

A Quadratic Programming Approach to Quasi-Static Whole-Body Manipulation

Krishna Shankar, Joel W. Burdick and Nicolas H. Hudson

Abstract This paper introduces a local motion planning method for robotic systems with manipulating limbs, moving bases (legged or wheeled), and stance stability constraints arising from the presence of gravity. We formulate the problem of selecting local motions as a linearly constrained quadratic program (QP), that can be solved efficiently. The solution to this QP is a tuple of locally optimal joint velocities. By using these velocities to step towards a goal, both a path and an inverse-kinematic solution to the goal are obtained. This formulation can be used directly for real-time control, or as a local motion planner to connect waypoints. This method is particularly useful for high-degree-of-freedom mobile robotic systems, as the QP solution scales well with the number of joints. We also show how a number of practically important geometric constraints (collision avoidance, mechanism self-collision avoidance, gaze direction, etc.) can be readily incorporated into either the constraint or objective parts of the formulation. Additionally, motion of the base, a particular joint, or a particular link can be encouraged/discouraged as desired. We summarize the important kinematic variables of the formulation, including the stance Jacobian, the reach Jacobian, and a center of mass Jacobian. The method is easily extended to provide sparse solutions, where the fewest number of joints are moved, by iteration using Tibshirani's method to accommodate an l_1 regularizer. The approach is validated and demonstrated on SURROGATE, a mobile robot with a TALON base, a 7 DOF serial-revolute torso, and two 7 DOF modular arms developed at JPL/Caltech.

K. Shankar (✉) · J.W. Burdick · N.H. Hudson
California Institute of Technology, Pasadena, USA
e-mail: krishna@robotics.caltech.edu

J.W. Burdick
e-mail: jwb@robotics.caltech.edu

N.H. Hudson
e-mail: nhudson@jpl.nasa.gov

1 Introduction

Consider one or more (possibly redundant) serial chain manipulator arms mounted on a mobile robot base. The base could be a wheeled or tracked vehicle, or it may be a multi-legged walker (see Fig. 1a, b). The mechanism may also contain a neck upon which visual and range sensors are mounted. We are particularly interested in the cases where the arms have sufficient reach and mass such that the mobile vehicle may tip over in the presence of gravity when they are extended too far during a manipulation task.

Suppose the robot must manipulate an object, where the manipulation task can be described by tool frame locations.

1. What arm configurations satisfy the manipulation constraints?
2. How do we apportion base and limb motion to achieve the goal?
3. How do we guard against vehicle tip-over—can we move the limbs in such a way as to keep the system center of mass over a safe region of support? Can we move in a way to *improve* stability with respect to gravitational forces?
4. How do we incorporate natural task constraints, such as mechanism self-collision, obstacle avoidance, and preferred camera gaze direction?

These problems form a generalized inverse kinematic problem, where the distal end of the manipulator(s) must be placed at specified locations, while incorporating numerous constraints, as well as optimality criteria which resolve ambiguities in the case of multiple possible solutions. The optimality criteria also endow the solution with desirable properties. Since the analysis in this paper is limited to quasi-static motions, the key kinematic variables governing arm motions and center of mass stability can be formulated in terms of appropriate Jacobian matrices (see Sect. 2).

Because systems of the type seen in Fig. 1a, b are kinematically redundant, we propose a solution which is intellectually related to the classical methods of redundancy resolution in fixed based redundant manipulator arms [1, 2]. However, instead of

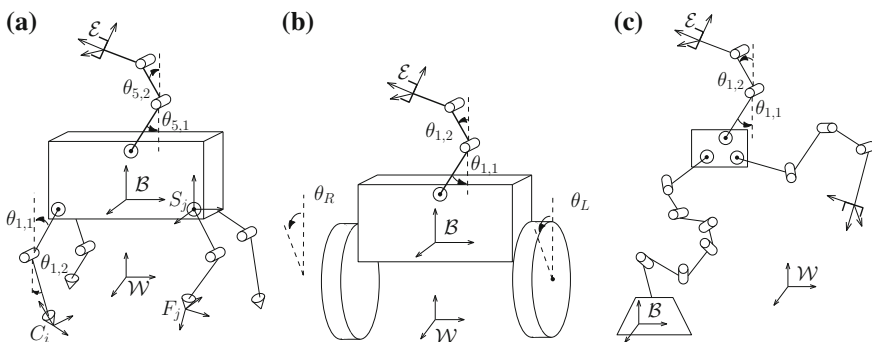


Fig. 1 Key reference frames for, **a** a legged-base robot, **b** a wheeled-base robot, **c** manipulator base robot

using a classical Jacobian pseudo-inverse type of solution, we formulate the problem as a convex optimization problem, specifically a constrained Quadratic Programming problem. Like the task-priority method of redundancy resolution [2], the formulation allows for multiple task priorities to be encoded as constraints or optimality criteria. However, unlike Jacobian pseudo-inverse methods, our QP formulation readily incorporates hard constraints and multiple optimality criteria, has better performance near singularities, and in practice tends to avoid awkward solutions for large kinematic chains [3]. Its real-time performance renders obsolete the need for heuristics [4] or look-up tables [5] to circumvent the curse of dimensionality with highly articulated systems.

The problem of mobile manipulation planning [6, 7] and whole body motion planning for humanoids [8] or multi-legged systems has attracted researchers for several decades. Theoretical advances in Convex Programming, and the associated introduction of efficient numerical optimization codes, allow us to propose new approaches which have not only serious computational speed advantages, but also allow added flexibility and generality in specifying the task objectives.

We are not the first to propose the use of Convex Optimization or Quadratic Programming techniques for solving inverse kinematics problems, for local motion planning of highly articulated mechanisms, or whole body manipulation planning. Zhang et al. [9, 10] used quadratic programming to solve kinematically redundant manipulator redundancy resolution problems. Kanehiro et al. [11, 12] show the use of QPs to incorporate fast collision avoidance calculations as part of humanoid whole body motion planning. Our method incorporates many additional constraints and optimality criteria, and our explicit formulation of several key kinematic equations [13] gives us significant advantages in terms of reported computation time (even adjusting for Moore's law). Very recently, MIT's DARPA Robotics Challenge team [14] used a sparse nonlinear optimization software that employs a sequential quadratic programming approach, to find inverse kinematic solutions to pose the Atlas humanoid robot, or to solve local motion planning problems involving manipulation with Atlas. We use a different objective function, which has several advantages, incorporate additional task criteria, and our explicit kinematic formulae also give us a reported computational speed advantage. We also provide solution existence and uniqueness results, and local feasibility certificates. Finally, our method can be readily adapted to provide *sparse* solutions, where only a minimum number of joints are moved in solving the motion planning problem. For mechanisms configured with brakes on the joint actuators, this option allows for energetically efficient mechanism motions, as much of the gravitational load on the mechanism can be supported by the friction forces at the brakes, instead of active joint torques.

We do not consider dynamic effects in this paper. However, many (e.g. [15] and citations therein) have obtained controllers for full dynamic models of humanoids with contact from simple convex QPs.

Structure of the Paper: Sect. 2 reviews the kinematics required to explicitly describe all possible motions of any robot link as well as the center of mass as a function of joint motions. In Sect. 3 we describe the optimization based approach to finding paths in configuration space that reach a given task-frame goal, and pro-

vide some analysis and describe extensions. In Sect. 4, we validate our ideas on the SURROGATE platform, and provide details related to computation time.

2 Stance and Reach Kinematics for Mobile Robots

We are interested in developing a whole body local planning framework which can be applied to

- multi-limbed robots, such as RoboSimian (see [13]), which can use its limbs either for walking or manipulating (Fig. 1a).
- wheeled or tracked vehicles mounted with one or more manipulators, and possibly articulated torsos (Fig. 1b and Sect. 4), such as the SURROGATE robot described in Sect. 4.
- multi-limbed robots with a fixed base, but a possible articulated torso (Fig. 1c). While such robots are not mobile in a conventional sense, their articulated torso presents a similar problem of apportioning the task-solving motions between the limb and the torso. Also, movements of the system’s center-of-mass far from the base places very large strain on the torso motors.

For this class of problems, we are concerned with describing the motions of a tool frame affixed to the manipulator(s), a frame describing the base’s location, and the location of the center of mass (since its position affects quasi-static stability of the vehicle), all with respect to a world frame, \mathcal{W} . This section reviews and derives the basic kinematic relationships between the movement of the frames and the mechanism joint motions. We also need to incorporate knowledge of the contact forces between feet and the terrain to ensure stability in the legged case. Let \mathcal{B} denote a right-handed orthogonal coordinate system fixed to the robot’s base, and let reference frame \mathcal{E}_i be affixed to the end-effector of the i th manipulating arm. We will call the point at which the manipulator is attached to the base a *shoulder*, and to it we associate the reference frame \mathcal{S}_i . We use conventions and notions from [16], which describes rigid body transformations using homogeneous transforms and velocities using twists. We will first develop some general relationships that govern this problem, and then specialize them for the particular classes of robots in Fig. 1.

2.1 Reach Jacobian

The location of the end effector in the world frame is given by $g_{\mathcal{W}\mathcal{E}} \in SE(3)$,

$$g_{\mathcal{W}\mathcal{E}} = g_{\mathcal{W}\mathcal{B}}g_{\mathcal{B}\mathcal{S}}g_{\mathcal{S}\mathcal{E}}$$

where g_{MN} describes the homogeneous transformation between reference frames M and N . The *body velocity* of the end effector, is a *twist*, defined by

$$V_{\mathcal{W}\mathcal{E}}^b = (g_{\mathcal{W}\mathcal{E}}^{-1} \dot{g}_{\mathcal{W}\mathcal{E}})^\vee = \left((g_{\mathcal{W}\mathcal{B}} g_{\mathcal{B}\mathcal{S}} g_{\mathcal{S}\mathcal{E}})^{-1} \frac{d}{dt} (g_{\mathcal{W}\mathcal{B}} g_{\mathcal{B}\mathcal{S}} g_{\mathcal{S}\mathcal{E}}) \right)^\vee = \text{Ad}_{g_{\mathcal{E}\mathcal{B}}} V_{\mathcal{W}\mathcal{B}}^b + V_{\mathcal{S}\mathcal{E}}^b$$

where the adjoint $\text{Ad}_{g_{MN}}$ transforms velocities from frame N to frame M . If we have a kinematic model (a map between joint velocities and robot motion) of the base, we can write

$$V_{\mathcal{W}\mathcal{E}}^\mathcal{E} = \text{Ad}_{g_{\mathcal{E}\mathcal{B}}} J_{\mathcal{B}}(\theta_{\mathcal{B}}, x_0) \dot{\theta}_{\mathcal{B}} + J_l(\theta_l) \dot{\theta}_l$$

where $J_{\mathcal{B}}(\theta_{\mathcal{B}}, x_0)$ is the *Base Jacobian*, x_0 includes additional necessary configuration and contact information (e.g. contact frame location and orientation) and the mechanism joint variables are divided into *base joint variables*, $\theta_{\mathcal{B}}$, and manipulating *limb joint variables*, θ_l , i.e. $\theta \triangleq (\theta_{\mathcal{B}}, \theta_l)^T$. Transforming this result to the Base Frame yields:

$$V_{\mathcal{W}\mathcal{E}}^{\mathcal{B}} = \begin{bmatrix} J_{\mathcal{B}}(\theta_{\mathcal{B}}, x_0) & 0 \\ 0 & \text{Ad}_{g_{\mathcal{B}\mathcal{E}}} J_l(\theta_l) \end{bmatrix} \dot{\theta} \triangleq J_{\mathcal{R}}(\theta, x_0) \dot{\theta}$$

where $J_{\mathcal{R}}(\theta, x_0)$ is the *Reach Jacobian*, which describes how base motions and limb motions contribute to tool frame motion, as described to an observer in frame \mathcal{B} .

2.2 The Center of Mass Jacobian

This section will derive the twist velocity of the center of mass for arbitrary motions of the base, supporting legs and any number of free arms. Suppose that our robot moves quasi-statically, and we want to ensure that it does not fall over during motion. We do this by ensuring that *static equilibrium* is satisfied at every instant. That is, the sum of moments and forces on any point of the robot are always zero.

Given a description of the robot's physical contacts with the world, the conditions for static equilibrium define a *support region* of all possible center of mass locations. For example, if the robot's feet only support point contacts without friction, and we assume that the robot can produce infinite torque at its joints, then the support region is the convex hull of the contacts.

Recall that the center of mass for a system of particles is located at the mass weighted average of the component particles' position. We can attach reference frames to each link whose origins coincide with the link center of mass. The transformation from the world-fixed frame to the center of mass in frame \mathcal{C} is then

$$g_{\mathcal{W}\mathcal{C}} = \frac{m_{\mathcal{B}}}{M} g_{\mathcal{W}\mathcal{B}} + \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^{n_i} m_{i,j} g_{\mathcal{W}(i,j)}$$

where $m_{i,j}$ is the mass of the j th link in the i th limb, and $g_{\mathcal{W}(i,j)}$ is a transformation from the world frame to the link frame. We can obtain the body velocity of the center

of mass by differentiating this expression:

$$\hat{V}_{wC} = g_{wC}^{-1} \dot{g}_{wC} = \frac{m_B}{M} g_{CB} \hat{V}_{wB}^B g_{CB}^{-1} + \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^{n_i} m_{i,j} g_{C(i,j)} \hat{V}_{w(i,j)}^{(i,j)} g_{C(i,j)}^{-1}$$

Converting to twist coordinates, and transforming to the base frame yields:

$$\begin{aligned} V_{wC}^B &= \text{Ad}_{g_{BC}} V_{wC}^C = \frac{m_B}{M} V_{wB}^B + \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^{n_i} m_{i,j} \left(V_{wB}^B + V_{B(i,j)}^B \right) \\ &= \frac{m_B}{M} V_{wB}^B + \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^{n_i} m_{i,j} \left(V_{wB}^B + \text{Ad}_{g_{BS_i}} J_{i,j}(\theta_{i \rightarrow j}) \dot{\theta}_{i \rightarrow j} \right) \\ &= V_{wB}^B + \frac{1}{M} \sum_{i=1}^N \sum_{j=1}^{n_i} m_{i,j} \text{Ad}_{g_{BS_i}} J_{i,j}(\theta_{i \rightarrow j}) \dot{\theta}_{i \rightarrow j} \end{aligned} \quad (1)$$

where

$$J_{i,j}(\theta_{i,1}, \dots, \theta_{i,j}) = \left[\left(\frac{\partial g_{S_i(i,j)}}{\partial \theta_{i,1}} g_{S_i(i,j)}^{-1} \right)^\wedge \dots \left(\frac{\partial g_{S_i(i,j)}}{\partial \theta_{i,j}} g_{S_i(i,j)}^{-1} \right)^\wedge \right]$$

is the (i, j) th link's Jacobian with respect to S_i (it is analogous to the spatial Jacobian of a manipulator with respect to its base). It maps the joint velocities of the first j joints in the i th limb ($\dot{\theta}_{i \rightarrow j}$) to the velocity of the j th link frame in the base B frame.

Following [13], this expression can be reorganized by introducing the “mass-weighted Jacobian”. The *mass-weighted Jacobian* is defined as

$$\bar{J}_k = \left[\left(\sum_{j=1}^{n_k} \frac{m_{k,j}}{M} \right) \xi_{1,k} \dots \left(\sum_{j=i}^{n_k} \frac{m_{k,j}}{M} \right) \xi'_{i,k} \dots \left(\frac{m_{k,n_k}}{M} \right) \xi'_{n_k,k} \right], \quad (2)$$

where M is the total robot mass and ξ_i is the twist associated with the i th joint at zero configuration, with

$$\xi'_i = \text{Ad}_{(e^{\hat{\xi}_1} \theta_1 \dots e^{\hat{\xi}_{i-1}} \theta_{i-1})} \xi_i.$$

We substitute it into Eq. (1) to get

$$V_{wC}^B = J_B(\theta) \dot{\theta}_B + \sum_{i=1}^N \text{Ad}_{g_{BS_i}} \bar{J}_i(\theta_i) \dot{\theta}_i = J_C(\theta, x_0) \begin{pmatrix} \theta_B \\ \theta_1 \\ \vdots \\ \theta_n \end{pmatrix}$$

where $J_C(\theta, x_0)$ is the “center-of-mass Jacobian”:

$$J_C(\theta, x_0) = \left[J_B(\theta_B, x_0) \text{Ad}_{g_{BS_1}} \bar{J}_1 \dots \text{Ad}_{g_{BS_n}} \bar{J}_N \right]$$

These building blocks are all that are required to solve a *very large class of local constrained inverse kinematics and planning problems*.

2.3 Kinematics for a Legged Base

The key specializations for the case of a legged base with arbitrarily many legs is summarized here, details are available in [13]. The kinematics for a legged robot arise from contact constraints—these prevent the feet from moving in directions that frictional forces can be applied in. Applying these constraints, we find that

$$S^T V_{\mathcal{WB}}^{\mathcal{B}} = J_{\mathcal{B}}(\theta_{\mathcal{B}}, x_0) \dot{\theta}_{\mathcal{B}} \tag{3}$$

where S , given by

$$S = - \left[\text{Ad}_{g_{\mathcal{A}c_1}}^{-T} B_{c_1} \cdots \text{Ad}_{g_{\mathcal{A}c_{n-1}}}^{-T} B_{c_{n-1}} \right]$$

is the *Stance Map*—a map between end effector forces and forces on the base (its transpose maps velocities at the contact frame to velocities at the base frame), and B_i is the wrench basis at the i th contact.¹ In the case of a legged robot, the θ 's don't split cleanly into a body component and limb components, as the body's motion is due to three or more supporting limbs. Define θ_i to the joint angles in the i th limb, an let n_i be the number of joints in the i th limb. Suppose that the robot has N limbs, and $M < N$ limbs making contact with the terrain at contact frames c_i , $i = 1 \dots m$. Then, for an M -limbed quasi static-walking robot, the base Jacobian takes the form:

$$J_{\mathcal{B}}(x_0, \theta_1, \dots, \theta_M) = - \begin{bmatrix} B_{c_1}^T \text{Ad}_{g_{s_1 c_1}}^{-1} J_1(\theta_1) & & & 0 \\ & \ddots & & \\ 0 & & B_{c_M}^T \text{Ad}_{g_{s_M c_M}}^{-1} J_M(\theta_M) & \end{bmatrix}. \tag{4}$$

The center of mass Jacobian is given by

$$J_{\mathcal{C}}(\theta, x_0) = \begin{bmatrix} B_{c_1}^T \text{Ad}_{g_{c_1 s_1}} \bar{J}_1(\theta_1) & B_{c_1}^T \text{Ad}_{g_{c_1 s_2}} \bar{J}_2(\theta_2) & \dots & B_{c_1}^T \text{Ad}_{g_{c_1 s_M}} \bar{J}_M(\theta_M) & \dots & B_{c_1}^T \text{Ad}_{g_{c_1 s_N}} \bar{J}_N(\theta_N) \\ B_{c_2}^T \text{Ad}_{g_{c_2 s_1}} \bar{J}_1(\theta_1) & B_{c_2}^T \text{Ad}_{g_{c_2 s_2}} \bar{J}_2(\theta_2) & & & & \\ \vdots & & \ddots & & & \\ \dots & & & B_{c_M}^T \text{Ad}_{g_{c_M s_M}} \bar{J}_M(\theta_M) & \dots & B_{c_M}^T \text{Ad}_{g_{c_M s_N}} \bar{J}_N(\theta_N) \end{bmatrix},$$

where \bar{J}_i is the weighted Jacobian defined in (2) and

¹Refer to [16], Chap. 5 for background.

$$\tilde{J}_k = \left[\left(\sum_{j=1}^{n_k} \frac{m_{k,j}+M}{M} \right) \xi_{k,1} \cdots \left(\sum_{j=i}^{n_k} \frac{m_{k,j}+M}{M} \right) \xi'_{k,i} \cdots \left(\frac{m_{k,n_k}+M}{M} \right) \xi'_{k,n_k} \right].$$

It is straightforward to show that [13]

$$S^T V_{WC}^B = J_C(\theta, x_0)\dot{\theta}. \tag{5}$$

2.4 Kinematics for a Wheeled Base

Suppose that we have manipulator arms attached to a differential-drive base with unit width. The base’s configuration is restricted to a plane, and consists of position and heading, (x, y, ϕ) . The motion of the base is described by²

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{\phi} \end{pmatrix} = \frac{1}{2} \begin{bmatrix} \cos \phi & \cos \phi \\ \sin \phi & \sin \phi \\ -1 & 1 \end{bmatrix} \begin{pmatrix} \dot{\theta}_L \\ \dot{\theta}_R \end{pmatrix}$$

where θ_L is the left wheel angle, and θ_R the right (see Fig. 1). We obtain the Base Jacobian by rewriting the kinematics in the body frame, and lifting to twists in 3 dimensions. One finds that the *base Jacobian* is given by

$$J_B = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ -\frac{1}{2} & \frac{1}{2} \end{bmatrix},$$

so that

$$V_{WB}^B = J_B \begin{pmatrix} \dot{\theta}_L \\ \dot{\theta}_R \end{pmatrix}$$

Suppose that the robot has N arms, and let the joint angles in the i th arm be denoted by θ_i . Then the Center of Mass Jacobian is simply

$$J_C(\theta_1, \theta_2, \dots, \theta_N) = [J_B \bar{J}_1(\theta_1) \dots \bar{J}_N(\theta_N)]$$

²These kinematics are well known, and arise from writing the no-slip conditions for each wheel with respect to the base frame.

2.5 Kinematics for a Serial Chain Torso

In this case, the spatial and base frame can be coincident, and the Base Jacobian is simply the Base’s spatial Jacobian:

$$J_{\mathcal{B}}(\theta_{\mathcal{B}}) = \left[\xi_1 \ \xi_2^\dagger \ \dots \ \xi_{n_{\mathcal{B}}}^\dagger \right]$$

where ξ_i is the twist associated with the i th joint at zero configuration and

$$\xi_i^\dagger = \text{Ad}_{(e^{\hat{\xi}_1 \theta_1} \dots e^{\hat{\xi}_{n_{\mathcal{B}}} \theta_{n_{\mathcal{B}}}})}^{-1} \xi_i .$$

The center of mass Jacobian in this case is

$$J_C(\theta, x_0) = \left[J_{\mathcal{B}}(\theta_{\mathcal{B}}, x_0) \text{Ad}_{g_{\mathcal{B}S_1}} \bar{J}_1(\theta_1) \ \dots \ \text{Ad}_{g_{\mathcal{B}S_n}} \bar{J}_N(\theta_N) \right]$$

3 The Local Motion Planning Problem: A Quadratic Program

This section motivates and describes the formulation of the explicit QP that is the crux of this paper. A basic problem is formulated, and then extended for more general use.

Recall that our task is described as an end-effector pose. Suppose that at every instant, we move optimally based only on knowledge of the current configuration, and the system kinematics. Naively, we might try to ‘*move towards the goal as much as possible, without falling down*’. This statement is very naturally translated into a constrained minimization problem:

$$\begin{aligned} &\text{minimize } \|V_{\mathcal{W}\mathcal{E}}^{\mathcal{B}} - \tilde{V}_{\mathcal{W}\mathcal{E}}^{\mathcal{B}}\|_{2, P_{\mathcal{E}}} + \|V_{\mathcal{W}C}^{\mathcal{B}} - \tilde{V}_{\mathcal{W}C}^{\mathcal{B}}\|_{2, P_C} + \|\dot{\theta}\|_{2, P_{\theta}} \\ &\text{subject to } J_R(\theta, x_0)\dot{\theta} = V_{\mathcal{W}\mathcal{E}}^{\mathcal{B}} \\ &\qquad\qquad J_C(\theta, x_0)\dot{\theta} = V_{\mathcal{W}C}^{\mathcal{B}} \end{aligned} \tag{6}$$

The objective indicates that we want to choose $V_{\mathcal{W}\mathcal{E}}^{\mathcal{B}}$ to be close (with respect to a weighted 2-norm defined by $\|x\|_{2, P} = \sqrt{x^T P x}$ where the weighting matrices are nominally diagonal, e.g. $P_{\mathcal{E}} = \text{diag}[w_1^{\mathcal{E}} \ \dots \ w_6^{\mathcal{E}}]$) to a desired end-effector velocity $\tilde{V}_{\mathcal{W}\mathcal{E}}^{\mathcal{B}}$ and the true center of mass velocity should be close to a specified velocity $\tilde{V}_{\mathcal{W}C}^{\mathcal{B}}$. The desired center of mass velocity might be determined so that the resulting motion of the robot’s center of mass remains fully within the support region (e.g. towards the center of support—a more natural way to control center of mass motion, using constraints, is given below). The desired end-effector velocity is specified as

the tangent to the desired end-effector path in $SE(3)$.³ The path can be any c^2 curve. The weights' relative magnitude corresponds to the importance of each term and component of motion in a given problem. Generally, weights in $P_{\mathcal{E}}$ are chosen to be significantly larger than the other weights, as the end effector goal is the highest priority. We have an exhaustive understanding of existence and uniqueness of solutions to this problem, and we state it as a proposition:

Proposition 1 *The constrained minimization (6) has a solution when $P_{\mathcal{E}}$, P_C and P_{θ}*

1. *are positive definite, or*
2. *are positive semi-definite, and both $J_{\mathcal{R}}$ and J_C have full rank.*

Moreover, (6) has a unique solution whenever there is no local motion that keeps the center of mass and end effector stationary.

The proof of this fact is straightforward, and follows from rank analysis of the KKT matrix.⁴ For the reader interested in a more detailed argument, we provide and prove this fact explicitly for legged robots in [13]. The argument is almost identical for any robot. Practically, this means is that singularities do not have as large an impact as they do for existing iterative IK (inverse kinematics) solvers. Intuitively, this makes sense, since we are *not enforcing* velocities, but instead simply *encouraging* them. Geometrically, the solution is a oblique projection of possible joint velocities onto affine subspaces defined by the kinematics.

3.1 Fully Integrated Planning

We expand this framework and rewrite the problem in order to easily and extensibly handle a broad variety of constraints and goals, and to make the problem size as compact as possible in the interest of efficiency.⁵ We do this by substituting the equality constraints into the objective directly, and by adding linear inequality constraints to accommodate hard limits.

Let F_i^g be a frame (attached anywhere to the mechanism) with an associated motion goal and let $\tilde{V}_{\mathcal{B}F_i^g}^{\mathcal{B}}$ be the corresponding desired instantaneous velocity of this frame. In order to write the motion goals efficiently in the objective, we can square

³A *desired velocity* for the end-effector can be determined from the transformation between the current pose and the desired pose; the velocity (twist) corresponding to the error is determined by the matrix logarithm. See [16] Chap. 2.

⁴This is the coefficient matrix one obtains when the KKT conditions for this problem posed as a QP in standard form are written as a linear equation in primal and dual variables. See [17] for background.

⁵The worst case complexity of solving a QP with linear constraints is shown to be $O(n^3L)$ where n is the size of the decision variable, and L is the program input size [18].

norms (without changing anything), and expand the residual between desired and true motion as

$$\|V_{BF_i^g}^{\mathcal{B}} - \tilde{V}_{BF_i^g}^{\mathcal{B}}\|_{2, P_{F_i^g}}^2 = \dot{\theta}^T J_{F_i^g}^T P_{F_i^g} J_{F_i^g} \dot{\theta} - 2\dot{\theta}^T J_{F_i^g}^T \tilde{V}_{BF_i^g}^{\mathcal{B}} + (\tilde{V}_{BF_i^g}^{\mathcal{B}})^T \tilde{V}_{BF_i^g}^{\mathcal{B}}$$

In order to efficiently represent *hard* constraints, we notice that we can restrict the motion of a frame F_i^r in the ‘direction’ of \tilde{V}_i^r by enforcing

$$(\tilde{V}_i^r)^T V_{BF_i^r}^{\mathcal{B}} = (\tilde{V}_i^r)^T J_{F_i^r} \dot{\theta} \geq \alpha_i,$$

where $\alpha_i \geq 0$.

Suppose that we have n motion goals, and p hard constraints. Define

$$P = \sum_{i=1}^n J_{F_i^g}^T P_{F_i^g} J_{F_i^g} + P_\theta, \quad \beta = -2 \sum_{i=1}^n J_{F_i^g}^T \tilde{V}_{BF_i^g}^{\mathcal{B}},$$

$$A = \begin{bmatrix} -(\tilde{V}_1^r)^T J_{F_1^r} & - \\ - & \vdots & - \\ -(\tilde{V}_p^r)^T J_{F_p^r} & - \end{bmatrix}, \quad \alpha = \begin{pmatrix} \alpha_1 \\ \vdots \\ \alpha_p \end{pmatrix}$$

We can now write a much more general constrained minimization problem:

$$\begin{aligned} &\text{minimize } \dot{\theta}^T P \dot{\theta} + \dot{\theta}^T \beta \\ &\text{subject to } A \dot{\theta} \leq \alpha \end{aligned} \tag{7}$$

where the inequality constraint holds element wise. With this more general formulation, a vast number of manipulation goals, subgoals and constraints can be naturally included. Some of these include:

Pointing The z -axis of a given link frame F_i can be pointed in particular direction by adding the link’s velocity and the twist in the desired direction to the objective. One neglects rotation in the pointing direction by letting

$$P_{F_i} = \text{Ad}_{g_{F_i \mathcal{B}}}^T \text{diag}[w_1^{F_i}, w_2^{F_i}, \dots, w_5^{F_i}, 0] \text{Ad}_{g_{F_i \mathcal{B}}}$$

(the instantaneous rotation about the frame-fixed z -axis is ignored). This could be used, for example, to encourage a gaze direction.

Tracking A link frame F_i can be moved along a desired trajectory by adding it to the objective, along with the tangent to the trajectory at the current configuration.

Collision Repulsion If the robot is in a configuration at which it makes contact with obstacles or itself, the links in contact can be forced to move away by defining a repulsive velocity as the normal to the collision plane [19], and adding a hard

inequality constraint forcing the link to move in the repulsive direction by making the corresponding α_i a positive number. For self-collisions, one or both collision links can be made to move away from collision.

Hard Static Equilibrium Constraints The center of mass can be kept within the robot's support region using linear inequality constraints. Let v_i be the twist corresponding to pure translation towards the i th side of the support region. Let the distance to the i th side be d_i . If we enforce the constraint

$$v_i^T V_{BC}^B = v_i^T J_C(\theta, x_0) \dot{\theta} \leq d_i$$

for every side of the support region, and if the robot follows velocity for much less than 1 s, the center of mass will not leave the support region.

Frame Boundaries A frame (or the difference between frames) can be kept in any polyhedral region in space using inequality constraints on frame velocity; these are constructed in the same way as the hard static equilibrium constraints. This could be used, for example, to keep the robot within some workspace boundaries, or to enforce hard constraints on gaze.

Joint Limit and Singular Configuration Avoidance These are straightforward to implement as inequality constraints on joint velocities.

Configuration Biasing The robot can be biased towards a known nominal configuration by adding a body velocity bias to nominal pose, and a joint angle bias that penalizes motions away from nominal joint angles.

Fewest Joints Moving We can look for solutions that move as few joints as possible by using a weighted 1-norm on $\dot{\theta}$ (defined by $\|x\|_{1,P} = \sum_{i=1}^n |Px|_i$ in the objective of (6)). This problem tends to provide solutions that are *sparse* in joint velocities. The solver will remain quite fast in this case, as the problem can be solved by a few iterations (approximately the same number as the number of joints) of the problem formed without $\dot{\theta}$ in the objective [20].

3.2 Feasibility Certificates and Optimal Constraints

The ability to certify feasibility or lack thereof is crucial for ensuring that partial plans that end in unsafe configurations are not executed. The problem (7) is infeasible if and only if the constraints cannot be met. We can check constraint feasibility very rapidly by solving the following linear program (the parameters are the same as those in (7)):

$$\begin{aligned} & \text{minimize } -t \\ & \text{subject to } A\dot{\theta} - \alpha \leq t \\ & \quad t \leq 1. \end{aligned} \tag{8}$$

The optimal solution value is 1 if the constraints are feasible and 0 otherwise.

In order to choose α in a clever way, one might ensure that feasibility is satisfied using (8) for the minimum values of α , and thereafter select an ‘optimal’ α in the sense of the following problem:

$$\begin{aligned} & \text{minimize } \mathbf{1}^T \alpha \\ & \text{subject to } A\dot{\theta} \leq \alpha \\ & \quad \quad -1 \leq \alpha \leq 1 \end{aligned} \tag{9}$$

Normalizing the resulting optimal α , we get the constraints that most aggressively avoid the limits we put on the system.

3.3 Iterative Algorithm

This section describes a method that integrates the ideas presented in this paper so far (this is the algorithm we use in the experiment of Sect.4). We define a robot configuration data structure \mathcal{C} , that contains the transformations to every joint and link frame as well as the center of mass frames, the instantaneous twists of every joint in the robot as seen in the base frame. We assume there are n frames that are following trajectories, and up to p inequality constraints; when there are fewer than p constraints for an iteration, the unused rows and elements of A and α are chosen to be trivially satisfied. In addition, for checking feasibility, α is set to the minimum reasonable value (e.g. a very small positive number for collision avoidance, or exactly the distance to a support region boundary). We also assume the existence of the following functions.

GoalDist(\mathcal{C}) Returns the distance to the goal (e.g. distance between end-effector current and desired poses).

SupportRegionVector(\mathcal{C}, i) Returns direction from the center of mass to the i th face of the support region in the base frame, for $i \in 1 \dots s$ where s is the number of faces of the support region—the locus of center-of-mass locations at which the robot is quasi-statically stable.

COMDist(\mathcal{C}, i) Computes the distance from the center of mass to the i th face of the support region.

CheckFeasibility(A, α) Returns the solution to the LP (8).

SetAlpha(\mathcal{C}, A) Returns a sensible choice of α using (9) or otherwise.

SolveQP(P, β, A, α) Returns the solution to (7). A key part of any algorithm that works using the ideas presented here is the QP solver. We used CVXGEN [21], to generate a fast, custom, primal-dual interior point method solver (a small, stand-alone C code).

ComputeStepSize($\mathcal{C}, \dot{\theta}$) Computes a step size Δt by searching $[0, 1]$ and ensuring that stepping by $\Delta t \times \dot{\theta}$ does not result in violation of constraints. We found that this function is unnecessary in most cases (the step size can be set to 1).

Update($\mathcal{C}, \dot{\theta}, \Delta t$) Updates the configuration after moving by $\Delta t \times \dot{\theta}$.

```

Initialize  $\mathcal{C}$ ;
while  $GoalDist(\mathcal{C}) > \epsilon$  do
  for  $i = 1 \dots n$  do
    Compute desired twists  $\tilde{V}_{B_{F_i}}^B$ . Select weighting matrices  $P_i$ ;
  for  $i = 1 \dots s$  do
     $\tilde{V}_i^r = SupportRegionVector(\mathcal{C}, i)$ 
     $\alpha_i = COMDist(c, i)$ 
  if In Collision then
    for  $i = 1 \dots \# of Collisions$  do
      Set  $\tilde{V}_{B_{(i+s)}}^r$  to be collision normal
    Construct  $P, \beta, A, \alpha$  as given in (7);
  if CheckFeasibility( $A, \alpha$ ) then
    SetAlpha( $\mathcal{C}, A$ );
     $\hat{\theta} = SolveQP(P, \beta, A, \alpha)$ ;
     $\Delta t = ComputeStepSize(\mathcal{C}, \hat{\theta})$ ;
    Update( $\mathcal{C}, \hat{\theta}, \Delta t$ )
  else
    Return Failure

```

Algorithm 1: QP-based path-planning and goal configuration search.

4 Implementation with Surrogate

Surrogate is a highly redundant 21 DOF robot torso mounted on differential drive mobile base (Figs. 2, 3 and 4). The following experiments demonstrate the use of iteratively solving the local quadratic program, and using the resulting velocities to move the robot end effector closer to the desired goal while also moving the torso and non-manipulating arm to maintain balance. At the end of each iteration, the velocities are integrated (multiplied by the time constant used to compute velocity constraints in the QP) to form small joint position displacements. Joint motions are limited to 0.1 radians per iteration.

Collisions between robotic bodies are computed using Bullet Collision Detection [22]. If collisions are detected after applying a joint position update, the previous QP is run again with additional velocity constraints enforcing repulsive motion between the bodies found in contact. Bullet provides a pairwise list of bodies in collision, and approximate collision locations.

The average run time for a single iteration (computing all Jacobians, constraints, and collisions, and solving the QP) was 348 μ s. On average just solving the QP took 267 μ s, or 78 % of the computation time. All computations were restricted to a single processing core of a 2.4 GHz i7. In the reported cases (a–f) in Fig. 2 the maximum average iteration time was for case (a), which ends in a near singular case, at 584 μ s, while the minimum was case (b), at 274 μ s. The iterations were stopped when the end effector displacement error was less than 0.001 m and 0.001 rad. Figure 2 case

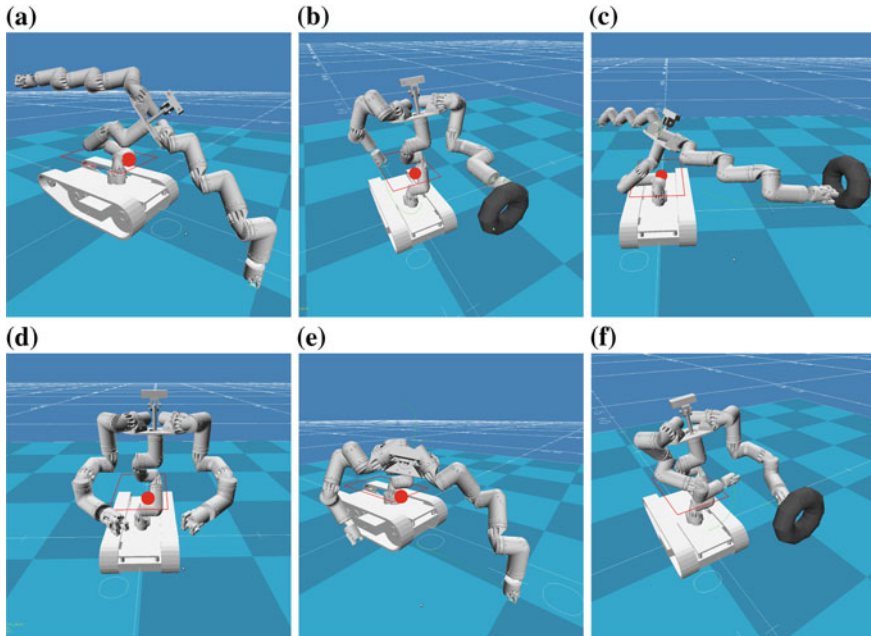


Fig. 2 Inverse Kinematics (IK) computed to specified end goals. The robot differential drive base is fixed. The *red sphere* is the robot center of mass (COM), and the *red rectangle* is the support polygon projected to the height of the COM. **a** Reaching to a point 1 m in front of the robot, and 0.2 m below the drive plane. **b** Reaching to a torus 0.5 m in front of the robot. **c** Reaching to a torus 1.25 m to the side of the robot. **d** Starting pose for all IK searches. **e** IK computed to (a) without using balance constraints. **f** IK computed to (b) stopping on detected collisions

(a) required 247 iterations to complete at a total time of 0.14 s, and case (b) took 17 iterations at a total time of 0.004 s.

The Surrogate robot has 7 degrees of freedom (DOF) in each limb and the torso. The serial chain from the robot base to the primary end effector is 14 DOF, with an extra 7 DOF on the free arm which can be used for balancing. This leaves 8 redundant DOF in the main serial chain, with an extra 7 DOF in the free limb. The limbs and torso on the Surrogate robot do not have a kinematic wrist, which makes deriving analytic inverse kinematics difficult.

As a comparison, IKfast (http://openrave.org/docs/latest_stable/openravepy/ikfast/) was used to compute analytic IK for the limb and the torso. Each IKfast call for the limb or torso requires fixing one joint, and solves for the IK of the remaining 6 joints in the limb or torso (resulting in up to 8 configurations). Each IKfast call for a Surrogate limb takes approximately 1000 ms, in contrast to a Barrett limb (with a wrist) which takes approximately 5 ms [23]. As the redundant space in the main serial chain is so high (8 DOF), searching over this space and using analytic IK to solve for joint angles is intractable. Figure 4 shows snapshots from a SURROGATE effort to turn a valve 90°.

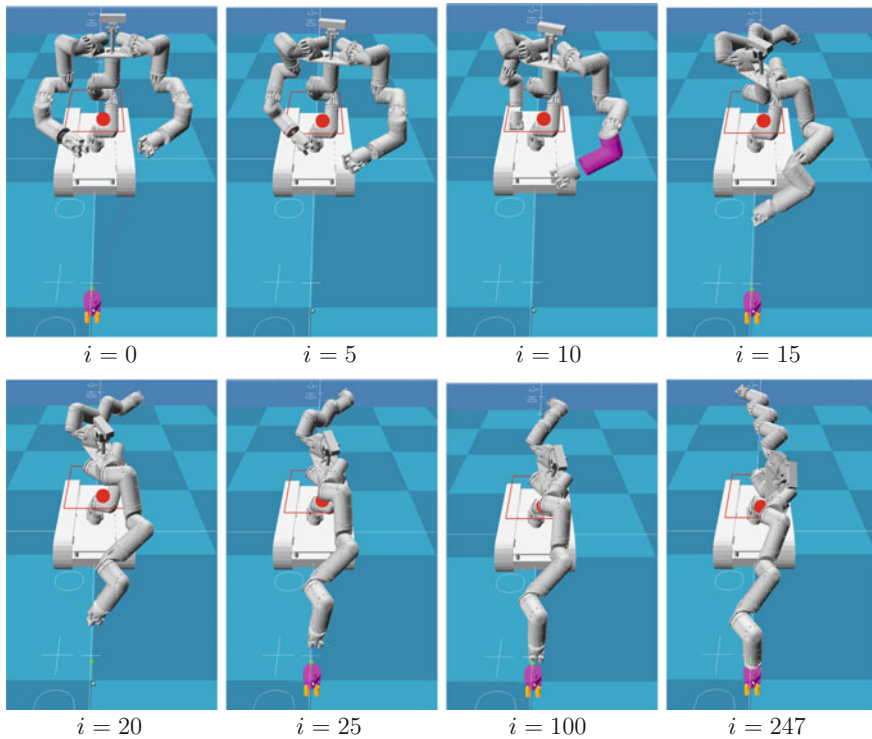


Fig. 3 Sequential iterations of solving the local QP to compute inverse kinematics for the *left* limb to a point 1 m in front and 0.2 m below robot base. The desired end effector location is shown by the highlighted hand at iteration $i = 0$. Subsequent iterations show the output robot state redisplayed. Iterations 5 and 10 show the free *right* limb being constrained from contacting the robotic torso. The robot center of mass (*red ball*) is within the support polygon (*red rectangle*) for all iterations

5 Conclusion and Future Work

This paper introduced a Quadratic Programming (QP) method to plan the local motions of robots with (possibly redundant) manipulator arms mounted on mobile bases (wheeled and legged bases, or highly articulated torsos) in the case where gravitational effects limit the possible stable locations of the system center of mass. We posed a generalized redundancy resolution approach which incorporates several optimality factors, while handling several types of constraints, such as self-collision, stable center of mass movement, local obstacle avoidance, and sensor gaze constraints. The locally optimal joint velocities produced by the QP solver can be used in real-time feedback control, or as a component of a global motion planning approach. Our application of the method to the 21 DOF SURROGATE robot resulted in surprisingly fast real-time solution performance. In part this is due to the efficiency of modern QP codes, but in part we believe the speed arose from our explicit formulation of

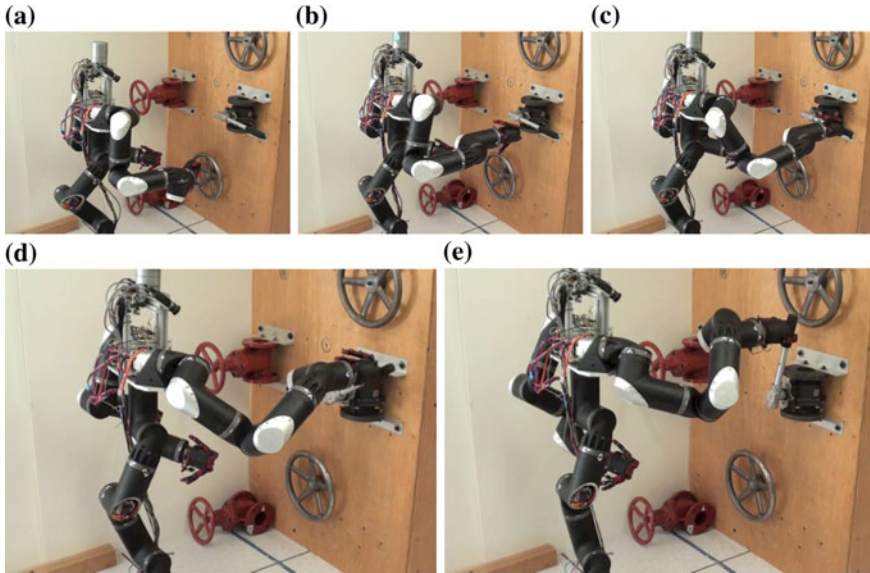


Fig. 4 Turning a valve using QP inverse kinematics: The robot’s motion is computed by iteratively solving the QP, and then executed on the robot in real time. **a** Starting pose. **b** Pre-contact with the valve. **c** Contact with the valve. **d** Half way through turning the valve. **e** The valve is fully open. The robotic torso significantly extended to achieve required end effector goals, and the free limb has contracted to maintain balance stability. Video available at <http://krishna.caltech.edu/WBM>

the key kinematic relationships. The QP formulation also leads to solution existence, uniqueness and infeasibility results. We are currently investigating how this local solution can be integrated into a receding horizon control and planning framework. Based on prior work, this combination should have the excellent real-time local performance demonstrated in this paper, coupled with completeness and correctness of a global motion planner.

References

1. Siciliano, B.: Kinematic control of redundant robot manipulators: a tutorial. *J. Intell. Robot. Syst.* **3**, 201–212 (1990)
2. Nakamura, Y., Hanafusa, H., Yoshikawa, T.: Task-priority based redundancy control of robot manipulators. *Int. J. Robot. Res.* **6**(2), 2–15 (1987)
3. Buss, S.R.: Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods
4. Bertram, D., Kuffner, J., Dillmann, R., et al.: An integrated approach to inverse kinematics and path planning for redundant manipulators. In: *Proceedings of the IEEE International Conference on Robotics and Automation*, pp. 1874–1879 (2006)
5. Satzinger, B.W., Reid, J.I., Bajracharya, M., et al.: More solutions means more problems: resolving kinematic redundancy in robot locomotion on complex terrain

6. Yamamoto, Y., Yun, X.: Coordinating locomotion and manipulation of a mobile manipulator. In: Proceedings of the 31st IEEE Conference on Decision and Control, pp. 2643–2648 (1992)
7. Brock, O., Khatib, O., Viji, S.: Task-consistent obstacle avoidance and motion behavior for mobile manipulation. In: Proceedings of the IEEE International Conference Robotics and Automation, pp. 388–393 (2002)
8. Harada, K., Kajita, S., Kanehiro, F., et al.: Real-time planning of humanoid robot's gait for force-controlled manipulation. *IEEE/ASME Trans. Mech.* **12**(1), 53–62 (2007)
9. Zhang, Y., Ge, S., Lee, T.: A unified quadratic-programming-based dynamical system approach to joint torque optimization of physically constrained redundant manipulators. *IEEE Trans. Syst. Man, Cybern. Part B: Cybern.* **34**(5), 2126–2132 (2004)
10. Zhang, Y., Zhang, Z.: *Repetitive Motion Planning and Control of Redundant Robot Manipulators*. Springer (2013)
11. Kanehiro, F., Lamiraux, F., Kanoun, O., et al.: A local collision avoidance method for non-strictly convex polyhedra. In: Proceedings of the Robotics: Science and Systems IV (2008)
12. Schulman, J., Duan, Y., Ho, J., et al.: Motion planning with sequential convex optimization and convex collision checking. *Int. J. Robot. Res.* (2014) (0278364914528132)
13. Shankar, K., Burdick, J.W.: Kinematics and methods for combined quasi-static stance/reach planning in multi-limbed robots. In: Proceedings of the IEEE International Conference on Robotics and Automation (2014)
14. Fallon, M., Kuindersma, S., Karumanchi, S., et al.: An architecture for online affordance-based perception and whole-body planning. Technical Report MIT-CSAIL-TR-2014-003, CSAIL, MIT, Boston, MA, March 2014
15. Kuindersma, S., Permenter, F., Tedrake, R.: An efficiently solvable quadratic program for stabilizing dynamic locomotion. arXiv preprint [arXiv:1311.1839](https://arxiv.org/abs/1311.1839) (2013)
16. Murray, R.M., Li, Z., Sastry, S.S.: *A Mathematical Introduction to Robotic Manipulation*. CRC Press (1994)
17. Boyd, S.P., Vandenberghe, L.: *Convex Optimization*. Cambridge University Press (2004)
18. Goldfarb, D., Liu, S.: An $O(n^3 \ln)$ primal interior point algorithm for convex quadratic programming. *Math. Progr.* **49**(1–3), 325–340 (1990)
19. Maciejewski, A.A., Klein, C.A.: Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *Int. J. Robot. Res.* **4**(3), 109–117 (1985)
20. Tibshirani, R.: Regression shrinkage and selection via the Lasso. *J. R. Stat. Soc. Series B (Methodological)* 267–288 (1996)
21. Mattingley, J., Boyd, S.: CVXGEN: a code generator for embedded convex optimization. *Optim. Eng.* **13**(1), 1–27 (2012)
22. Courmans, E.: Bullet physics engine. <http://bulletphysics.org/Bullet/BulletFull/>
23. Hudson, N., Ma, J., Hebert, P., et al.: Model-based autonomous system for performing dexterous, human-level manipulation tasks. *Auton. Robot.* **36**(1–2), 31–49 (2014)