2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

# Weighted Barrier Functions for Computation of Force Distributions with Friction Cone Constraints

Per Henrik Borgstrom*, Maxim A. Batalin§, Gaurav S. Sukhatme‡ and William J. Kaiser*
*Department of Electrical Engineering, University of California, Los Angeles
†Department of Mechanical Engineering, University of California, Los Angeles
‡Department of Computer Science, University of Southern California
§Center for Embedded Networked Sensing, University of California, Los Angeles
Email: hborgstr@ucla.edu

*Abstract*— **We present a novel Weighted Barrier Function (WBF) method of efficiently computing optimal grasping force distributions for multifingered hands. Second-order conic friction constraints are not linearized, as in many previous works. The force distributions are smooth and rapidly computable, and they enable flexibility in selecting between firm, stable grasps or looser, more efficient grasps. Furthermore, fingers can be disengaged and re-engaged in a smooth manner, which is a critical capability for a large number of manipulation tasks. We present efficient solution methods that do not incur the increased computational complexity associated with solving the Semi-Definite Programming formulations presented in previous works. We present results from static and dynamic simulations which demonstrate the flexibility and computational efficiency associated with WBF force distributions.**

## I. Introduction

Redundantly-actuated parallel manipulators have been widely studied for use in grasping robots [1]. Multifingered grasping robots promise improved flexibility in performing industrial tasks and are critical in systems such as bomb-defusing or rescue robots, in which the uncertain nature of interactions with the physical world requires flexibility of manipulator systems.

In an early work on multifingered hands [2], Kerr and Roth determined three primary problems in the design of grasping robots: Manipulation, determination of hand workspace, and force optimization. Force optimization is the selection of optimal contact forces exerted by fingertips on the end-effector given a particular grasp configuration. The contact forces are characterized by a set of constraints:

- Fingers must act in unison to exert the desired wrench on the end-effector.
- Fingers can only exert positive or pushing forces.
- Joint efforts must not exceed hardware limits.
- Contact forces must not be excessively shallow relative to the contact plane.

The first three requirements represent linear constraints on contact forces, while the final requirement is a nonlinear Second-order Conic (SOC) constraint that guards against slippage of fingers relative to the end-effector. The focus of this paper is the efficient generation of optimal grasp force distributions subject to these constraints.

This work was supported by the US National Science Foundation (NSF) under Grants ANI-0331481 and ECCS-0725441

There is considerable prior work in force optimization for grasping robots. In [2], Kerr and Roth linearized the nonlinear friction-cone constraint and posed the force optimization problem as a Linear Program (LP). Linear equality constraints were removed by decomposing grasping forces into particular and homogeneous force vectors and optimizing purely on these homogeneous force vectors. These statically indeterminate components are also known as internal force vectors. The optimization objective was to minimize the proximity to violation of constraints in a method roughly analogous to finding the Chebyshev approximation of the center of the feasible polyhedron. Furthermore, Cheng and Orin [3] described efficient methods of solving LP problems using a compact dual method.

There are two significant shortcomings associated with the LP approach suggested in [2] and [3]. First, the linearization of friction-cone constraints is not particularly accurate unless many linear constraints are used to approximate each cone. If many inequalities are used, the associated computational burden increases. Second, as noted by the authors, the resulting force profiles are prone to discontinuous behavior in response to infinitesimal changes in robot configuration. This is particularly problematic if an end-effector is to be held steady on the border between two different operating regions. The resulting oscillations between force regions can result in instability.

Buss *et al.* observed that the nonlinear friction-cone constraint was equivalent to the positive definiteness of a particular matrix $\mathbf{P}$ [4]. By using this property, the force optimization problem could be formulated as a Semi-Definite Program (SDP), which is a convex optimization problem. In [5], a linearly constrained gradient flow solution method is proposed. An improved solution method is presented in [6], wherein a Dikin-type algorithm is employed. However, in these methods (as with all SDP solvers), the computation of descent directions is expensive, and an appropriate step-size must be determined by a linesearch. Thus, the per-iteration computational burden can be high unless special properties of the bounding matrices are exploited [7]. In fact, all SOC problems can be solved as SDP problems, but doing so is inadvisable both on numerical grounds and due to increased computational complexity [8].

The authors of [9] note that the SDP methods proposed in [6] require feasible starting points and propose a gradient method for generating such initial points.

In [10], a recurrent neural network is proposed that minimizes a quadratic objective function subject to nonlinear friction-cone constraints.

The objective function proposed in [4] was composed of a linear sum of contact wrenches and a barrier term that approaches infinity as the matrix $\mathbf{P}$ approaches singularity i.e. as the friction-cone constraints are approached. By weighting the two elements of the objective function, users could favor low joint effort (at the cost of reduced robustness to slip) or increased robustness to slip (at the cost of increased joint effort). This tradeoff is critical in enabling platforms to grasp both heavy and delicate objects.

In [11], Schlegl *et al.* provide experimental results on a four-fingered hand. Forces are found using an SDP formulation, and the ability to disengage and re-engage fingers in a smooth fashion is demonstrated. This is a critical capability for many manipulation tasks.

In this paper, we present the Weighted Barrier Function (WBF) force distribution method, which minimizes the weighted sum of two barrier functions to efficiently compute force distributions for redundantly-actuated parallel manipulators subject to non-linear friction constraints. The presented methods are characterized by:

- Smooth and continuous force profiles.
- Flexibility in selecting operating region along the grasp-stability/joint-effort tradeoff curve.
- Smooth regrasp capabilities.
- Improved computational efficiency by avoiding the increased per-iteration cost associated with SDP methods.

The remainder of this paper is structured as follows: In Section II, we formalize the grasp force distribution problem. In Section III, we describe two barrier functions and introduce our novel WBF method of computing grasps. Results are presented in Section IV, as is analysis of the computational burden incurred by WBF and SDP solution methods. In Section V, we conclude the paper.

## II. The Grasping Force Distribution Problem

A schematic diagram of a multifingered grasping robot is shown in Fig. 1. The primary objective of force distribution is to exert the desired wrench, $\mathbf{f} \in \Re^6$, on the end-effector. This can be expressed as follows:

$$\mathbf{W}\mathbf{c} = \mathbf{f}, \tag{1}$$

where $\mathbf{W}$ is a grasp map matrix and $\mathbf{c}$ is the vector of contact forces exerted by the fingers on the object. If the finger contacts are modeled as Frictionless Point Contacts (FPC), then $\mathbf{c}$ consists of only forces normal to the grasped object and is an $n$-dimensional vector, where $n$ is the number of fingers. If Point Contact With Friction (PCWF) is assumed, $\mathbf{c} \in \Re^{3n}$ consists of one normal and two tangent forces for each finger. For the $i^{th}$ finger, we define these as $\mathbf{c}_i = \begin{bmatrix} c_{i_{norm}} & c_{i_x} & c_{i_y} \end{bmatrix}^T$. Finally, if Soft Finger Contacts (SFC) are considered, $\mathbf{c} \in \Re^{4n}$, and each finger also applies a wrench

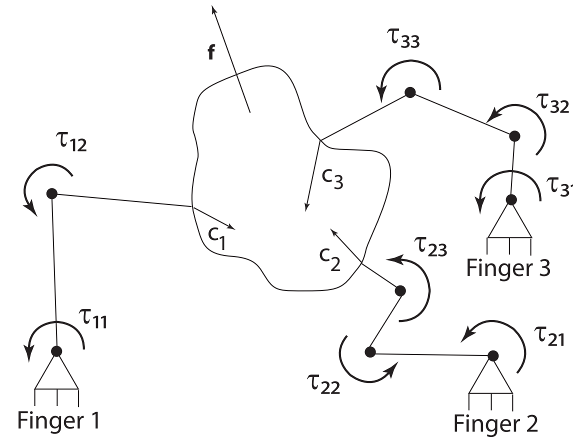


Fig. 1. A three-fingered hand grasping an object. Joint torques, $\tau_{ij}$, contact forces, $\mathbf{c}_i$, and the desired wrench, $\mathbf{f}$, are shown.

about the axis normal to the contact plane. In this paper, we use the PCWF contact model, but all methods apply directly to the FPC and SFC contacts as well. Furthermore, we define the dimensionality of $\mathbf{c}$ as $m$.

In addition to the linear equality constraint posed by Eqn. (1), grasp forces are also constrained by actuator and hardware limits. In particular, each joint must not exceed its torque limits. If $\boldsymbol{\tau}^L \in \Re^m$ and $\boldsymbol{\tau}^U \in \Re^m$ represent lower and upper joint torque limits, then we have

$$\boldsymbol{\tau}^L \leq \mathbf{J}^T\mathbf{c} \leq \boldsymbol{\tau}^U, \tag{2}$$

where $\mathbf{J}$ is the hand Jacobian.

It is also critical that fingers not exert forces that are excessively shallow relative to the normal plane, which would result in slippage. In the SFC contact model, this is represented by the following SOC constraint:

$$\mu_i c_{i_{norm}} \geq \sqrt{c_{i_x}^2 + c_{i_y}^2}, \quad i = 1 \dots n, \tag{3}$$

where $\mu_i$ is the coefficient of friction between the $i^{th}$ finger and the grasped object. This friction-cone constraint dictates that the forces exerted by finger $i$ must lie within the Lorentz cone whose radius increases at a rate of $\mu_i$ relative to the height. This is shown in Fig. 2.

Thus, the grasp optimization problem becomes:

$$\begin{aligned} \overset{*}{\mathbf{c}} &= \underset{\mathbf{c}}{\operatorname{argmin}} f(\mathbf{c}) \\ s.t.\ & \mathbf{W}\mathbf{c} = \mathbf{f} \\ & \boldsymbol{\tau}^L \leq \mathbf{J}^T\mathbf{c} \leq \boldsymbol{\tau}^U \\ & \mu_i c_{i_{norm}} \geq \sqrt{c_{i_x}^2 + c_{i_y}^2}, \quad i = 1 \dots n, \end{aligned} \tag{4}$$

where $f(\mathbf{c})$ is some objective function. In order to reduce the risk of slippage, $f(\mathbf{c})$ should penalize proximity to friction-cone constraints. This is particularly important in real-world applications, where friction coefficients and other grasp properties may not be precisely known. Furthermore,

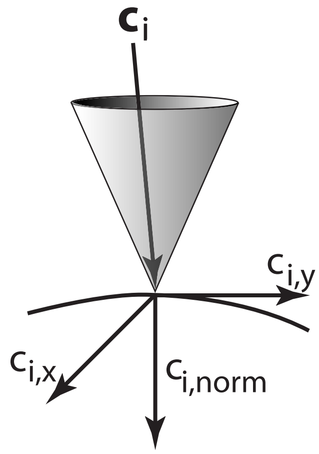


Fig. 2. A contact point showing the contact force vector, $\mathbf{c}_i$, its normal and tangential components, and the Lorentz cone within which the contact vector must reside.

to reduce joint effort and avoid excessive internal forces, which may crush delicate objects, the magnitude of the contact forces should also be minimized. These goals are contrasting, as reducing contact forces generally decreases friction margins. Therefore, a weighted sum of the two penalty terms should be used that enables end-users to select an operating region along the tradeoff curve between grasp stability and joint effort. The objective functions proposed in [4] and [12] behave in this manner, but the iterative methods used to solve the associated SDP formulations are characterized by a high computational burden, as is discussed in Section IV-B.3. In the following section, we propose a novel objective function that enables this flexibility without incurring the increased computational burden associated with SDP solvers.

## III. Weighted Barrier Functions for Force Distribution

In this section, we present a novel Weighted Barrier Function (WBF) formulation that efficiently generates grasping force distributions that possess a number of positive characteristics. Before doing so, it is necessary to introduce two barrier functions that are frequently used in constrained optimization problems. In Section III-A, we present a barrier function that is commonly employed for Linear Programs (LPs) and provide expressions for its gradient and Hessian matrix. In Section III-B, we introduce a barrier function commonly used in solving Second-Order Conic programs (SOCPs). Thereafter, in Section III-C, we present our novel formulation and discuss methods used to rapidly compute optimal solutions.

### A. Polyhedral Barrier Function

Consider the polyhedral set $\mathcal{Q} \in \Re^u$ that satisfies:

$$\mathcal{Q} = \{\mathbf{x} | \mathbf{A}\mathbf{x} \leq \mathbf{b}\}, \tag{5}$$

where $\mathbf{x} \in \Re^u$, and $\mathbf{A} \in \Re^{v \times u}$ and $\mathbf{b} \in \Re^v$ represent a set of $v$ linear inequality constraints on $\mathbf{x}$. A barrier function commonly used to represent such linear constraints is given by

$$\Phi_{lin}(\mathbf{x}) = -\sum_{i=1}^{v} ln(r_i), \tag{6}$$

where $r_i$ is the $i^{th}$ element of $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$. The gradient, $\nabla\Phi_{lin}(\mathbf{x}) \in \Re^u$, and Hessian matrix, $\nabla^2\Phi_{lin}(\mathbf{x}) \in \Re^{u \times u}$, are required for the optimization methods discussed below, so we provide them here:

$$\begin{aligned} \nabla\Phi_{lin}(\mathbf{x}) &= \mathbf{A}^T\mathbf{d} \\ \nabla^2\Phi_{lin}(\mathbf{x}) &= \mathbf{A}^T(\mathbf{diag}(\mathbf{d}))^{\mathbf{2}}\mathbf{A}, \end{aligned} \tag{7}$$

where $\mathbf{d} \in \Re^v$ is a vector whose $i^{th}$ entry is given by $\frac{1}{r_i}$ [13].

### B. SOC Barrier Function

If a vector $\mathbf{y} = [y_0|\hat{\mathbf{y}}]^T$ satisfies $\mu y_0 \geq \|\hat{\mathbf{y}}\|$, then $\mathbf{y}$ lies within the Lorentz cone defined by the parameter $\mu$, and we write $\mathbf{y} \succeq_{soc} \mathbf{0}$. Using this notation and splitting the vector $\mathbf{y}$ into the scalar $y_0$ and subvector $\hat{\mathbf{y}}$ is a common practice in SOCPs [8]. For a set of $\beta$ vectors, written as

$$\tilde{\mathbf{y}} = \begin{bmatrix} \mathbf{y}_1 \\ \vdots \\ \mathbf{y}_\beta \end{bmatrix}, \tag{8}$$

we can define the barrier function

$$\Phi_{soc}(\tilde{\mathbf{y}}) = -\sum_{i=1}^{\beta} ln((\mu_i y_{i_0})^2 - \|\hat{\mathbf{y}}_i\|^2), \tag{9}$$

which is finite only if all vectors lie within their corresponding cones.

In order to provide expressions for the gradient and Hessian matrix of $\Phi_{soc}(\tilde{\mathbf{y}})$, we first introduce the matrix

$$\mathbf{R}_{\mu_i} = \begin{pmatrix} -\mu_i^2 & 0 & \dots & 0 \\ 0 & 1 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 1 \end{pmatrix}, \tag{10}$$

which is frequently used in SOCPs. The gradient of $\Phi_{soc}(\mathbf{y}_i)$is given by

$$\nabla\Phi_{soc}(\mathbf{y}_i) = \frac{2}{g_i}\mathbf{R}_{\mu_i}\mathbf{y}_i, \tag{11}$$

where $g_i = \mu_i^2 y_{i_0}^2 - \|\hat{\mathbf{y}}_i\|^2$. The gradient of $\Phi_{soc}(\tilde{\mathbf{y}})$ is given by:

$$\nabla\Phi_{soc}(\tilde{\mathbf{y}}) = \begin{bmatrix} \nabla\Phi_{soc}(\mathbf{y}_1) \\ \vdots \\ \nabla\Phi_{soc}(\mathbf{y}_\beta) \end{bmatrix}. \tag{12}$$

The Hessian matrix of $\Phi_{soc}(\mathbf{y}_i)$ is given by:

$$\nabla^2 \Phi_{soc}(\mathbf{y}_i) = \nabla \Phi_{soc}(\mathbf{y}_i) \nabla^T \Phi_{soc}(\mathbf{y}_i) + \frac{2}{g_i} \mathbf{R}_{\mu_i}, \quad (13)$$

and the Hessian matrix of $\Phi_{soc}(\tilde{\mathbf{y}})$ is given by:

$$\nabla^2 \Phi_{soc}(\tilde{\mathbf{y}}) = \underset{i=1\ldots\beta}{\mathbf{Blockdiag}}(\nabla^2 \Phi_{soc}(\mathbf{y}_i)). \quad (14)$$

*C. Force Distribution by Weighted Barrier Functions*

*1) Weighted Barrier Function Formulation:* Recall from Eqn. (4) that the force optimization problem consisted of minimizing an objective function, $f(\mathbf{c})$, subject to linear equality constraints, linear inequality constraints, and nonlinear friction cone constraints. The linear inequality constraints can be written as:

$$\begin{aligned} &\mathbf{A}\mathbf{x} \leq \mathbf{b}, \text{ where} \\ &\mathbf{A} = \begin{bmatrix} \mathbf{J}^T \\ -\mathbf{J}^T \end{bmatrix}, \text{ and} \\ &\mathbf{b} = \begin{bmatrix} \boldsymbol{\tau}^U \\ -\boldsymbol{\tau}^L \end{bmatrix}. \end{aligned} \quad (15)$$

Furthermore, the nonlinear friction-cone constraint can be written as $\mathbf{c}_i \succeq_{soc} \mathbf{0}$, $i = 1 \ldots n$. Thus, we propose the following formulation for grasp force optimization:

$$\begin{aligned} &\overset{*}{\mathbf{c}} = \underset{\mathbf{c}}{\operatorname{argmin}}\, \bar{\Phi}(\mathbf{c}) \\ &\text{s.t. } \mathbf{W}\mathbf{c} = \mathbf{f}, \end{aligned} \quad (16)$$

where $\bar{\Phi}(\mathbf{c}) = \Phi_{soc}(\mathbf{c}) + \alpha \Phi_{lin}(\mathbf{c})$. We call this formulation the Weighted Barrier Function method because the objective function consists solely of barrier terms weighted relative to each other. The first term penalizes proximity to slippage conditions, and the second penalizes excessive joint effort. The effect of varying $\alpha$, is to adjust the relative weight of the two terms. For large $\alpha$, large joint efforts are penalized more severely, and lower grasp forces are expected, with the associated cost of increased probability of slip. For small $\alpha$, proximity to slip conditions is penalized more severely, and larger joint efforts are favored. This is discussed in greater detail in Section IV-A.

In Eqn. (16), the objective function and inequalities in Eqn. (4) have been replaced by a barrier funtion. The subsequent optimization still generates optimal force distributions, in that they are "centered" in the feasible space and thereby minimize proximity to constraint violations. This is akin to the work in [2], except that the notion of "centering" in [2] is roughly analagous to the Chebyshev center of a polyhedron, which is prone to step discontinuities.

It is noted here that interior solutions to all convex optimization problems follow smooth trajectories in response to smooth changes in input data [13]. For finite weights, logarithmic barrier functions result in strictly interior solutions. Thus, the use of analytic centering guarantees continuous force profiles.

*2) Reduced Formulation:* As noted in [2], linear equality constraints on $\mathbf{c}$ can be eliminated with the additional benefit of reducing the number of optimization variables. Any $\mathbf{c}$ satisfying $\mathbf{W}\mathbf{c} = \mathbf{f}$ can be rewritten as $\mathbf{c} = \mathbf{c}_p + \mathbf{c}_h$, where $\mathbf{c}_p$ represents a particular solution such that $\mathbf{W}\mathbf{c}_p = \mathbf{f}$, and where $\mathbf{c}_h$ lies in the nullspace of $\mathbf{W}$. A particular solution is readily found using $\mathbf{c}_p = \mathbf{W}^+\mathbf{f}$, where $^+$ indicates the pseudovinverse. Given a matrix $\mathbf{N}$ spanning the nullspace of $\mathbf{W}$, we can write

$$\mathbf{c} = \mathbf{W}^+\mathbf{f} + \mathbf{N}\boldsymbol{\lambda}, \quad (17)$$

where $\boldsymbol{\lambda}$ weights the vectors spanning the nullspace of $\mathbf{W}$. Eqn. (16) now becomes

$$\overset{*}{\boldsymbol{\lambda}} = \underset{\boldsymbol{\lambda}}{\operatorname{argmin}}\, \bar{\Phi}(\mathbf{c}_p + \mathbf{N}\boldsymbol{\lambda}). \quad (18)$$

If $\mathbf{W}$ is of full row rank, as it is in all nonsingular configurations, then $\mathbf{N} \in \Re^{m \times (m-6)}$ and $\boldsymbol{\lambda} \in \Re^{m-6}$. Thus, the dimension of the optimization vector has decreased from $m$ to $m-6$, which results in significant reductions in computational burden. It should be noted that the gradients and Hessian matrices will also change to reflect the change in variables. We now have:

$$\begin{aligned} \nabla \bar{\Phi}(\boldsymbol{\lambda}) &= \mathbf{N}^T(\nabla \Phi_{soc}(\mathbf{c}) + \alpha \nabla \Phi_{lin}(\mathbf{c})), \text{ and} \\ \nabla^2 \bar{\Phi}(\boldsymbol{\lambda}) &= \mathbf{N}^T(\nabla^2 \Phi_{soc}(\mathbf{c}) + \alpha \nabla^2 \Phi_{lin}(\mathbf{c}))\mathbf{N}. \end{aligned} \quad (19)$$

*3) Regrasping:* In many grasping applications, manipulators are required to disengage and re-engage fingers in order to execute otherwise impossible tasks. For example, Schlegl *et al.* consider a four-fingered hand screwing a light bulb into a socket [11]. It is clear that, to execute this task, a hand must disengage fingers and regrasp the light bulb in order to repeatedly wrench it. During regrasping, the desired wrench, $\mathbf{f}$, must still be delivered, and all joint torque and friction cone constraints must be met. Furthermore, contact forces must be continuous throughout, or grasp integrity may be compromised.

To enable smooth regrasping, we augment the linear inequalities given in Eqn. (15) with an additional upper constraint on the normal contact forces of each finger. The constraint assumes the following form:

$$c_{i_{norm}} \leq w_i(t) c_{max}, \quad (20)$$

where $c_{max}$ is a loose upper bound on the maximum normal contact force, and $0 \leq w_i(t) \leq 1$ is a weight associated with each finger. Under normal operating conditions, all fingers are active, and $w_i = 1$, $i = 1 \ldots n$. $c_{max}$ is considerably larger than any normally occurring contact forces, so, when $w_i = 1$, this constraint has a minimal impact on $\overset{*}{\mathbf{c}}_{\mathrm{i}}$. However, if finger $i$ is to be disengaged, $w_i(t)$ gradually shrinks from 1 to 0, causing a decrease in $c_{i_{norm}}$. When $w_i(t)$ becomes equal to 0, finger $i$ is entirely removed from the grasp, and the hand is treated as an $(n-1)$-fingered hand. When finger $i$ re-engages, its corresponding constraints are placed back into the force optimization, and $w_i(t)$ increases gradually from 0 to 1.

It is known that interior solutions to all convex optimization problems vary smoothly in response to smooth changes in input data. Smooth changes in $w_i(t)$ meet this smoothness criteria, and adding or removing fingers whose contact force is zero has no effect on the grasp. Thus, continuity of contact forces during regrasping is guaranteed.

*4) Computation Method:* Because the gradient and Hessian matrix of $\bar{\Phi}(\boldsymbol{\lambda})$ are readily computable, the well-known Newton method can be used to compute $\overset{*}{\boldsymbol{\lambda}}$. The Newton method is a globally convergent iterative algorithm that boasts quadratic convergence near the optimum [13]. During each iteration of the algorithm, the current guess $\boldsymbol{\lambda}_k$ is updated as follows:

$$\boldsymbol{\lambda}_{k+1} = \boldsymbol{\lambda}_k - \gamma \nabla^2 \bar{\Phi}_k^{-1} \nabla \bar{\Phi}_k, \tag{21}$$

where $\gamma \leq 1$ is a factor used to ensure that the Newton step does not leave the feasible region and that the new estimate has a lower objective value than the current one. In our implementation, $\gamma$ is found using a backtracking line search.

*5) Initialization:* The Newton method converges globally from any feasible initial point, and quadratic convergence is guaranteed near the optimum [13]. Thus, we must somehow generate a feasible starting iterant before the optimization can begin. $\overset{*}{\boldsymbol{\lambda}}(t)$ varies smoothly with smooth changes in the input data, as is the case for interior solutions to all convex optimization problems [13]. This property is tremendously useful in generating near-optimal starting iterants, in that the optima from the two previous iterations can be used to linearize the force profile and generate a starting iterant via extrapolation. As will be seen in Section IV-B, the proximity of this starting iterant to the global optimum yields very high efficiency.

For problems in which the feasible space is small, this extrapolated initial iterant may be infeasible. In this event, a Phase-1 solver can be used with the extrapolated starting iterant as its initial point. Because this starting point is generally near-feasible, in most cases only one iteration of the Phase-1 solver is required.

Typical Phase-1 solvers find feasible starting points by introducing a slack variable that expands the feasible space [13]. This slack variable is subsequently reduced until either a feasible starting point has been found or the problem constraints are found to be infeasible. In the latter case, the Phase-1 solver returns the solution for which the violation of problem constraints is minimized. Thus, infeasible finger configurations are managed gracefully.

## IV. Results

Simulations were performed in which a four-fingered hand manipulated a cube with a mass of 250 g and a side length of .075 m. Each finger is composed of two links of length .075 m. The first joint, which connects the first link to the rigid palm, is a two-DOF joint which can rotate about the axis normal to the hand and control the angle of the finger relative to the palm. The second joint has one DOF, namely the retraction or extension of the finger. The fingers extend from the sides of a .10 m square centered at the origin. All actuators are characterized by $\boldsymbol{\tau}^L = -.5$ N·m and $\boldsymbol{\tau}^U = .5$ N·m, and all links are considered massless. The coefficient of friction for all contact points is taken to be .5, and the contact points occur at the centers of the vertical sides of the cube. This is shown in Fig. 3.

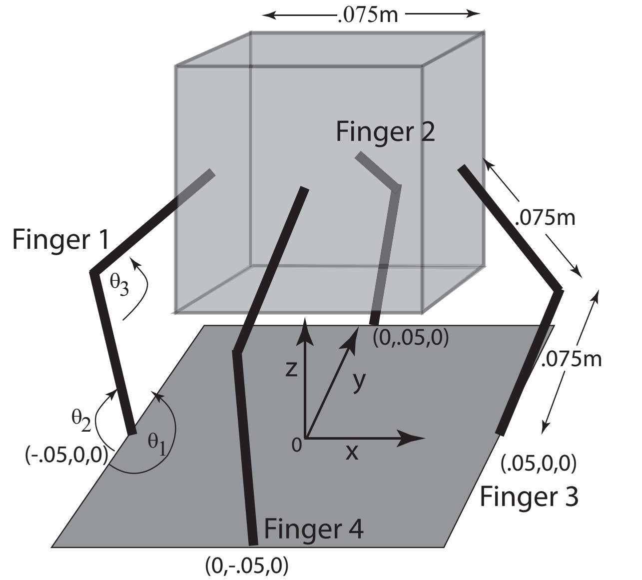


Fig. 3. The robotic grasping platform described in Section IV.

### A. Static Simulations

In order to demonstrate the effect that the weighting factor $\alpha$, described in Section III-C.1, has on grasps, we considered a static grasp of the cube, which was held with its center at $[.025 \ \ -.05 \ \ .1]$ m with a rotation of .1 radians about its vertical axis. $\alpha$ was varied from .01 to 100, and the corresponding optimal grasps were recorded. Fig. 4 shows the resulting normal forces of all fingers as a function of $\alpha$. It is apparent that increasing $\alpha$ results in significant decreases in normal forces and a much lighter grip.

We define the torque residual $r_\tau$ as the minimum difference between the vector of joint torques and their limits. Further, we define the friction residual, $r_f$, as follows:

$$r_f = \min_{i=1:n} (\mu_i^2 c_{i_{norm}}^2 - (c_{i_x}^2 + c_{i_y}^2)). \tag{22}$$

Thus, $r_f$ is a measure of proximity to violation of the friction cone constraints.

In Fig. 5, the normalized torque and friction residuals are plotted against $\alpha$. For small $\alpha$, large $r_f$ and small $r_\tau$ are observed. The reduced weighting of $\Phi_{lin}$ relative to $\Phi_{soc}$ yields high torques and a firm grasp. Thus, $r_f$ is large, and the grasp is resistant to slippage. For large $\alpha$, $r_\tau$ increases at the cost of a decrease in $r_f$, as the high weight of $\Phi_{lin}$ relative to $\Phi_{soc}$ forces joint torques away from their limits.

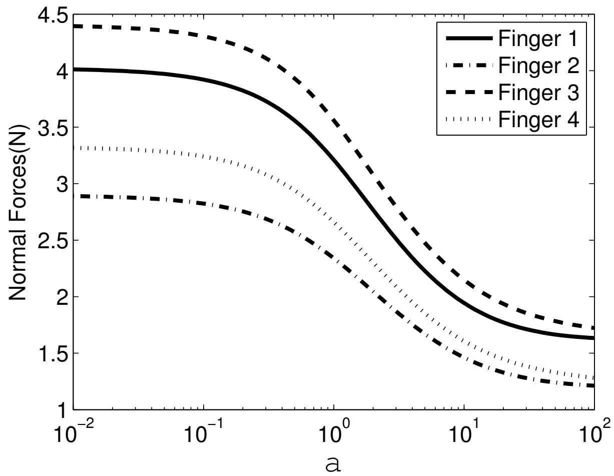


Fig. 4. Normal contact forces shown as a function of $\alpha$.

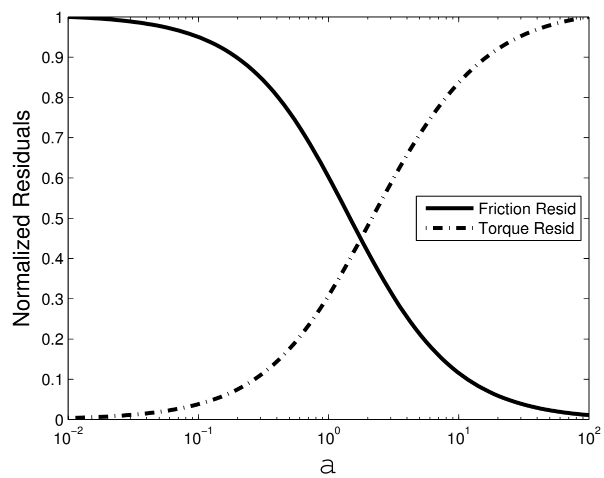


Fig. 5. Torque and friction residuals shown against $\alpha$.

It is also evident that most of the variation occurs between $\alpha = .1$ and $\alpha = 10$ and that significant flexibility is enabled by moderate values of $\alpha$.

*B. Dynamic Simulations*

In order to demonstrate the numerical efficiency of computing WBF solutions, we simulated the execution of a dynamic trajectory on the same grasping mechanism as considered above. In a two-second span, the cube executes a horizontal circle with a radius of .025 m centered at $[.025\ 0\ .05]$ m, while $\theta_z$ varies sinusoidally with an amplitude of $\pi/8$ and a frequency of 1 Hz. We consider $\alpha$=5 and a servo rate of 250 Hz.

*1) Four-fingered Simulations:* Execution of the trajectory was simulated, and WBF forces were computed and recorded. Fig. 6 shows normal contact forces, and Fig. 7 shows maximum joint torques. It is apparent that all joints remain well below their torque limits. A plot of $r_f(t)$, shown

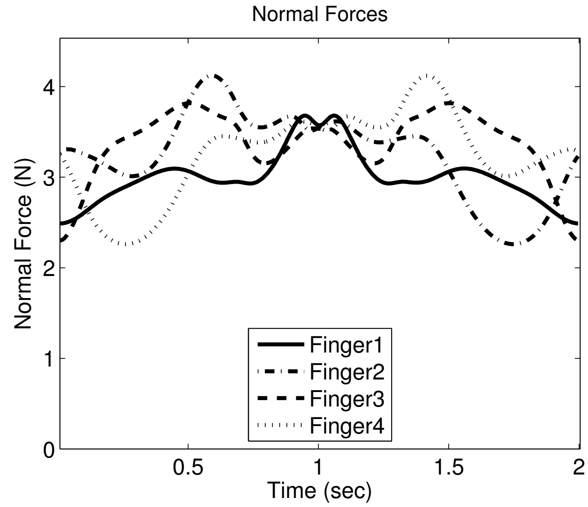


Fig. 6. Normal forces during four-fingered simulations described in Section IV-B.1.

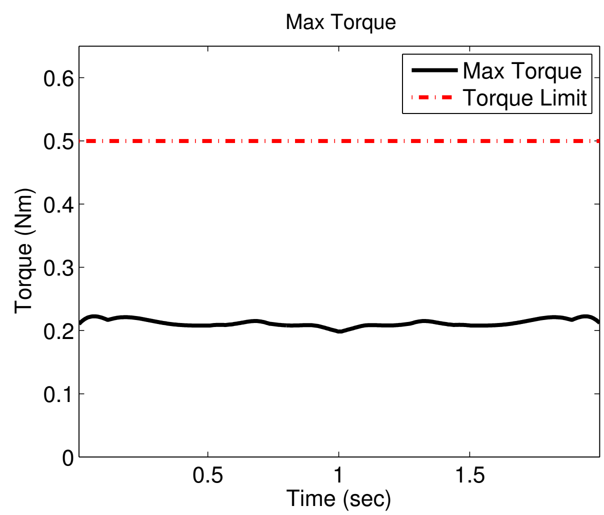


Fig. 7. Maximum joint torque during four-fingered simulations described in Section IV-B.1.

in Fig. 8, indicates that the friction residual remains safely above 0 and that slippage conditions are avoided. During this simulation, each computation of WBF forces required only a single iteration of the Newton method, and the mean computation time was roughly 700 $\mu$s on an Intel Pentium M processor running at 1.5GHz. It should be noted that these results were generated using code written in an interpretive language and that significant reductions in computation time are expected for compiled versions. While this computational burden is expected to rise in real systems, where measurement errors and control inputs may cause more fluctuation in $\mathbf{f}$, we still expect the extrapolation method to produce near-optimal starting iterates well within the region of quadratic convergence of Newton's method. Thus, the increase in computational burden would be moderate.

*2) Regrasping Simulations:* In order to demonstrate regrasping capabilities, we consider the simulated execution

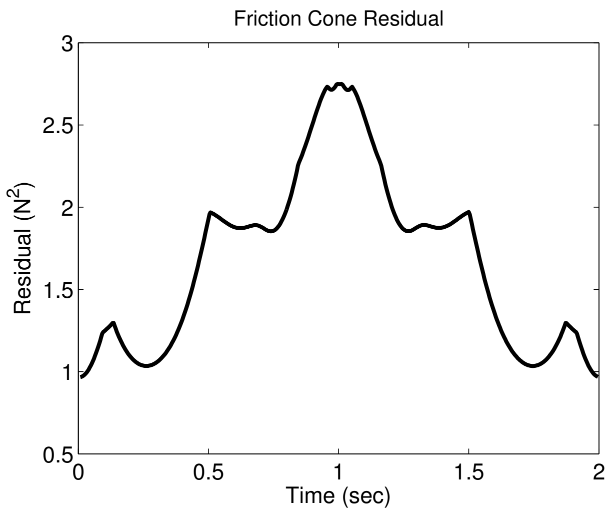


Fig. 8. Friction residual during four-fingered simulations described in Section IV-B.1.

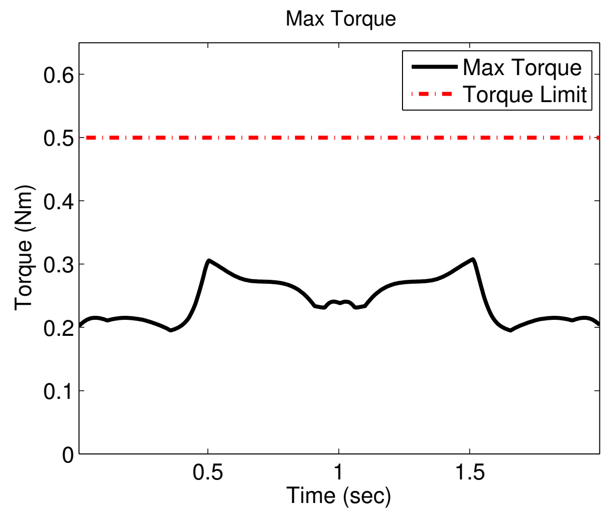


Fig. 10. Maximum joint torque during regrasping simulations described in Section IV-B.2.

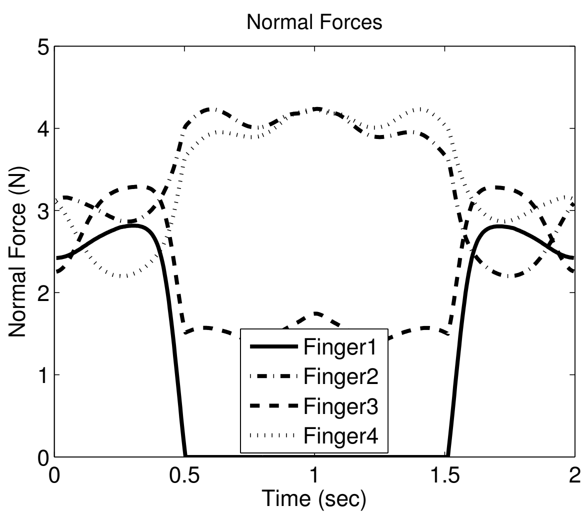


Fig. 9. Normal forces during regrasping simulations described in Section IV-B.2.

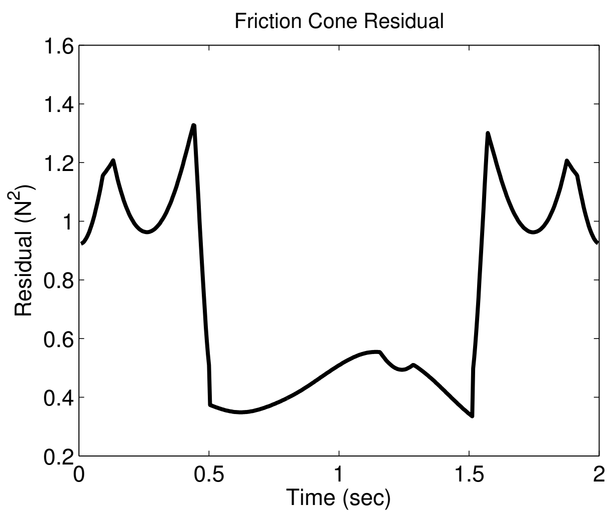


Fig. 11. Friction residual during regrasping simulations described in Section IV-B.2.

of the same trajectory described above. However, in this simulation, finger 1 is removed between $t = .5$ s and $t = 1.5$ s. Because the removal of finger 1 can not occur instantly, $w_1(t)$ begins to decrease from 1 at $t = .25$ s. Similarly, the regrasping phase occurs over a period of .25 s.

Fig. 9 shows the normal contact forces during execution of the trajectory. Continuous grasping forces are observed, even during the removal and re-engaging of finger 1. When finger 1 is disengaged, finger 3, which otherwise provides much of the force opposing finger 1, reduces its normal force. The other two fingers increase their grasping forces in order to maintain grasp stability. The plot of maximum joint torque in Fig. 10 shows that the removal of finger 1 results in an increase in maximum joint torque from $t = .5$ s to $t = 1.5$ s.

Fig. 11 shows the friction residual, $r_f(t)$, during execution of the trajectory. Despite the efforts of fingers 2-4 to stabilize the grasped object, the absence of finger 1 significantly reduces the robustness of the grasp. In Fig. 12, the number of Newton iterations required to compute $\overset{*}{\mathbf{c}}$ is shown along $w_1(t)$. During typical operation, only one Newton iteration is required to converge to the optimum. However, when $w_1$ is very close to 0, the feasible region is small, and the problem becomes ill-conditioned. As a result, the number of required iterations increases. To mitigate this, finger $i$ could be removed from the grasping formulation when $w_i$ becomes sufficiently small. During four-fingered operation, typical computation times were roughly 700 $\mu$s, and, during three-fingered operation, computation times shrank to roughly 600 $\mu$s.

*3) Analysis of Computational Complexity:* In computing WBF grasping forces, the Newton step must be computed using Eqn. (21) during each iteration. The number of floating-

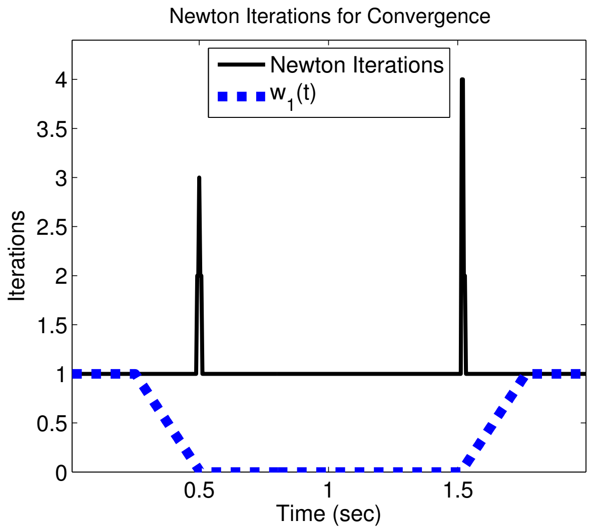


Fig. 12. Newton iterations required for convergence shown with $w_1(t)$ during simulated execution of a test trajectory.

point operations (flops) required to compute the gradient and Hessian and then solve the linear system for the four-fingered robot considered in Section IV-B.1 is roughly 12 kflops. In our trials, only one iteration of the Newton method was required during each servo-loop, so we expect the cost of computing WBF force distributions to be on the order of 12 kflops.

SDP formulations, such as those presented in [5], are typically solved using iterative projection methods, in which much of the computational effort is dedicated to computing the solution to a particular least-squares problem [7]. If no special matrix structures are exploited, then each Linear Matrix Inequality (LMI) contributes $\frac{p^6}{4}$ flops, where $p$ is the row dimension of the matrix used to represent the LMI. For the four-fingered case considered above, using the formulation in [12], there are four LMIs with $p = 3$ (to represent the friction cone constraints), and two LMIs with $p = 6$ (to represent the force equality and torque constraints). This results in roughly 24kflops, twice the burden of computing a Newton step. It should be noted that this burden represents only a fraction of the per-iteration cost of the SDP solver. In [12], convergence typically occurs in roughly 5 iterations for a four-finger grasper slightly simpler than ours, resulting in a computational burden of over 120kflops. Thus, the cost of computing grasp forces using our WBM method is at least an order of magnitude lower than that incurred by SDP solvers.

## V. Conclusion

In this paper, we have presented the Weighted Barrier Function (WBF) method of computing grasping force distributions. WBF distributions are continuous and rapidly computable, and they enable flexibility in selecting between firm, stable grasps or looser, more efficient grasps. Furthermore, fingers can be disengaged and re-engaged in a smooth manner, which is a critical capability for a large number of manipulation tasks. Previous methods with similar capabilities [4], [12], have been elegantly formulated as SDPs, but the computational burden posed by SDP solvers vastly exceeds that required of the SOCP formulation presented in this paper.

## References


[1] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *International Conference on Robotics and Automation (ICRA)*, 2000, pp. 348–353.

[2] J. Kerr and B. Roth, "Analysis of multifingered hands," *International Journal of Robotics Research*, vol. 4, no. 4, pp. 3–17, 1986.

[3] F.-T. Cheng and D. Orin, "Efficient algorithm for optimal force distribution-the compact-dual lp method," *Robotics and Automation, IEEE Transactions on*, vol. 6, no. 2, pp. 178–187, Apr 1990.

[4] M. Buss, H. Hashimoto, and J. Moore, "Dextrous hand grasping force optimization," *Robotics and Automation, IEEE Transactions on*, vol. 12, no. 3, pp. 406–418, Jun 1996.

[5] M. Buss, L. Faybusovich, and J. Moore, "Recursive algorithms for real-time grasping force optimization," in *1997 IEEE International Conference on Robotics and Automation*, vol. 1, Apr 1997, pp. 682–687 vol.1.

[6] M. Buss, L. Faybusovich, and J. B. Moore, "Dikin-type algorithms for dextrous grasping force optimization," *International Journal of Robotics Research*, vol. 17, no. 8, pp. 831–839, 1998.

[7] A. Nemirovskii and P. Gahinet, "The projective method for solving linear matrix inequalities," in *American Control Conference, 1994*, vol. 1, June-1 July 1994, pp. 840–844 vol.1.

[8] F. Alizadeh and D. Goldfarb, "Second-order cone programming," *Mathematical Programming*, vol. 95, pp. 3–51, 2001.

[9] G. Liu and Z. Li, "Real-time grasping-force optimization for multi-fingered manipulation: theory and experiments," *IEEE/ASME Transactions on Mechatronics*, vol. 9, no. 1, pp. 65–77, March 2004.

[10] Y. Xia, J. Wang, and L.-M. Fok, "Grasping-force optimization for multifingered robotic hands using a recurrent neural network," *IEEE Transactions on Robotics and Automation*, vol. 20, no. 3, pp. 549–554, June 2004.

[11] T. Schlegl, M. Buss, T. Omata, and G. Schmidt, "Fast dextrous regrasping with optimal contact forces and contact sensor-based impedance control," in *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, May 1994, pp. 103–108.

[12] L. Han, J. C. Trinkle, and Z. Li, "Grasp analysis as linear matrix inequality problems," *IEEE Transactions on Robotics and Automation*, vol. 16, no. 6, pp. 663–674, December 2000.

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