2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Versatile and Robust 3D Walking with a Simulated Humanoid Robot (Atlas): a Model Predictive Control Approach

Salman Faraji, Soha Pouya, Christopher G. Atkeson[‡], and Auke Jan Ijspeert
Biorobotics Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland
[‡]Robotics Institute, Carnegie Mellon University, Pittsburgh, USA

***Abstract*— In this paper, we propose a novel walking method for torque controlled robots. The method is able to produce a wide range of speeds without requiring off-line optimizations and re-tuning of parameters. We use a quadratic whole-body optimization method running online which generates joint torques, given desired Cartesian accelerations of center of mass and feet. Using a dynamics model of the robot inside this optimizer, we ensure both compliance and tracking, required for fast locomotion. We have designed a foot-step planner that uses a linear inverted pendulum as simplified robot internal model. This planner is formulated as a quadratic convex problem which optimizes future steps of the robot. Fast libraries help us performing these calculations online. With very few parameters to tune and no perception, our method shows notable robustness against strong external pushes, relatively large terrain variations, internal noises, model errors and also delayed communication.**

I. INTRODUCTION

Among legged robots, bipeds are difficult to control compared to quadruped and multi-legged robots. Keeping the Center of Mass (CoM) inside support polygon (statically-stable walking) produces an un-natural and slow motion. The concept of Zero Moment Points (ZMP) ([1], [2]) is then used which enables the CoM to move more freely while still having dynamic stability. Maintaining the ZMP inside the support polygon is the key rule, preventing feet to roll or tilt. For instance, [3] and [4] use inverse kinematics to produce and track stable ZMP trajectories.

Model-free methods like [5], [6] and [7] optimize a walking pattern off-line, targeting stability, fast motion, energy efficiency or similarity to human walking. Other approaches incorporate a kinematics model of the robot into the loop, which enables them to convert Cartesian positions, velocities or forces to joint variables and vice-versa. Among these are [8] and [3] that use inverse kinematics to reproduce trajectories while [9] and [10] use this data to convert Cartesian forces to joint torques. The latter is called Virtual Model Control, which balances the robot and performs locomotion by means of virtual spring/dampers, connected between the robot and a moving frame.

Using dynamics-model allows us to predict future motion of the robot at a wide range of speeds. This additional information is more useful than pure kinematics regarding dynamics coupling effects, since we can calculate required torques for a specific motion. It requires smaller feedback gains for perturbation rejection and behaves more compliant compared to kinematics-based methods which follow trajectories using high gains. The benefits of using robot's dynamics model to reduce the need for high-gain position controllers have been discussed and demonstrated via simulation as well as real robot experiments in [11] and [12]. Compliance is important because in legged robots, each step makes an impact on the whole body. These shocks can be harmful for the environment and for the robot itself if it behaves stiffly. There are some mechanical techniques to alleviate these effects such as using soft feet. Designers use spring/damper sets either mechanically or virtually in joints to decrease stiffness of a position-controlled robot. The former can also be used in torque-controlled robots, but it generally complicates the control problem and adds more dynamics to the system.

The current platform we use to test our algorithms is the Atlas robot simulated in Gazebo, an ODE-based rigid body simulator. Fig.1 shows the real robot built by Boston Dynamics [1] that serves as a platform for the DARPA Robotic Challenge. Using strong hydraulic actuators, the robot is capable of performing many dexterous tasks which makes it a good platform for our fast walking method as well.

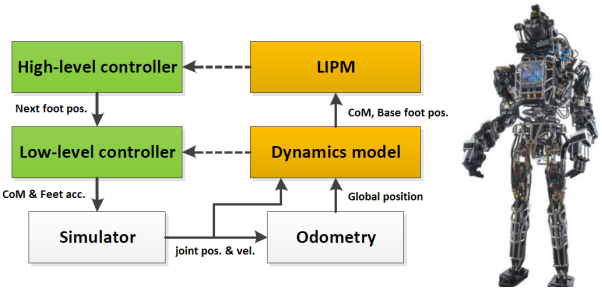

Fig. 1: The fully torque controlled Atlas robot by Boston Dynamics. It is 190cm tall and weighs 150Kg. Leveraging high power hydraulic actuators, it is capable of doing many dexterous tasks. The 2-layers control architecture is shown on left which will be described in sections II and III.

In this work, we propose a walking control framework for torque-controlled humanoids (See Fig.1) which is not based on joint trajectories. Here we only have joint accelerations in the control loop and follow trajectories in the Cartesian or task space. For the purpose of walking, planning these trajectories is simpler as we are interested in footstep locations. The problem formulation is presented

[1] http://www.bostondynamics.com/

in section II. Given task-space accelerations produced by PD tracker controllers, we run an optimization algorithm which optimizes joint accelerations and contact forces while considering friction cones, center of pressure and joint torque limits as constraints. The outcome will be joint torques calculated from equations of motion (EOM). In [13], [14] the same approach is taken, however they first optimize contact forces with those constraints separately and then minimize joint accelerations by pseudo-inversion in next step. We prefer the approach of [15] as it simply solves the problem in one step and enables us to consider torque limits as well. We also use a convex optimization library called CVXGEN [16] in which we formulate the problem efficiently tailored to our own specific application. CVXGEN avoids solving the problem by forming standard large matrices which are sparse in our case. We can then solve the problem very rapidly and avoid the computationally expensive matrix operations used in [13]. In comparison to previous works, the main features and strengths of our low-level controller are two-folded: (i) it operates in task-space and does not depend on prescribed joint-space trajectories and (ii) it is computationally efficient and capable of solving the corresponding optimization problem online. This controller will be described in section II.

In section III, we explain the high-level algorithm we use to plan Cartesian task-space trajectories for the above-mentioned low-level controller. We use a state machine to determine the pattern of motion and also the swing-phase trajectories. The high-level planner uses Linear Inverted Pendulum Model (LIPM) [17], [18] to produce CoM motion which helps us reduce problem dimensions and plan for multiple future steps. This planning is done in the form of a discrete-time Model Predictive Control (MPC) which determines future footstep locations.

To compare, [19] has similar control architecture. At a low level, they run a quadratic kinematics-based problem which optimizes joint torques given Cartesian forces. We use a dynamics model however which brings more compliance and improves task space tracking properties. The method in [19] optimizes half-cycle full body trajectories off-line and replicates them online. We plan online however with our LIPM simplified model which is more robust and fast enough. Whitman in [14] uses the same low-level control, however it plans body motion off-line and builds a library of motions supporting a wide range of states and required speeds. We avoid such computationally expensive off-line optimization and plan the motion online.

Other works such as Foot Point Indicator (FPI) [20] and Instantaneous Capture Point (ICP) [18],[21], [22] and [4] plan a motion for taking the next step, where the robot goes to rest condition after being pushed. FPI takes full body dynamics into account while ICP simplifies the robot's model into a LIPM. The latter is computationally less expensive which allows calculating rest condition after multiple steps [18]. Inspired by this idea, we formulate a MPC problem which brings the robot to a desired speed instead of rest condition in multiple steps. Our work is therefore similar to [23] except that we use LIPM instead of SLIP (Spring Loaded Inverted Pendulum), where we can linearly express the state of the system in multiple footsteps and optimize it per time-step.

Our low-level controller is slightly better than similar formulations in the sense that it can consider torque limits as well. However the novelty of the present work is in high-level footstep planning rather than the low-level controller. The proposed discrete-time MPC formulation in high-level is based on the LIPM and lets us deal with the footstep planning constraints in a systematic way. We provide a linear convex optimization that allows online planning over multiple future steps regarding the current state of the CoM. Section IV is dedicated to evaluate this approach through different scenarios, in which the robot is subject to various internal perturbations (like sensor noises or model errors) or external perturbations (like pushes or unevenness in the terrain). The conclusion and future works are discussed in section V.

II. Low Level Controller

In this section, we present our approach for generating the joint torques, given task-space trajectories. Using a dynamics-model based method which calculates the required torques for the given motion, we can reduce tracking feedback gains and make the robot more compliant. On top of this controller layer, a planner determines Cartesian trajectories and tracks them by PD controllers which produce Cartesian accelerations. Controlling a floating based robot with many degrees of freedom is difficult in joint space. We are interested in giving task-space trajectories however to reduce the problem size in trajectory planning level.

Our whole body optimization approach takes Cartesian accelerations of CoM and feet as the inputs and considers the lower body joint accelerations and contact forces as open parameters subject to optimization. Ultimately, the joint torques can be extracted by replacing the above-mentioned parameters in the equation of motion. Such task space formulation eliminates the need to use pre-optimized joint trajectories that are usually driven off-line. Hereafter, we formulate our problem for the specific type of robot under control.

Beside Gazebo as the main simulator, the controller is running in a separate process on the same computer while ROS[2] is used to exchange the robot's states and commands. Atlas publishes its states at 1KHz over ROS and our controller subscribes to these packets. The robot in Gazebo has on-board controllers for each joint which generate joint commands according to the following PID rule:

$$\tau = \tau_{des} + K_p(q_{des} - q) + K_d(\dot{q}_{des} - \dot{q}) + K_i \int (q_{des} - q) \quad (1)$$

So for each joint, our controller can give a feed-forward torque τ_{des} and desired trajectories q_{des} and $\dot{q}_{des}$ as well as feedback gains K_p, K_d and K_i. In this work, we merely give desired torques τ_{des} to leg joints (lower body) and keep torso joints and arms (upper body) fixed. This is done by

[2] http://wiki.ros.org/

giving constant q_{des} and $\dot{q}_{des} = 0$ to fixed joints and disabling PID for leg joints. However we already calculate gravity compensation torques for the upper body of the robot. Thus, we have 12 DoF for the legs and 16 DoF fixed in the robot.

The equation of motion (EoM) is:

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) = \mathbf{S}\boldsymbol{\tau} + \mathbf{J}_\mathbf{C}^\mathbf{T}(\mathbf{q})\boldsymbol{\lambda} \tag{2}$$

With $n = 28$ to be the total number of joints and k to be the number of constraint equations, variables are defined as:

- $\mathbf{q} \in \mathbb{R}^{n+6}$: All degrees of freedom
- $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{(n+6)\times(n+6)}$: The floating base inertia matrix
- $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n+6}$: The floating base centripetal, Coriolis and gravity forces
- $\mathbf{S} \in \mathbb{R}^{(n+6)\times n}$: The actuated joint selection matrix
- $\boldsymbol{\tau} \in \mathbb{R}^n$: The vector of actuated joint torques
- $\mathbf{J_C} \in \mathbb{R}^{k\times(n+6)}$: The Jacobian of k linearly independent constraints
- $\boldsymbol{\lambda} \in \mathbb{R}^k$: The vector of k linearly independent constraint forces if any exists

We have 6 DoF in global coordinates, $k = 6$ in single support phase and $k = 12$ in double support phase when both feet are on the ground. Regarding each foot, we have:

$$\begin{aligned} \dot{\mathbf{x}}_\mathbf{c} &= \mathbf{J_c}\dot{\mathbf{q}} \\ \ddot{\mathbf{x}}_\mathbf{c} &= \mathbf{J_c}\ddot{\mathbf{q}} + \dot{\mathbf{J}}_\mathbf{c}\dot{\mathbf{q}} \end{aligned} \tag{3}$$

Positions, velocities and accelerations should be consistent in both Cartesian and Joint spaces. In the case of ground contact, these equations should be equal to zero.

As mentioned before, the inputs to this low level layer of the controller are Cartesian accelerations for the feet and CoM and the outputs are joint torques. In Eqn.2, variables are decomposed as:

- $\ddot{\mathbf{q}} = \begin{bmatrix} \ddot{\mathbf{q}}_p \in \mathbb{R}^3 & \ddot{\mathbf{q}}_o \in \mathbb{R}^3 & \ddot{\mathbf{q}}_u \in \mathbb{R}^{16} & \ddot{\mathbf{q}}_l \in \mathbb{R}^{12} \end{bmatrix}$
- $\boldsymbol{\tau} = \begin{bmatrix} \mathbf{0} \in \mathbb{R}^\mathbf{6} & \boldsymbol{\tau}_u \in \mathbb{R}^{16} & \boldsymbol{\tau}_l \in \mathbb{R}^{12} \end{bmatrix}$
- $\boldsymbol{\lambda} = \begin{bmatrix} \boldsymbol{\lambda}_{left} \in \mathbb{R}^6 & \boldsymbol{\lambda}_{right} \in \mathbb{R}^6 \end{bmatrix}$

Where $\ddot{\mathbf{q}}_p$ and $\ddot{\mathbf{q}}_o$ refer to position and orientation of the robot's frame respectively. The subscript u refers to the upper body and l refers to the lower body of the robot. Our specific way of formulating the problem takes the following steps:

1) **Given by the high level controller:**
   - $\ddot{\mathbf{x}}_{\mathbf{c.l}} \in \mathbb{R}^\mathbf{6}$: Cartesian acceleration of the left foot
   - $\ddot{\mathbf{x}}_{\mathbf{c.r}} \in \mathbb{R}^\mathbf{6}$: Cartesian acceleration of the right foot
   - $\ddot{\mathbf{x}}_{\mathbf{p,com}} \in \mathbb{R}^\mathbf{3}$: CoM translational acceleration
   - $\ddot{\mathbf{q}}_\mathbf{o} \in \mathbb{R}^\mathbf{3}$: Torso's angular acceleration
   - desired hybrid state (left, right or double support)
2) **Set to zero:** $\ddot{\mathbf{q}}_u$.
3) **Subject to optimization:** $\ddot{\mathbf{q}}_p$, $\ddot{\mathbf{q}}_l$, $\boldsymbol{\lambda}_{left}$ and $\boldsymbol{\lambda}_{right}$.
4) **Calculated from EoM:** $\boldsymbol{\tau}_u$ as upper-body gravity compensation and $\boldsymbol{\tau}_l$ as leg actuator torques.

We implicitly formulate a quadratic problem with the standard form of Eqn.4:

$$\begin{aligned} &min_\mathbf{x}\ \mathbf{x}^\mathbf{T}\mathbf{Q}\mathbf{x} \\ &s.t. \\ &\mathbf{A}\mathbf{x} + \mathbf{B} = \mathbf{0} \\ &\mathbf{C}\mathbf{x} + \mathbf{D} \geq \mathbf{0} \end{aligned} \tag{4}$$

Where $\mathbf{Q}$ is a positive semi-definite matrix, $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ and $\mathbf{D}$ are coefficient matrices, and $\mathbf{x}$ is a concatenation of variables subject to optimization in step 3. We also have various constraints that should be considered:

Equality:

- **EoM as a constraint:** Considering Eqn.2, the 6 equations where there is no actuation on global coordinates are equality constraints.
- **Joint/Cartesian consistency:** Joint accelerations should be consistent with the given Cartesian accelerations of the CoM and both feet, either being in contact or in swing phase.
- **Existence of foot constraint:** When we want the robot to switch to right support for example, we should force left forces to zero in the optimization. Torque limits on ankles are proportional to ground normal forces as mentioned in [14] regarding the CoP limits. Thus setting normal forces to zero prevents the existence of non-zero torques as well.

Inequality:

- **Center of Pressure (CoP):** Ankle torques should be limited regarding the size of feet to avoid tilting or rolling effects.
- **Ground friction:** CoM accelerations should not exceed maximum static ground friction.
- **Actuator torque limits:** We can also take care of joint torque magnitudes and rate limits as we can calculate them from EoM, knowing joint accelerations and constraint forces.

Since the matrices generated by Eqn.2 are mostly sparse, we only select proper blocks which are used in calculations and not multiplied by zeros. The library we use for solving this optimization problem is called CVXGEN [16] which generates C codes, tailored for the specific formulation of our problem. It avoids explicitly forming matrices of Eqn.2 and instead, it generates codes which do math operations only on none-zero elements.

A small difference that our formulation has with [15] and [13] is the orientation regulation which is done here directly on the torso. Herzog in [15] uses whole body momentum rate to regulate $\mathbf{q_o}$ (torso orientation), whereas we directly use torso's angular acceleration ($\ddot{\mathbf{q}}_o$) to regulate its orientation. Thus, the inputs and outputs of our PD orientation regulator are on the same variable.

So far we have described the general form of formulating the optimization used in the low level controller. On top of this, we have some more blocks that restrict motion of the robot, but greatly reduce problem dimensions.

- **Orientation control:** In general, we keep postures of both feet and torso upright by a PD controller which uses $\ddot{\mathbf{x}}_{\mathbf{o.left}}$, $\ddot{\mathbf{x}}_{\mathbf{o.left}}$ and $\ddot{\mathbf{q}}_o$ as control inputs.
- **Acceleration regulator:** It might happen that the given desired accelerations exceed maximum friction and ankle torques available. This block thresholds incoming accelerations.
- **Force distribution:** In double support, the optimization

equally distributes contact forces between the two feet which is not a realistic assumption. We determine corresponding elements in quadratic cost matrix $\mathbf{Q}$ such that the weighting becomes proportional to closeness of the CoM to each foot similar to [14].

This formulation enables us to abstract a high dimensional problem in joint space to a low dimensional one in Cartesian space. The high-level controller block now only generates translational accelerations of three points in the robot and also imposes the discrete state.

III. High Level Controller

In the previous section, we described the low level controller used to generate joint torques, given the high level Cartesian accelerations. We have thus transferred trajectories to task space where they are easier to formulate and generate. As mentioned in the previous section, we only determine Cartesian accelerations and the hybrid state of the system in the high level controller and keep posture of the torso and the feet always upright, using a PD controller. The pattern of motion in our method is determined via a finite state machine with fixed frequency and phase.

To plan future motions of the robot, we prefer to simplify its model to make the problem computationally less expensive. Mordatch et al. [23] use SLIP model which is more realistic in terms of contact forces [7]. This model lets the CoM change height which looks more natural especially during running. However since the SLIP model is not linear, predicting the future motion with this model is not easy in closed form. The method in [23] approximates trajectories with polynomials and expresses states of the system across hybrid transitions in closed form. Optimizing the evolution of variables over future steps is not linear however and has many local optima. Although their planning produces more natural motion, it takes considerable time while reducing performance and responsiveness to perturbations. This motivates us to choose a simpler model to describe robot's state for future planning during walking.

A. Model simplification

In this work, we use the Linear Inverted Pendulum Model [18] without feet and inertial mass to simplify the robot's motion. This model is composed of a mass and a telescopic actuator which is basically massless. The equation for acceleration of this structure is:

$$\ddot{x} = \frac{g}{z_0}(x - x_{base}) \tag{5}$$

Where x refers to CoM and x_{base} to base positions, z_0 is the CoM height and g is gravity. We can write same equations for the y direction assuming no steering and rotation. The assumption in LIPM is that the telescopic actuator keeps the CoM height constant (z_0). This assumption makes the equation linear and enables us to solve it in the time domain as:

$$x(t) = Ae^{-t/\tau} + Be^{t/\tau} + C \tag{6}$$

By examining $x(0)$ and $\dot{x}(0)$ in this equation we can obtain:

$$\tau = \sqrt{\frac{z_0}{g}},\ C = x_{base} \tag{7}$$
$$A = \frac{-\tau\dot{x}(0) + x(0) - x_{base}}{2}$$
$$B = \frac{\tau\dot{x}(0) + x(0) - x_{base}}{2}$$

If we approximate the robot in single support phase with this simplified model and assume a fixed swing duration of T, we can express the CoM state evolution by defining $h = e^{T/\tau}$:

$$\begin{aligned} x(T) &= x(0)(\frac{1}{2h} + \frac{h}{2}) + \dot{x}(0)(\frac{-1}{2h} + \frac{\tau h}{2}) \\ &+ x_{base}(\frac{-1}{2h} + \frac{-h}{2} + 1) \end{aligned} \tag{8}$$

which is a linear relation between the state of CoM in the beginning and at the end of a swing phase. One can obtain similar relations for y, $\dot{x}$ and $\dot{y}$ as well. Expressing the full state of the CoM with these variables as $\mathbf{q} \in \mathbb{R}^4$, we can write:

$$\mathbf{q(T)} = \mathbf{A(T)q(0)} + \mathbf{B(T)P_{base}} \tag{9}$$

Where $\mathbf{A}$ and $\mathbf{B}$ are functions of T and thus constant. Note that a similar evolution could be calculated for the CoM in the middle of a swing phase with T as the nominal remaining time.

Performing such linearization in double support phase is not easy because we have two supports and the force distribution policy makes the problem nonlinear. Thus we prefer to omit this phase so that the robot switches between the left and right support phases directly. Instead of leveraging ankle torques to control the CoM path, we let the robot follow LIPM behavior and basically fall while the swing leg touches down in a location which captures the energy of the robot. Thus we convert the CoM control problem to a footplacement problem with this strategy. The obvious benefit is avoiding dependency on having feet and ankle torques and thus, not being concerned about the CoP (this constraint is considered in the low level controller).

We can now formulate up to N future steps and predict the motion of the robot between these steps:

$$\begin{aligned} \mathbf{q[1]} &= \mathbf{A_0 q[0]} + \mathbf{B_0 P[0]} \\ \mathbf{q[2]} &= \mathbf{Aq[1]} + \mathbf{BP[1]} \\ \mathbf{q[3]} &= \mathbf{Aq[2]} + \mathbf{BP[2]} \\ &\vdots \\ \mathbf{q[N+1]} &= \mathbf{Aq[N]} + \mathbf{BP[N]} \end{aligned} \tag{10}$$

The first equation in Eqn.10 slightly differs as $\mathbf{q[0]}$ corresponds to the current state of the system in the middle of a swing phase. To obtain $\mathbf{q[1]}$ which is the state at the end of same swing phase, the remaining phase time is less than T which changes nominal matrices $\mathbf{A}$ and $\mathbf{B}$ to $\mathbf{A_0}$ and $\mathbf{B_0}$.

Given the current state of CoM at each time-step $\mathbf{q[0]}$, our approximate model lets us predict the evolution of future states with a linear sequence where footsteps act like discrete

inputs. Note that one can easily verify controllability [24] of the system matrices $\mathbf{A}$ and $\mathbf{B}$ with our choice of $T = 0.5s$ which results in a natural and fast motion. Additionally, simulating the robot walking over several steps ($150s$ simulation time) verifies the walking stability. [3]

B. Model predictive control

Having the discrete time dynamical system, we can now formulate a Model Predictive Control (MPC) problem to plan future footsteps. The robot is principally falling during the single support phase while the swing leg is navigating toward a desired location. This location is the first control input $\mathbf{P}[\mathbf{1}]$ in Eqn.10 which captures the extra energy in the system. Swing trajectories in task space are defined using sinusoidal arcs and a soft exponential transitions between the previous and the next desired footstep locations which guaranty zero speeds at the two ends of the arc. These arcs are then tracked by PD controllers in the task space, producing Cartesian accelerations. The x and y components of the CoM acceleration are determined according to the LIPM model (Eqn.5) in the z_0 plane and this height is controlled via a PD controller.

We require the robot to track a certain average velocity $\mathbf{v_{ref}}$. Our MPC minimizes the least square error between this velocity and future states of the robot ($\mathbf{q}[i]$ in Eqn.10). It also minimizes the least square errors between control inputs (future footsteps $\mathbf{P}[i]$ in Eqn.10) and a desired sequence defined in Eqn.11 regarding the average speed.

$$\begin{aligned} \mathbf{P_{des}}[i] &= \mathbf{P_{base}} + 2iT \times \mathbf{v_{ref}} \\ &+ (i \ mod \ 2) \begin{bmatrix} 0 & dk \end{bmatrix}^T, \quad 1 \leq i \leq N \end{aligned} \tag{11}$$

The variable d is inter-feet clearance distance determined manually and k is either 1 or -1, if the planning is done in right or left support phases respectively. If we cancel this similarity minimization, foot steps become closer in the lateral plane (left/right) and finally overlap to minimize lateral motion of the CoM which is not desired due to self collision. We thus try to keep the feet separated in the ideal foot sequence. Our MPC problem is again formulated in CVXGEN up to a horizon of $N = 5$ future steps which is enough regarding the range of speeds we are targeting. Using the CoM state at each time-step, we solve the MPC problem to find the next footstep location, used to generate arc trajectories.

The only top-level variable to be determined is the average speed ($\mathbf{v_{ref}}$) which we regulate with a simple PD controller to track the desired path and speed profile for the robot. With this formulation, the robot stabilizes dynamically and recovers from perturbations by taking proper steps rather than relying on ankle torques. Herdt et al. [25] formulate a similar MPC problem that optimizes both footstep locations and the CoP trajectories, but their low level controller is based on inverse kinematics. We do not use ankle torque to modulate the motion like [25], rather we use them for perturbation rejection (in our low level controller) which yields in a simpler problem formulation and enhanced robustness in different scenarios.

In this layer, the restricting assumption is constant CoM height which essentially makes the problem linear. One can use more complex models if the target is to reach higher speeds and a more natural motion. Also, feet are always assumed to be in full contact (both toes and heels) if placed on the ground which again restricts the motion at high speeds.

IV. RESULTS AND DISCUSSIONS

In this section, we will characterize the performance and robustness of the two blocks we described in the previous sections. We first test compliance of the low level controller by testing the robot's reaction/behavior against external pushes. To this end, in stance mode we have two PD controllers regulating CoM position and torso's posture. We make the position controller stiffer so that (i) the CoM does not move considerably and (ii) the force distribution between the feet does not change. The resulting behaviour after applying a constant push of $50N$ on the pelvis is shown in Fig.2 where the posture changes and helps the robot withstand such a strong external push.

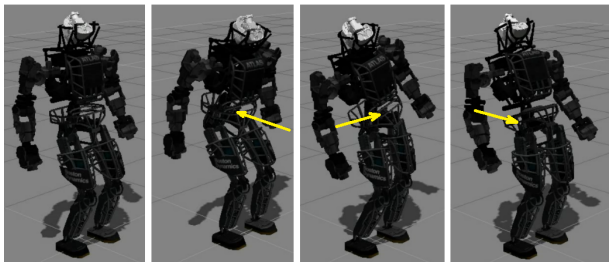

Fig. 2: Static push recovery scenario in (a) normal, (b) backward, (c) left and (d) front directions. The posture change helps the robot withstand a maximum constant push of $50N$ on its pelvis. The CoM controller is stiffer and keeps the CoM between the two feet.

This test shows the compliance of the robot and the fact that it can handle large pushes without reacting stiffly. The complex joint space control problem is converted to task space where PD controllers are more meaningful with respect to the desired tasks.

For walking, we show the sequence of footsteps planned by the MPC controller in Fig.3. Regarding the duration of the single support phase which is $T = 0.5s$, we plot the planned sequence every $0.1s$ to show its emergence. Fig.3 shows 6 snapshots of this sequence where the robot is in left support phase in the beginning. When it switches to right support, the blue rectangle shows the next desired footstep location ($\mathbf{P}[1]$) where the swing foot should touch down. This point is then used to generate arc trajectories.

Note that the slight bending to left and right in the sequence of Fig.3 is because of the PD average speed regulator on top of the high level controller. Such behavior is due to the regulation of CoM lateral motion which is not

[3] Movies for all the presented scenarios in this work are online available at: `http://biorob.epfl.ch/page-96274-en.html` .

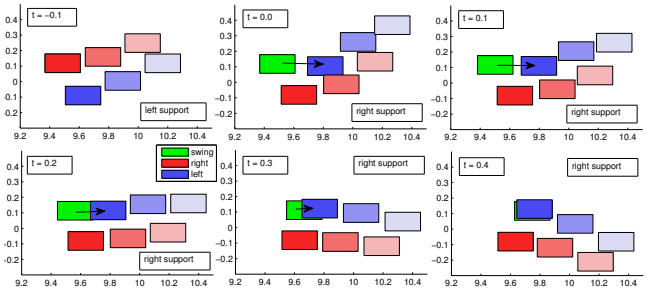

Fig. 3: Sequence of steps planned by MPC during a single support phase. The figure shows snapshots from this sequence every $0.1s$ where the duration of single support is $T = 0.5s$. In the upper left plot, the robot is in left support and about to switch the contacting foot. The swing foot (green) is trying to reach the blue location $\mathbf{P}[1]$ gradually in next plot.

normally needed except when the robot is on a lateral slope. The role of this PD speed regulator is thus correcting biased errors in both lateral and coronal directions. Note also that the choice of N does not have big impacts on the sequence. We observe that choosing larger N will produce the same repeated motion. Smaller N causes the sequence to capture energy very fast which produces large steps.

With this control architecture and manual tuning of few parameters (mostly Cartesian PD controllers), we can obtain a notable controllability over forward speed, namely from $-0.2m/s$ up to $0.5m/s$ which is remarkable. This ability is more highlighted when comparing to off-line optimized solutions where the gait can be sustained only around a specific nominal solution. Achieving higher speeds would require releasing some restricting assumptions like full foot contact and knee bending which are future works.

There could be various uncertainties and perturbations in the real robot such as frictions, damping, actuator dynamics, noises, model errors and etc. In this part we analyze the robustness against some important internal and external perturbations.

A. External pushes

Methods like [4] and [25] depend on ankle torques to control the CoP and provide motion. These approaches produce good tracking, but they are not robust to terrain variations and depend also on contact properties. In this work however, we react to perturbations by taking steps. Our MPC maintains the system stability dynamically by capturing extra energy in the robot. It does not rely solely on ankle torques and thus makes locomotion robust to moderate variations in the terrain and contact model. Here we expose the robot to external pushes in four directions during walking. Fig.4 shows the resulting foot steps when $\mathbf{v_{ref,x}} = 0.4m/s$. The robot successfully recovers from these pushes and continues stepping forward.

B. Delayed communication

The controller almost runs in about $2.3ms$ with both layers of optimizations. The simulation and controller processes run on the same machine with a Quad-Core Intel Core-i5 processor running at 1.7GHz. We have no parallelization for the controller as its layers work sequentially. A remarkable property of our controller is that it tolerates a total delay of about $10.7ms$ caused by the packet based communication method (ROS). This can be further improved when running the controller on-board on the real robot.

C. System noise

Another important and inevitable issue in real robot control is the sensor noise. To examine our controller performance in the presence of noise, we have added a Gaussian noise of zero mean and standard deviation of σ to all joint state variables during in-place stepping. Such noise can affect the task-space motion of a kinematic chain severely. The maximum noise tolerable before falling or self collisions is characterized in Table.I.

TABLE I: Maximum tolerable noise.

Variable	Noise strength σ
position sensors & IMU	1.8^o
velocity sensors & IMU	$9.9^o/s$
Output torques	$3N.m$

This test shows that the robot is able to tolerate a reasonable amount of noise, although various methods exist in the literature to filter such per time-step noise.

D. Model errors in link masses and lengths

In the previous test, we tested the effect of sensor noises on the robustness of our method. However, there are constant errors which can not be predicted, measured or filtered. Modeling error is one of these problems, which is important in our model based controller. To test the sensitivity of the method to the modeling errors, here we change some link masses in the simulator while they are kept intact in our robot model. These changes make the robot heavier and asymmetric. Here we assume the inertia matrix of a link to be the same as before. In Table.II we have listed maximum extra mass that could be added to different links while in-place stepping is still stable.

TABLE II: Maximum tolerable mass added to different links as model error.

scenario	link name	maximum mass [Kg]
(a)	pelvis	3
(b)	left thigh	5
(c)	left foot	3
(d)	left and right feet	3 each

Our controller is in fact tolerating a large amount of unknown mass added to links, but limit cycles change considerably. We can also perform similar tests during walking where $\mathbf{v_{ref}} \neq 0$. Generally the maximum tolerable mass is less than the case of stepping in place, but similar effects are observed. We have also tested the algorithm against link length errors in a specific scenario where we make the left thigh $10cm$ longer and the left shank $10cm$ shorter to keep the total leg length the same. This is the maximum tolerable error for the robot while still being stable. Our algorithm is capable of tolerating such a large internal geometric error without affecting task-space tracking considerably.

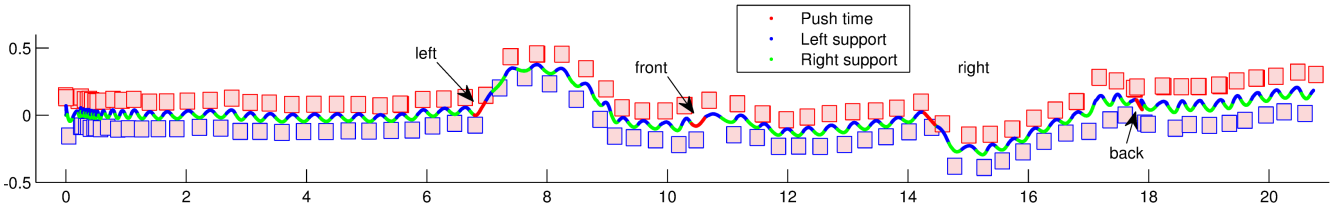

Fig. 4: Push recovery scenario with $\mathbf{v_{ref}} = 0.4m/s$. Pushes of $40N$ each lasting $0.5s$ are applied to the system at $t = 30s$ to left, $t = 40s$ to front, $t = 50s$ to right, $t = 60s$ to back. The CoM is usually outside the support polygons which is verified by the fact that we do not have double support phase. Note that in this figure, $x-y$ scale is not true as robot's rectangular feet are appearing to be square.

E. Walking on uneven terrain and slopes

In this section we test our method against unperceived unevenness and gradual changes of slopes in the terrain. The maximum tolerable roughness (difference between maximum and minimum height) is 10% of the robot's leg length while the robot can go up a slope of 4.6^o at maximum. In both scenarios the terrain is smooth to avoid contact problems and instability in simulations.

Fig.5 and Fig.6 show the uneven terrain scenario and slopes respectively. In the former case, we give $\mathbf{v_{ref}} = 0.2m/s$ to the robot and start on a hill. It speeds up on a downhill slope while slowing down on a uphill slope. It can also shift left or right if it finds a non-negative sideways component in the gradient of the terrain. Note that one can achieve better performance by either choosing higher desired speeds or using larger feedback gains for the PD regulator on top of the high level controller. In the second scenario, the

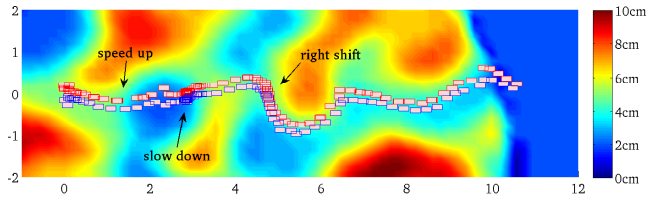

Fig. 5: The performance of our controller with the same configuration of parameters as before. In this figure, the robot is given desired speed of $\mathbf{v_{ref}} = 0.2m/s$. It speeds up on negative slopes and slows down when reaching an uphill slope. If the gradient of the terrain has a sideway component, it can cause the robot to shift to the left or right and bypass a hill.

slope in front of the robot increases $0.02rad \simeq 1.14^o$ every $3m$. We gradually increase slopes, since placing foot on a discontinuity in the surface will change its posture and stability of the simulation in Gazebo.

One can use larger feedback gains here, but this prevents the robot from responding flexibly to the the environment, such as the lateral shifts appearing in Fig.5 for instance. Maximizing performance is a matter of parameter tuning, which is out of the scope of this research.

V. CONCLUSION

In this work, we have chosen a dynamics-model based method as we target versatile biped locomotion which requires knowledge on robot dynamics to allow both compliant behavior and good tracking performance in a wide range of states. Although one could use simpler methods in terms of computation and optimize a gait off-line for fast motion, the resulting solution will only work in a limited range of speeds and terrain conditions. Using inverse dynamics, we predict future motion of the robot and calculate the required torques which reduces feedback gains and thus provides more compliant behavior. We consider various constraints in the control loop for which a whole body optimization method is suitable. Given desired Cartesian accelerations for the CoM and feet, our optimization generates feed-forward torques for the robot.

To generate task space swing foot trajectories, we have a simple online strategy, unlike many approaches which need off-line optimization. We only determine an arc shape in the Cartesian space and follow it while we do not require any desired joint trajectory. Many other approaches need to optimize these trajectories, replicate them online and maintain constraint consistency which results in only a locally optimal and robust pattern generation. We have used a simplified model of the robot which enables us to plan multiple future steps. Although our model assumes constant CoM height and torso posture which results in a less natural motion, the method is generic and can be produce a wide range of speeds.

In terms of calculations, in both low level and high level controllers we have quadratic convex optimizations running per time-step. However our novel problem formulation and simplification has enabled us to do all calculations in almost $2.3ms$, resulting in a fast control loop. Our footstep planner runs per time-step and enables us to react against external pushes almost without delay. On top of the high level controller we have a simple PD controller which regulates the desired average speed.

The final controller is able to withstand large perturbations either externally from pushes or terrain variations, or internally from noise, model errors or delays. A notable example is recovery from strong external pushes while methods depending on ankle torques can not recover from those easily. This performance is without any external perception and thus shows the intrinsic properties of our method. In future work we will add perception and modify strategies and methods to be able to deal with larger terrain variations.

The total of 34 degrees of freedom are guided by 2 task-space average speeds. We can also have control over the feet and CoM in task space which improves navigation and obstacle avoidance. A drawback in our footstep planning method is the assumption on exact foot placement. Although

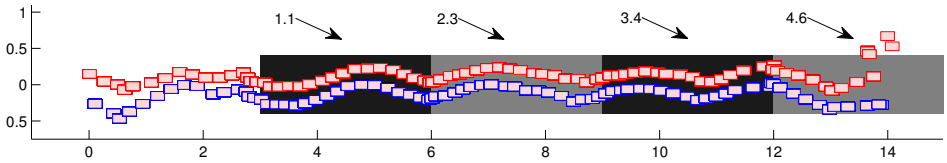

Fig. 6: The performance of our controller with addition of position feedback in the x direction. The robot slows down on slope transitions, but recovers its speed and even accelerates more to reach the desired $x(t)$ due to position feedback. It can go maximally on the slope of 4.6^o and then falls out.

the first step in the sequence could be imposed, the method needs more freedom for further steps to recover from pushes and capture the extra energy. Our method could be improved by adding steering, using more complicated IPM, including torso joints and arms and also adding non-convex constraints in footstep planning to handle self collisions in a more systematic way. We are currently transferring this algorithm to our torque-controlled humanoid robot Coman.

VI. ACKNOWLEDGMENTS

The authors would like to thank Eric Whitman for his kind collaboration and feedbacks in this work.

REFERENCES

[1] P. Sardain and G. Bessonnet, "Forces acting on a biped robot. center of pressure-zero moment point," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 34, no. 5, pp. 630–637, 2004.

[2] M. Vukobratović and B. Borovac, "Zero-moment point—thirty five years of its life," *International Journal of Humanoid Robotics*, vol. 1, no. 01, pp. 157–173, 2004.

[3] Y. Choi, B.-J. You, and S.-R. Oh, "On the stability of indirect ZMP controller for biped robot systems," in *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, vol. 2. IEEE, 2004, pp. 1966–1971.

[4] J. Englsberger, C. Ott, M. A. Roa, A. Albu-Schaffer, and G. Hirzinger, "Bipedal walking control based on capture point dynamics," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011, pp. 4420–4427.

[5] G. Endo, J. Morimoto, T. Matsubara, J. Nakanishi, and G. Cheng, "Learning CPG-based biped locomotion with a policy gradient method: Application to a humanoid robot," *The International Journal of Robotics Research*, vol. 27, no. 2, pp. 213–228, 2008.

[6] C.-S. Park and J.-H. Kim, "Stable modifiable walking pattern algorithm with constrained optimized central pattern generator," in *Robot Intelligence Technology and Applications 2012*. Springer, 2013, pp. 223–230.

[7] H. Geyer and H. Herr, "A muscle-reflex model that encodes principles of legged mechanics produces human walking dynamics and muscle activities," *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, vol. 18, no. 3, pp. 263–273, 2010.

[8] S. Kajita, T. Nagasaki, K. Kaneko, K. Yokoi, and K. Tanie, "A running controller of humanoid biped HRP-2LR," in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 2005, pp. 616–622.

[9] J. Pratt, C.-M. Chew, A. Torres, P. Dilworth, and G. Pratt, "Virtual model control: An intuitive approach for bipedal locomotion," *The I. J. of Robotics Research*, vol. 26, no. 2, pp. 129–143, February 2001.

[10] A. Lauber, "Virtual model control on ALoF," master thesis, Swiss Federal Institute of Technology Zurich, Spring 2011.

[11] L. Righetti, J. Buchli, M. Mistry, M. Kalakrishnan, and S. Schaal, "Optimal distribution of contact forces with inverse-dynamics control," *The International Journal of Robotics Research*, vol. 32, no. 3, pp. 280–298, 2013.

[12] J. Buchli, M. Kalakrishnan, M. Mistry, P. Pastor, and S. Schaal, "Compliant quadruped locomotion over rough terrain," in *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*. IEEE, 2009, pp. 814–820.

[13] B. J. Stephens and C. G. Atkeson, "Dynamic balance force control for compliant humanoid robots," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 1248–1255.

[14] E. C. Whitman and C. G. Atkeson, "Control of instantaneously coupled systems applied to humanoid walking," in *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*. IEEE, 2010, pp. 210–217.

[15] A. Herzog, L. Righetti, F. Grimminger, P. Pastor, and S. Schaal, "Momentum-based balance control for torque-controlled humanoids," *arXiv preprint arXiv:1305.2042*, 2013.

[16] J. Mattingley and S. Boyd, "CVXGEN: a code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.

[17] S. Kajita and K. Tani, "Study of dynamic biped locomotion on rugged terrain-derivation and application of the linear inverted pendulum mode," in *Robotics and Automation, 1991. Proceedings., IEEE International Conference on*. IEEE, 1991, pp. 1405–1411.

[18] T. Koolen, T. De Boer, J. Rebula, A. Goswami, and J. Pratt, "Capturability-based analysis and control of legged locomotion, part 1: Theory and application to three simple gait models," *The International Journal of Robotics Research*, vol. 31, no. 9, pp. 1094–1113, 2012.

[19] J.-c. Wu and Z. Popović, "Terrain-adaptive bipedal locomotion control," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, p. 72, 2010.

[20] P. van Zutven, D. Kostic, and H. Nijmeijer, "Foot placement for planar bipeds with point feet," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 983–988.

[21] J. Pratt, T. Koolen, T. De Boer, J. Rebula, S. Cotton, J. Carff, M. Johnson, and P. Neuhaus, "Capturability-based analysis and control of legged locomotion, part 2: Application to M2V2, a lower-body humanoid," *The International Journal of Robotics Research*, vol. 31, no. 10, pp. 1117–1133, 2012.

[22] J. Pratt, J. Carff, S. Drakunov, and A. Goswami, "Capture point: A step toward humanoid push recovery," in *Humanoid Robots, 2006 6th IEEE-RAS International Conference on*. IEEE, 2006, pp. 200–207.

[23] I. Mordatch, M. De Lasa, and A. Hertzmann, "Robust physics-based locomotion using low-dimensional planning," *ACM Transactions on Graphics (TOG)*, vol. 29, no. 4, p. 71, 2010.

[24] K. Ogata and Y. Yang, "Modern control engineering," 1970.

[25] A. Herdt, N. Perrin, and P.-B. Wieber, "Walking without thinking about it," in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 190–195.