mathbf{s} &= -\mathbf{L}^{-1}(\mathbf{F}_3 + \mathbf{S}\Delta\lambda) \end{aligned} \tag{16}$$

where $\mathbf{G} = \mathbf{A}_1 + \mathbf{A}_2^T(\mathbf{S}^{-1}\mathbf{L})\mathbf{A}_2$. Since $\mathbf{G}$ is a 6×6 matrix, $\mathbf{G}^{-1}$ can be computed in constant time. Moreover, $\mathbf{S}$ and $\mathbf{L}$ are $n \times n$ diagonal matrices, so $\mathbf{S}^{-1}$ and $\mathbf{L}^{-1}$ can be computed in linear time. Thus we can compute all the inverses in $O(n)$. Moreover, noting the dimensions of $\mathbf{A}_1$ and $\mathbf{A}_2$, we can see by inspection that the matrix multiplication also requires $O(n)$ operations. So Equation 16 can be evaluated in $O(n)$ time, or in other words, the computation of the Newton step takes $O(n)$ time. Note that we have not made any assumptions regarding the primitive surface describing the object. Thus this analysis is valid for any implicit surface (including planes, quadrics, superquadrics, hyperquadrics, etc.) and for intersections of these implicit surfaces.

For general functions, there is no known bound on the total number of iterations (including both while loops). However, if the log barrier function of the implicit surface constraints is a self-concordant function (refer to Section III), the number of Newton iterations (which is the number of times Equation 15 must be solved) is polynomial in a parameter depending on the structure of the function. For polyhedral constraints and quadric constraints the number of Newton iterations required for converging to the optimal solution is $O(n^{0.5})$. This implies that the theoretical complexity of our approach for polyhedra and quadrics is $O(n^{1.5})$. Although algorithms with theoretical linear time guarantees are available for polyhedra, for the case of quadrics, as far as we know, this is the best known bound. Moreover, our experiments indicate the algorithm exhibits linear time behavior in practice, as shown in Figure 6 and Figure 7.

For superquadrics and hyperquadrics, the log barrier function might not be self-concordant in the general case. However, note that superquadric and hyperquadric functions have self-concordant barriers because they define convex regions, and every convex region has a self-concordant barrier (its universal barrier). If this barrier is too hard to find to be useful computationally, an alternative is to decompose the function into simpler functions (for example, as second order cones [41]) such that the sum of the barriers for the simpler functions gives a barrier for the original superquadric or hyperquadric. However these representations may lead to computationally slower solutions due to the increased number

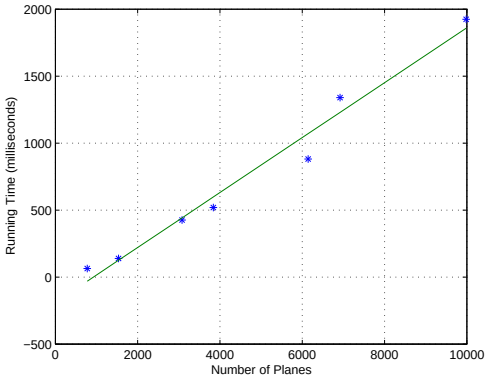


Fig. 6. Plot showing observed linear time behavior of the interior point algorithm for polyhedra.

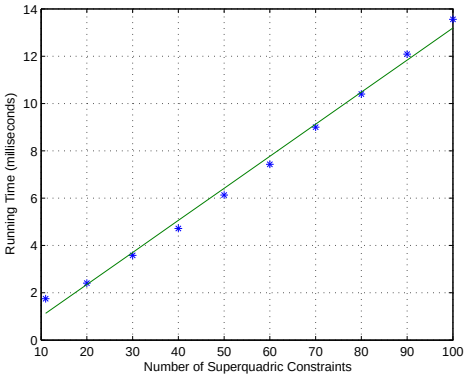


Fig. 8. Plot showing observed linear time behavior of the interior point algorithm for superquadrics.

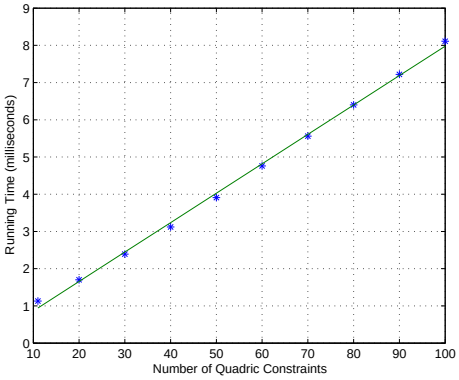


Fig. 7. Plot showing observed linear time behavior of the interior point algorithm for quadrics.

of variables and constraints. Glineur and Terlaky [23] provide self-concordant formulations for l_p-norm minimization that apply to superquadrics and hyperquadrics. However the computational performance of these formulations has not yet been explored in the literature. Moreover, the observed time complexity of the interior point algorithm is *linear* for this class of shapes (Figure 8), which implies that in practice the number of iterations is constant, i.e., independent of the size of the problem. The observed linear time behavior of the interior point algorithm even without self-concordant representations further justifies the use of an interior point-based solver for this generic nonlinear programming formulation.

VII. Continuous Proximity Queries for Translating Objects

We now address the problem of continuous proximity queries for two linearly translating objects. Such queries can be useful in identifying feasible object motions during assembly planning. The goal is to determine the exact time at which the two moving objects are closest, without discrete sampling of their configurations. This computation of the closest distance between two swept objects is closely related to the problem of continuous collision detection ([12], [9], [61], [44], [45], [55]), where the time of first contact between two colliding objects is to be determined; however most prior work has been restricted to polyhedral objects. The advantage of such continuous collision detection methods is the ability to detect collisions even for fast moving objects in the presence of thin obstacles.

We address the continuous collision detection problem for linearly translating objects by solving two related convex optimization problems. Assume the objects are moving along piecewise linear paths. Let X and Y be two objects described by $f_X(\mathbf{x}_l) \leq 0$ and $f_Y(\mathbf{y}_l) \leq 0$. Let the two objects be linearly translating along the directions specified by the unit vectors $\hat{\mathbf{g}}_x$ and $\hat{\mathbf{g}}_y$ with constant velocities v_x and v_y respectively. The following optimization problem finds the minimum distance between the two objects in the time interval $[0, t_{max}]$, where each object is moving along a single line segment. If the minimum distance is greater than zero, the solution provides the closest points and the time t at which the objects are closest. See Figure 9.

$$\begin{aligned} \text{Minimize} \quad & \|\mathbf{x}_g - \mathbf{y}_g\|_2^2 \\ \text{subject to:} \quad & \mathbf{x}_g = \mathbf{R}_x \mathbf{x}_l + \mathbf{p}_x + v_x t \hat{\mathbf{g}}_x \\ & \mathbf{y}_g = \mathbf{R}_y \mathbf{y}_l + \mathbf{p}_y + v_y t \hat{\mathbf{g}}_y \\ & f_X(\mathbf{x}_l) \leq 0 \\ & f_Y(\mathbf{y}_l) \leq 0 \\ & t \leq t_{max} \\ & t \geq 0 \end{aligned} \tag{17}$$

When the objects intersect, the problem above has multiple solutions corresponding to zero distance. The time of first contact can be obtained by solving another convex optimization problem using the solution of Equation 17. Let Q be

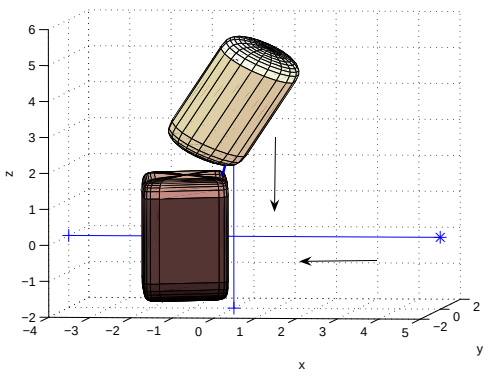


Fig. 9. Computing the instant of closest distance using the continuous proximity query. The bold blue line connects the closest points on the two objects, as they translate along the indicated line segments.

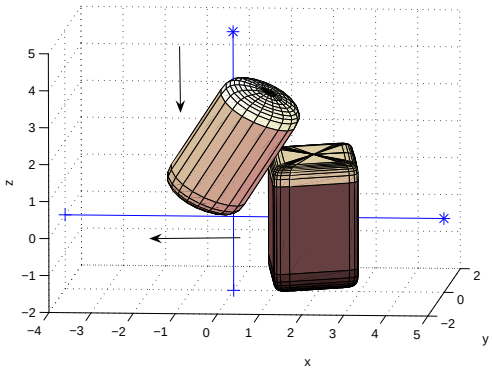


Fig. 10. Computing the time of first contact using the continuous proximity query gives the solution to the continuous collision detection problem.

the set of intersecting points and $\mathbf{q} \in Q$ be a point specified in the global coordinate system. To obtain the time of first contact, we solve the problem below, starting from an initial feasible guess that is the solution of Equation 17.

$$\begin{aligned} \text{Minimize} \quad & t \\ \text{subject to:} \quad & \mathbf{q} = \mathbf{R}_x \mathbf{x}_l + \mathbf{p}_x + v_x t \hat{\mathbf{g}}_x \\ & \mathbf{q} = \mathbf{R}_y \mathbf{y}_l + \mathbf{p}_y + v_y t \hat{\mathbf{g}}_y \\ & f_X(\mathbf{x}_l) \leq 0 \\ & f_Y(\mathbf{y}_l) \leq 0 \\ & t_p \geq t \geq 0 \end{aligned} \tag{18}$$

where t_p is the time obtained from the solution of Equation 17. See the example in Figure 10.

We now establish that the computational complexity of solving the Newton step in the continuous collision detection problem along a single linear segment is $O(n)$, which is the same as for the static query problem. We can eliminate $\mathbf{x}_l$ and $\mathbf{y}_l$ from Equation 17 and can write it in the form of Equation 9 where $\mathbf{x}$ is a 7×1 column vector that includes t. Thus in Equation 15, $\mathbf{A}_1$ is a 7×7 matrix and $\mathbf{A}_2$ is an $n \times 7$ matrix. This implies that $\mathbf{G}$ is still a constant sized 7×7 matrix and the complexity argument for solving Equation 16 in Section VI applies directly. Similarly in Equation 18, $\mathbf{x}$ is a 4×1 vector consisting of $\mathbf{q}$ and t. Thus $\mathbf{A}_1$ is a 4×4 matrix and $\mathbf{A}_2$ is an $n \times 4$ matrix and $\mathbf{G}$ is a 4×4 matrix. Hence the computation of the Newton step in both problems takes $O(n)$ time irrespective of the implicit primitive. Moreover, as the constraints have a self-concordant log barrier function for the case of planes and quadrics, the overall complexity is $O(n^{1.5})$ in these cases.

Obtaining the collision interval for motion with known linear paths: Computing the path intervals over which two robots can collide is useful in multiple robot coordination problems where the robots travel along known paths [42]. When we are given the paths traversed by the objects and wish to determine the path intervals over which the objects could collide with each other, we solve the following optimization problem.

$$\begin{aligned} \text{Minimize} \quad & s_x \\ \text{subject to:} \quad & \mathbf{x}_g = \mathbf{R}_x \mathbf{x}_l + \mathbf{p}_x + s_x \hat{\mathbf{g}}_x \\ & \mathbf{y}_g = \mathbf{R}_y \mathbf{y}_l + \mathbf{p}_y + s_y \hat{\mathbf{g}}_y \\ & f_X(\mathbf{x}_l) \leq 0 \\ & f_Y(\mathbf{y}_l) \leq 0 \\ & f_X(\mathbf{y}_g) \leq 0 \\ & f_Y(\mathbf{x}_g) \leq 0 \\ & s_x, s_y \geq 0 \end{aligned} \tag{19}$$

Here s_x and s_y represent the path lengths traversed by the objects. The inequalities $f_X(\mathbf{y}_g) \leq 0$ and $f_Y(\mathbf{x}_g) \leq 0$ describe all points in the intersection of both objects, and also encode the constraint that the distance between the objects is zero. In fact, we have to solve three more variations of the above problem with the objective function minimizing $-s_x$, s_y, and $-s_y$ to find the minimum and maximum values of s_x and s_y. These define corresponding collision intervals over the path for each object.

VIII. Results

We now present results illustrating our approach. To solve the distance computation problem, we used KNITRO 5.0, a commercially available interior point based solver ([8], [59]). We use the primal-dual feasible interior point method in KNITRO, where all the iterates are feasible. We have an initial feasible solution trivially from points at the centers of the objects. The barrier parameter μ is initially set to 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 [38].

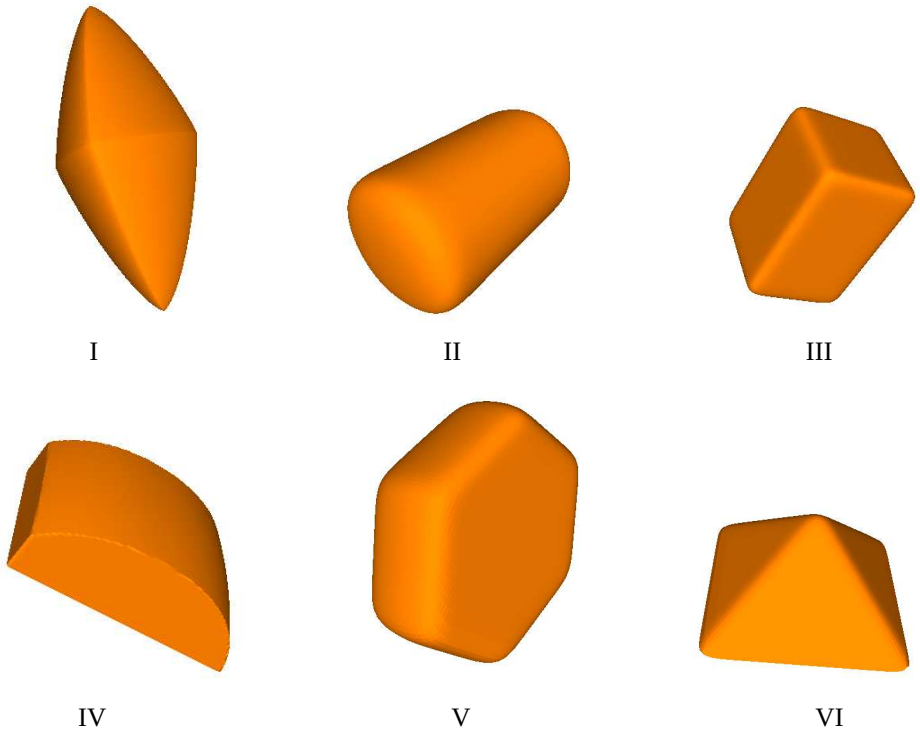


Fig. 11. Example objects. Objects I–III are superquadrics, IV is an intersection of superquadrics and halfspaces, and V–VI are hyperquadrics.

We depict six example objects in Figure 11, three of which are superquadrics. The indices and semiaxes 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 $(1, 1, 1.7)$ for Object II (a soda can), and $(\frac{76}{9}, \frac{71}{5}, \frac{179}{13})$ and $(1, 1, 1.5)$ 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 semiaxes 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, with some test cases depicted in Figure 3. All data was obtained on a 2.2 GHz Athlon 64 X2 4400+ machine with 2 GB of RAM. The running times demonstrate that the distance computation rate is about 1 kHz, which is sufficiently fast for real-time dynamic simulations and interactive haptic simulations. We also generated triangulations of these objects with about 19,000 triangles and found that the distance computation time (not collision detection time) taken by PQP [28], a popular collision detection software, was comparable for our examples. Note that our randomly generated object configurations do not provide the benefits of coherence. We also compared our algorithm on quadric surfaces against SOLID [53], [54], which supports proximity queries for quadrics without discretization. SOLID runs about 80 times faster than our approach for the case of ellipsoids. However, our algorithm has a theoretical guarantee, and SOLID cannot deal with general implicit surfaces like superquadrics or hyperquadrics without discretization.

The timing data for translation-only continuous collision detection is shown in Table II. In cases where there is no collision, the query time is the time to solve Equation 17. In cases with collision, we compute the exact time of first contact by solving the two optimization problems described in Equation 17 and Equation 18.

Deforming Objects: As stated in Section IV, the linear constraints in Equation 7 can represent a general affine transformation. Thus this framework can easily handle global deformations such as nonuniform scaling. Figure 12 shows nonuniform scaling of Object I and Object III in 10 steps. The scaling matrices used for the two objects are diagonal matrices whose diagonal entries are $(1, 0.5, 2)$ and $(0.5, 3, 0.67)$ respectively. The approach can also handle any global deformation where the convexity of the object is preserved. For example, consider the global shape deformation for su-

TABLE I

SAMPLE RUN TIMES, IN MILLISECONDS, FOR PROXIMITY QUERIES BETWEEN PAIRS OF OBJECTS USING KNITRO 5.0. THE RUN TIMES WERE COMPUTED FOR EACH PAIR BY AVERAGING THE RUN TIMES OVER 100,000 RANDOM CONFIGURATIONS. ALL DATA WAS OBTAINED ON A 2.2 GHZ ATHLON 64 X2 4400+ MACHINE WITH 2 GB OF RAM.

	Objects	Number of constraints	Proximity query time (millisecs)
1	I, II	2	0.84
2	I, III	2	0.91
3	II, III	2	0.70
4	III, IV	6	0.85
5	II, IV	6	0.76
6	III, V	2	0.78
7	V, VI	2	0.89
8	III, VI	2	0.89

TABLE II

SAMPLE CONTINUOUS PROXIMITY QUERY RUN TIMES BETWEEN PAIRS OF OBJECTS USING KNITRO 5.0. THE RUN TIMES WERE COMPUTED FOR EACH PAIR BY AVERAGING THE RUN TIMES OVER 100,000 PAIRS OF RANDOM CONFIGURATIONS. FOR THE TIME OF FIRST CONTACT QUERIES, ONLY THOSE CONFIGURATION PAIRS THAT RESULTED IN COLLISIONS WERE USED AND THE REPORTED QUERY TIME IS THE TOTAL QUERY TIME FOR SOLVING BOTH PROBLEMS.

Objects	Query type	Number of constraints	Query time (millisecs)
I, III	Closest distance	2	1.32
	Time of first contact	2	2.03
II, III	Closest distance	2	0.97
	Time of first contact	2	1.51

TABLE III

SAMPLE PROXIMITY QUERY RUN TIMES BETWEEN DEFORMING PAIRS OF OBJECTS USING KNITRO 5.0. THE RUN TIMES WERE COMPUTED FOR EACH PAIR BY AVERAGING THE RUN TIMES AT EACH OF 10 STEPS IN THE SHAPE CHANGE, OVER 100,000 RANDOM CONFIGURATIONS.

Objects	Type of Deformation	Number of constraints	Proximity query time (millisecs)
I, III	Nonuniform Scaling	2	1.04
II, III	Nonuniform Scaling	2	0.75
I $\rightarrow$ III, III	Index Change	2	0.87
II $\rightarrow$ III, III	Index Change	2	0.84

perquadrics (or hyperquadrics) due to changes in the indices. Note that if instead a polygonal representation of the object had been used for this kind of shape change, the polygonal representation would have to be recomputed. Figure 13 shows three snapshots of Object I being deformed to Object III in 10 steps. Table III shows the average distance computation times for both nonuniform scaling and index deformation. This data shows that both affine and index changing deformations can be performed with similar running times to the rigid object proximity query.

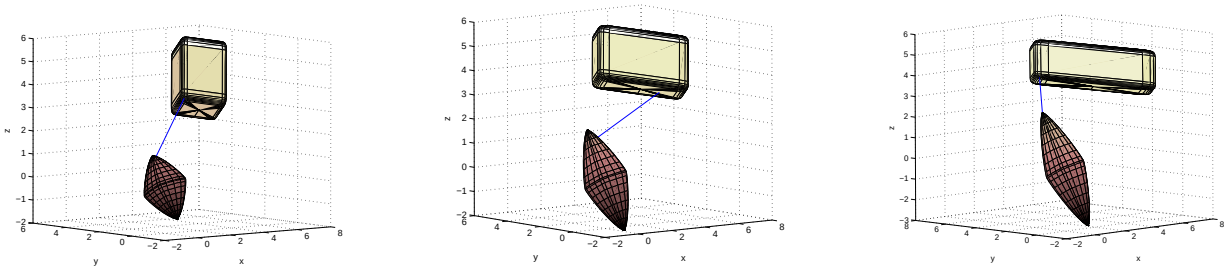
Numerical Robustness Issues: We have observed a small number of cases where KNITRO failed to converge to the optimal solution. For most objects, the failure rates were typically less than 0.01%. The largest failure rate observed was 0.4% when Object I was one of the objects. This may be because (as is evident from our formulation) the algorithm needs the second derivatives of the functions representing the objects, but the second derivative does not exist everywhere for Object I. Despite this, the solver converges to the optimal solution in most cases. Switching to a variant of the interior point method that uses a conjugate gradient method, available within KNITRO, enabled the solver to converge for some of the failure cases. Therefore adaptively using the two variants of the method could improve robustness even further.

IX. 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 proximity query approach complements the proximity query approaches of GJK [21], Lin-Canny [31], and similar algorithms, since they focus on polyhedra while we focus on smooth implicit surfaces. We demonstrated our algorithm on example implicit surface objects including convex polyhedra, quadrics, superquadrics, hyperquadrics, and their intersections. The global convergence properties of interior point algorithms make them robust even in the absence of any initial information about the closest points. For the class of (convex polyhedra and) convex quadric surfaces, this approach has a theoretical complexity of $O(n^{1.5})$, where n is the number of implicit function constraints. To the best of our knowledge, this is the first bound on the running time of proximity queries for convex quadrics. Moreover, the practical running time behavior is linear in the number of constraints for all the classes of implicit surfaces that we have studied. The speed at which distance computations can be performed enables real-time dynamic simulations and haptic interactions at 1 KHz rates.

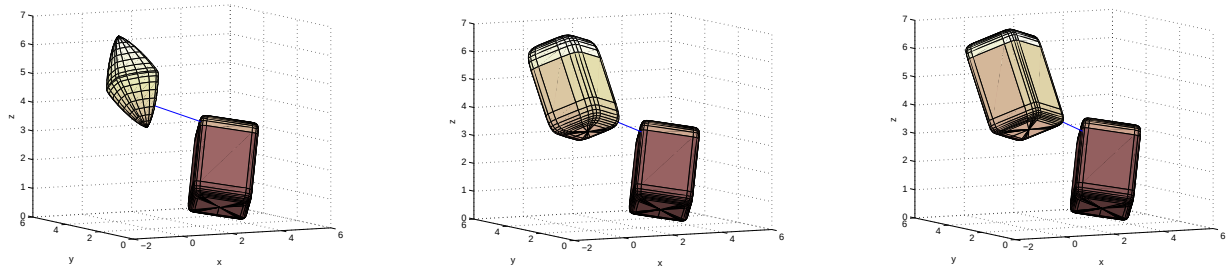
An important additional advantage of this method is that it provides a uniform framework for distance computation between convex objects described as arbitrary intersections of polyhedra, quadrics, or any convex implicit surface. Furthermore, within this framework we can handle global affine deformations of implicit surface objects, and index change deformations of superquadrics (or hyperquadrics) without significant computational overhead. Finally, we show that continuous collision detection for linearly translating implicit surface objects can be performed by solving two related convex optimization problems. For polyhedra and quadrics, we establish that the computational complexity of this continuous collision detection problem is $O(n^{1.5})$.

Future Work: There are several directions for future work.



(a) (b) (c)

Fig. 12. Proximity queries on deforming (superquadric) objects, with the deformation described by monotonic scaling. The deformation is performed in 10 steps. (a) The original objects. (b) The objects midway through the scaling. (c) The scaled objects.



(a) (b) (c)

Fig. 13. Proximity queries on deforming superquadric objects, with the deformation governed by monotonic change of exponents. Object I is transformed to Object III in 10 steps. (a) The original objects. (b) Midway through the deformation, deformed Object I has indices $(\frac{196}{45}, \frac{39}{5}, \frac{97}{13})$. (c) The final objects.

We plan to explore alternative interior point algorithms (LOQO [56] and IPOPT [58], for example) to test their performance on the minimum distance problem. Performing warm starts, where a good initial estimate for the solution is available, can potentially improve the running time when there is coherence. This may be best achieved by a combination of interior point methods and sequential quadratic programming approaches. We would like to extend this approach to nonconvex objects, modeled as unions of convex shapes and incorporate it in a hierarchical framework. Longer term directions for future research include tracking closest points continuously for haptics applications, and extending this approach to performing continuous collision detection with both rotational and translational motion. The approach can be extended to compute the translational penetration depth in a given motion direction for two intersecting objects. We are working on developing an efficient method to compute the minimum translation penetration depth of two intersecting objects when the motion direction is unknown. Finally, we wish to integrate the proximity query algorithm with a nonlinear complementarity formulation of dynamic systems with intermittent contact. This would allow us to study the effects of shape variations and shape approximations in such systems.

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, Steve Berard, Binh Nguyen, and Frank Luk 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] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty. *Nonlinear Programming: Theory and Algorithms*. John Wiley, New York, second edition, 1993.

[4] S. Berard, B. Nguyen, J. Fink, J. C. Trinkle, and V. Kumar. daVinci Code physical simulation library, 2006. http://www.cs.rpi.edu/ sberard/dvc.

[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. Collision detection by four-dimensional intersection testing. *IEEE Transactions on Robotics and Automation*, 6(3):291–302, June 1990.

[10] 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.

[11] M.-P. Cani-Gascuel and M. Desbrun. Animation of deformable models using implicit surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 3(1):39–50, 1997.

[12] J. Canny. Collision detection for moving polyhedra. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):200–209, Mar. 1986.

[13] N. Chakraborty, J. Peng, S. Akella, and J. Mitchell. Proximity queries between convex objects: An interior point approach for implicit surfaces. In *2006 IEEE International Conference on Robotics and Automation*, pages 1910–1916, Orlando, FL, May 2006.

[14] Y.-K. Choi, W. Wang, Y. Liu, and M.-S. Kim. Continuous collision detection for two moving elliptic disks. *IEEE Transactions on Robotics*, 22(2):213–224, Apr. 2006.

[15] D. Constantinescu, S. Salcudean, and E. Croft. Haptic rendering of rigid contacts using impulsive and penalty forces. *IEEE Transactions on Robotics*, 21(3):309–323, June 2005.

[16] 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.

[17] J. Czyzyk, M. Mesnier, and J. More. The NEOS server. *IEEE Journal on Computational Science and Engineering*, 5:68–75, 1998. http://www.mcs.anl.gov/neos/Server/.

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

[19] 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.

[20] 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.

[21] 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.

[22] P. E. Gill, W. Murray, and M. A. Saunders. SNOPT: An SQP algorithm for large-scale constrained optimization. *SIAM Journal on Optimization*, 12(4):979–1006, 2002.

[23] F. Glineur and T. Terlaky. Conic formulation for l_p-norm optimization. *Journal of Optimization Theory and Applications*, 122(2):285–307, Aug. 2004.

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

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

[26] 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.

[27] P. G. Kry and D. K. Pai. Continuous contact simulation for smooth surfaces. *ACM Transactions on Graphics*, 22(1):106–129, Jan. 2003.

[28] 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.

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

[30] 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.

[31] 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.

[32] 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.

[33] Q. Luo and J. Xiao. Physically accurate haptic rendering with dynamic effects. *IEEE Computer Graphics and Applications*, 24(6):60–69, Nov./Dec. 2004.

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

[35] 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.

[36] D. J. Montana. The kinematics of contact and grasp. *International Journal of Robotics Research*, 7(3):17–32, 1988.

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

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

[39] V. Patoglu and R. B. Gillespie. Feedback stabilized minimum distance maintenance for convex parametric surfaces. *IEEE Transactions on Robotics*, 21(5):1009–1016, Oct. 2005.

[40] 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.

[41] J. Peng. *Multiple Robot Coordination: A Mathematical Programming Approach*. PhD thesis, Department of Mathematical Sciences, Rensselaer Polytechnic Institute, May 2005.

[42] J. Peng and S. Akella. Coordinating multiple robots with kinodynamic constraints along specified paths. *International Journal of Robotics Research*, 24(4):295–310, Apr. 2005.

[43] 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.

[44] S. Redon, A. Kheddar, and S. Coquillart. Fast continuous collision detection between rigid bodies. *Computer Graphics Forum (Eurographics 2002 Proceedings)*, 21(3), 2002.

[45] S. Redon, Y. J. Kim, M. C. Lin, and D. Manocha. Fast continuous collision detection for articulated models. In *Proceedings of ACM Symposium on Solid Modeling and Applications*, 2004.

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

[47] J. Sauer, E. Schömer, 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.

[48] E. Schömer, J. Reichel, T. Warken, and C. Lennerz. Efficient collision detection for curved solid objects. In *Proceedings of 7th ACM Symposium on Solid Modeling and Applications (SM 02)*, pages 321–328, Saarbrucken, Germany, June 2002.

[49] 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.

[50] P. Song, J. Trinkle, V. Kumar, and J. Pang. Design of part feeding and assembly processes with dynamics. In *IEEE International Conference on Robotics and Automation*, pages 39–44, New Orleans, LA, May 2004.

[51] 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.

[52] 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.

[53] 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.

[54] G. van den Bergen. *Collision Detection in Interactive 3D Environments*. Morgan Kaufmann, 2004.
[55] G. van den Bergen. Ray casting against general convex objects with application to continuous collision detection. *Journal of Graphics Tools*, 2004. Submitted.
[56] R. J. Vanderbei. LOQO user's manual – version 3.10. Technical Report SOR 97-08, Statistics and Operations Research, Princeton University, Dec. 1997.
[57] R. J. Vanderbei and D. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.
[58] A. Wachter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.
[59] R. A. Waltz. *KNITRO 4.0 User's Manual*. Ziena Optimization, Inc., Evanston, IL, Oct. 2004.
[60] S. J. Wright. *Primal-Dual Interior-Point Methods*. Society for Industrial and Applied Mathematics, Philadelphia, 1997.
[61] P. G. Xavier. Fast swept-volume distance for robust collision detection. In *IEEE International Conference on Robotics and Automation*, pages 1162–1169, Albuquerque, NM, Apr. 1997.
[62] 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.