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 E. Mitchell

***Abstract*—This paper presents a general method for 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 intermittent contact, such as in dynamic simulations and in haptic interactions. 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. We use an interior point method to solve the optimization problem and demonstrate that for general convex objects represented as implicit surfaces, interior point approaches are globally convergent, and fast in practice. Further, they provide polynomial-time guarantees for implicit surface objects when the implicit surfaces have self-concordant barrier functions. 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 also $O(n^{1.5})$.**

***Index Terms*—Proximity query, closest points, implicit surfaces, interior point algorithms, collision detection.**

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 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. Example applications include multibody dynamic simulation [1], [32], [40], computer animation [10], dextrous manipulation [6], and haptics [30], [13]. Applications where collision avoidance is the primary goal, such as robot path planning [35] and spacecraft safe volume computations [14], can also make use of the knowledge of the closest distance information.

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 $\mathbb{R}^2$ 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 [23], [29], mainly for polyhedral object representations [16], [19], [28], [31]. 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 we are given an implicit surface model of each object. Our choice of object representation is motivated by the goal of simulating systems with smooth objects, where polygonal discretizations may not be desirable. The literature on distance computation between general implicit surfaces is relatively sparse because, with a few notable exceptions [18], [41], 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 presents the first general method for 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 [4], [5], 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 having self-concordant barriers. We apply a recently developed interior point algorithm [7], [46] to compute the distance between convex implicit surface objects and demonstrate that it is particularly effective for this class of problems. For

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) 421-0800, Fax: (518) 276-2529, Email: s.akella@ieee.org.

polyhedral and quadric surfaces, we exploit the problem structure to show 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; 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 C^2 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 also $O(n^{1.5})$.

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 [12].

II. Related Work

Proximity queries for polyhedra: Proximity queries and collision detection algorithms have an extensive literature in computational geometry [16], [27], robotics [19], [28], and computer graphics [1], [41]. We provide a sampling of the related work in these areas; see [29] and [23] 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 [19], Lin-Canny [28], and V-Clip [31]. GJK [19] is an iterative algorithm for distance computation between two convex polyhedra. Without any preprocessing, it takes time linear in the number of vertices and uses a support function description of the polyhedra. Lin-Canny [28] 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 [4] 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. Since this approach can suffer from convergence issues, Zeghloul et. al. [48] propose a method to improve its convergence.

Proximity queries for implicit surfaces: The literature on distance computation between general implicit surfaces is relatively sparse because, with the exception of GJK [18], methods for polyhedral representations do not easily generalize to implicit surfaces. van den Bergen [41] discusses in detail a GJK implementation for convex quadric objects. This algorithm is globally convergent for convex objects and numerical experiments for two quadric surfaces indicate that for a given tolerance ϵ, the algorithm converges in $O(-log(\epsilon))$ steps. Most other algorithms [1], [27], [26], [14], [39], [25] use the collinearity properties of the surface normals at the closest points to form a set of nonlinear algebraic equations whose solution gives the closest points. The differences between the different algorithms lie in the approaches they use to solve the system of nonlinear equations. All these methods require a good initial guess to converge to the correct solution. Although superquadrics are a generalization of quadrics, the problem in generalizing the methods in [1], [19], [14], [39] 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. A different approach based on interval arithmetic techniques was used to detect contact points [17], [38]. Duff [17] presented an interval arithmetic approach to perform collision detection between objects modeled as constructive solid geometry combinations of implicit functions, using a binary subdivision of space along each axis. Snyder et al. [38] used the tangency constraints at the touching points to form the system of equations and solve it using an interval Newton method.

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 $\lambda \mathbf{u}_1 + (1-\lambda)\mathbf{u}_2 \in U$ for any two points $\mathbf{u}_1, \mathbf{u}_2 \in U$ and any λ with $0 \leq \lambda \leq 1$.

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 $f(\lambda \mathbf{u}_1 + (1-\lambda)\mathbf{u}_2) \leq \lambda f(\mathbf{u}_1) + (1-\lambda) f(\mathbf{u}_2)$. for all $\mathbf{u}_1, \mathbf{u}_2 \in$ *dom* f and any λ with $0 \leq \lambda \leq 1$.

Superquadric: A superquadric [22] is defined by the equation

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

Although the definition here differs slightly from that in [2], the two definitions are equivalent [22]. Convex superquadrics are a broad class of shapes that include 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, a_i

controls the length along the ith axis, 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 [22] is defined by the equation

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

Hyperquadrics are a more general class of shapes than superquadrics. In particular, they include asymmetric shapes. In this case also, the intersection of the halfspaces $H_i(\mathbf{x}) \leq 1$ 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 $x \in domf$. 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 [5] for details).

IV. Problem Formulation

We now 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], [27], [39] and relate it to an optimization formulation. Let f_X and f_Y be two twice continuously differentiable functions representing objects X and Y respectively, and let $\mathbf{x}_g$, $\mathbf{y}_g$ be the global coordinates of points in X and Y. 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

$$\begin{aligned} & \mathbf{x}_g - \mathbf{y}_g = -\lambda_X \nabla f_X(\mathbf{x}_g) = \lambda_Y \nabla f_Y(\mathbf{y}_g) \\ & f_X(\mathbf{x}_g) = 0, \quad f_Y(\mathbf{y}_g) = 0 \end{aligned} \tag{4}$$

where λ_X and λ_Y are scalars. The conditions given by Equation 4 are precisely the Karush-Kuhn-Tucker (KKT) conditions for solving the following optimization problem:

$$\begin{aligned} \text{Minimize} \quad & \|\mathbf{x}_g - \mathbf{y}_g\|_2 \\ \text{subject to:} \quad & f_X(\mathbf{x}_g) = 0, \quad f_Y(\mathbf{y}_g) = 0. \end{aligned} \tag{5}$$

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. 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

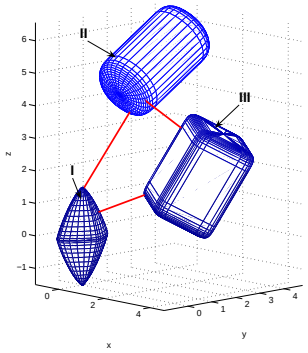


Fig. 1. Three example objects. The closest points of each pair of objects are shown connected by a line segment.

Equation 1 is then given by

$$\begin{aligned} \text{Minimize} \quad & \|\mathbf{x}_g - \mathbf{y}_g\|_2^2 \\ \text{subject to:} \quad & f_i(\mathbf{x}_g) \leq 0 \quad i = 1, \ldots, m \\ & f_j(\mathbf{y}_g) \leq 0 \quad j = m+1, \ldots, n \end{aligned} \tag{6}$$

where f_k ($k = i, j$) are twice continuously differentiable functions representing the implicit surfaces in the global reference frame. If the description of the surfaces are in their local frame we can transform them to a global frame by $\mathbf{x}_g = \mathbf{R}\mathbf{x}_l + \mathbf{p}$, where $(\mathbf{R}, \mathbf{p})$ is the transformation from local to global frame. Note that this transformation 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. The objective function in Equation 6 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.

The solution to the minimum distance problem of Equation 6 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 [7] for solving this problem. See Figure 1 for an example solution generated using an interior point algorithm. Interior point methods [5] 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 [4], 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 [20]. 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 [15] 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 minimum distance problem as an optimization problem. The Karush-Kuhn-Tucker (KKT) conditions give necessary and sufficient conditions for solving the minimum distance problem (Equation 6), since it is a convex optimization problem and satisfies Slater constraint qualification [3]. For ease of presentation, we rewrite Equation 6 in a general nonlinear program format as

$$\begin{aligned} \text{Minimize} \quad & f_0(\mathbf{x}) \\ \text{subject to:} \quad & \mathbf{f}(\mathbf{x}) + \mathbf{s} = \mathbf{0}, \quad \mathbf{s} \geq \mathbf{0} \end{aligned} \tag{7}$$

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 KKT conditions for Equation 7 are the system of nonlinear equations below:

$$\begin{aligned} \nabla f_0(\mathbf{x}) + (\nabla \mathbf{f}(\mathbf{x}))^T \lambda &= \mathbf{0} \\ \mathbf{f}(\mathbf{x}) + \mathbf{s} &= \mathbf{0} \\ \mathbf{LSe} &= \mathbf{0} \end{aligned} \tag{8}$$

Here $\mathbf{L}$ is an $n \times n$ diagonal matrix of the Lagrange multipliers λ, $\mathbf{S}$ is an $n \times n$ diagonal matrix of the slack variables $\mathbf{s}$, and $\mathbf{e}$ is an n-vector of ones. Equation 8 can be solved by Newton's method for solving systems of nonlinear equations, if the initial guess is *near enough* [33]. 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 ensuring $\mathbf{s} \geq 0$, which may lead to very small step lengths that result in convergence problems.

Interior point methods, in essence, 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 [5], we present the interior point method by reformulating Equation 7 as a *barrier* problem.

$$\begin{aligned} \text{Minimize} \quad & f_0(\mathbf{x}) - \mu \sum_{i=1}^{n} \ln(s_i) \\ \text{subject to:} \quad & \mathbf{f}(\mathbf{x}) + \mathbf{s} = \mathbf{0} \end{aligned} \tag{9}$$

where μ is called the barrier parameter, with $\mu > 0$. Equation 9 differs from Equation 7 in that the nonnegativity constraints on $\mathbf{s}$ are not present explicitly, but are implicit in the objective. The KKT conditions can be written as

$$\begin{aligned} \nabla f_0(\mathbf{x}) + (\nabla \mathbf{f}(\mathbf{x}))^T \lambda &= \mathbf{0} \\ \mathbf{f}(\mathbf{x}) + \mathbf{s} &= \mathbf{0} \\ \mathbf{LSe} - \mu \mathbf{e} &= \mathbf{0} \end{aligned} \tag{10}$$

This is a system of $2n + 6$ nonlinear equations in $2n + 6$ variables and can be approximately solved for a given μ. Note that Equation 8 and Equation 10 differ in the complementarity conditions. As the barrier parameter μ approaches 0, the KKT conditions for the barrier problem (Equation 10) approach the KKT conditions of the original problem (Equation 8).

The general structure of interior point methods is indicated in Algorithm 1. For our problem, the immediate choice for the initial guess is the center of the involved objects. The termination criterion 1 is the ending condition for the whole problem (Equation 8) and termination criterion 2 is the ending condition for approximately solving Equation 10 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 10 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 10 for a fixed value of μ. The different interior point implementations (KNITRO [7], [46], LOQO [44], IPOPT [45]) 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.

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 ← 0
while termination criterion 1 not satisfied do
  while termination criterion 2 not satisfied do
    Solve system of linear equations for Newton direction
    Determine step length α_k by line search
    x_{k+1} ← x_k + α_k Δx_k
    s_{k+1} ← s_k + α_k Δs_k
    λ_{k+1} ← λ_k + α_k Δλ_k
    k ← k + 1
  end while
  μ ← cμ    // c < 1, may be constant or adaptive
end while
return 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

$$\begin{pmatrix} \mathbf{A_1} & \mathbf{A_2}^T & \mathbf{0}_{6\times n} \\ \mathbf{A_2} & \mathbf{0}_{n\times n} & \mathbf{I}_{n\times n} \\ \mathbf{0}_{6\times n} & \mathbf{S} & \mathbf{L} \end{pmatrix} \begin{pmatrix} \Delta\mathbf{x} \\ \Delta\lambda \\ \Delta\mathbf{s} \end{pmatrix} = \begin{pmatrix} -\mathbf{F}_1 \\ -\mathbf{F}_2 \\ -\mathbf{F}_3 \end{pmatrix} \tag{11}$$

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 $\mathbf{F}_1 = \nabla f_0(\mathbf{x}) + (\nabla \mathbf{f}(\mathbf{x}))^T \lambda$, $\mathbf{F}_2 = \mathbf{f}(\mathbf{x}) + \mathbf{s}$, $\mathbf{F}_3 = \mathbf{LSe} - \mu \mathbf{e}$.

We now show Equation 11 can be solved in $O(n)$ time, although the computational cost of solving a system of n linear equations in n unknowns is $O(n^3)$ in general. By simple algebraic manipulation of the above equations, we obtain the following formulas for $\Delta\mathbf{x}$, $\Delta\lambda$, and $\Delta\mathbf{s}$:

$$\begin{aligned} \Delta\mathbf{x} &= \mathbf{G}^{-1}(-\mathbf{F}_1 - \mathbf{A}_2^T(\mathbf{S}^{-1}\mathbf{L})(\mathbf{F}_2 + \mathbf{L}^{-1}\mathbf{F}_3)) \\ \Delta\lambda &= (\mathbf{S}^{-1}\mathbf{L})(\mathbf{A}_2\Delta\mathbf{x} - \mathbf{F}_2 + \mathbf{L}^{-1}\mathbf{F}_3) \\ \Delta\mathbf{s} &= -\mathbf{L}^{-1}(\mathbf{F}_3 + \mathbf{S}\Delta\lambda) \end{aligned} \tag{12}$$

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 12 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 type describing the object. Thus this analysis is valid for any C^2 implicit surface (including planes, quadrics, superquadrics, hyperquadrics) 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 11 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 $O(n^{0.5}log(c/\epsilon))$ [5], where c is a problem dependent constant and ϵ is the error tolerance. This implies that the theoretical complexity of our approach for polyhedra and quadrics is $O(n^{1.5})$, for a given ϵ. Alternatively, if n is constant (for example, if we consider one quadric surface describing each object, i.e., $n = 2$) then the solution is obtained in $O(-log(\epsilon))$ time. For the case of quadrics, this $O(n^{1.5})$ complexity is the best known bound. Moreover, our experiments indicate the algorithm exhibits linear time behavior in practice, as shown in Figure 2(a) and Figure 2(b).

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 [34]) 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 of variables and constraints. Glineur and Terlaky [21] 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 2(c)), 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.

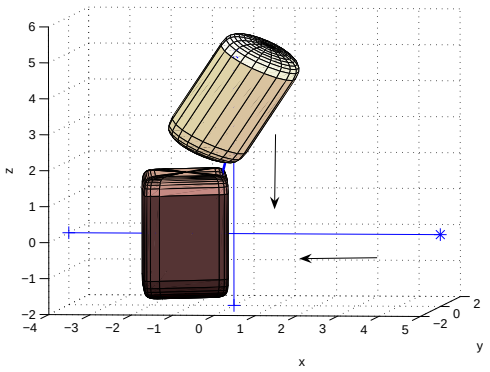


Fig. 3. Computing the instant of closest distance using the continuous proximity query. The bold blue line connects the closest points on the two objects, as they translate along the indicated line segments.

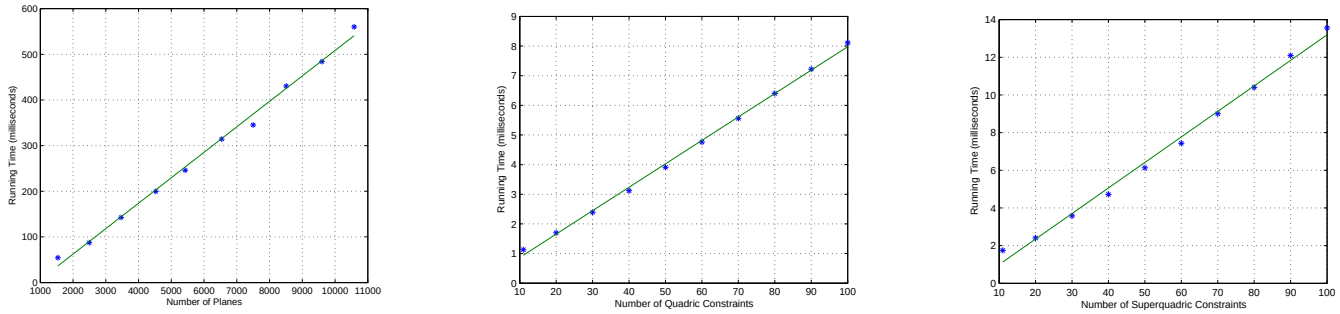
VII. Continuous Proximity Queries for Translating Objects

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 [11], [8], [47], [36], [37], [43] 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 3.

$$\begin{aligned} \text{Minimize} \quad & \|\mathbf{x}_g - \mathbf{y}_g\|_2^2 \\ \text{subject to:} \quad & \mathbf{x}_g = \mathbf{R}_x\mathbf{x}_l + \mathbf{p}_x + v_x t\hat{\mathbf{g}}_x \\ & \mathbf{y}_g = \mathbf{R}_y\mathbf{y}_l + \mathbf{p}_y + v_y t\hat{\mathbf{g}}_y \\ & f_X(\mathbf{x}_l) \leq 0, \quad f_Y(\mathbf{y}_l) \leq 0 \\ & 0 \leq t \leq T_{max} \end{aligned} \tag{13}$$

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 13. Let Q be the set of



(a) (b) (c)

Fig. 2. Plots showing observed linear time behavior of the interior point algorithm for different classes of surfaces. (a) Planes (b) Quadrics (c) Superquadrics. Each data point was generated by averaging over 10,000 random configurations. All data was obtained on a 2.2 GHz Athlon 64 X2 4400+ machine.

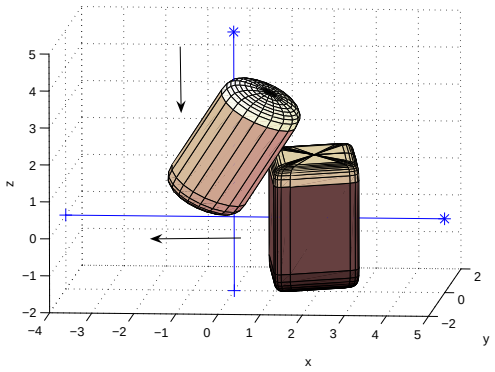


Fig. 4. Computing the time of first contact using the continuous proximity query gives the solution to the continuous collision detection problem.

intersecting points and $\mathbf{q} \in Q \subset \mathbb{R}^3$ 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 13.

$$\begin{aligned} \text{Minimize} \quad & t \\ \text{subject to:} \quad & \mathbf{q} = \mathbf{R}_x \mathbf{x}_l + \mathbf{p}_x + v_x t \hat{\mathbf{g}}_x \\ & \mathbf{q} = \mathbf{R}_y \mathbf{y}_l + \mathbf{p}_y + v_y t \hat{\mathbf{g}}_y \\ & f_X(\mathbf{x}_l) \leq 0, \quad f_Y(\mathbf{y}_l) \leq 0 \\ & 0 \leq t \leq T_p \end{aligned} \tag{14}$$

where T_p is the time obtained from the solution of Equation 13. See the example in Figure 4.

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 13 and can write it in the form of Equation 7 where $\mathbf{x}$ is a 7×1 column vector that includes t. Thus in Equation 11, $\mathbf{A}_1$ 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 12 in Section VI applies directly. Similarly in Equation 14, $\mathbf{x}$ 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.

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 [7], [46]. 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 [33]. The absolute and relative error tolerances used in our simulations are 10^{-8} and 10^{-6} respectively.

We depict six example objects in Figure 5, 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 four halfspaces. The indices and semiaxes of the superquadric are $(\frac{23}{11}, \frac{11}{5}, \frac{17}{7})$ and $(2, 1, 1.7)$. The halfspaces 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 1. All data was obtained on a 2.2 GHz Athlon 64 X2 4400+ machine with 2 GB of RAM and averaged over $100,000$ random configurations. The running times demonstrate that the distance computation rate is about 1 kHz,

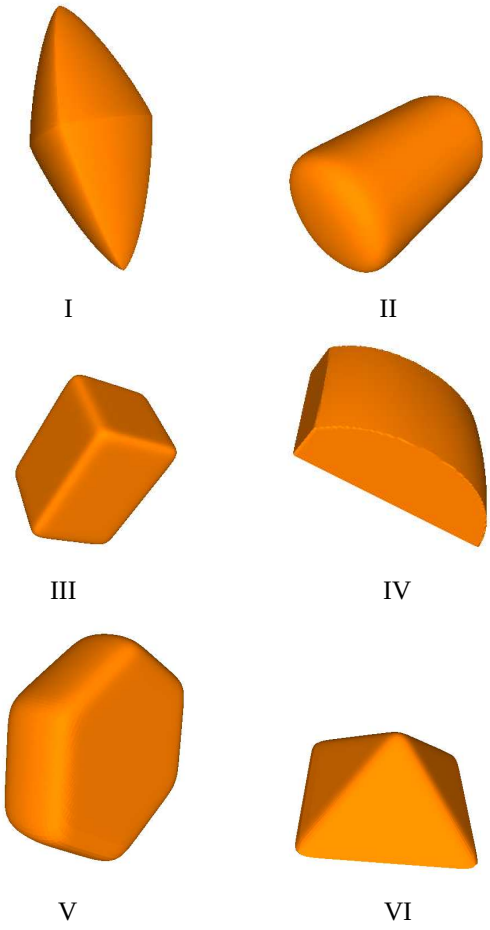


Fig. 5. Example objects. Objects I–III are superquadrics, IV is an intersection of superquadrics and halfspaces, and V–VI are hyperquadrics.

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 [24], 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 [41], [42], 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 of running time in terms of the number of intersecting surfaces and error tolerance. Moreover, SOLID cannot deal with general implicit surfaces like superquadrics or hyperquadrics without discretization.

The timing data for translational continuous collision detection is shown in Table II. In cases where there is no collision, the query time is the time to solve Equation 13. In cases with collision, we compute the exact time of first contact by solving the two optimization problems described in Equation 13 and Equation 14.

Deforming Objects: As stated in Section IV, we can easily handle global deformations such as nonuniform scaling in our framework. Figure 6 shows nonuniform scaling of Object I and Object III in 10 steps. The final scaling matrices 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 superquadrics (or hyperquadrics) due to changes in the indices. Note that if a polygonal representation of the object had instead been used for this kind of shape change, the polygonal representation would have to be recomputed. Figure 7 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 changing deformations, indicating they can be performed with similar running times to the rigid object proximity query.

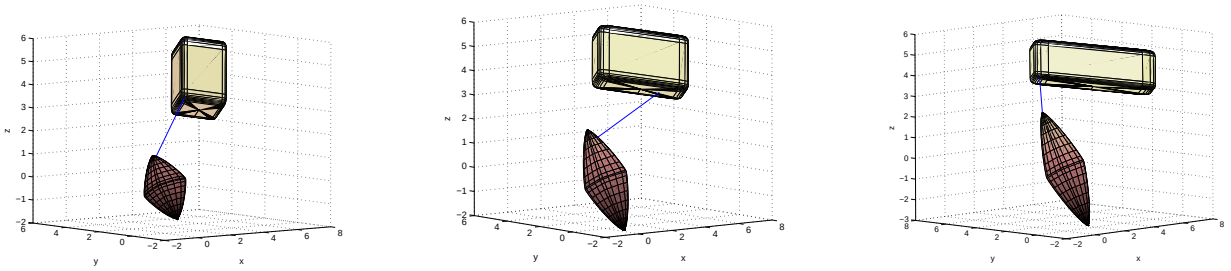
TABLE I
SAMPLE RUN TIMES, IN MILLISECONDS, FOR PROXIMITY QUERIES BETWEEN PAIRS OF OBJECTS.

	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

Numerical Robustness Issues: We have observed a small number of cases where KNITRO failed to converge to the optimal solution (with a 10^{-8} absolute tolerance and a 10^{-6} relative tolerance). 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. The number of failure cases decreases as the error tolerance is increased. For relative and absolute tolerance values of 10^{-2}, we have observed no failures. We also found that 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 (with a 10^{-8} absolute tolerance and a 10^{-6} relative tolerance). Therefore adaptively using the two variants of the method could improve robustness even further.

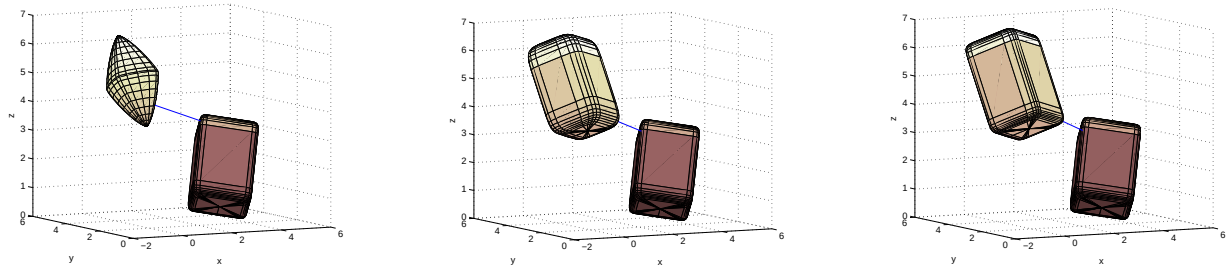
IX. CONCLUSION

This paper develops a general approach for computing the distance between two or more convex objects, where each object is described as an intersection of implicit surfaces, and establishes the theoretical and practical effectiveness of recently developed interior point algorithms for this problem. This proximity query approach complements most existing proximity query algorithms since they (with the exception of GJK [19]) focus on polyhedra while we focus on smooth im-



(a) (b) (c)

Fig. 6. 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. 7. Proximity queries on deforming superquadric objects, with the deformation governed by a linear 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 $(\frac{44}{9}, \frac{39}{5}, \frac{97}{13})$. (c) The final objects.

TABLE II
SAMPLE CONTINUOUS PROXIMITY QUERY RUN TIMES BETWEEN PAIRS OF OBJECTS. 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.

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

plicit 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, we establish a theoretical complexity of $O(n^{1.5})$ for this approach, where n is the number of implicit function constraints. This is the first bound on the running time of proximity queries for intersections of 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. 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 also $O(n^{1.5})$.

Future Work: There are several directions for future work. We plan to explore alternative interior point algorithms (LOQO [44] and IPOPT [45], 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.

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] 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.

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

[6] 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.

[7] R. Byrd, J. Nocedal, and R. A. Waltz. KNITRO: An integrated package for nonlinear optimization. In G. de Pillo and M. Roma, editors, *Large Scale Nonlinear Optimization*, 35–59, 2006, Springer Verlag.

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

[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] 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.

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

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

[13] 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.

[14] 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.

[15] 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/.

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

[17] T. Duff. Interval arithmetic and recursive subdivision for implicit functions and constructive solid geometry. *Computer Graphics*, 26(2):131–138, 1992.

[18] 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.

[19] 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.

[20] 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.

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

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

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

[24] 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.

[25] C. Lennerz and E. Schömer. Efficient distance computation for quadratic curves and surfaces. In *Geometric Modeling and Processing (GMP 2002)*, pages 60–69, Saitama, Japan, July 2002.

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

[27] 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.

[28] 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.

[29] 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.

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

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

[32] 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.

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

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

[35] 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.

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

[37] 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.

[38] J. Snyder, A. Woodbury, K. Fleischer, B. Currin, and A. Barr. Interval methods for multi-point collisions between time-dependent curved surfaces. *Computer Graphics*, 27(2):321–334, 1993.

[39] 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.

[40] 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.

[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. *Collision Detection in Interactive 3D Environments*. Morgan Kaufmann, 2004.

[43] G. van den Bergen. Ray casting against general convex objects with application to continuous collision detection, 2004. http://www.detecta.com.

[44] R. J. Vanderbei and D. Shanno. An interior-point algorithm for nonconvex nonlinear programming. *Computational Optimization and Applications*, 13:231–252, 1999.

[45] 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.

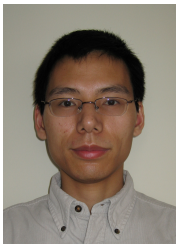
[46] R. A. Waltz and T. D. Plantenga. *KNITRO 5.0 User's Manual*. Ziena Optimization, Inc., Evanston, IL, June 2006.

[47] 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.

[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.

Nilanjan Chakraborty is a Ph.D. student in the Department of Computer Science at Rensselaer Polytechnic Institute. He received his B.E. in Mechanical Engineering from Jalpaiguri Government Engineering College in 1998, and his M.Sc.(Engg.) from the Department of Mechanical Engineering, Indian Institute of Science, Bangalore in 2003.

His research interests are in mobile robotics, motion planning for multi-robot systems, multibody dynamics, and application of optimization based techniques in these areas.



Jufeng Peng Jufeng Peng received the B.S. degree in mathematics from Wuhan University, Wuhan, China, in 2000, and the Ph.D. degree in mathematics from Rensselaer Polytechnic Institute, Troy, USA, in 2005. He worked in the Algorithmic Robotics Laboratory under the supervision of Srinivas Akella from 2001-2005. He was awarded the Joaquin B. Diaz Memorial Prize in Mathematics at RPI in 2005 for his enthusiasm in mathematics research. He is currently an R&D analyst at the Progressive Insurance Company. His current research lies in optimal risk pricing using Operations Research and Statistics Techniques.

Srinivas Akella is an Assistant Professor in the Computer Science department at Rensselaer Polytechnic Institute, Troy, New York. He received his B.Tech. in Mechanical Engineering from the Indian Institute of Technology, Madras in 1989 and his Ph.D. in Robotics from the School of Computer Science at Carnegie Mellon University in 1996. He was a Beckman Fellow at the Beckman Institute for Advanced Science and Technology, University of Illinois at Urbana-Champaign, from 1996 to 1999. He is a recipient of the NSF CAREER award, and the National Talent Search Scholarship from the Government of India.

His research interests are in developing geometric and optimization algorithms for robotics, automation, design, and bioinformatics applications.

John Mitchell received the B.A. (Hons) degree in Mathematics from Cambridge University in 1983 and the M.S. and Ph.D. degrees in Operations Research and Industrial Engineering from Cornell University in 1986 and 1988, respectively.

He joined the department of Mathematical Sciences at Rensselaer Polytechnic Institute in 1988, where he is currently a Professor. His research interests lie in optimization, particularly interior point methods, integer programming, and their application.

LIST OF FIGURES

LIST OF TABLES