Generation of Dynamic Humanoid Behaviors Through Task-Space Control with Conic Optimization

Patrick M. Wensing, David E. Orin

Abstract—This paper presents a new formulation of prioritized task-space control for humanoids that is used to develop a dynamic kick and dynamic jump in a 26 degree of freedom simulated system. The demonstrated motions are controlled through a real-time conic optimization scheme that selects appropriate joint torques and contact forces. More specifically, motions are characterized in appropriate task spaces, and the real-time optimizer solves the task-space control problem while accounting for user-defined priorities between the tasks. In contrast to previous solutions of the Prioritized Task-Space Control (PTSC) problem for humanoids, the solution presented here satisfies the ZMP constraint and ground friction limitations at all levels of priority, and is general to periods of flight as well as support. All generated motions include control of the system’s centroidal angular momentum, which leads to emergent whole-body behaviors, such as arm-swing, that are not specified by the designer. In addition, compared to a previous quadratic programming solution of the PTSC problem, our approach gains a factor of 2 speedup in its required computational time. This speedup allows the control approach to operate at real-time rates of approximately 200 Hz.

I. INTRODUCTION

While the field of humanoid robots has received much attention over the past decade, control of dynamic whole-body humanoid behaviors remains a difficult objective. Although challenging, the proper coordination of humanoids’ many degrees of freedom (DOF) offers many benefits such as in the recovery from disturbances through arm-swinging [1] or to redirect the torso during a fall through managing inertial couplings from the legs [2]. While these additional DOFs provide more flexibility to whole-body motion, they also add complexity to select appropriate joint torques in real time. Additionally, the large number of DOFs make it difficult to author motions by hand, which has lead to widespread use of motion capture techniques for whole-body motion generation [1], [3], [4], [5].

Task-space (or operational-space) control [6], [7] provides a framework that significantly eases the burden of authoring whole-body behaviors. Task-space trajectories can be designed in intuitive motion spaces. For instance, walking motions can be specified through design of foot and center of mass trajectories. Without the need for any inverse kinematics, online task-space controllers provide an elegant solution to generate whole-body walking behaviors [8].

In this paper, Prioritized Task-Space Control (PTSC) is used to generate dynamic behaviors such as the kick and jump shown in Fig. 1. The common control structure used to create these behaviors is shown in Fig. 2. A high-level state machine manages the commanded task dynamics \( \dot{r}_{t,c} \) and priorities at each instant. A prioritized task-space controller then finds joint torques \( \tau \) which keep feet planted and prevent slipping while realizing the desired task dynamics. This control loop is closed at real-time rates of approximately 200 Hz.

Fig. 1. A new formulation of the Prioritized Task-Space Control problem allows us to control dynamic behaviors such as a kick and a jump at real-time rates and in challenging environments. The use of Centroidal Momentum control results in rich emergent arm motions to maintain balance without any upper-body motion authoring.

Fig. 2. Block diagram for behavioral controllers used in this work. A high-level state machine manages the commanded task dynamics \( \dot{r}_{t,c} \) and priorities at each instant. A prioritized task-space controller then finds joint torques \( \tau \) which keep feet planted and prevent slipping while realizing the desired task dynamics. This control loop is closed at real-time rates of approximately 200 Hz.
centroidal momentum control [9] is used to alleviate the need to design specific upper-body trajectories, which instead emerge from our control strategies.

**Related Work in Whole-Body Humanoid Control**

While we adopt a task-space approach here, many previous approaches assume access to pre-designed joint trajectories and apply an inverse dynamics controller to select joint torques. Under the assumption of sufficient friction on the feet, Mistry et al. [10] use a projection method to select appropriate torques and to resolve over-actuation in double support. Improvements to their approach [11] attempt to push ground reaction forces inside their frictional boundaries. Still, no motion modification is applied, which can lead to balance failure when aggressive joint accelerations are desired. This problem can be accounted for by the application of motion filtering techniques that modify a motion to remain balanced [5], [12]. Park et al. formulate a stringent motion filter as a conic optimization problem [13] which forces the feet to remain in full support. Other motion filtering techniques incorporate long-term balance strategies based on simple models while trying to follow motion [3]. Again, all these approaches require whole-body joint trajectories which are costly to plan in advance and even more difficult to modify online.

Task-space control provides an alternative to the aforementioned approaches, and can be solved in a variety of ways. Projection methods of Park and Khatib [7] and Sentis et al. [14] allow task-priorities to be enforced and deal well with support foot constraints as long as the desired task dynamics lead to balanced motion. Just as a motion filter can be applied to balance pre-designed joint trajectories that violate contact force constraints (such as ZMP), similar filters can be applied to physically unrealizable task-space trajectories. For example de Lasa et al. [8] and Salini et al. [15] both use a series of quadratic programs (QPs) to solve PTSC while satisfying ground reaction force constraints. This paper is inspired by their work and reformulates their QPs into a conic optimization problem where PTSC can be solved at real-time rates. This improvement allows high-bandwidth control of highly dynamic movements such as the kick and jump showcased here.

**II. Task-Space Control for Humanoids**

**A. Notation**

This section introduces notation and summarizes the previous approaches that are used to solve the PTSC problem for humanoids. Given an \( n \) degree-of-freedom (DOF) floating-base humanoid, as shown in Fig. 3, the system’s configuration can be described by:

\[
q = \begin{bmatrix} q_b^T & q_a^T \end{bmatrix}^T.
\]

Here \( q_b \in SE(3) \) is the unactuated position and orientation of the system’s floating base, described in Fig. 3 and \( q_a \) denotes the actuated joints’ configurations. The system’s joint rate \( \dot{q} \in \mathbb{R}^{n+6} \) and acceleration vectors and \( \ddot{q} \in \mathbb{R}^{n+6} \) are partitioned similarly. The standard dynamic equations of motion are:

\[
H(q)\ddot{q} + C(q, \dot{q})\dot{q} + G(q) = S_a^T \tau + J_s(q)^T F_s
\]

(2)

where \( H, C, G \) are the familiar mass matrix, velocity product terms, and gravitational terms, respectively [16]. Here \( F_s \) collects ground reaction forces (GRFs) for appendages in support, as described in Fig. 3, and \( J_s \) is a combined support Jacobian. The matrix \( S_a = [0_{n \times 6} \ 1_{n \times n}] \) is a selection matrix for the actuated joints.

In order to author whole-body behaviors, it is often convenient to characterize the system’s desired dynamics in a task (or operational) space [6]. Task velocities \( \dot{x}_t \) are related to joint-rates \( \dot{q} \) by the standard relationship:

\[
\dot{x}_t = J_t(q) \dot{q}
\]

(3)

where \( J_t(q) \) is a task Jacobian. Alternatively, our task may not arise from a Jacobian relationship, such as in the control of the system’s net angular momentum. To accommodate such a generalization, we relax this definition of a task to any relationship of the form:

\[
r_t = A_t(q) \dot{q}.
\]

(4)

Given commanded instantaneous task dynamics \( \dot{r}_{t,c} \), the task-space control problem is to find joint torques \( \tau \) that result in joint accelerations \( \ddot{q} \) with:

\[
A_t \dot{q} + A_{\ddot{q}} \ddot{q} = \dot{r}_t
\]

(5)

such that \( \dot{r}_t \) most closely matches \( \dot{r}_{t,c} \). Depending on the choice of \( \dot{r}_{t,c} \) additional freedoms may be used to match lower-priority task dynamics. For systems in support, there are a variety of ways to solve this problem, which differ in how they account for the presence of external forces \( F_s \) in (2). These different methods are discussed in the following subsections.
B. Projection Methods for Task-Space Control During Support

In periods of support, the equations of motion (2) are often considered under the constraint of zero acceleration at the support bodies:

$$\ddot{x} = J_s \ddot{q} + \dot{J_s} q = 0.$$  \hfill (6)

The support forces $F_s$ required to ensure (6) can be substituted into (2) to produce a set of constrained dynamic equations of motion:

$$H \ddot{q} + N_s^T (C \ddot{q} + G) + \gamma(q, \dot{q}) = N_s^T s_a^T \tau$$  \hfill (7)

where $\gamma = J_s^T (J_s H^{-1} J_s^T) \dot{J_s} \dot{q}$ is a velocity dependent term due to the constraints and

$$N_s^T = 1 - J_s^T (J_s H^{-1} J_s^T)^\dagger J_s H^{-1}$$  \hfill (8)

is the dynamically-consistent null-space projector for support [7]. The symbol $(\cdot)^\dagger$ denotes the Moore-Penrose pseudo-inverse of the enclosed matrix. In a rough sense, these equations of motion are a projection of the original equations of motion onto the subset of the configuration space that is consistent with the support constraint. Since no constraints are placed on the GRFs, these dynamic equations are in fact not correct if torques are supplied that would lift the foot off the ground, or cause the foot to slip.

Starting from (7) the works of Park and Khatib [7] and Sentis et al. [14] extend the traditional prioritized operational-space control framework for manipulators to the case of humanoids in support. Just as in the above derivation, all their control laws are predicated on the assumption that given a joint input $\tau$, the ground is capable of supplying support forces $F_s$ to ensure the foot remains stationary. This assumption is powerful, and often valid on level terrain since GRFs are dominated by a gravity compensation force. Still, for highly dynamic motions, additional care has to be taken to ensure that the commanded task dynamics do not require GRFs outside their unidirectional or frictional boundaries. This becomes increasingly difficult on more challenging terrains. Their approach uses a nested series of task nullspace projectors to enforce task priority [14].

C. Ground Reaction Force Constraint Modeling

To account for the true constraints on ground reaction forces (GRFs) during the selection of joint torques, others ([8], [15], [17], [18]) have proposed the use of constrained quadratic programming to solve the PTSC problem. These approaches treat $F_s$ as a control variable and optimize the selection of $\tau$ and $F_s$ under appropriate constraints. In order to approximate the constraints on each net foot wrench $f_{sij}$, it is customary to represent each wrench as a combination of pure forces that act at the corners of the feet, as shown in Fig. 4. Unidirectional and frictional constraints can then easily be enforced on the individual vertex forces $f_{sij} \in \mathbb{R}^3$. Given a coefficient of friction $\mu_i$ for foot $i$, these forces must reside inside a friction cone:

$$\mathcal{C}_i := \left\{ (f_x, f_y, f_z) \in \mathbb{R}^3 \left| \sqrt{f_x^2 + f_y^2} \leq \mu_i f_z \right\} \right..$$  \hfill (9)

To simplify optimization, previous work has approximated these cones by a friction pyramid $\mathcal{P}_i \subset \mathbb{R}^3$ (shown in gray) as in previous work.

D. Quadratic Programming Methods for Task-Space Control During Support

Previous work [8], [18] formulates a quadratic program (QP) to find contact forces, joint accelerations $\ddot{q}$, and joint torques $\tau$ that are consistent with the dynamic equations of motion, and most closely match the commanded task dynamics. An example of such a QP is given below.

$$\min_{\ddot{q}, \tau, f_{sij}} \frac{1}{2} \| A_i \ddot{q} + \dot{A}_i \dot{q} - \ddot{r}_{t,c} \|^2$$  \hfill (10)

subject to $H \ddot{q} + C \dot{q} + G = S_a^T \tau + \sum_{i=1}^{N_S} \sum_{j=1}^{N_{P_i}} J_{sij} f_{sij}$

$$f_{sij} \in \mathcal{P}_i \ \forall i \in \{1, \ldots, N_S\}, \ \forall j \in \{1, \ldots, N_{P_i}\}$$

$$J_s \ddot{q} + \dot{J}_s \dot{q} = 0$$  \hfill (12)

$$\tau \leq \tau \leq \mathcal{T}$$

Here each $J_{sij} \in \mathbb{R}^{3 \times (6+6)}$ is the Jacobian for the contact point where $f_{sij}$ acts. Joint torque limits are described by the vectors $\mathcal{T}$ and $\mathcal{F}$. The unidirectional constraints also imposed on the GRFs through (11) combined with the support acceleration constraint (12) assure that the optimized $\ddot{q}$ satisfy the ZMP constraint.

To optimize for lower priority tasks, the QP may be modified and additional constraints included to ensure that high-priority task dynamics are not corrupted. This nested series of QPs (or stack of QPs) replaces the nested series of nullspace projectors that are required for projection based methods. In the work of Mansard [17], an elimination of variables is employed that reduces the size of the above QP at each step, but this reduction is specific to periods of
support. In the next section, we propose a reduction that is
general to periods of flight as well as support.

### E. Net Momentum Control

As a quick aside, we note that the previous formulation is
general to control features that cannot be described with
a Jacobian relationship. Most notably, this generality can
be exploited to control the system’s centroidal momentum
\( h_G \) [9]. The centroidal momentum \( h_G = [k_G^T, I_G^T]^T \)
is a spatial momentum comprised of the system’s net linear
momentum \( I_G \in \mathbb{R}^3 \) and angular momentum \( k_G \in \mathbb{R}^3 \) about
the Center of Mass (CoM). The centroidal momentum is
given by the relationship

\[
h_G = A_G(q) \dot{q}
\]

where \( A_G \) is known as the Centroidal Momentum Matrix
[9]. While this looks like a Jacobian relationship, the
system’s net angular momentum does not admit any function
\( g(q) \) with Jacobian \( J_g = \frac{\partial g}{\partial q} \) such that \( k_G = J_g \dot{q} \). This
result is due to the fact that the conservation of angular
momentum imposes a nonholonomic constraint [19]. Methods
to compute \( A_G \) and \( A_G \dot{q} \) are given in [1] and [9].

### III. Conic Formulation of Prioritized Optimization

This section provides a reformulation of the PTSC problem
as a conic optimization problem. This reformulation results in a reduced
number of variables, reduced number of
constraints, and is general to periods of flight as well as support. This reduction allows our reformulation to be solved
about twice as fast as (10) for the examples considered.
Suppose that the task matrix \( A_t \) contains all possible tasks
regardless of priority. The error in achieving a commanded
700 task dynamics is given by:

\[
e = \dot{r}_{t,c} - \dot{r}_t
\]

(14)

\[
e = \dot{r}_{t,c} - A_t \dot{q} - A_t \dot{q}.
\]

(15)

Solving (2) for \( \dot{q} \), the task dynamics error \( e \) obeys:

\[
b_t = A_t H^{-1} S_t^T \tau + A_t H^{-1} J_{sp}^T F_{sp} + e
\]

where:

\[
b_t = \dot{r}_{t,c} + A_t H^{-1} (C \dot{q} + G) - A_t \dot{q}.
\]

(17)

In (16), \( F_{sp} \) collects all support forces \( f_{s,j} \) and \( J_{sp} \) is
the combined support point Jacobian. This support point
Jacobian relates \( \dot{q} \) to the linear velocity of all the support
vertices and differs from the support Jacobian \( J_{s} \) which
related \( \dot{q} \) to the linear and angular velocity of all support
bodies. We define:

\[
\Lambda_{ts}^{-1} = A_t H^{-1} S_t^T \quad \text{and} \quad \Lambda_{ts}^{-1} = A_t H^{-1} J_{sp}^T
\]

(18)

(19)

with symbols \( \Lambda^{-1} \) since each of these is a cross-coupling
inverse operational-space (task-space) inertia matrix [20].

Here, the PTSC problem with \( K \) levels of prioritization is
solved with a series of \( K \) conic optimization problems. At
each level \( k \in \{1, \ldots, K\} \) it is assumed that the subset of
the tasks to be optimized is encoded in a task optimization
selector matrix \( S_{o,k} \). Each row of \( S_{o,k} \) is a unit vector
that selects a single task error, and the number of rows corresponds to the number of tasks concurrently
optimized at level \( k \). The optimal error from the previous level is defined as \( e_{k-1} \). A similar task constraint selector \( S_{c,k} \) selects those
tasks from all previous levels, as well as any additional hard
constraints. The problem for priority level \( k \) is then:

\[
\begin{align*}
\min_{\tau, F_{sp}, e_k, z} & \quad z \\
\text{s.t.} & \quad \Lambda_{ts}^{-1} \tau + \Lambda_{ts}^{-1} F_{sp} + e_k = b_t \\
& \quad \|S_{o,k} e_k\| \leq z \\
& \quad S_{c,k} e_k = S_{c,k} e_{k-1} \\
& \quad f_{s,j} \in C_i \\
& \quad \tau \leq T \leq \tau^*
\end{align*}
\]

(20)

(21)

(22)

(23)

Minimization of the scalar \( z \) in (20) results in a minimization
of error for the current task dynamics to be optimized
due to constraint (21). The incorporation of \( z \) also provides
a linear objective, which is required for the conic solver
used here. The constraint (22) ensures that the optimal
errors for the higher priority tasks are not corrupted. Table I
describes the entire algorithm for this series of optimization
problems in more explicit detail. Note that for application
to position controlled systems, an optimal \( \dot{q} \) can be obtained
by solving (2) or through a forward dynamics computation.

<table>
<thead>
<tr>
<th>Inputs:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task Dynamics Descriptors:</td>
</tr>
<tr>
<td>System Dynamics Descriptors:</td>
</tr>
<tr>
<td>Task Priority Descriptors:</td>
</tr>
</tbody>
</table>

**Algorithm:**

\[
\begin{align*}
\Lambda_{ts}^{-1} & = A_t H^{-1} S_t^T \\
\Lambda_{ts}^{-1} & = A_t H^{-1} J_{sp}^T \\
b_t & = \dot{r}_{t,c} + A_t H^{-1} (C \dot{q} + G) - A_t \dot{q} \\
& \quad \text{for } k = 1 \text{ to } K \text{ do} \\
& \quad (\tau^*, F_{sp}^*, e_k^*, z^*) := \arg \min_{\tau, F_{sp}, e_k, z} (20) \\
& \quad S_{c,k}^{(k+1)} := |S_{c,k}^T S_{o,k}|^{-1} \\
& \quad \text{end for } k
\end{align*}
\]

**Output:**

\( \tau := \tau^* \)

**TABLE I**

**Prioritized Task-Space Control (PTSC) Algorithm**

The advantages of this strategy over the QP in (10) is
multi-pronged. First, the variables \( \dot{q} \) are eliminated in favor
of task error \( e \). Since at each level only a few tasks are
optimized, this reduces the number of optimization variables,
and simplifies the objective. Salini [18] eliminates \( \dot{q} \) in
an alternative QP formulation, but does not include \( e \),
resulting in a dense objective function Hessian. As a result,
the formulation of [18] and the QP here were found to perform comparably. Second, polygonal approximations to
the friction cones, which grow in complexity as the fidelity
of the approximation is increased, are replaced by a single
constraint per force \( f_{sij} \). For a 4-sided polygonal approximation, the use of friction cones was not found to have any substantial effect on the computation times or simulation results. Beyond a 4-sided polygonal approximation, it was found to be advantageous to use cone constraints to enable faster solutions. Solution of all optimization formulations was provided by the interior point optimizer in MOSEK [21]. This optimizer employs a primal-dual method that handles the cone constraints (21) and (23) directly and efficiently.

IV. APPLICATION TO THE CONTROL OF DYNAMIC BEHAVIORS

A. Model and Simulation

The model used in this work is a 26 DoF (20 actuated DoFs) humanoid as shown in Fig. 3. Spherical joints are modeled at the hips and shoulders, providing 6 DoFs in each leg, and 4 DoFs in each arm. Degrees of freedom in the hands, wrists, and head are not modeled. The mass distribution to each segment is modeled after a 50-th percentile male [22], with segment dimensions based on the model presented in [23]. Inertia tensors are estimated based on a simple equidensity mass distribution for each segment.

Full 3D simulation of the system is carried out using the DynaMechs [24] simulation package. This package provides an extremely efficient implementation of the articulated-body algorithm [25] with recursive steps of the algorithm optimized for each type of joint. Contact dynamics are simulated with a penalty-based spring damper model. No force feedback or information about the environment (e.g. surface normal or surface height) is provided to the controller.

B. Control of a Dynamic Kick

As a first example, PTSC is used along with a state machine, shown in Fig. 5, to control a dynamic kicking motion. During all states, the task prioritization (1) Feet, (2) Linear Momentum, (3) Pose & Centroidal Angular Momentum. All state transitions are based on time in this example.

Fig. 5. State machine used for Kick control. All states use the task priorities (1) Feet, (2) Linear Momentum, (3) Pose & Centroidal Angular Momentum. All state transitions are based on time in this example.

where \( e_\theta \in \mathbb{R}^3 \) is an angle-axis representation of error between a desired and actual orientation, as used in [4]. Desired positions and orientations are derived from hand-authored motion, such as the kick trajectory in the kick state.

For Centroidal Momentum, a rate of change \( \dot{r}_c = [k_{G,c}^T, I_{G,c}^T]^T \) is commanded separately for linear and angular momentum. For linear momentum, this command is from PD control on the desired CoM (G):

\[
\dot{l}_{G,c} = m_0[p_{G,d} + K_{D,l}(p_{G,d} - p) + K_{P,l}(p_{G,d} - p)]
\]

where \( p_G \) is the CoM position and \( m \) is the total mass of the system. The commanded rate of change in angular momentum takes a simpler form:

\[
\dot{k}_{G,c} = \dot{k}_{G,d} + D_{k,c} - \dot{k}_{G}.
\]

All of the balance states and the lift state employ \( k_{G,d} = 0 \) which provides a dampening of any excess angular momentum.

To achieve pose control, joint accelerations are commanded for actuated joints and the torso orientation. For all examples, this commanded acceleration takes the form of a PD law to a static nominal pose. For revolute joints:

\[
\dot{q}_{i,d} = l_{i,c} + K_{D,i}(q_{i,d} - q_i) + K_{P,i}(q_{i,d} - q_i),
\]

where \( \dot{q}_{i,d} = \dot{q}_{i,d} = 0 \) in all the examples here. For spherical joints and orientation of the torso, the law (24) is employed. Since the pose task is optimized at the last level of control, it is generally not possible to fulfill all the desired pose dynamics. Weighting factors are employed that promote closer tracking on certain joints than others. For instance, the shoulder and elbow are given lower weight to promote arm action in the resultant motion. These weights can be incorporated by replacing \( e \) with \( W e \) in (16) for an appropriate diagonal weighting matrix \( W \).

Simple spline trajectories are used throughout to generate the desired dynamic motions. Cubic spline trajectories on the CoM and right foot are used to generate the desired motions for the Balance and Lift states. In the kick state, the foot is commanded to move in an arc centered at the initial right hip position. The orientation of the right foot is commanded to remain tangent to this arc. A series of cubic splines on the desired right virtual leg angle \( \theta_d \) are used to produce the desired 3D accelerations of the right foot using standard formula that relate accelerations in polar coordinates to cartesian coordinates.

During a kick, the system predominantly rotates about the stance hip, resulting in non-zero centroidal angular momentum. For the example shown, the inertial \( -z \)-axis is opposite gravity and the \( y \)-axis is perpendicular to the sagittal plane. To promote a whole-body rotation about the \( y \)-axis, the system’s net moment of inertia about the \( y \)-axis \( I_{yy,0} \) is recorded at the beginning of the kick motion and desired centroidal angular momentum and rates are selected as:

\[
k_{G,des} = [0, \gamma I_{yy,0} \hat{\theta}_d, 0]^T,
\]

\[
\hat{k}_{G,des} = [0, \gamma I_{yy,0} \ddot{\theta}_d, 0]^T.
\]
Here $\gamma = 0.8$ is a factor that accounts for the stance leg remaining stationary. While crude, these desired centroidal momentum dynamics are some of the first non-trivial ones to be designed in the literature and are sufficient to produce rich motion. For a video of the kick motion, please see the attachment to this paper or view it at the link below.

http://www.go.osu.edu/WensingOrin_ICRA2013

Without authoring any upper-body trajectories, the control approach yields complex upper-body motion. Shoulder and torso angles during the kick state are shown in Fig. 6. During the kick motion, the left shoulder shows a larger displacement. This behavior emerges from the controller to regulate the angular momentum about the global vertical $z$-axis in response to the right foot’s kick trajectory.

The conic formulation of PTSC was found to be much faster than the QP formulation for this example. As shown in Fig. 7, the conic formulation (20) was able to be solved in nearly half the time as the QP formulation (10) (in 55% of the time on average). A 4-sided polygonal approximation to the friction cone was employed for the QP formulation. Thus, the majority of the improved speed results here can be attributed to the variable reductions and simplified objective due to the use of $\epsilon$. The times shown in this graph include the computation time of all quantities required by each algorithm ($H, \dot{A}, q, \Lambda_{\dot{A}}^{-1}$, etc.) as well as the optimization time of the solver. All computational experiments were run on a 2.3 GHz Intel Core i5 MacBook Pro.

### C. Control of a Standing Broad Jump

The PTSC framework was also applied to produce a standing broad jump using the state machine shown in Fig. 8. All states during stance use the task priorities (1) Feet, (2) Centroidal Momentum, (3) Pose. The state in red (Flight) uses task priorities (1) Feet, (2) Pose. State transition criteria are noted on the transition arrows, where an omission of a criterion indicates a transition that takes place based on time.

The PTSC framework was also applied to produce a standing broad jump using the state machine shown in Fig. 8. All states during stance use the task priorities (1) Feet, (2) Centroidal Momentum, (3) Pose. The state in red (Flight) uses task priorities (1) Feet, (2) Pose. State transition criteria are noted on the transition arrows, where an omission of a criterion indicates a transition that takes place based on time.

Without authoring any upper-body trajectories, the control approach yields complex upper-body motion. Shoulder and torso angles during the kick state are shown in Fig. 6. During the kick motion, the left shoulder shows a larger displacement. This behavior emerges from the controller to regulate the angular momentum about the global vertical $z$-axis in response to the right foot’s kick trajectory.

The conic formulation of PTSC was found to be much faster than the QP formulation for this example. As shown in Fig. 7, the conic formulation (20) was able to be solved in nearly half the time as the QP formulation (10) (in 55% of the time on average). A 4-sided polygonal approximation to the friction cone was employed for the QP formulation. Thus, the majority of the improved speed results here can be attributed to the variable reductions and simplified objective due to the use of $\epsilon$. The times shown in this graph include the computation time of all quantities required by each algorithm ($H, \dot{A}, q, \Lambda_{\dot{A}}^{-1}$, etc.) as well as the optimization time of the solver. All computational experiments were run on a 2.3 GHz Intel Core i5 MacBook Pro.

The PTSC framework was also applied to produce a standing broad jump using the state machine shown in Fig. 8. All states during stance use the task priorities (1) Feet, (2) Centroidal Momentum, (3) Pose. The state in red (Flight) uses task priorities (1) Feet, (2) Pose. State transition criteria are noted on the transition arrows, where an omission of a criterion indicates a transition that takes place based on time.

Without authoring any upper-body trajectories, the control approach yields complex upper-body motion. Shoulder and torso angles during the kick state are shown in Fig. 6. During the kick motion, the left shoulder shows a larger displacement. This behavior emerges from the controller to regulate the angular momentum about the global vertical $z$-axis in response to the right foot’s kick trajectory.

The conic formulation of PTSC was found to be much faster than the QP formulation for this example. As shown in Fig. 7, the conic formulation (20) was able to be solved in nearly half the time as the QP formulation (10) (in 55% of the time on average). A 4-sided polygonal approximation to the friction cone was employed for the QP formulation. Thus, the majority of the improved speed results here can be attributed to the variable reductions and simplified objective due to the use of $\epsilon$. The times shown in this graph include the computation time of all quantities required by each algorithm ($H, \dot{A}, q, \Lambda_{\dot{A}}^{-1}$, etc.) as well as the optimization time of the solver. All computational experiments were run on a 2.3 GHz Intel Core i5 MacBook Pro.

The PTSC framework was also applied to produce a standing broad jump using the state machine shown in Fig. 8. All states during stance use the task priorities (1) Feet, (2) Centroidal Momentum, (3) Pose. The state in red (Flight) uses task priorities (1) Feet, (2) Pose. State transition criteria are noted on the transition arrows, where an omission of a criterion indicates a transition that takes place based on time.

Without authoring any upper-body trajectories, the control approach yields complex upper-body motion. Shoulder and torso angles during the kick state are shown in Fig. 6. During the kick motion, the left shoulder shows a larger displacement. This behavior emerges from the controller to regulate the angular momentum about the global vertical $z$-axis in response to the right foot’s kick trajectory.

The conic formulation of PTSC was found to be much faster than the QP formulation for this example. As shown in Fig. 7, the conic formulation (20) was able to be solved in nearly half the time as the QP formulation (10) (in 55% of the time on average). A 4-sided polygonal approximation to the friction cone was employed for the QP formulation. Thus, the majority of the improved speed results here can be attributed to the variable reductions and simplified objective due to the use of $\epsilon$. The times shown in this graph include the computation time of all quantities required by each algorithm ($H, \dot{A}, q, \Lambda_{\dot{A}}^{-1}$, etc.) as well as the optimization time of the solver. All computational experiments were run on a 2.3 GHz Intel Core i5 MacBook Pro.
where hand authored upper-body motions were used [27]. The torso and shoulder angles for the level terrain jump are shown in Fig. 9. During thrust, the arms are swung upwards to prevent the torso from pitching backwards. If the arms are forced to remain locked, the lack of this behavior results in excess torso pitch as shown.

The jump controller for level terrain is applied without modification for a jump onto uneven terrain. Once the system lands, no knowledge of the terrain is assumed, and the feet are simply commanded to not accelerate. Since the control approach used here always attempts to push off the ground, corners of the foot that are originally not in contact are quickly pushed into contact with the unknown terrain. Based on these experiments, it appears that force feedback may not be required for reasonably stiff terrain.

In a final video demonstration, we show the performance of the jump controller when landing on a slippery surface. In this demonstration, the controller assumes that the ground has a coefficient of friction of $\mu = 0.6$, but the simulation employs a coefficient of $\mu = 0.4$. While the system does experience foot slip due to this inaccuracy, the balance controller is robust to this disturbance and results in non-authored arm-windmilling to maintain balance. The desired CoM is constantly updated in the example to remain over the middle of the support polygon.

V. Conclusion

Prioritized Task-Space Control provides a convenient framework in which to characterize dynamic behaviors for humanoid systems. Through consideration of the constraints on ground reaction forces, motions for level terrain are able to be easily adapted to uneven terrain scenarios. Reformulation of previous QP formulations for PTSC allows speed gains to be achieved while addressing friction constraints in their full complexity with conic optimization. This reformulation enables real-time control rates of 200 Hz. These algorithms are general to control quantities such as the system’s net angular momentum, which leads to rich upper-body behaviors that are not authored by the designer. Future work will focus on addressing joint limits and self-collision as well as analysis of generated torque data for both QP and conic formulations. Additionally, the use of these methods to support higher-level motion planners provides an exciting direction of future work that may provide humanoids with a broader vocabulary of dynamic maneuverability.

VI. Acknowledgments

This work was supported by a National Science Foundation Graduate Research Fellowship to Patrick Wensing, and by Grant No. CNS-0960061 from the NSF with a subaward to The Ohio State University. Special thanks to Yiping Liu for assistance with the simulation videos. We also wish to thank the reviewers for their helpful comments.

REFERENCES