CONTROL OF A CLIMBING ROBOT USING REAL-TIME CONVEX OPTIMIZATION

Teresa G. Miller [*] **Timothy W. Bretl** [*] **Stephen Rock** [*]

[*] *Department of Aeronautics and Astronautics, Stanford University, Stanford, CA 94305 USA*

Abstract:
This paper presents a controller for a free-climbing robot. Given a pre-planned path, a loop is closed around desired robot chassis position to generate cartesian feedback forces. A convex optimization problem is then solved in real-time to find a torque input to the robot that will achieve these feedback forces while not exceeding torque limits or violating friction constraints. The effectiveness of this controller is demonstrated both in simulation and with experimental hardware.

Keywords: control, control applications, convex optimization, robotics

1. INTRODUCTION

1.1 Overview

This work is part of an ongoing effort at Stanford University to enable a robot to free-climb vertical rock. Achieving this goal requires that issues be addresed in at least five topic areas: hardware design, sensing, planning, control and grasping (Bretl *et al.*, 2003*b*). Previous papers (Bretl, 2006; Bretl *et al.*, 2003*a*; Bretl *et al.*, 2003*b*) have focused primarily on the planning problem. This paper focuses on the design of a control system.

The target robot used for these studies is JPL's LEMUR robot, shown in Fig. 1. This four-limbed robot has sufficient strength to free-climb. An array of end effectors is available for LEMUR, of which simple "fingers" are used in this study. Joint angles as well as joint torques are sensed. Cameras are also available which can be mounted on the chassis of this robot.

In addition to LEMUR, a robot with similar geometry has been used to develop and test various control schemes for free-climbing. This robot is Stanford's Free-Flyer shown in Fig. 2. It operates in the horizontal plane by floating on an air cushion. Gravity is simulated by hanging weights over the edge of the granite table.

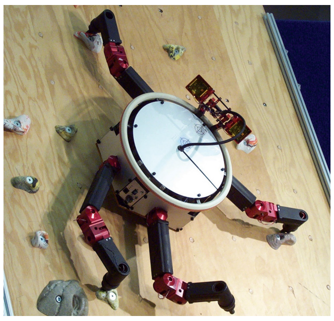
Fig. 1. Photograph of JPL's LEMUR robot.

1.2 Previous Results: Planning

Planning the motion of a free-climbing robot is a complex problem. The only thing that keeps the robot from falling is frictional contact with a carefully chosen set of *holds* (natural features such as holes or protrusions). The robot must apply contact forces at these holds that exactly compensate for gravity without causing hands or feet to slip. To take a single step upward, it must follow a continuous trajectory – consisting of joint angles, chassis position and orientation, contact

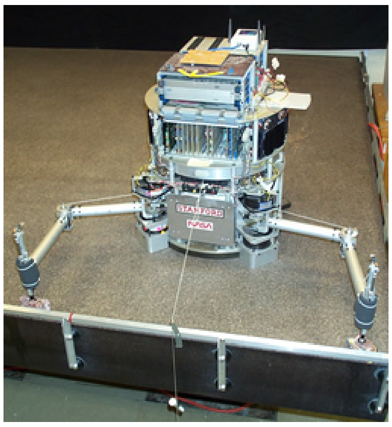

Fig. 2. Photograph of Stanford's Free-Flyer robot.

forces, and joint torques – that reaches a new hold without violating these constraints. To take many steps, it must follow a sequence of trajectories, avoiding those which might lead to dead-ends where progress is no longer possible.

Three basic constraints must be addressed by the planner. The first is that the robot's center of mass must remain above a support polygon to ensure that the robot does not fall. The second is that the contact forces between the robot's hands and the holds they are contacting remain within friction cones to ensure that the hands do not slip. The third is that the joint torques must remain below limits to ensure that joint motors do not saturate.

The feasibility of solving this planning problem, given a model of the robot and its environment, was developed and demonstrated in (Bretl, 2006; Bretl *et al.*, 2003*a*; Bretl *et al.*, 2003*b*). In particular, a long climb using JPL's LEMUR robot was demonstrated in (Bretl, 2006).

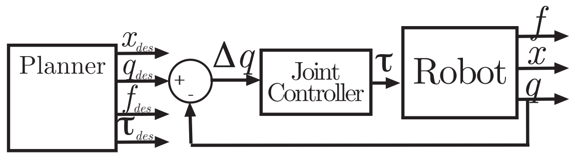

Fig. 3. Simplified block diagram of the controller used on LEMUR.

A simplified block diagram of the control system used in (Bretl, 2006) is presented in Fig. 3. In this approach, the joint-angles calculated in the planner are used as commands to feedback loops on the robot. This type of controller will be referred to as joint-angle control

The approach presented in Fig 3 can and did work well, but it is not robust to errors. In particular, it assumes perfect knowledge of the hold locations, zero error in the joint controllers, and no unmodeled disturbances. If these errors are small, and if there is sufficient margin designed into the planned trajectory, successful climbs are possible. However, in practice these assumptions are often violated, and a more robust control system is required.

The goal of this paper is to present and demonstrate such a modified control system.

1.3 Proposed Control System

The proposed control system is presented in Fig. 4. It incorporates two fundamental modifications to the system shown in Fig. 3. First, it replaces joint angle tracking with a control loop around the cartesian robot position. Second, it checks in real-time whether constraints are being satisfied based on measurements of the robot's configuration and modifies (or redistributes) joint torque commands to ensure that no constraint is violated. This is done by solving a linear programming problem inside the primary control loop.

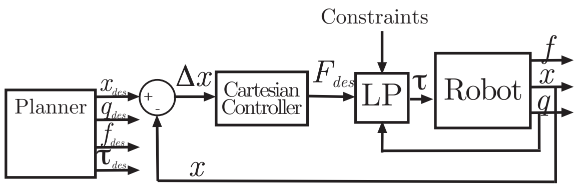

Fig. 4. Block diagram of the robot controller.

In this paper, the real-time constraint calculations include the limits on joint torques (i.e. saturation) and the requirement that the contact forces remain within specified friction cones (i.e. no slip). However, it is assumed that the plan has sufficient margin to ensure that the robot's center of mass (CM) remains over its support polygon (i.e. does not fall) given a cartesian robot position controller. This assumption is reasonable and is made for convenience. It is not required, however, and will be relaxed in future work.

This paper develops the controller shown in Fig. 4, referred to hereafter as Cartesian Force Convex (CFC) control. Results of using CFC control both in simulation and on the Free-Flyer hardware are presented.

2. RELATED WORK

Robotic climbing research has focused either on robots whose mechanical design is specialized for the climbing task or on robots that climb in a specialized environment. For example, the research described in Bevly *et al.* (2000) and Sunada *et al.* (1994) used variations on Jacobian Transpose control on a robot which interacted with its environment through a series of pegs and hooks. Because of the specialized end effectors, these researchers did not need to consider frictional force constraints for their application. The research of Shapiro *et al.* (2005) studied an application similar to climbing, but it focused on hold selection for a robot moving in a horizontal tunnel. By choosing holds that will immobilize the robot even when disturbed, they were able to control their robot using only joint-angle control.

Many aspects of the free climbing problem, particularly frictional constraints and redundancy caused by the existence of closed chains, make it similar to the problems of walking robots and object grasping. McGhee and Orin (1976) were among the first to note that it is possible to use mathematical programming to resolve redundancy in over-actuated systems, such as legged robots. Waldron (1986) developed simplifications to the walking robot problem that enable rapid, but suboptimal, calculations. Cheng and Orin (1989) and Nahon and Angeles (1992) developed algorithms for linear programming (LP) and quadratic programming (QP) respectively to control simulated grasping robots in real-time. This technique was implemented on a real robot hand by Schlegl *et al.* (2001), using LPs to optimize forces even while changing grasps. Fujimoto and Kawamura (1998) used a QP in their control loop to control not only the endpoint forces of a simulated bipedal walking robot, but also the position of the robot's center of mass.

3. ROBOT CONTROL

3.1 Configuration Definition

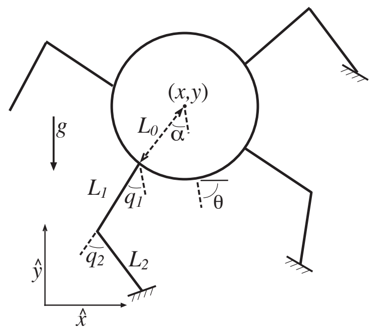

Fig. 5. Diagram of a 2D climbing robot showing lengths (L) and states. The robot chassis position is defined by $\boldsymbol{x} = (x, y, \theta)$. Joint angles are $\boldsymbol{q} = (q_1, q_2, ...)$.

A diagram of a climbing robot analogous to LEMUR is shown in Fig. 5. The robot moves in response to torques applied by motors at each joint, where each of these torques is limited to a maximum value. This robot interacts with its environment by placing the endpoints of its arms on frictional surfaces, or holds. The surface of each hold can be characterized by a friction cone. The edges of the friction cone define the maximum angle to the hold that force can be applied.

The sensors on-board the robot are assumed to be sufficient to measure or estimate joint angles, joint angular velocities, joint torques, body position, and body orientation. A typical sensor suite would include joint angle encoders, torque sensors, inclinometers, and accelerometers. Ideally the cartesian body position would be sensed using external sensors such as cameras or a localized position sensing system.

3.2 Controller Description

A diagram of the overall control architecture was shown in Fig. 4. The control of the robot takes place in two stages. In the first stage a desired force on the robot body, $\boldsymbol{F_{des}}$, is calculated based on the difference between the desired cartesian position of the robot chassis generated by the planner, $\boldsymbol{x_{des}}$, and the current best estimate of this value $\boldsymbol{x}$ as determined from the available sensors. This position $\boldsymbol{x}$ is assumed to have two translational and one rotational degree of freedom as shown in Fig. 5. This control can be of any type, but a simple PD controller proved adequate in the present study.

The second stage of control is to calculate control inputs, that is, joint torques $\boldsymbol{\tau}$, that will result in $\boldsymbol{F_{des}}$ being applied to the robot chassis. This mapping is not unique. There are more torque motors than there are dimensions in $\boldsymbol{F_{des}}$. Hence, a nullspace exists, and the existence of this nullspace makes it possible to compute a mapping that satisfiies the constraints on maximum torques and tip forces. For example, if the system has N 2-joint arms in contact with holds, the system then has only 3 degrees of freedom, but $2N$ torques to control. This means there is a $2N - 3$ dimensional nullspace that can be used for adjusting the magnitude and directions of forces on holds. For a robot with $N = 2$, the direction of this nullspace can be simply visualized: it is the direction that corresponds to the robot pinching its two holds. Any added force from the nullspace will not cause the robot to move but will change the direction of forces exerted by the hands on a hold.

3.3 LP Setup

Presented here is a method for formulating the objectives and constraints described above as a Linear-Programming (LP) problem and solving it using a convex optimization technique. The formulation is presented in two dimensions for clarity. However, this approach applies directly to the full three-dimensional problem (Bretl *et al.*, 2003*a*).

Each hold is modeled as a frictional point contact as shown in Fig. 6. Assuming dry Coulomb friction, the reaction force $\boldsymbol{f}_i$ at each hold i is constrained to vary within a friction cone FC_i centered about a normal vector. Defining unit vectors $\hat{f}_{i1}$ and $\hat{f}_{i2}$ along each edge of the cone, the sum of contact forces $f_{i1}\hat{f}_{i1} + f_{i2}\hat{f}_{i2} = \boldsymbol{f}_i$ lies within the friction cone FC_i if and only if $f_{i1}, f_{i2} \geq 0$. Therefore, for each hold i, the friction cone constraint on the reaction force can be expressed by this linear constraint on contact forces. That is, the robot hands are not slipping strictly if

$$\boldsymbol{f} \geq 0 \tag{1}$$

To describe the constraint that achieves $\boldsymbol{F_{des}}$, consider a stance with limbs in contact with N holds. Let $\boldsymbol{r}_i = (x_i, y_i)$ be the location of each hold i with respect to the robot chassis location (x_c, y_c). Assume that the robot has mass m, the vector from the chassis to the center of mass of the robot is $\boldsymbol{r}_{\text{CM}}$, the force of gravity acting at the CM is $\boldsymbol{F}_g$, and the desired force on the chassis is $\boldsymbol{F_{des}}$. Finally, let $\boldsymbol{f}_i = [f_{i1}\ f_{i2}]^T$ and define the following matrix for each hold:

$$\mathbf{F}_i = \begin{bmatrix} \hat{f}_{i1} & \hat{f}_{i2} \\ \left(\boldsymbol{r}_i \times \hat{f}_{i1}\right)_z & \left(\boldsymbol{r}_i \times \hat{f}_{i2}\right)_z \end{bmatrix}$$

where $\left(\boldsymbol{r}_i \times \hat{f}_{i1}\right)_z$ is the z component of the cross product. Then the robot can achieve $\boldsymbol{F}_{des}$ if $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_N \geq 0$ exist such that

$$\begin{bmatrix}\mathbf{F}_1 \cdots \mathbf{F}_N\end{bmatrix} \begin{bmatrix} \boldsymbol{f}_1 \\ \vdots \\ \boldsymbol{f}_N \end{bmatrix} + \begin{bmatrix} 0 \\ -mg \\ (\boldsymbol{r}_{\text{CM}} \times \boldsymbol{F}_g)_z \end{bmatrix} = \boldsymbol{F}_{des}$$

For convenience, define $\boldsymbol{f} = \left[\boldsymbol{f}_1^T \ldots \boldsymbol{f}_N^T\right]^T$ and $\mathbf{C}_1 = [\mathbf{F}_1 \ldots \mathbf{F}_N]$.

At a particular center of mass location define

$$\boldsymbol{d}_1 = \boldsymbol{F}_{des} - \begin{bmatrix} 0 \\ -mg \\ (\boldsymbol{r}_{\text{CM}} \times \boldsymbol{F}_g)_z \end{bmatrix}$$

So to achieve the desired force on the robot chassis, the vector of contact forces must satisfy

$$\mathbf{C}_1 \boldsymbol{f} = \boldsymbol{d}_1 \tag{2}$$

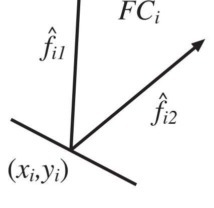

Fig. 6. Friction cone. A reaction force $\boldsymbol{f}_i = f_{i1}\hat{f}_{i1} + f_{i2}\hat{f}_{i2}$

In order to compute torques, it is also necessary to establish the relationship between the contact forces and the joint torques. Let τ_{i1} and τ_{i2} be the torque in the shoulder and elbow respectively of each limb j in contact with a hold. Let $\boldsymbol{\tau} = [\tau_{11}, \tau_{12}, \ldots, \tau_{N1}, \tau_{N2}]^T$. If the robot moves slowly through a climb, static equilibrium can be assumed and the relationship can be calculated by considering the sum of the forces on each joint of the arm. The result is an equation of the form:

$$\boldsymbol{\tau} = \mathbf{T}\boldsymbol{f} - \boldsymbol{g} = \begin{bmatrix} \mathbf{T_1} & \mathbf{0} & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{T_2} & \cdots & \mathbf{0} \\ \vdots & \vdots & \ddots & \\ \mathbf{0} & \mathbf{0} & & \mathbf{T_N} \end{bmatrix} \boldsymbol{f} - \boldsymbol{g} \tag{3}$$

where each $\mathbf{T}$ matrix is a two by two matrix and $\boldsymbol{g}$ is a vector of torques due to gravity on the arms.

Furthermore, each joint motor has a maximum allowable output, further constraining the problem such that

$$\boldsymbol{\tau} \leq \boldsymbol{\tau_{max}} \tag{4}$$

So, the problem of computing torques is the problem of finding contact forces $\boldsymbol{f}$ and joint torques $\boldsymbol{\tau}$ subject to the four constraints given in Eqs. 1-4.

These constraints still allow redundant solutions. A best one can be selected with a multiple objective cost function. A two term function is used in this study. The first term aims to minimize a weighted 1-norm of the joint torques. This linear cost can be written as:

$$J_1 = \left| \operatorname{diag}\left(\frac{1}{\tau_{max,11}}, \ldots, \frac{1}{\tau_{max,N2}}\right) \boldsymbol{\tau} \right|_1$$

The second term aims to center each contact force as much as possible in its friction cone. Ideally, if for each hold $f_{i1} = f_{i2}$, all contact forces would be in the center of their friction cones. This can be accomplished by minimizing (for the case of two holds in 2D):

$$J_2 = |f_{i2} - f_{i1}|_1 = \left| \begin{bmatrix} 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 \end{bmatrix} \boldsymbol{f} \right|_1$$

These two objectives are weighted by μ and can be minimized by solving the following linear program:

$$\begin{aligned} \min \quad & J_1 + \mu J_2 \\ \text{such that} \quad & \mathbf{C}_1 \boldsymbol{f} = \boldsymbol{d}_1 \\ & \boldsymbol{f} \geq 0 \\ & \boldsymbol{\tau} = \mathbf{T}\boldsymbol{f} - \boldsymbol{g} \\ & \boldsymbol{\tau} \leq \boldsymbol{\tau}_{max} \end{aligned}$$

This type of linear program can be solved using any of a number of existing software libraries. For this study it is solved by the simplex method using the open source software GNU Linear Programming Kit (GLPK).

4. SIMULATION AND EXPERIMENTS

In order to demonstrate the utility of the proposed control approach, simulations in Matlab and experiments with the Stanford Free-Flying robots were performed.

4.1 Simulation Results

A dynamic simulation in Matlab was performed of the Lemur robot executing a complex climbing maneuver. This simulation is performed at 200 Hz.

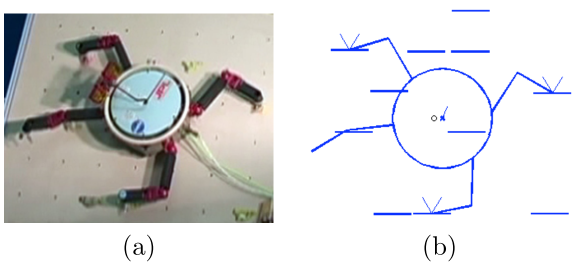

Fig. 7. (a) Photo of the Lemur robot. (b) A simulation of Lemur in this same position

The commanded maneuver is to begin at the position shown in Fig. 7 and to raise the robot chassis up while translating left. This maneuver is difficult because it requires torques in the joints of two arms that approach their limits.

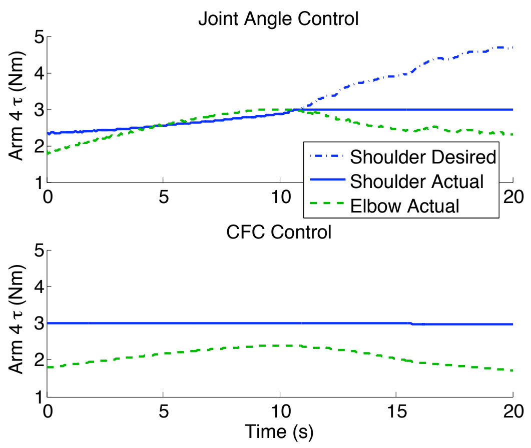

Fig. 8. Using joint control, the simulated Lemur exceeds its allowable torque (3 Nm) and comes to a near halt. Using CFC control, the limit is exactly achieved, but never exceeded.

The first control logic tested was joint-angle control. Joint torque limits of 3 Nm were assumed and Fig. 8 presents the results for a single representative arm. During this portion of the climb the motors required more than the allowable torque, which in turn resulted in the robot slowing to a halt.

When CFC control is applied to the same Lemur simulation, the climb is successful. The maximum tracking error of the robot chassis is 2 mm and the highest torques required do not exceed the 3 Nm maximum. Fig. 8 shows the joint torques used for this trajectory.

4.2 Experimental Results with the Free-Flyer

As another demonstration of CFC, a chimney climb using a Stanford Free-Flying robot was performed. This robot floats on an air cushion above a flat granite table. It has two two-link arms. To simulate gravity a weight is attached to the robot that hangs over the edge of the table via a pulley. The computer onboard the free-flying robots has a 200 MHz processor and runs a control loop at 120 Hz. Onboard batteries enable tetherless operation of the vehicle.

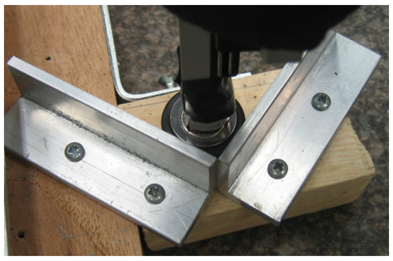

Fig. 9. Photograph of robot endpoint in contact with hold designed to simulate friction.

The endpoint of each robotic arm is a rubber wheel on a bearing. These endpoints make contact with angled holds, as shown in Fig. 9. This type of contact simulates a frictional hold where the normals to the flat surfaces define the edges of a friction cone. This type of hold is convenient for experiments in that it provides a clear visual display. Any motion of the wheel is equivalent to a violation of a friction cone constraint.

Chimney climbing is a difficult manuever. It is characterized by having vertical walls, and therefore horizontal hold normals. In this situation the climber must continually exert pressure normal to the hold in order to not slip out of the friction cones while exerting vertical pressure.

Using only joint-angle control, this chimney climb could not be completed successfully. The friction cone constraints were violated and the hands slipped.

Using CFC control the robot climbed successfully. The results are shown in Fig. 10. The robot was instructed to track a sinusoidal trajectory in the y direction having a 9 cm amplitude. The holds used in the experiment had a half-angle of 45 degrees. Visual observation showed that the hands did not slip and estimation of tip force angle from experimental data confirmed that forces were not applied that violated friction constraints.

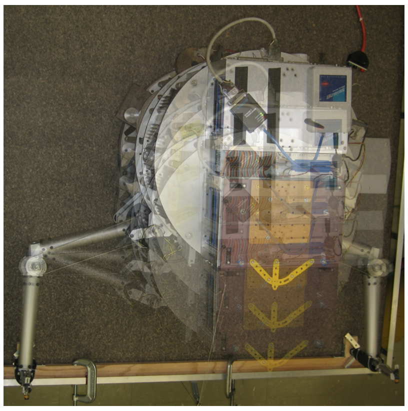

Fig. 10. The free-flying robot ascends using vertical holds.

One of the advantages of CFC control is its robustness to many types of uncertainties. In particular, a real climb will have imperfect knowledge of the holds. Unknown friction properties can be compensated for by assuming appropriate margins when calculating the forces to apply. Some error in translational location of a hold can also be compensated. To demonstrate this, climbing holds were placed in the same chimney configuration as described above, but the robot's right hand hold was displaced 2.5 cm in the negative x direction. To simulate unknown uncertainties, the controller was not changed to reflect the new configuration. The same trajectory was commanded (9 cm amplitude in the y direction) as performed in previous experiments. Visual observation and $\boldsymbol{f}$ calculation for this experiment again showed that the hands did not slip at any point during the trajectory.

5. CONCLUSION

This paper describes a controller for a free-climbing robot that tracks a desired trajectory while maintaining equilibrium and not violating motor constraints. This is done by feeding back the position of the robot, generating a desired force to drive this position to a desired location. A linear program is then solved in real-time to achieve this force without causing hands to slip out of their friction cones. The effectiveness of this controller is shown in simulation and experiment.

ACKNOWLEDGMENTS

This work was supported by NSF grant 0412884. Special thanks to the Mobility and Manipulation Group of JPL for their help and support with the Lemur robot.

REFERENCES

Bevly, D., S. Dubowsky and C. Mavroidis (2000). A simplified cartesian computed torque controller and its application to an experimental climbing robot. *Jrnl of Dynamic Systems, Measurement, and Control* **122**(1), 27–32.

Bretl, T. (2006). Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem. *Intl. Jrnl of Robotics Research.*

Bretl, T., J.C. Latombe and S. Rock (2003*a*). Toward autonomous free-climbing robots. In: *Int. Symp. of Robotics Research.*

Bretl, T., T. Miller, S. Rock and J.-C. Latombe (2003*b*). Motion planning for a three-limbed climbing robot in vertical natural terrain. In: *Intl. Conf. on Intelligent Robotics and Automation.*

Cheng, F.-T. and D. E. Orin (1989). Efficient algorithm for optimal forve distribution in multiple-chain robotic systems: The compact-dual lp method. In: *IEEE Intl. Conf. on Robotics and Automation.* Scottsdale, AZ. pp. 943–950.

Fujimoto, Y. and A. Kawamura (1998). Simulation of an autonomous biped walking robot including environmental force interaction. *IEEE Robotics and Automation Magazine* pp. 33–42.

McGhee, R.B. and D.E. Orin (1976). A mathematical programming approach to control of joint positions and torques in legged locomotion systems. In: *ROMANSY-76 Symposium.*

Nahon, M. and J. Angeles (1992). Real-time force optimization in parallel kinematic chains under inequality constraints. *IEEE Transactions on Robotics and Automation* **8**(4), 439–451.

Schlegl, T., M. Buss, T. Omata and G. Schmidt (2001). Fast dextrous regrasping with optimal contact forces and contact sensor-based impedance control. In: *IEEE Intl. Conf. on Robotics and Automation.* pp. 103–108.

Shapiro, A., E. Rimon and S. Shoval (2005). A foothold selection algorithm for spider robot locomotion in planar tunnel environments. *Intl. Jrnl. of Robotics Research* **24**(10), 823–844.

Sunada, C., D. Argaez, S. Dubowsky and C. Mavroidis (1994). A coordinated jacobian transpose control for mobile multilimbed robotic systems. In: *IEEE Intl. Conf. on Robotics and Automation.* p. 19101915.

Waldron, K. (1986). Force and motion management in legged locomotion. *IEEE Jrnl of Robotics and Automation* **RA-2**(4), 214–220.