# Task Space Control with Prioritization for Balance and Locomotion

Michael Mistry[1], Jun Nakanishi,[2,3] and Stefan Schaal[1,2]
[1]*Computer Science & Neuroscience, University of Southern California, Los Angeles, CA 90089, USA*
[2]*ATR Computational Neuroscience Laboratories, Kyoto 619-0228, Japan*
[3]*ICORP, Japan Science and Technology Agency, Saitama 332-0012, Japan*
*mmistry@usc.edu, jun@atr.jp, sschaal@usc.edu*

*Abstract*— **This paper addresses locomotion with active balancing, via task space control with prioritization. The center of gravity (COG) and foot of the swing leg are treated as task space control points. Floating base inverse kinematics with constraints is employed, thereby allowing for a mobile platform suitable for locomotion. Different techniques of task prioritization are discussed and we clarify differences and similarities of previous suggested work. Varying levels of prioritization for control are examined with emphasis on singularity robustness and the negative effects of constraint switching. A novel controller for task space control of balance and locomotion is developed which attempts to address singularity robustness, while minimizing discontinuities created by constraint switching. Controllers are evaluated using a quadruped robot simulator engaging in a locomotion task.**

## I. Introduction

Legged robots have advantages over wheeled platforms in traversing rough terrain where accurate foot placement is needed. However with increased mobility comes decreased stability, often requiring active control to maintain balance by positioning either the robot's center-of-gravity (COG) or zero-moment-point (ZMP) [2],[3]. Robots with multiple legs such as quadrupeds, can utilize static gaits with COG sway to always maintain the COG within the current support polygon. Traditional approaches to quadruped locomotion have relied on such static gaits, often using open loop gait patterns that position the COG in the appropriate place [1]. However to increase robustness, reactive COG control is still desirable, particularly for traversing rough terrain where environmental disturbances and slipping can perturb the open loop COG trajectories. In addition to maintaining balance, locomotion in rough terrain requires accurate foot placement.

This paper addresses task space control for simultaneous balancing and walking. The robot's COG and foot of the swing leg are treated as task space control points. We will discuss using a floating base representation of kinematics, that allows for the robot to freely traverse its environment. Floating base kinematics approaches have been used before for spacecraft control [12] and humanoid robots [11]. Here we will place emphasis on the constraints due to contacts with the ground, including the effects of constraint switching on locomotion. Additionally task-prioritization is discussed as a potential solution for singularity robustness. Three controllers are evaluated on a dynamic simulator of a the Boston Dynamics LittleDog robot.

## II. Floating Body Kinematics

A free-floating body is by definition not confined to any fixed location in space. In this section we will define the necessary notation and the Jacobians required for floating base inverse kinematics. We will also discuss the role of constraints in the control of unactuated degrees of freedom.

### A. Floating Base Jacobians

The complete configuration of a rigid-body robot with a floating base can be represented by the vector:

$$\mathbf{q} = \left[ \begin{array}{cc} \mathbf{q}_R^T & \mathbf{x}_B^T \end{array} \right]^T \tag{1}$$

where $\mathbf{q}_R \in \mathbb{R}^n$ is the joint configuration of the robot with $n$ joints and $\mathbf{x}_B \in \mathbb{R}^6$ is the position and orientation of a coordinate system attached locally to the robot base, and measured with respect to some fixed world coordinate system. If $\mathbf{x} = \mathbf{f}(\mathbf{q})$ is the position and orientation of any frame represented in the fixed world coordinate system, we can construct a free-floating base Jacobian as follows:

$$\mathbf{J} = \left[ \begin{array}{cc} \frac{\partial \mathbf{x}}{\partial \mathbf{q}_R} & \frac{\partial \mathbf{x}}{\partial \mathbf{x}_B} \end{array} \right] \tag{2}$$

which maps configuration velocities to the velocity of that point: $\dot{\mathbf{x}} = \mathbf{J}\dot{\mathbf{q}}$. In a quadruped robot, for example, for accurate foot placement we are interested in the control of the position and possibly orientation of the $i$th foot ($\mathbf{x}_{L_i}$) in world coordinates. If we segment the the robot configuration vector into the 4 separate leg configurations ($\mathbf{q}_{L_{1..4}}$)

$$\mathbf{q}_R = \left[ \begin{array}{cccc} \mathbf{q}_{L_1}^T & \mathbf{q}_{L_2}^T & \mathbf{q}_{L_3}^T & \mathbf{q}_{L_4}^T \end{array} \right]^T, \tag{3}$$

the floating base Jacobian for the linear velocity of the foot of leg 1 will be

$$\mathbf{J}_{L_1} = \left[ \begin{array}{ccccc} \frac{\partial \mathbf{x}_{L_1}}{\partial \mathbf{q}_{L_1}} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \frac{\partial \mathbf{x}_{L_1}}{\partial \mathbf{x}_B} \end{array} \right]. \tag{4}$$

Each component of the Jacobian can be computed geometrically (see [17]). For example, end-effector motion relative to base motion is

$$\frac{\partial \mathbf{x}_{L_i}}{\partial \mathbf{x}_B} = \left[ \begin{array}{cc} \mathbf{I} & \mathbf{R}_B \times (\bar{\mathbf{x}}_{L_i} - \bar{\mathbf{x}}_B) \\ \mathbf{0} & \mathbf{I} \end{array} \right] \tag{5}$$

where $\bar{\mathbf{x}} \in \mathbb{R}^3$ are the three positional components of $\mathbf{x}$ and $\mathbf{R}_B \times$ is a column-wise cross product of $\mathbf{R}_B$, the rotation matrix representing base orientation relative to the fixed world frame. Other important free-floating base Jacobians to consider are the Jacobian describing motion of the base itself:

$$\mathbf{J}_B = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{6}$$

as well as the center of gravity [16] of the robot:

$$\mathbf{J}_{\text{COG}} = \begin{bmatrix} \frac{\partial \bar{\mathbf{x}}_{\text{COG}}}{\partial \mathbf{q}_R} & \mathbf{I} & \mathbf{R}_B \times (\bar{\mathbf{x}}_{\text{COG}} - \bar{\mathbf{x}}_B) \\ \mathbf{0} \quad \mathbf{0} & \mathbf{0} \quad \mathbf{0} & \mathbf{0} \quad \mathbf{I} \end{bmatrix} \tag{7}$$

where $\bar{\mathbf{x}}_{\text{COG}}$ is the position of the robot's center of gravity, and

$$\frac{\partial \bar{\mathbf{x}}_{\text{COG}}}{\partial \mathbf{q}_R} = \frac{\sum_i m_i \mathbf{J}_{i,\text{COG}}}{\sum_i m_i} \tag{8}$$

is the mass weighted average of the Jacobians of COGs of individual links. Note that orientation of a COG is not a realistic concept, so for the formulation of (7) the orientation of the COG is defined to be the orientation of the robot's base.

## B. Inverse Kinematics with Floating Base Jacobians

In order to control an end effector (the COG for example) with floating base inverse kinematics, one could naively take the pseudo-inverse of the floating base Jacobian, and compute desired configuration velocities:

$$\dot{\mathbf{q}} = \mathbf{J}^{\#}\dot{\mathbf{x}} \tag{9}$$

where the pseudo-inverse is defined as $A^{\#} = A^T \left(AA^T\right)^{-1}$ when $\mathbf{J}$ has more columns than rows and has full row rank. When $\mathbf{J}$ is not full row rank, the pseudo-inverse is computable via a singular value decomposition (see [19]). However, using (9) to compute desired joint trajectories also creates desired trajectories for the base position and orientation, which may not be realizable due to underactuation. This problem can be overcome, provided our system is constrained sufficiently such that the unactuated degrees of freedom cannot move independently from the actuated joints (there must exist more constraints than unactuated degrees of freedom). Calling $\mathbf{J}_C$ the Jacobian of constraints, we can control an arbitrary end effector by:

$$\dot{\mathbf{q}}_R = \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{J}_C \\ \mathbf{J} \end{bmatrix}^{\#} \begin{bmatrix} \mathbf{0} \\ \dot{\mathbf{x}} \end{bmatrix} \tag{10}$$

where the first matrix is used to select $\dot{\mathbf{q}}_R$ from $\dot{\mathbf{q}}$. This equation will provide the correct velocities for the actuated joints that result in the precise desired end effector velocity, provided that the desired end effector velocity, $\dot{\mathbf{x}}$, is compatible with the constraints and the augmented matrix $\begin{bmatrix} \mathbf{J}_C^T & \mathbf{J}^T \end{bmatrix}^T$ is not singular. It is important to note that when constraints change (for example when feet make and break contact with the ground during locomotion), $\mathbf{J}_C$ will abruptly change (possibly in size as well if new constraints are added or removed), creating discontinuities in $\dot{\mathbf{q}}_R$. These discontinuities can potentially destabilize the robot and are therefore undesirable.

## III. Task Prioritization

Robots can utilize redundancy, if available, to achieve multiple tasks simultaneously. A robot may have a set of $n$ tasks, each represented by the vector $\mathbf{x}_i$, $i = 1...n$. It may be possible to compute a configuration velocity, that will achieve all $n$ tasks simultaneously by utilizing the pseudo-inverse of an augmented Jacobian matrix, created by stacking the $n$ task Jacobians together:

$$\dot{\mathbf{q}} = \begin{bmatrix} \mathbf{J}_1 \\ \mathbf{J}_2 \\ \vdots \\ \mathbf{J}_n \end{bmatrix}^{\#} \begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \\ \vdots \\ \dot{\mathbf{x}}_n \end{bmatrix}. \tag{11}$$

For convenience we will write augmented Jacobians and velocity vectors, stacked with $i$ components, as $\hat{\mathbf{J}}_i$ and $\hat{\dot{\mathbf{x}}}_i$ respectively, and can therefore rewrite (11) as

$$\dot{\mathbf{q}} = \hat{\mathbf{J}}_n^{\#}\hat{\dot{\mathbf{x}}}_n \tag{12}$$

However, occasions can arise when two or more tasks will be in conflict, i.e. one task cannot be completed without violating another task. In this case, the augmented Jacobian of (12) will drop rank and become singular. The pseudo-inverse can create unreasonably high joint velocities when approaching singularities, and discontinuities when rank is changed [18]. Note that we can potentially deal with singularities by using a damped pseudo-inverse, as discussed in [7]:

$$\mathbf{J}^{\#\lambda} = \mathbf{J}^T \left(\mathbf{J}\mathbf{J}^T + \lambda^2\mathbf{I}\right)^{-1}$$

at the cost of inaccurate control of task variables. Another way to prevent singularities, is to employ a task-priority framework, whereby lower priority tasks are guaranteed not to interfere with the achievement of higher priorities. In addition, if it becomes necessary to sacrifice a lower priority task for a higher one, then the lower priority task should be executed with a minimum amount of error.

Task-priority control of redundant manipulators have been studied in several papers [4], [5], [6], [7], [8]. In one of the originals [4], task-priority based control of two tasks is defined the following way:

$$\dot{\mathbf{q}} = \mathbf{J}_1^{\#}\dot{\mathbf{x}}_1 + (\mathbf{J}_2\mathbf{N}_1)^{\#}\left(\dot{\mathbf{x}}_2 - \mathbf{J}_2\mathbf{J}_1^{\#}\dot{\mathbf{x}}_1\right). \tag{13}$$

where $\mathbf{N}_i = \mathbf{I} - \mathbf{J}_i^{\#}\mathbf{J}_i$ is the null-space projection matrix of $\mathbf{J}_i$, and the task defined by $\dot{\mathbf{x}}_1$ is claimed to have higher priority than the task defined by $\dot{\mathbf{x}}_2$. The framework is generalized, recursively, in [5] for the prioritization of $n$ tasks:

$$\begin{aligned} \dot{\mathbf{q}}_1 &= \mathbf{J}_1^{\#}\dot{\mathbf{x}}_1 \\ \dot{\mathbf{q}}_i &= \dot{\mathbf{q}}_{i-1} + \left(\mathbf{J}_i\hat{\mathbf{N}}_{i-1}\right)^{\#}(\dot{\mathbf{x}}_i - \mathbf{J}_i\dot{\mathbf{q}}_{i-1}) \\ \dot{\mathbf{q}} &= \dot{\mathbf{q}}_n \end{aligned} \tag{14}$$

where $\hat{\mathbf{N}}_{i-1}$ is defined to be the null space projection of the augmented Jacobian $\hat{\mathbf{J}}_{i-1}$ :

$$\hat{\mathbf{N}}_{i-1} = \mathbf{I} - \begin{bmatrix} \mathbf{J}_1 \\ \mathbf{J}_2 \\ \vdots \\ \mathbf{J}_{i-1} \end{bmatrix}^{\#} \begin{bmatrix} \mathbf{J}_1 \\ \mathbf{J}_2 \\ \vdots \\ \mathbf{J}_{i-1} \end{bmatrix},$$

and task $i-1$ is claimed to have higher priority than task $i$. Unfortunately this formulation also has difficulty handling task conflicts, since so called algorithmic singularities are created when one task will violate another (and at least one $\mathbf{J}_i\hat{\mathbf{N}}_{i-1}$ term will become singular). By using a manipulability measure, [8] is able to show that algorithmic singularities in the two task case of (13) occur if and only if the augmented Jacobian of both tasks, i.e. (12), is singular. Thus the formulations of (12) and (14) have two striking similarities. They both achieve accurate tracking of all $n$ tasks whenever possible, and when two or more tasks conflict, they both run into singularity difficulties. It will be shown in the appendix of this paper that when $\hat{\mathbf{J}}_n$ has full row rank, for an arbitrary $n > 1$ tasks, the two formulations are precisely equivalent, i.e.:

$$\dot{\mathbf{q}}_n = \dot{\mathbf{q}}_{n-1} + \left(\mathbf{J}_n\hat{\mathbf{N}}_{n-1}\right)^{\#}(\dot{\mathbf{x}}_n - \mathbf{J}_n\dot{\mathbf{q}}_{n-1}) = \hat{\mathbf{J}}_n^{\#}\hat{\dot{\mathbf{x}}}_n. \quad (15)$$

Thus as the matrix $\hat{\mathbf{J}}_n$ approaches a singularity, so will at least one $\mathbf{J}_i\hat{\mathbf{N}}_{i-1}$ term of (14). Consequently, the only benefit of (14) over (12) occurs precisely at singularities, where (14) is still able to guarantee task achievement of higher priority tasks. However, practically, operating at singularities is unrealistic since approaching them can generate unreasonably high joint velocities, as well as large discontinuities at the rank transition. Therefore in its current form, (14) is no more useful as a task-priority formulation than (12), and we must turn to singularity robust methods. [6] mentions an alternative form of task prioritization:

$$\dot{\mathbf{q}} = \mathbf{J}_1^{\#}\dot{\mathbf{x}}_1 + \mathbf{N}_1\mathbf{J}_2^{\#}\dot{\mathbf{x}}_2, \quad (16)$$

which [7] claims has poor tracking of lower priorities (even if redundancy is available for the achievement of all tasks). However, when two tasks of different priorities conflict there are no problematic singularities. This technique, coupled with augmented Jacobians of (12), and damped pseudo-inverses, will be employed in the development of our control algorithms.

## IV. Controller Formulation

Our primary concern in balance and static locomotion is the control of the COG position. The position of the COG must be accurately controlled so that it does not drift outside the support polygon. However, in order to do so with floating base kinematics, we must have sufficiently many constraints (to allow for control of unactuated DOFs) and our controllers must always maintain those constraints. If at any time, a constraint is allowed to be violated, the COG will no longer be controllable. Therefore, we must always place the COG and constraints in the same and highest level of priority, i.e. together in an augmented Jacobian. It may be possible that the COG trajectory can conflict with constraints: the most common example would be when the COG is commanded to go as high as possible, which is only achievable if the feet lift from the floor (thus breaking constraints). Therefore, we will only place the x and y (floor plane) components of the COG Jacobian in this augmented Jacobian. The z position (height) of the COG will be controlled at a lower level of priority. It may be still possible that an x or y component of the COG will violate the constraints, however we will assume that these postures only occur when the COG is outside the support triangle, when stability is lost anyway. Our secondary concern is for the position of the swing leg, for as accurate foot placement as possible. We will investigate placing control of this task variable both in primary and secondary levels. Finally, any remaining degrees of freedom (COG height and orientation of the body), must never be able to interfere with constraints, COG, or swing leg, and will always be placed at the lowest level of priority.

### A. Control Method 1

In our first controller implementation, we place the swing leg in the same level of priority as the constraints and COG. The main advantages are that the structure of the primary Jacobian does not change with the constraints, and thus desired joint velocities remain continuous (provided desired end-effector velocities remain continuous). The major disadvantage with this approach is that the swing leg can potentially conflict with the COG (for example, when reaching for a target outside of the leg's workspace), and thus swing leg trajectories must be planned in advance to not interfere with COG trajectory. We will define:

$$\hat{\mathbf{J}}_A = \begin{bmatrix} \hat{\mathbf{J}}_{\text{stance}} \\ \mathbf{J}_{\text{swing}} \\ \mathbf{J}_{\text{COG,XY}} \end{bmatrix},$$

$$\hat{\mathbf{J}}_P = \begin{bmatrix} \mathbf{J}_{\text{RPY}} \\ \mathbf{J}_{\text{COG,Z}} \end{bmatrix}$$

where $\hat{\mathbf{J}}_{\text{stance}}$ is the augmented Jacobian, containing a stack of all legs in contact with the ground, $\mathbf{J}_{\text{swing}}$ is the Jacobian of the swing leg, $\mathbf{J}_{\text{RPY}}$ dictates base orientation (in roll, pitch, yaw angles), $\mathbf{J}_{\text{COG,XY}}$ is the Jacobian for the projection of the COG on the XY plane, and $\mathbf{J}_{\text{COG,Z}}$ for the height of the COG. Similarly we will define the vector of desired end effector velocities:

$$\hat{\dot{\mathbf{x}}}_A = \begin{bmatrix} \mathbf{0} \\ \dot{\mathbf{x}}_{\text{swing}} \\ \dot{\mathbf{x}}_{\text{COG,XY}} \end{bmatrix},$$

$$\hat{\dot{\mathbf{x}}}_P = \begin{bmatrix} \dot{\mathbf{x}}_{\text{RPY}} \\ \dot{\mathbf{x}}_{\text{COG,Z}} \end{bmatrix}$$

Finally we compute desired joint velocities as

$$\dot{\mathbf{q}}_1 = \hat{\mathbf{J}}_A^{\#}\hat{\dot{\mathbf{x}}}_A + \hat{\mathbf{N}}_A\hat{\mathbf{J}}_P^{\#}\hat{\dot{\mathbf{x}}}_P, \quad (17)$$

and $\hat{\mathbf{N}}_A = \mathbf{I} - \hat{\mathbf{J}}_A^{\#}\hat{\mathbf{J}}_A$ is the null space projection matrix of $\hat{\mathbf{J}}_A$.

## B. Control Method 2

Because Method 1 leaves us in danger of commanding the swing leg to a posture that conflicts with the COG (creating a kinematic singularity), we would like to move the swing leg controller into the null space of the COG controller and constraints. Defining:

$$\hat{\mathbf{J}}_B = \begin{bmatrix} \hat{\mathbf{J}}_{\text{stance}} \\ \mathbf{J}_{\text{COG,XY}} \end{bmatrix},$$

and

$$\hat{\dot{\mathbf{x}}}_B = \begin{bmatrix} \mathbf{0} \\ \dot{\mathbf{x}}_{\text{COG,XY}} \end{bmatrix},$$

the 2nd controller will be:

$$\dot{\mathbf{q}}_2 = \hat{\mathbf{J}}_B^{\#}\hat{\dot{\mathbf{x}}}_B + \hat{\mathbf{N}}_B\mathbf{J}_{\text{swing}}^{\#}\dot{\mathbf{x}}_{\text{swing}} + \hat{\mathbf{N}}_A\hat{\mathbf{J}}_P^{\#}\hat{\dot{\mathbf{x}}}_P. \tag{18}$$

Note that we project $\hat{\mathbf{J}}_P^{\#}\hat{\dot{\mathbf{x}}}_P$ into the null-space of $\hat{\mathbf{J}}_A$ since this is the intersection of the $\hat{\mathbf{J}}_B$ and $\mathbf{J}_{\text{swing}}$ null-spaces. Thus the term $\hat{\mathbf{N}}_A\hat{\mathbf{J}}_P^{\#}\hat{\dot{\mathbf{x}}}_P$ will not interfere with the constraints, COG or swing leg. Also note that computation of $\hat{\mathbf{N}}_A$ requires a pseudo-inverse of $\hat{\mathbf{J}}_A$, and thus eliminates the singularity robustness we were seeking by moving the swing leg out of the top priority. To deal with this issue we will compute $\hat{\mathbf{N}}_A$ with a damped pseudo-inverse:

$$\hat{\mathbf{N}}_A^{\lambda} = \mathbf{I} - \hat{\mathbf{J}}_A^{\#\lambda}\hat{\mathbf{J}}_A$$

Because $\hat{\mathbf{N}}_A^{\lambda}$ does not perfectly project into the null-space of $\hat{\mathbf{J}}_A$, we will need to premultiply with $\hat{\mathbf{N}}_B$ to be certain the constraints and COG are not violated. Thus the final version of the method 2 controller is:

$$\dot{\mathbf{q}}_2 = \hat{\mathbf{J}}_B^{\#}\hat{\dot{\mathbf{x}}}_B + \hat{\mathbf{N}}_B\mathbf{J}_{\text{swing}}^{\#}\dot{\mathbf{x}}_{\text{swing}} + \hat{\mathbf{N}}_B\hat{\mathbf{N}}_A^{\lambda}\hat{\mathbf{J}}_P^{\#}\hat{\dot{\mathbf{x}}}_P. \tag{19}$$

## C. Control Method 3

Jacobian switching problems arise because at the moment of transfer between a 3 and 4 leg stance: $\dot{\mathbf{q}}_1 \neq \dot{\mathbf{q}}_2$. One question we can ask is if it is possible to add a term to $\dot{\mathbf{q}}_2$ such that at the moment of constraint switching:

$$\dot{\mathbf{q}}_1 = \dot{\mathbf{q}}_2 + \hat{\mathbf{N}}_B\xi, \tag{20}$$

where $\xi$ is an arbitrary vector. The additional term will not violate the constraints of COG trajectory (although it may interfere with the swing leg). Substituting equations (17) and (18) into (20), and solving for $\hat{\mathbf{N}}_B\xi$ gives us:

$$\hat{\mathbf{N}}_B\xi = \hat{\mathbf{J}}_A^{\#}\hat{\dot{\mathbf{x}}}_A - \hat{\mathbf{J}}_B^{\#}\hat{\dot{\mathbf{x}}}_B - \hat{\mathbf{N}}_B\mathbf{J}_{\text{swing}}^{\#}\dot{\mathbf{x}}_{\text{swing}}. \tag{21}$$

We can multiply both sides of (21) by $\hat{\mathbf{N}}_B$ and because $\hat{\mathbf{N}}_B\hat{\mathbf{J}}_B^{\#} = \mathbf{0}$ and $\hat{\mathbf{N}}_B$ is idempotent, we have:

$$\hat{\mathbf{N}}_B\xi = \hat{\mathbf{N}}_B\hat{\mathbf{J}}_A^{\#}\hat{\dot{\mathbf{x}}}_A - \hat{\mathbf{N}}_B\mathbf{J}_{\text{swing}}^{\#}\dot{\mathbf{x}}_{\text{swing}}. \tag{22}$$

Now adding this new term to (18) gives us

$$\dot{\mathbf{q}}_3 = \hat{\mathbf{J}}_B^{\#}\hat{\dot{\mathbf{x}}}_B + \hat{\mathbf{N}}_B\hat{\mathbf{J}}_A^{\#}\hat{\dot{\mathbf{x}}}_A + \hat{\mathbf{N}}_A\hat{\mathbf{J}}_P^{\#}\hat{\dot{\mathbf{x}}}_P.$$

Again, because we wish to be robust to singularities of $\hat{\mathbf{J}}_A$ we will use the damped pseudo-inverse:

$$\dot{\mathbf{q}}_3 = \hat{\mathbf{J}}_B^{\#}\hat{\dot{\mathbf{x}}}_B + \hat{\mathbf{N}}_B\hat{\mathbf{J}}_A^{\#\lambda}\hat{\dot{\mathbf{x}}}_A + \hat{\mathbf{N}}_B\hat{\mathbf{N}}_A^{\lambda}\hat{\mathbf{J}}_P^{\#}\hat{\dot{\mathbf{x}}}_P. \tag{23}$$

If $\lambda = 0$, and because the relation of equation (20) holds, and $\dot{\mathbf{q}}_3 = \dot{\mathbf{q}}_1$. Thus as long as $\hat{\mathbf{J}}_A$ is not near a singularity, we can operate with $\lambda = 0$ and maintain continuous joint velocities and accurate swing leg tracking. As $\hat{\mathbf{J}}_A$ approaches a singularity (manipulability becomes low), we can increase $\lambda$ to maintain robustness.

## V. Evaluations

### A. Platform

We evaluate the three controllers on a dynamic simulator of the Boston Dynamics LittleDog quadruped robot. The LittleDog robot is a 12 degree of freedom robot (3 per leg) with point feet contacts. We attempt a locomotion task on flat ground where a smooth, twice-differentiable, COG trajectory for locomotion, is generated automatically based on foot movement [10]. Foot motion is planned to reach predefined targets on the floor (for moving straight ahead). Trajectories for the feet to reach targets are generated via 5th order splines (which provide minimum jerk trajectories). Desired task velocities (for input into the controllers) are generated from the trajectories $\dot{\mathbf{x}}_d$, $\mathbf{x}_d$, as follows:

$$\dot{\mathbf{x}} = K_V\dot{\mathbf{x}}_d + K_P(\mathbf{x}_d - \mathbf{x}) \tag{24}$$

The controllers compute desired joint velocities, which are then integrated to produce desired joint positions. The desired joint position and velocity set points are sent to low level joint PD controllers on the robot at a rate of 500 Hz.

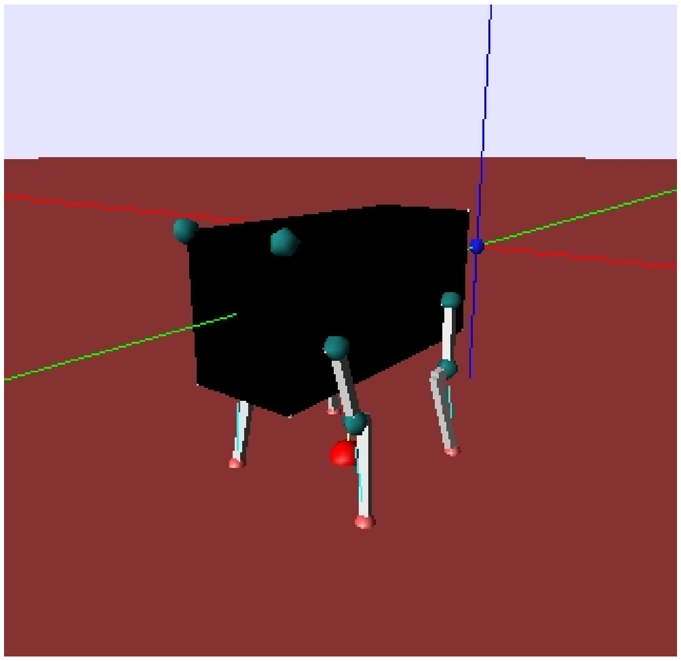

Fig. 1. LittleDog Robot Simulator

### B. Locomotion Performance

The COG trajectory is automatically placed within the support triangle of the stance legs, and since the COG trajectory is parameterized by foot position, velocity, and acceleration, any foot movement, including slipping, will readjust the trajectory (see [10]). Oscillations in the desired trajectories are caused by ground contacts, which are modeled in the simulator as spring mass dampers. Figure 2 shows the COG tracking performance for each of the three controllers during a locomotion task on flat terrain. The robot moves at a speed of one gait cycle per 6 seconds. Controller 3 uses a constant $\lambda = 0.05$ for the entire motion (although a more intelligent approach may be to vary $\lambda$). Tracking for all controllers is good enough for maintaining stability during a locomotion task.

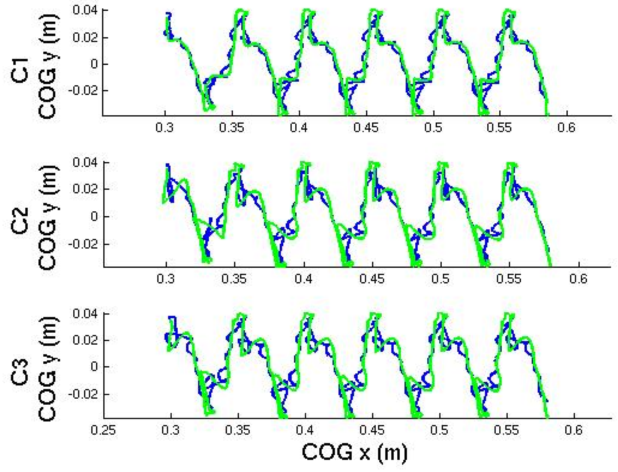

Fig. 2. x and y coordinates (overhead view) of the COG trajectory (in blue) for the 3 controllers plotted with the desired COG trajectory in green, during walking on flat ground. the robot moves from left to right.

The swing leg is guided to its target with a trajectory created by a 5th order spline. Figure 3 shows the swing leg tracking performance in the z direction of the left front leg for each of the three controllers during locomotion. In green is the trajectory generated by the splines for the leg to lift off and touch down, and in blue is the foot's actual trajectory.

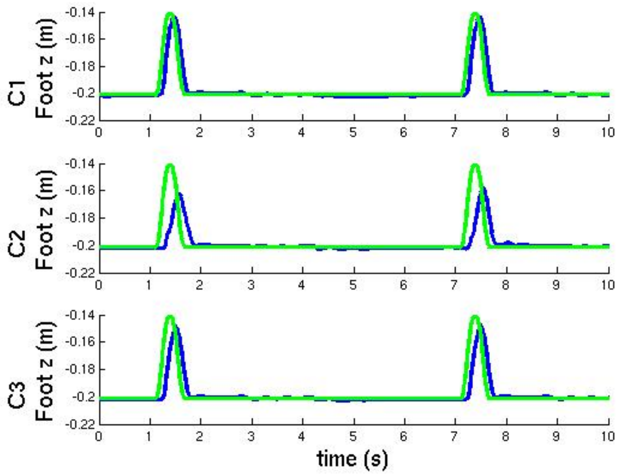

Fig. 3. left front leg's tracking performance in the z direction over time during locomotion for each of the three controllers. Controller 2 is not able to track that well.

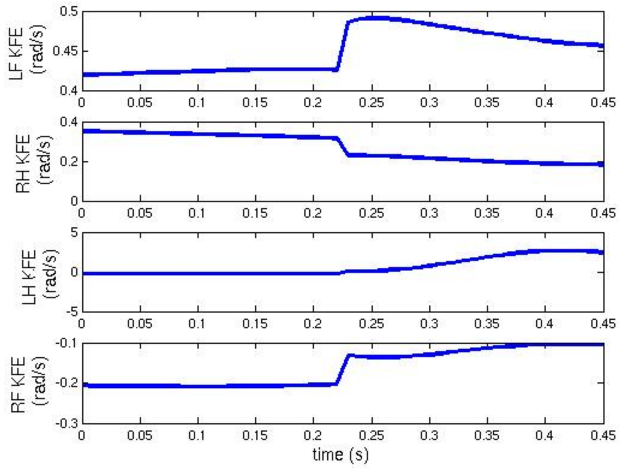

Fig. 4. desired knee velocities computed by controller 2. There are clearly discontinuities in the trajectories, caused by a sudden constraint switch.

## C. Constraint Switching

Controllers 2 and 3 suffer from constraint switching discontinuities. Figure 4 shows the effect of constraint switching, on controller 2 during the transition from 4 foot support phase to 3 foot support. Shown in the graph are plots of the computed desired knee velocities for each of the for knees, and the discontinuity created by the switching Jacobian.

Figure 5 shows the same time period for Controller 3 (with $\lambda = 0.05$). Although, this controller is also going though a constraint switch, the effect is nearly negligible.

## D. Swing Leg Tracking

Since controller 1 places the swing leg in the highest level of priority, we know that swing leg control will not be distorted (provided $\hat{\mathbf{J}}_A$ remains non-singular). In order to quantify the amount of degradation caused by placing the swing leg controller in the null space of the balance controller, we can compute the error in $\dot{\mathbf{x}}_{\text{swing}}$ reconstruction. For controller 2, it is:

$$\begin{aligned}\mathbf{e}_2 &= \dot{\mathbf{x}}_{\text{swing}} - \mathbf{J}_{\text{swing}}\hat{\mathbf{N}}_B\mathbf{J}^{\#}_{\text{swing}}\dot{\mathbf{x}}_{\text{swing}} \\ &= \left(\mathbf{I} - \mathbf{J}_{\text{swing}}\hat{\mathbf{N}}_B\mathbf{J}^{\#}_{\text{swing}}\right)\dot{\mathbf{x}}_{\text{swing}} \\ &= \mathbf{E}_2\dot{\mathbf{x}}_{\text{swing}}\end{aligned}$$

where $\mathbf{E}_2$ is independent of the desired swing foot velocity. For method 3:

$$\mathbf{E}_3 = \mathbf{I} - \mathbf{J}_{\text{swing}}\hat{\mathbf{N}}_B\hat{\mathbf{J}}_A^{\#\lambda}\mathbf{S}_{\text{swing}},$$

where $\mathbf{S}_{\text{swing}}$ is used to convert $\dot{\mathbf{x}}_{\text{swing}}$ into a higher dimensional vector.

$$\mathbf{S}_{\text{swing}} = \begin{bmatrix}\mathbf{0}\\ \mathbf{I}\\ \mathbf{0}\end{bmatrix} \tag{25}$$

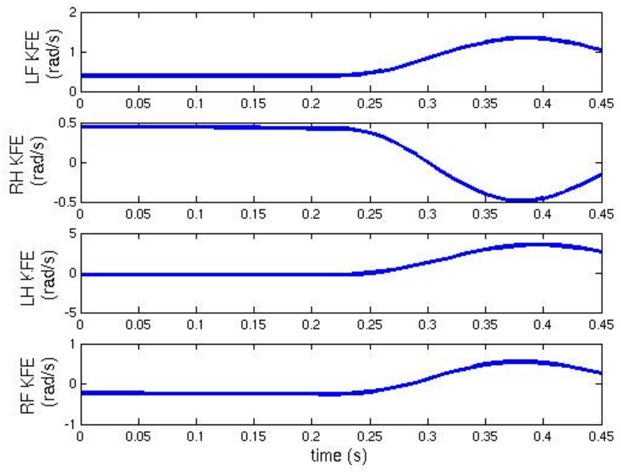

Fig. 5. desired knee velocities computed by controller 3. There is a constraint switch but this controller maintains a rather smooth trajectory though it.

The matrix $\mathbf{E}$ should give some indication of the controller's performance of swing leg tracking, independently of the particular trajectory ($\mathbf{E} = \mathbf{0}$ means perfect swing leg tracking). Figure 6 shows plots of the Frobenius norm of $\mathbf{E}_2$ (in blue), $\mathbf{E}_3$ (in green) with $\lambda = 0.05$ during the swing phase of the left front leg. Note that the norms of the first controller (or third controller with $\lambda = 0$) are zero. The norm of controller 3 is consistently less than that of 2, indicating that controller 3 should be better at swing leg tracking. Reducing $\lambda$ even further will likewise reduce the norm, and should improve swing leg tracking even further.

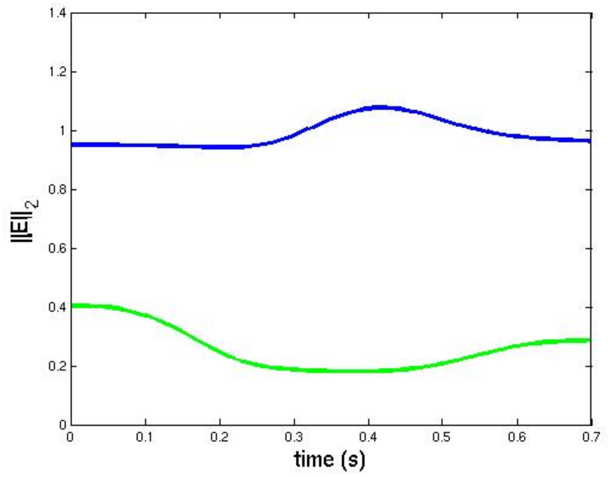

Fig. 6. The Frobenius norms of $\mathbf{E}_2$ (blue) and $\mathbf{E}_3$ (green) plotted during the swing phase of the left front leg. Smaller norm means less distortion of the desired swing leg trajectory.

## VI. Discussion

Controller 1 had the best performance for locomotion. However, in order to use this controller, foot placement must be carefully planned to avoid conflicts with the COG. Using a damped pseudo-inverse in this controller, is not an option since we will lose the ability to maintain the constraints (and thus lose COG controllability). Controller 2 no longer has these singularity issues, but as the results show, it performs poorly in swing leg tracking and suffers from discontinuities during constraint switches. Controller 3 seems to be a nice tradeoff between these two approaches, however it requires the additional tuning of the $\lambda$ parameter to achieve the best performance.

## VII. Conclusion

We have demonstrated the feasibility using floating base inverse kinematics, with task-prioritization to address the issue of simultaneous balance and walking of a quadruped robot. A controller developed here seems to be a promising trade-off between singularity problems and constraint switching. Future work will focus on the evaluation of these controllers on more complex terrain, torque based control via inverse dynamics, and other robot platforms including humanoids.

## VIII. Appendix

### A. *Proof of (15)*

Since (14) is a recursive formulation, we will use induction to prove (15). The induction hypothesis will be:

$$\dot{\mathbf{q}}_i = \begin{bmatrix} \mathbf{J}_1 \\ \mathbf{J}_2 \\ \vdots \\ \mathbf{J}_i \end{bmatrix}^{\#} \begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \\ \vdots \\ \dot{\mathbf{x}}_i \end{bmatrix} = \hat{\mathbf{J}}_i^{\#} \hat{\dot{\mathbf{x}}}_i, \tag{26}$$

and the base case ($i = 1$) is trivially verified ($\dot{\mathbf{q}}_1 = \mathbf{J}_1^{\#} \dot{\mathbf{x}}_1$). Therefore we only need to prove the $(i+1)$th step:

$$\dot{\mathbf{q}}_i + \left(\mathbf{J}_{i+1} \hat{\mathbf{N}}_i\right)^{\#} (\dot{\mathbf{x}}_{i+1} - \mathbf{J}_{i+1} \dot{\mathbf{q}}_i) = \begin{bmatrix} \mathbf{J}_1 \\ \mathbf{J}_2 \\ \vdots \\ \mathbf{J}_i \\ \mathbf{J}_{i+1} \end{bmatrix}^{\#} \begin{bmatrix} \dot{\mathbf{x}}_1 \\ \dot{\mathbf{x}}_2 \\ \vdots \\ \dot{\mathbf{x}}_i \\ \dot{\mathbf{x}}_{i+1} \end{bmatrix} \tag{27}$$

The right hand side of (27) can be rewritten by using the augmented Jacobian notation:

$$\dot{\mathbf{q}} = \begin{bmatrix} \hat{\mathbf{J}}_i \\ \mathbf{J}_{i+1} \end{bmatrix}^{\#} \begin{bmatrix} \hat{\dot{\mathbf{x}}}_i \\ \dot{\mathbf{x}}_{i+1} \end{bmatrix} \tag{28}$$

Assuming the augmented Jacobian of (28) is full row rank, we can expand its pseudo-inverse to get the equation:

$$\dot{\mathbf{q}} = \begin{bmatrix} \hat{\mathbf{J}}_i^T & \mathbf{J}_{i+1}^T \end{bmatrix} \begin{bmatrix} \hat{\mathbf{J}}_i \hat{\mathbf{J}}_i^T & \hat{\mathbf{J}}_i \mathbf{J}_{i+1}^T \\ \mathbf{J}_{i+1} \hat{\mathbf{J}}_i^T & \mathbf{J}_{i+1} \mathbf{J}_{i+1}^T \end{bmatrix}^{-1} \begin{bmatrix} \hat{\dot{\mathbf{x}}}_i \\ \dot{\mathbf{x}}_{i+1} \end{bmatrix}. \tag{29}$$

The inversion in (29) can be solved using the Strassen blockwise inversion formula:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} W & X \\ Y & Z \end{bmatrix}. \tag{30}$$

where

$$\begin{aligned}
W &= A^{-1} + A^{-1}B\left(D - CA^{-1}B\right)^{-1}CA^{-1} \\
X &= -A^{-1}B\left(D - CA^{-1}B\right)^{-1} \\
Y &= -\left(D - CA^{-1}B\right)^{-1}CA^{-1} \\
Z &= \left(D - CA^{-1}B\right)^{-1}
\end{aligned}$$

By setting $A, B, C$, and $D$ to the corresponding element of the inverted matrix in (29) we can solve for $W, X, Y$, and $Z$. First we expand $Z$:

$$\begin{aligned}
Z &= \left(\mathbf{J}_{i+1}\mathbf{J}_{i+1}^T - \mathbf{J}_{i+1}\hat{\mathbf{J}}_i^T\left(\hat{\mathbf{J}}_i\hat{\mathbf{J}}_i^T\right)^{-1}\hat{\mathbf{J}}_i\mathbf{J}_{i+1}^T\right)^{-1} \\
&= \left(\mathbf{J}_{i+1}\mathbf{J}_{i+1}^T - \mathbf{J}_{i+1}\hat{\mathbf{J}}_i^{\#}\hat{\mathbf{J}}_i\mathbf{J}_{i+1}^T\right)^{-1} \\
&= \left(\mathbf{J}_{i+1}\left(\mathbf{I} - \hat{\mathbf{J}}_i^{\#}\hat{\mathbf{J}}_i\right)\mathbf{J}_{i+1}^T\right)^{-1} \\
&= \left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\right)^{-1}. \qquad (31)
\end{aligned}$$

Next we will expand the diagonal elements of (30):

$$\begin{aligned}
Y &= -\left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\right)^{-1}\mathbf{J}_{i+1}\hat{\mathbf{J}}_i^T\left(\hat{\mathbf{J}}_i\hat{\mathbf{J}}_i^T\right)^{-1} \\
&= -\left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\right)^{-1}\mathbf{J}_{i+1}\hat{\mathbf{J}}_i^{\#}, \qquad (32)
\end{aligned}$$

$$X = -\left(\hat{\mathbf{J}}_i\hat{\mathbf{J}}_i^T\right)^{-1}\hat{\mathbf{J}}_i\mathbf{J}_{i+1}^T\left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\right)^{-1}. \qquad (33)$$

And the final element is expanded:

$$\begin{aligned}
W &= \left(\hat{\mathbf{J}}_i\hat{\mathbf{J}}_i^T\right)^{-1} \\
&\quad + \left(\hat{\mathbf{J}}_i\hat{\mathbf{J}}_i^T\right)^{-1}\hat{\mathbf{J}}_i\mathbf{J}_{i+1}^T\left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\right)^{-1}\mathbf{J}_{i+1}\hat{\mathbf{J}}_i^{\#} \qquad (34)
\end{aligned}$$

Replacing the matrix inversion in (29) with (30) results in:

$$\begin{aligned}
\dot{\mathbf{q}} &= \begin{bmatrix} \hat{\mathbf{J}}_i^T & \mathbf{J}_{i+1}^T \end{bmatrix}\begin{bmatrix} W & X \\ Y & Z \end{bmatrix}\begin{bmatrix} \hat{\dot{\mathbf{x}}}_i \\ \dot{\mathbf{x}}_{i+1} \end{bmatrix} \\
&= \left(\hat{\mathbf{J}}_i^T W + \mathbf{J}_{i+1}^T Y\right)\hat{\dot{\mathbf{x}}}_i + \left(\hat{\mathbf{J}}_i^T X + \mathbf{J}_{i+1}^T Z\right)\dot{\mathbf{x}}_{i+1} \\
&= P\hat{\dot{\mathbf{x}}}_i + Q\dot{\mathbf{x}}_{i+1} \qquad (35)
\end{aligned}$$

First we will expand the right most term of (35) and substitute (31) and (33):

$$\begin{aligned}
Q &= \hat{\mathbf{J}}_i^T X + \mathbf{J}_{i+1}^T Z \\
&= -\hat{\mathbf{J}}_i^T\left(\hat{\mathbf{J}}_i\hat{\mathbf{J}}_i^T\right)^{-1}\hat{\mathbf{J}}_i\mathbf{J}_{i+1}^T\left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\right)^{-1} \qquad (36) \\
&\quad + \mathbf{J}_{i+1}^T\left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\right)^{-1} \\
&= -\hat{\mathbf{J}}^{\#}\hat{\mathbf{J}}_i\mathbf{J}_{i+1}^T\left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\right)^{-1} \\
&\quad + \mathbf{J}_{i+1}^T\left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\right)^{-1} \\
&= \left(\mathbf{I} - \hat{\mathbf{J}}_i^{\#}\hat{\mathbf{J}}_i\right)\mathbf{J}_{i+1}^T\left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\right)^{-1} \\
&= \hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\right)^{-1} \\
&= \hat{\mathbf{N}}_i^T\mathbf{J}_{i+1}^T\left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\hat{\mathbf{N}}_i^T\mathbf{J}_{i+1}^T\right)^{-1} \\
&= \left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\right)^{\#} \qquad (37)
\end{aligned}$$

The last two steps are possible because $\hat{\mathbf{N}}$ is symmetric and idempotent. Next we expand the left hand term of (35) and substitute (32) and (34):

$$\begin{aligned}
P &= \hat{\mathbf{J}}_i^T W + \mathbf{J}_{i+1}^T Y \\
&= \hat{\mathbf{J}}_i^T\left(\hat{\mathbf{J}}_i\hat{\mathbf{J}}_i^T\right)^{-1} \\
&\quad + \hat{\mathbf{J}}_i^T\left(\hat{\mathbf{J}}_i\hat{\mathbf{J}}_i^T\right)^{-1}\hat{\mathbf{J}}_i\mathbf{J}_{i+1}^T\left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\right)^{-1}\mathbf{J}_{i+1}\hat{\mathbf{J}}_i^{\#} \\
&\quad - \mathbf{J}_{i+1}^T\left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\right)^{-1}\mathbf{J}_{i+1}\hat{\mathbf{J}}_i^{\#} \\
&= \hat{\mathbf{J}}_i^{\#} + \hat{\mathbf{J}}_i^{\#}\hat{\mathbf{J}}_i\mathbf{J}_{i+1}^T\left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\right)^{-1}\mathbf{J}_{i+1}\hat{\mathbf{J}}_i^{\#} \\
&\quad - \mathbf{J}_{i+1}^T\left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\right)^{-1}\mathbf{J}_{i+1}\hat{\mathbf{J}}_i^{\#} \\
&= \hat{\mathbf{J}}_i^{\#} - \hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\mathbf{J}_{i+1}^T\right)^{-1}\mathbf{J}_{i+1}\hat{\mathbf{J}}_i^{\#} \\
&= \hat{\mathbf{J}}_i^{\#} - \left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\right)^{\#}\mathbf{J}_{i+1}\hat{\mathbf{J}}_i^{\#} \qquad (38)
\end{aligned}$$

Substituting (37) and (38) into (35) yields:

$$\begin{aligned}
\dot{\mathbf{q}} &= \hat{\mathbf{J}}_i^{\#}\hat{\dot{\mathbf{x}}}_i - \left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\right)^{\#}\mathbf{J}_{i+1}\hat{\mathbf{J}}_i^{\#}\hat{\dot{\mathbf{x}}}_i + \left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\right)^{\#}\dot{\mathbf{x}}_{i+1} \\
&= \hat{\mathbf{J}}_i^{\#}\hat{\dot{\mathbf{x}}}_i + \left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\right)^{\#}\left(\dot{\mathbf{x}}_{i+1} - \mathbf{J}_{i+1}\hat{\mathbf{J}}_i^{\#}\hat{\dot{\mathbf{x}}}_i\right) \qquad (39)
\end{aligned}$$

Finally, we invoke the induction hypothesis and substitute (26) into (39):

$$\dot{\mathbf{q}} = \dot{\mathbf{q}}_i + \left(\mathbf{J}_{i+1}\hat{\mathbf{N}}_i\right)^{\#}\left(\dot{\mathbf{x}}_{i+1} - \mathbf{J}_{i+1}\dot{\mathbf{q}}_i\right)$$

$\square$

## IX. Acknowledgments

This research was supported in part by National Science Foundation grants ECS-0325383, IIS-0312802, IIS-0082995, ECS-0326095, ANI-0224419, the DARPA program on Learning Locomotion, a NASA grant AC98-516, an AFOSR grant on Intelligent Control, the ERATO Kawato Dynamic Brain Project funded by the Japanese Science and

Technology Agency, and the ATR Computational Neuroscience Laboratories. We thank the reviewers for referring us to [18].

## References

[1] F. Hardarson, "Stability analysis and synthesis of statically balanced walking for quadruped robots," in Mechatronics Lab, Department of Machine Design. Stockholm: Royal Institute of Technology (KTH), 2002.

[2] M. Vukobratovic and J. Stepanenko, "On the stability of anthropomorphic systems," Journal of Mathematical Bioscience, vol. 15, pp. 1-37, 1972.

[3] M. Vukobratovic and B. Borovac, "Zero-mement point – Thirty five years of its life," International Journal of Humanoid Robotics, vol. 1, pp. 157-173, 2004.

[4] Y. Nakamura, H. Hanafusa, and T. Yoshikawa. Task-priority based redundancy control of robot manipulators, International Journal of Robotics Research, 6(2):3-15, 1987.

[5] B. Siciliano and J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems, Fifth International Conference on Advance Robotics, Pisa, Italy, 1991, pp. 1211-1216.

[6] S. Chiaverini. "Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators", IEEE Transactions on Robotics and Automation, Vol 13, No. 3:398-410, 1997.

[7] P. Baerlocher, and R. Boulic, "Task-priority formulations for the kinematic control of highly redundant articulated structures", Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 323-329, 1998.

[8] G. Marani, J. Kim, J. Yuh, and W.K. Chung "Algorithmic singularities avoidance in task-priority based controller for redundant manipulators", Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems, Las Vegas, Nevada, 3570-3574, 2003.

[9] J. Nakanishi, R. Cory, M. Mistry, J. Peters, S. Schaal, "Comparative Experiments on Task Space Control with Redundancy Resolution", 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, Canada, 2005, pp. 3901-3908.

[10] D. Pongas, M. Mistry, and S. Schaal, "A Robust Quadruped Walking Gait for Traversing Rough Terrain", *2007 International Conference on Robotics and Automation*, Rome, Italy, 2007.

[11] L. Sentis and O. Khatib, "Control of Free-Floating Humanoid Robots Through Task Prioritization", *in Proceedings of the 2005 International Conference on Robotics and Automation*, Barcelona, Spain, 2005, pp.1730-1735.

[12] S. Dubowsky and E. Papadopoulos. The kinematics, dynamics, and control of free-flying and free-floating space robotic systems. IEEE Transactions on Robotics and Automation, 9(5), October 1993.

[13] O. Khatib, A unified approach to motion and force control of robot manipulators: The operational space formulation, *International Journal of Robotics Research*, 3(1):43-53, 1987.

[14] J. Nakanishi, M. Mistry, and S. Schaal, "Inverse dynamics control with floating base and constraints", *submitted to the 2007 International Conference on Robotics and Automation*, Rome, Italy, 2007.

[15] J. Peters, M. Mistry, F. Udwadia, R. Cory, J. Nakanishi, S. Schaal., "A unifying methodology for the control of robotic systems"*2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Edmonton, Canada, 2005, pp. 1824-1831.

[16] Sugihara,T. Mobility Enhancement Control of Humanoid Robot based on Reaction Force Manipulation via Whole Body Motion. PhD thesis, University of Tokyo, 2004.

[17] L. Sciavicco, B. Siciliano, *Modeling and Control of Robot Manipulators*, McGraw-Hill, Inc., New York, 1996.

[18] A. Maciejewski and C. Klein, Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments, The International Journal of Robotics Research, Vol. 4, No. 3, pp.109–117, 1985.

[19] G. Strang, *Linear Algebra and its Applications*, 3rd ed., Harcourt Brace Jovanovich College Publishers, Orlando, 1988.