Robust and Agile 3D Biped Walking With Steering Capability Using a Footstep Predictive Approach

Salman Faraji
Biorobotics Laboratory
Ecole Polytechnique
Fédérale de Lausanne, Switzerland
Email: salman.faraji@epfl.ch

Soha Pouya
Neuromuscular Biomechanics Laboratory
Stanford University, USA
Email: spouya@stanford.edu

Auka Jan Ijspeert
Biorobotics Laboratory
Ecole Polytechnique
Fédérale de Lausanne, Switzerland
Email: auke.ijspeert@epfl.ch

***Abstract*—In this paper, we formulate a novel hierarchical controller for walking of torque controlled humanoid robots. Our method uses an online whole body optimization approach which generates joint torques, given Cartesian accelerations of different points on the robot. Over such variable translation, we can plan our desired foot trajectories in Cartesian space between starting and ending positions of the foot on the ground. On top level, we use the simplified Linear Inverted Pendulum Model to predict the future motion of the robot. With LIPM, we derive a formulation where the whole system is described by the state of center of mass and footstep locations serve as discrete inputs to this linear system. We then use model predictive control to plan optimal future footsteps which resemble a reference plan, given desired sagittal and steering velocities determined by the high-end user. Using simulations on a child-size torque controlled humanoid robot, the method tolerates various disturbances such as external pushes, sensor noises, model errors and delayed communication in the control loop. It can perform robust walking over slopes and uneven terrains blindly and turn rapidly at the same time. Our generic dynamics model-based method does not depend on any off-line optimization, being suitable for typical torque controlled humanoid robots.** [1]

I. INTRODUCTION

Among various kinds of robots, legged robots are difficult to control as one needs to maintain stability all the time while performing desired tasks. Although quadruped robots have some intrinsic stability, the case is more restricted for bipeds or humanoids since the support region for Center of Mass (CoM) is smaller comparatively. A traditional way to control the robot is to keep the CoM inside the support region all the time while taking steps (statically-stable walking). However this produces an un-natural motion in terms of high coronal motion, non-smooth sagittal motion and stepping speed, compared to real humans. One can use the concept of Zero Moment Points (ZMP) [20] to allow the CoM to move more freely while maintaining dynamic stability. Keeping the ZMP inside the support region will prevent the feet from tilting or rolling when the COM is outside.

Various methods are introduced to perform locomotion by modulating ZMPs. Based on Inverted Pendulum Model (IPM) [7], one can produce desired motions for the CoM using ankle joints and controlling ZMP ([2, 3]). More complicated forms of the IPM which assume inertia for the base of the pendulum are also used in literature to improve walking stability. In [18], the inertia mass is used to rest after taking a step while in [24], it compensates swing leg's dynamics. Inverted pendulum models are useful for simplifying bipeds in single support phase, aiming at predicting future motion of the robot with lower computational cost compared to using a full model of the system. An example of exploiting such simplification is [9] where the concept of capture points is introduced. Simple linear dynamics of capture points let us predict future motion of CoM and plan the next footstep position which will absorb the energy in the robot.

Planning locomotion and performing low-level joint control of the robot are two interleaved topics. Using the Jacobian of robot's state vector, one can translate Cartesian variables to joint variables and vice-versa. Virtual Model Control method ([17]) is an example of translating forces while various methods translate velocities as well [2] or integrate them to obtain joint positions. However when we target agile and versatile locomotion, it becomes helpful to also incorporate the knowledge from dynamics of the robot into the loop. While Jacobian merely provides information about the geometry of the robot, dynamics-models can predict required joint torques to realize the desired motion. Without this knowledge, one should rely on the performance of individual joint controllers and their tracking performance specially in case of position controlled robots. Using high gains for better tracking normally leads to stiff behavior which could be harmful for the environment and the robot itself, especially in legged robots which deal with impacts at each step. Besides, stability, compliance and accuracy depend on tuning of various parameters, posture of the robot and speed of the desired motion resulting in a complicated trade-off problem. Thus, one prefers to use fewer and weaker feedback gains to reject perturbations and rely more on dynamics-based information of the robot to avoid high stiffness and being more compliant.

Inspired from operational space formulation of Khatib [8] and unified formulation of Aghili [1], inverse-dynamics methods have been widely used on humanoid robots either using joint-space trajectories like [19] or using Cartesian trajectories like [25, 22]. In such formulation, one optimizes joint torques and constraint forces, given desired joint space or Cartesian

[1]This work was funded by the WALK-MAN project (European Community's 7th Framework Programme: FP7-ICT 611832)

accelerations. Although the equality constraint of contact acceleration is considered in closed form solutions [19], one can not include inequality constraints like Center of Pressure (CoP), friction cones or joint torque limits easily. This motivates solving a quadratic constrained problem using fast QP solvers per time step where one can consider all constraints at the same time ([6], [25], [24] and [11]). With such formulation of the joint controller, we can track desired trajectories at end-effectors with simple PD controllers while being compliant, maintaining balance and satisfying all constraints.

In this paper, we propose a hierarchical control architecture for locomotion of torque controlled humanoids similar to our previous work [4] but with steering capabilities. The method is composed of three layers:

1) Whole body optimization: generates joint torques given Cartesian accelerations of the feet and the CoM.
2) Trajectory pattern generator: produces and tracks task-space trajectories for feet and CoM, given the next footstep position and orientation. The outputs of this layer are Cartesian accelerations, given to the previous layer.
3) Foot-step planner: produces discrete footstep patterns, given the desired sagittal, coronal and yaw speeds that the user determines. The first footstep position is then given to the previous layer.

The first layer is described in section II where we formulate the quadratic problem we solve to generate joint torques. Distinguished features of our formulation and comparison with other implementations will be then discussed, though the main contribution of this work is not at this level. Next in section III, we describe our smooth Cartesian trajectory generation and tracking policy in the second layer. These trajectories are defined between the initial and the final footstep positions at the end of the current swing phase. The latter is determined by the third layer introduced in section IV. This layer optimizes future steps of the robot according to the LIPM and a reference footstep plan, based on Cartesian velocities that the user determines with joystick. The optimization in this layer is written as a linear discrete Model Predictive Control (MPC) problem [10] over future footsteps. Fig.1 visualizes the different control layers and the information flow between them.

The novelty of this work is mainly in the third layer where we plan future footsteps in discrete space. In [11], first optimal trajectories are found off-line by inverse kinematics and a small QP for the configuration of the robot (based on [16]). Using these trajectories, the method proposed in [11] then finds optimal ZMP trajectories that satisfy robots frictional and torque limit constraints. Using these trajectories and a linear Time Variant LQR, the method solves for CoM dynamics over an arbitrary horizon and then CoM accelerations found are given to their low level active-set based QP solver to generate joint torques. The major difference of [11] with our method is the fact that we do not have any off-line optimization. Rather we optimize the next footstep positions on-line, considering the aforementioned constraints and a reference footstep pattern

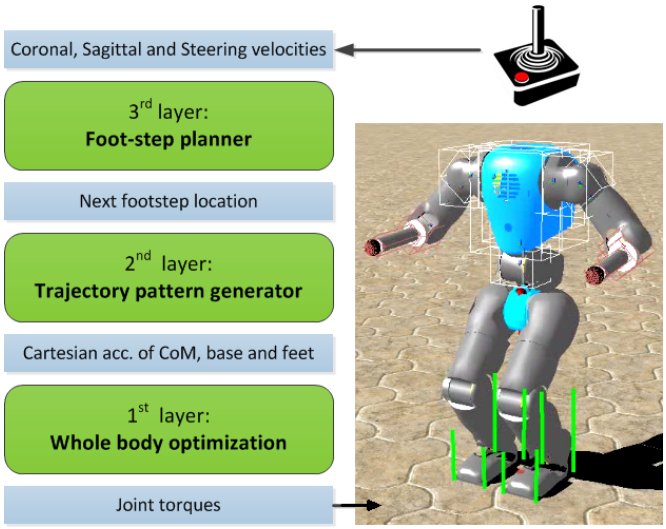

Fig. 1. In this figure, the hierarchical architecture of the control is shown together with a picture of the real robot. We have three different layers which receive simple commands from user and generate joint torques for the robot.

which is defined in a closed form. Moreover, although the time-variant LQR in [11] allows change in CoM height with a certain policy, keeping the CoM height constant has minimal practical effect on their robot's balance performance. Comparatively, we make the latter assumption and obtain a simple closed form discrete linear system describing future steps. In [11], authors incorporate a simple terrain estimation algorithm to determine the desired CoM height profile. However we will show that our blind method with constant CoM height is notably robust against terrain variations. Overall, we try to reject such disturbances by taking proper footsteps, rather than modulating ZMP and relying on ankle torques [11, 3, 23].

While bringing the system to the rest conditions after pushes exists in literature over a single step [23] or multiple steps [9], we aim at a more generic scenario, i.e. bringing the system to a desired non-zero velocity (given by a joystick). Besides, the idea of capture points in [9] is similar in terms of formulating future steps of the robot in closed form by simplifying it into LIPM. However we consider both the CoM position and speed rather than a single quantity, i.e. capture point. Our approach can also be taken by using Spring Loaded Inverted Pendulum (SLIP) model as a simplified model of running. Hertzmann et al. in [14] predict up to four future half cycles combining single support, double support and ballistic motion phases with nonlinear formulations of the SLIP model. Although it can perform walking and running on various terrains, the planning itself takes considerable time, making the robot not responsive to strong perturbations such as pushes which change the CoM state rapidly. Our footstep planner instead takes less than $0.2ms$ to solve the problem.

After the next three introductory sections on controller layers, we will shortly introduce the simulation platform and the Coman robot [21] used to validate our controller in section V. Results of simulations over a wide range of forward and

steering speeds given by joystick is then presented in the same section. We will also test our method against various perturbations either internally like noise or model errors or externally like pushes or terrain variations.

II. First layer: Whole body optimization

In this section, we present our QP problem formulation used for the low-level joint control. Similar to several previous works ([25, 23, 22, 6]), the objective function has a quadratic form while constraints consist of the Equation of Motion (describing dynamics of the robot), and other physical constraints (such as end-effector constraints, CoP being inside feet polygon and frictions polyhedrals). The difference of our implementation however is that, we solve the problem by including the joint torque limits in the constraints. This is in contrast to Dynamic Balance Force Control (DBFC) methods [22, 23] which divide the problem in two levels: (i) finding optimal contact force distributions and then (II) finding joint torques via pseudo-inversion and treating the torque limits by saturation.

To formulate our problem, we first define the state vector of our robot as $\mathbf{q} = \begin{bmatrix}\mathbf{p_b} & \mathbf{o_b} & \mathbf{q_j}\end{bmatrix}^{\mathrm{T}}$, where $\mathbf{p_b} \in \mathbb{R}^{\mathbf{3}}$ and $\mathbf{o_b} \in \mathbb{R}^{\mathbf{4}}$ represent position and quaternion orientation of the robot's reference frame attached to the pelvis in the global coordinates and $\mathbf{q_j} \in \mathbb{R}^{\mathbf{n}}$ represents $n = 23$ joint angles. Throughout this paper, we use a convention to show positions by $\mathbf{p}$, quaternions by $\mathbf{o}$ and both of them together with $\mathbf{x}$. We update the global variables $\mathbf{p_b}$ and $\mathbf{o_b}$ by an internal odometry based on IMU and joint sensors. We write the Equation of Motion (EoM) for this rigid body in Eq.1:

$$\begin{aligned}\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) &= \boldsymbol{\tau} + \mathbf{J}_{\mathbf{C}}^{\mathrm{T}}(\mathbf{q})\boldsymbol{\lambda} \\ \ddot{\mathbf{x}}_{\mathbf{C}} &= \mathbf{J}'_{\mathbf{C}}(\mathbf{q})\ddot{\mathbf{q}} + \dot{\mathbf{J}}'_{\mathbf{C}}(\mathbf{q})\dot{\mathbf{q}}\end{aligned} \tag{1}$$

Where $\mathbf{M}(\mathbf{q}) \in \mathbb{R}^{(n+6)\times(n+6)}$ is the inertia matrix, $\mathbf{h}(\mathbf{q}, \dot{\mathbf{q}}) \in \mathbb{R}^{n+6}$ represents the floating base centripetal, Coriolis and gravity forces, $\boldsymbol{\tau} = \begin{bmatrix}\mathbf{0} \in \mathbb{R}^6 & \boldsymbol{\tau}_{\mathbf{j}} \in \mathbb{R}^{\mathrm{n}}\end{bmatrix}^T$ is the vector of actuated joint torques, $\mathbf{J_C}(\mathbf{q}) \in \mathbb{R}^{k\times(n+6)}$ is the Jacobian of k linearly independent constraints and $\boldsymbol{\lambda} \in \mathbb{R}^k$ is the vector of k constraint forces. Note that there is no need to consider the 4th element of the quaternion vector $\mathbf{o_b}$ in derivations. The variable $\ddot{\mathbf{x}}_{\mathbf{C}} \in \mathbb{R}^{\mathrm{k}'}$ denotes the Cartesian translational and rotational accelerations of the controlled points on the robot's body (namely robot's end-effectors and COM). Here k' is the total number of constraint equations introduced by these points. Note that $\mathbf{J}'_{\mathbf{C}}(\mathbf{q}) \in \mathbb{R}^{\mathrm{k}'\times(\mathrm{n}+6)}$ corresponds to all controlled Cartesian points being in contact or moving freely and thus, $\mathbf{J_C}(\mathbf{q})$ is a sub-matrix of $\mathbf{J}'_{\mathbf{C}}(\mathbf{q})$. For the points being in contact and fixed, the corresponding $\ddot{\mathbf{x}}_{\mathbf{C}}$ is zero while for the floating points, this reference acceleration is determined by the second layer of our controller. Thus for walking, k' is always 12 (as we have two feet, each introducing 6 constraint equations). The parameter k (number of contacting points) is either 6 in single support or 12 in double support phase.

Given the Cartesian accelerations ($\ddot{\mathbf{p}}_{\mathbf{com}}$, $\ddot{\mathbf{o}}_{\mathbf{b}}$ and $\ddot{\mathbf{x}}_{\mathbf{C}}$), we use a quadratic program (as formulated in Eq.2) to minimize $\ddot{\mathbf{q}}$, $\boldsymbol{\tau}$ and $\boldsymbol{\lambda}$ under various physical constraints.

$$\begin{aligned}\min_{\ddot{\mathbf{q}},\boldsymbol{\tau}_{\mathbf{j}},\boldsymbol{\lambda},\boldsymbol{\sigma}} \; & \mathbf{V}_{\mathbf{Q}_q}(\ddot{\mathbf{q}} - \ddot{\mathbf{q}}_d) + \mathbf{V}_{\mathbf{Q}_\tau}(\boldsymbol{\tau}) + \mathbf{V}_{\mathbf{Q}_\lambda}(\boldsymbol{\lambda}) + \mathbf{V}_{\mathbf{Q}_\sigma}(\boldsymbol{\sigma}) \\ & \mathbf{M}\ddot{\mathbf{q}} + \mathbf{h} = \boldsymbol{\tau} + \mathbf{J}_{\mathbf{C}}^{\mathrm{T}}\boldsymbol{\lambda} \\ & \ddot{\mathbf{x}}_{\mathbf{C}} = \mathbf{J}'_{\mathbf{c}}\ddot{\mathbf{q}} + \dot{\mathbf{J}}'_{\mathbf{c}}\dot{\mathbf{q}} \\ & \textstyle\sum \boldsymbol{\lambda}_i = m\ddot{\mathbf{p}}_{\mathbf{com}} + \boldsymbol{\sigma} \\ & \mathbf{A}\begin{bmatrix}\boldsymbol{\tau}_{\mathbf{j}}^{\mathrm{T}} & \boldsymbol{\lambda}^{\mathrm{T}}\end{bmatrix}^{\mathrm{T}} \leq \mathbf{B}\end{aligned} \tag{2}$$

Where $\mathbf{V_Q}(\mathbf{v}) = \mathbf{v^T Q v}$, the variable m is robot's mass and the matrices $\mathbf{Q_i}$ are diagonal quadratic costs and $\boldsymbol{\sigma}$ induces a soft constraint on CoM dynamics. We encapsulate coefficients introduced by physical inequality constraints into $\mathbf{A}$ and for conciseness. The joint torques should fall within a certain bound determined by our real robot's actuator specifications. The CoP limits the torque available at each contact, proportional to the normal force (refer to [25] and [24]). We also include the friction polyhedrals at each contact similar to [11]. In Eq.2, $\ddot{\mathbf{q}}_d$ is zero except for the base rotational acceleration which is $\ddot{\mathbf{o}}_{\mathbf{b}}$. In fact, we control the base orientation directly, while the method in [6] controls it by regulation of total angular momentum rate. Note that the equality constraint for $\ddot{\mathbf{p}}_{\mathbf{com}}$ in Eq.2 is a simpler alternative of using CoM Jacobian for relating $\ddot{\mathbf{p}}_{\mathbf{com}}$ to joint accelerations which makes the optimization slower.

With such problem definition, robot's dynamics, feet accelerations and inequality constraints are defined as hard constraints. However, Cartesian accelerations determined by the second layer of our controller are followed by soft constraints with large quadratic costs. If the second layer gives infeasible accelerations in terms of available frictions or torque limits for example, the solution of this QP lies on the margin of the constraints so that they are not violated. A similar approach is taken in [11] where the second equality constraint (contact accelerations) is soft. In their own formulation, this soft constraint has the same effects on infeasible $\ddot{\mathbf{x}}_{\mathbf{C}}$. However in our formulation, the softness on the CoM ($\ddot{\mathbf{p}}_{\mathbf{com}}$) and the base orientation ($\ddot{\mathbf{o}}_{\mathbf{b}}$) deal with infeasible Cartesian accelerations ($\ddot{\mathbf{x}}_{\mathbf{C}}$) as well. We consider larger quadratic costs for the CoM compared to the base orientation. As a result, the Cartesian tracking of feet is more precise with the cost of small variations mainly on base orientation, which happen in case of large perturbations and swing dynamics. Note that this tracking is more important for us compared to the robot's posture, since we want our full robot to match with the LIPM used for planning.

It is worth mentioning that for walking, we keep the upper body joints of the robot fixed, i.e. assuming zero joint accelerations. A PD position controller keeps them fixed with the aim of feed-forward gravity compensation torque extracted from the full dynamics model. The quadratic costs in Eq.2 determine our weightings for torques vs. contact forces and accelerations. Our robot in single support phase is fully actuated, considering 6 degrees of freedom in each leg and the number of constraints. So there exist a unique solution, regarding analysis provided in [1] and [13]. Note however that the robot is under-actuated in

the double support phase and our force distribution weighting between the two feet in this phase follows similar policy described in [25]. We give larger weight to the foot closer to the CoM. Double support phase only happens shortly before starting rhythmic stepping, described in the next section.

Benefiting from the well known QP solver, CVXGEN, we are able to solve the whole problem including all joints in less than $1.2ms$ (6 iterations on average) on a Core i5 1.7GHz machine, coded in C++. With fixed upper body assumption however, there is no point in considering corresponding elements in the mass matrix and the Jacobians, since accelerations are zero. Therefore we avoid defining such sparse matrices and break them into several blocks in CVXGEN which reduce the problem size drastically. These fixed joints are around 50% of all joints in most of humanoid robots including Coman. So we can increase the performance up to around $0.7ms$ in total, still being able to calculate gravity compensation torques for fixed joints. For closed form solutions however, one needs to reformulate the alternative pseudo-inversion formula in [13] to make it efficient. Note that the active-set based QP solver in [11] uses the solution found previously to speed up convergence and has superior average performance, compared to the QP solver of the present work.

In this section we presented how we calculate joint torques, given Cartesian accelerations of controlled points on the robot, i.e. CoM, base orientation and feet. We also described strengths of the proposed problem formulation, tailored to our own objectives we want to achieve in the upper layers of the controller. In next section, we explain how Cartesian accelerations are generated in the second layer.

III. Second layer: Trajectory Pattern generator

In this section, we explain the policy we use to generate gait trajectories as well as the rhythm with which the support leg is changed. In this layer, a state machine with fixed timing switches between left and right support. There is only a short double support phase in the beginning when the robot starts from normal posture and shifts left or right to start rhythmic stepping properly. The input to this layer of our controller is the next footstep position in global coordinates at the end of each swing phase. This layer is responsible for generating Cartesian accelerations of the CoM, base orientation and feet.

As will be described later in the third layer, we assume that the robot follows a simple foot-less LIPM model in left or right support phase which induces weak ankle assumption. In our method, ankles mainly serve as controlling posture of the robot rather than inducing motions on the CoM. This assumption is the key factor in making our blind robot robust against terrain variations. However to enhance the accuracy of the model, we consider the non-flat orientation of the contacting foot in calculating friction polyhedrals or CoP constraints of Eq.2. Considering LIPM, the x and y components of the variable $\ddot{\mathbf{p}}_{\mathbf{com}}$ are determined by:

$$\ddot{\mathbf{p}}_{\mathbf{com}} = \frac{\mathrm{g}}{\mathrm{z}_0}(\mathbf{p}_{\mathbf{com}} - \mathbf{p}_{\mathbf{base}}) \tag{3}$$

Where z_0 is the reference constant CoM height and $\mathbf{p}_{\mathbf{base}}$ is the center location of the stance foot. Note that the z component of $\ddot{\mathbf{p}}_{\mathbf{com}}$ is determined by a PD feedback, tracking constant CoM height reference z_0.

In order to investigate the coupling of x, y and yaw motions, from [9] one can write a full IPM equation of motion with ankles and inertia mass by:

$$\begin{aligned} m\ddot{\mathbf{p}}_{\mathbf{com}} &= \mathbf{f} + m\mathbf{g} \\ \mathbf{J}\dot{\boldsymbol{\omega}} &= \boldsymbol{\tau}_{\mathbf{hip}} - \boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega}) \end{aligned} \tag{4}$$

Where $\mathbf{J}$ is the moment of inertia and $\boldsymbol{\omega}$ is the angular velocity of the body. The moment balance for the IPM based massless leg is:

$$-(\mathbf{p}_{\mathbf{com}} - \mathbf{p}_{\mathbf{base}}) \times \mathbf{f} - \boldsymbol{\tau}_{\mathbf{hip}} + \tau_{\mathbf{ankle}} = \mathbf{0} \tag{5}$$

Substituting $\boldsymbol{\tau}_{hip}$ and $\mathbf{f}$ from Eq.4 to Eq.5, we obtain:

$$\begin{aligned} (\mathbf{p}_{\mathbf{com}} - \mathbf{p}_{\mathbf{base}}) \times (\mathrm{m}\mathbf{g} - \mathrm{m}\ddot{\mathbf{p}}_{\mathbf{com}}) + \\ \tau_{\mathbf{ankle}} = \mathbf{J}\dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\mathbf{J}\boldsymbol{\omega}) \end{aligned} \tag{6}$$

Note that x and y components of $\ddot{\mathbf{p}}_{com}$ are in proportion to those components in $(\mathbf{p}_{\mathbf{com}} - \mathbf{p}_{\mathbf{base}})$, described in Eq.3. So the left-hand-side cross product has no z component. One can easily prove that if $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\omega}}$ are zero, $\boldsymbol{\tau}_{\mathbf{ankle}}$ will also be zero which is essentially the basic ankle-less and inertia-less LIPM used in Eq.3.

For the complex model of Eq.6 however, we should make sure that $\boldsymbol{\tau}_{\mathbf{ankle}}$ will compensate all right-hand-side moments so that our prediction which is based on simple LIPM stays valid. It is difficult to find bounds for $\dot{\boldsymbol{\omega}}$ since $\mathbf{J}$ is time variant and depends on the posture of the robot. Assuming diagonal $\mathbf{J}$ however, one can write the approximate validity condition as:

$$\left| \begin{bmatrix} J_{xx}\dot{\omega_x} + (J_{zz} - J_{yy})\omega_z\omega_y \\ J_{yy}\dot{\omega_y} + (J_{xx} - J_{zz})\omega_x\omega_z \\ J_{zz}\dot{\omega_z} + (J_{yy} - J_{xx})\omega_y\omega_x \end{bmatrix} \right| \leq \begin{bmatrix} w_{foot,y}/2 \\ w_{foot,x}/2 \\ \mu_{rot} \end{bmatrix} |m\mathbf{g}| \tag{7}$$

Where $w_{foot,x}$ and $w_{foot,y}$ are the robot's foot width in x and y directions and μ_{rot} is the rotational friction of the feet with the ground. If x and y components of $\boldsymbol{\omega}$ are negligible as well as non-diagonal elements of $\mathbf{J}$, then large yaw acceleration values will only appear in z component of $\boldsymbol{\tau}_{\mathbf{ankle}}$ which has larger bounds. Since μ_{rot} is much larger than foot width of our robot, (in SI units, 0.5 vs. 0.08 respectively) and J_{zz} is smaller than J_{xx} and J_{yy}, the method can tolerate larger yaw motions compared to pitch and roll. By increasing steering speed up to a certain point, the whole body optimization in the first layer generates torques that completely track reference orientation. Above that, tracking will not be precise and the induced motion will be on the margins of constraints.

So far we discussed how we generate CoM accelerations. At the end of each swing phase, we assume that the base and swing foot are rotated by $\omega_{joy}\Delta t$ around z axis where Δt is the duration of swing phase and ω_{joy} is the reference steering velocity determined by joystick. The final location of the swing foot (i.e. the next footstep location) is also

determined by the third layer. Thus, the second layer generates trajectories between initial and final orientations of the base and also positions and orientations of the swing foot. We use Spherical linear interpolation (Slerp) transition function to generate smooth Cartesian trajectories and use quaternions to avoid singularities of Euler rotations during steering. For the purpose of lifting the swing foot and having enough clearance, we use sinusoids of the form:

$$\mathbf{p_{C,z}}(t) = \frac{z_{cl}}{\frac{1}{2\omega} - \frac{1}{6\omega}}\left(\frac{\sin(\omega t)}{2\omega} - \frac{\sin(3\omega t)}{6\omega}\right), \quad \omega = \frac{\pi}{\Delta t} \tag{8}$$

Where $p_{C,z}(t)$ is the reference height for the swing foot, t is the time counted from the beginning of each phase and z_{cl} is the clearance distance between the foot and the ground, when it is in the apex of foot trajectory arc. Such function ensures zero position, velocity and acceleration of the beginning and the end of the arc. The generated reference trajectories are all tracked by PD controllers in this layer which generate $\ddot{\mathbf{o}}_\mathbf{b}$ and $\ddot{\mathbf{x}}_\mathbf{C}$, given to the first layer together with $\ddot{\mathbf{p}}_\mathbf{com}$ described earlier. Note that alternation of stance legs and the corresponding Jacobians ($\mathbf{J_C(q)}$) in Eq.2 are also based on the pattern of phases produced in this layer. In section V, we will show the performance of such pattern generation policy over uneven terrains where the assumption of flat ground is violated. This assumption is in fact used when keeping CoM height constant and generating arc trajectories of Eq.8.

So far, we have defined the Cartesian trajectory and tracking of feet, CoM and base orientations. In the next section we will describe how the next footstep location is determined by the third layer.

IV. Third layer: Foot-step planner

In this section, we will formulate a Model Predictive Control (MPC) problem [10] to find a stabilizing future plan of footsteps. The inputs to this layer of our controller are sagittal ($v_{joy,x}$), coronal ($v_{joy,y}$) and steering velocities (ω_{joy}) determined by a joystick (assumed to be in robot's coordinate frame).

At this level of the controller, the robot is simplified using a foot-less LIPM. The prismatic actuator in the leg keeps the CoM height always constant. This model allows us to predict the future motion of the robot, assuming weak ankles. Our formulation forms a discrete time model of the robot where CoM position and velocity are the state of the system and footsteps locations serve as inputs. Recall Eq.3 where the CoM acceleration depends on its distance from the stance foot. One can solve this differential equation and obtain the solution in time for x and y components of $\mathbf{p_{com}}$, expressed in Eq.9.

$$\begin{aligned}
\ddot{\mathbf{p}}_\mathbf{com} &= \frac{g}{z_0}(\mathbf{p_{com}} - \mathbf{p_{base}}) \\
\mathbf{p_{com}}(t) &= \mathbf{a}e^{-t/\tau} + \mathbf{b}e^{t/\tau} + \mathbf{p_{base}} \\
\tau &= \sqrt{z_0/g} \\
\mathbf{a} &= 0.5(-\tau\dot{\mathbf{p}}_\mathbf{com}(0) + \mathbf{p_{com}}(0) - \mathbf{p_{base}}) \\
\mathbf{b} &= 0.5(\tau\dot{\mathbf{p}}_\mathbf{com}(0) + \mathbf{p_{com}}(0) - \mathbf{p_{base}})
\end{aligned} \tag{9}$$

We can then predict x and y components of CoM at time Δt using their current value.

$$\begin{aligned}
\mathbf{p_{com}}(\Delta t) = \mathbf{p_{base}}(\tfrac{-1}{2h} + \tfrac{-h}{2} + 1) + \\
+\mathbf{p_{com}}(0)(\tfrac{1}{2h} + \tfrac{h}{2}) + \dot{\mathbf{p}}_\mathbf{com}(0)(\tfrac{-1}{2h} + \tfrac{\tau h}{2})
\end{aligned} \tag{10}$$

Where $h = e^{\frac{\Delta t}{\tau}}$. One can follow the same procedure and obtain x and y components of $\dot{\mathbf{p}}_\mathbf{com}(\Delta t)$. We define the simplified robot's state $\hat{\mathbf{q}}$ as:

$$\hat{\mathbf{q}} = \begin{bmatrix}\mathbf{p_{com,x}} & \mathbf{p_{com,y}} & \dot{\mathbf{p}}_\mathbf{com,x} & \dot{\mathbf{p}}_\mathbf{com,y}\end{bmatrix}^\mathbf{T} \tag{11}$$

So using current state of the CoM ($\hat{\mathbf{q}}(\mathbf{0})$) and the remaining time of current phase $\hat{\Delta}t$, we can express the state of CoM at the end of the current swing phase by:

$$\hat{\mathbf{q}}(\hat{\Delta}t) = \mathbf{A}(\hat{\Delta}t)\hat{\mathbf{q}}(0) + \mathbf{B}(\hat{\Delta}t)\mathbf{p_{base}} \tag{12}$$

Where $\mathbf{p_{base}} \in \mathbb{R}^2$ and $\mathbf{A} \in \mathbb{R}^{4\times4}$ and $\mathbf{B} \in \mathbb{R}^{4\times2}$ are matrices containing all the coefficients of Eq.10. Since $\mathbf{p_{base}}$ and $\hat{\mathbf{q}}(\mathbf{0})$ are already available at each moment, $\hat{\mathbf{q}}(\hat{\Delta}t)$ could be calculated and we call it $\hat{\mathbf{q}}[1]$ in discrete space. Following the same procedure, we can predict the future motion by:

$$\hat{\mathbf{q}}[i+1] = \mathbf{A}\hat{\mathbf{q}}[i] + \mathbf{B}\mathbf{p_{base}}[i], \quad i \geq 1 \tag{13}$$

Assuming fixed phase durations, now $\mathbf{A}$ and $\mathbf{B}$ are functions of Δt and future footsteps are expressed by $\mathbf{p_{base}}[i]$. It is straightforward to show that matrices $\mathbf{A}$ and $\mathbf{B}$ form a controllable system by checking the rank of $\begin{bmatrix}\mathbf{B} & \mathbf{AB}\end{bmatrix}$ ([15]). Eq.13 serves as the basis model used in our MPC to plan future footsteps. The next footstep position given to the second layer of our controller is therefore $\mathbf{p_{base}}[1]$ determined in this layer.

We aim at guiding the future footsteps so that they induce the average velocity vector given by the joystick. To this end, we first define a reference footstep plan. One expects coronal motions of the CoM to be minimal in natural walking. Define the variable $s = 1$ if the robot is in right support or $s = -1$ if it is in left support. Assume also that footsteps normally have distance of $2d$ in coronal plane during walking. We define delta-motion for each step by $\Delta x = 2v_{joy,x}\Delta t$, $\Delta y = 2v_{joy,y}\Delta t$ and $\Delta\theta = \omega_{joy}\Delta t$. If we define the rotation vector $\mathbf{R}(\eta) = \begin{bmatrix}cos(\eta) & sin(\eta)\end{bmatrix}^T$, we can plan a reference discretized path over N horizon by initializing :

$$\begin{aligned}
\theta[0] &= \mathbf{o_{b,z}} \\
\mathbf{p_{des}}[0] &= \mathbf{p_{base}} \\
\mathbf{m}[0] &= \mathbf{p_{des}}[0] + d\ \mathbf{R}(\theta[0] + s\pi/2)
\end{aligned} \tag{14}$$

And defining the steps for $1 \leq i \leq N$ as:

$$\begin{aligned}
\theta[i] &= \theta[i-1] + \Delta\theta \\
\mathbf{m}[i] &= \mathbf{m}[i-1] + \\
&+\Delta x\ \mathbf{R}(\theta[i-1]) + \Delta y\ \mathbf{R}(\theta[i-1] + \pi/2) \\
\mathbf{p_{des}}[i] &= \mathbf{m}[i] + d\ \mathbf{R}(\theta[0] - s(-1)^i\pi/2) \\
\mathbf{v}[i] &= v_{joy,x}\ \mathbf{R}(\theta[i]) + v_{joy,y}\ \mathbf{R}(\theta[i] + \pi/2)
\end{aligned} \tag{15}$$

In these notations, θ represents steering angle, $\mathbf{m}$ represents midpoint of the two feet, $\mathbf{p_{des}}$ represents ideal footsteps plan

and $\mathbf{v}$ represents ideal CoM speed. The goal of the MPC controller in the third layer is to track such sequence of ideal footsteps and CoM desired velocities given by joystick. Therefore, we define a quadratic optimization problem of the form:

$$\begin{aligned}\min_{\hat{\mathbf{q}}[i],\mathbf{p_{base}}[i]} \sum V_{\mathbf{Q_q}}(\hat{\mathbf{q}}_\mathbf{p}[i+1] + \hat{\mathbf{q}}_\mathbf{p}[i] - 2\ \mathbf{m}[i]) + \\ V_{\mathbf{Q_{dq}}}(\Delta\hat{\mathbf{q}}_\mathbf{p}[i+1,i] - \Delta t\ \mathbf{v}[i]) + \\ V_{\mathbf{Q_p}}(\mathbf{p_{base}}[i] - \mathbf{p_{des}}[i]) + \\ V_{\mathbf{Q_{dp}}}(\Delta\mathbf{p_{base}}[i+1,i] - \Delta\mathbf{p_{des}}[i+1,i]) \\ s.t. \\ \hat{\mathbf{q}}[i+1] = \mathbf{A}\hat{\mathbf{q}}[i] + \mathbf{B}\mathbf{p_{base}}[i], \quad i \geq 1 \end{aligned} \tag{16}$$

Where $\mathbf{V_Q(v)} = \mathbf{v^T Q v}$, the matrices $\mathbf{Q_i}$ are quadratic costs, $\hat{\mathbf{q}}_\mathbf{p}[i]$ denotes position components (x and y) of the state vector $\hat{\mathbf{q}}[i]$ and the operator Δ denotes the difference between consecutive $i+1$ and i indexes. Such objective function implicitly minimizes both states and input control variables, together with their derivatives in discrete space. The first two quadratic costs are written over two consecutive CoM positions so that they preserve symmetry of the limit-cycle and that in normal conditions when the robot is moving solely in forward direction, the value of objective function becomes zero. Note that this formulation in fact translates coronal and sagittal speeds which are defined in robot's local frame to global coordinate frame. In practice however, we always give zero reference coronal speed to the planner. A sample planned and reference paths are shown in Fig.2.

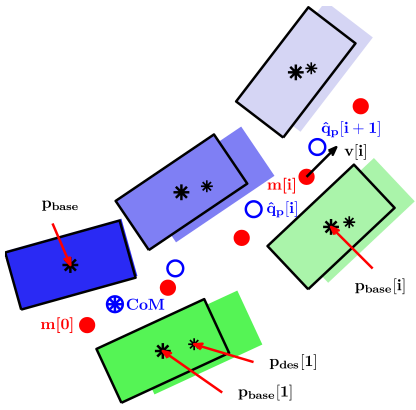

Fig. 2. Sequence of steps planned by MPC during the left support phase. Ideal sequence planned in Eq.15 is shown by filled shapes while the optimal plan extracted from Eq.16 is solid bounded. Note that the difference between the two plans decreases over time.

In practice, we use higher weights for the matrices $\mathbf{Q_{dp}}$ and $\mathbf{Q_{dq}}$ so that the robot does not react aggressively or take very large steps, when it is perturbed with strong pushes. Such weighting policy is more robust for maintaining average speeds, determined by the joystick. One can easily introduce additional constraints to this problem on individual footsteps. For example CoM position should not go further than $|\mu z_0|$ of stance foot position where μ is friction coefficient. In this case, the friction vector will fall out of the friction polyhedral. We have added this constraint, though it is rarely triggered in the range of speeds that our method is able to produce. Note also that the inter-foot distance ($2d$) is chosen to be large and we tune foot tracking gains so that self collision between feet does not happen.

Our MPC controller of this level is implemented by using CVXGEN [12] up to horizon of $N = 5$ footsteps and takes around $0.2ms$ on average to solve. Once the next footstep $\mathbf{p_{base}}[\mathbf{1}]$ is optimized, it is transfered to the second layer of the controller and treated as a target point for the swing leg. Note that if the CoM is perturbed, the corresponding $\mathbf{p_{base}}[\mathbf{1}]$ will adapt in single time-step while in many methods such as [14], the corrective response is delayed more. In the next section, we will briefly introduce the robot we use to validate our method and analyze its performance.

V. RESULTS

In this section we want to design and perform a set of experiments to evaluate the performance of our control approach. The Coman robot [21] which serves as the main platform for simulating this walking method is a child-sized torque controlled robot with electric motors and series-elastic elements in pitch joints. It weights around $30kg$ and has total of 23 degrees of freedom, 6 per leg, 4 per arm and 3 between the pelvis and the torso. For the purpose of this work, we keep upper body of the robot fixed with the policy described before. The robot carries usual joint position and velocity sensors as well as an IMU unit on the pelvis and contact force sensors. In the proposed controller however we do not use contact sensors, accelerometers or any perception. Our method is robust against Coman's series-elastic elements in simulations, where the torque tracking performance is ideal. Although the control loop delay in the real setup is negligible (around $2-3ms$), but actuator dynamics are not yet responsive and fast enough to be suitable for this controller. They are typically slow, introducing about $50ms$ of delay with respect to the given torque profile. As improving the real robot is still ongoing, we show simulation results in this section (the accompanying movie is found at http://infoscience.epfl.ch/record/198512?), using an ODE based simulator software environment.

A. Range of speeds

With the same set of parameters, we are able to cover a wide range of backward and forward speeds between $-0.2m/s$ to $0.4m/s$, shown in Fig.3. However, the tracking of the desired average speeds given by the joystick depends on various parameters. This tracking acceptable for forward speeds, however it deteriorates for backward motions due the ankle asymmetry.

We additionally test the performance of our controller for the tasks combining steering and forward motions. To this end, we perform four tests shown in Fig.4 to find the maximum reference velocity combinations that can be achieved. High forward and steering velocities at the same time is highly challenging. One can clearly see in Fig.2 that the outer foot must take larger steps compared to the inner foot and thus, steering motion limits maximum forward speed.

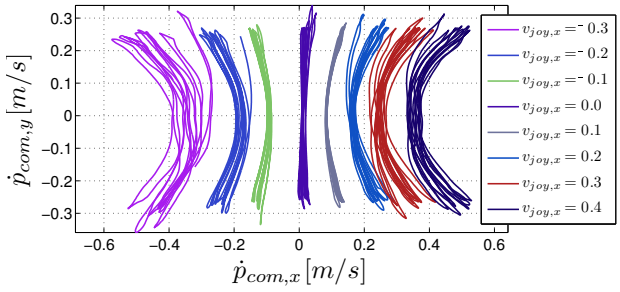

Fig. 3. Resulting limit-cycles of CoM speeds plotted for a wide range of desired forward velocities. The controller tracks the desired velocity, however the repeatability of limit-cycles is affected by foot trajectory tracking performance and contact simulation in high speeds.

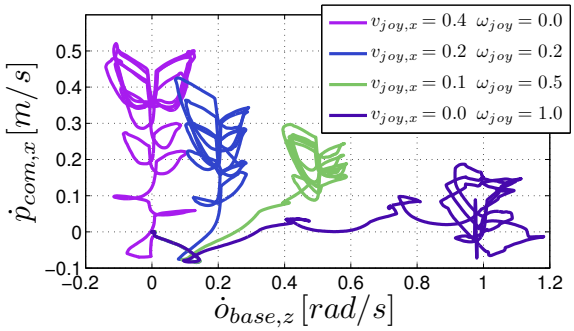

Fig. 4. Evolution of the CoM forward ($\dot{p}_{com,x}$) and the base yaw velocities ($\dot{o}_{base,z}$). In each test, the reference joystick velocity linearly increases up to the indicated value and saturates after about $3s$. The robot starts from rest condition and goes to the final limit-cycle. These four tests show maximum reference speeds that could be given to the robot for stepping stably. We can see that steering limits the forward velocity as the outer foot must take larger steps.

The walking performance of our controller on Coman is comparable in relative terms to real-sized humanoids which are taller and can walk faster. For example the method presented in [5] uses on-line trajectory optimization and can perform up to $1.14m/s$ on the flat ground by Atlas robot, which is two times taller than Coman. A restricting assumption for our method is in fact the full foot contact while Feng in [5] uses toes at lift-offs. Including a phase for the toe lift-off is a possible improvement for our method. So far we characterized the region of stable functionality for the robot without any perturbation. In next parts, we will examine its performance against perturbations, starting from external pushes.

B. External pushes

One important an challenging task for the current humanoids is to stabilizes after being pushed in different directions while performing walking. Here we consider a scenario where the robot is going forward with a moderate speed, i.e. $0.2m/s$. Our impulsive pushes are applied to the torso of the robot, each having $3N.s$ of strength. The robot takes corrective steps to capture the accumulated energy rather than relying on ankle torques and maintaining the CoP inside the limited support polygon. Note that the external pushes perturb the CoM state and the method simultaneously changes the future footstep plan, starting from Eq.12. Since our planning is done per time-step, the robot is able to react as fast as possible. The resulting behavior of the robot is shown in Fig.5-A.

C. Model Errors

In model based methods, one needs to make sure that the internal dynamics model of the robot matches the real robot as much as possible. The aim is to reduce feedback gains which correct these errors and thus, making the robot more compliant. We test our method in some scenarios where the robot carries additional weight in simulations and the controller is blind. The aim is to know how robust the method is against unknown errors. The resulting limit-cycles at the nominal speed of $0.2m/s$ and maximum tolerable errors are shown in Fig.5-B as well as walking on a slope of 15^o degrees. The latter test evaluates the performance if the assumption of flat-ground used in our second and third layers is violated. This slope is the maximum that the method can tolerate while being blind. We observe that the walking is still stable, even though the limit cycles become skewed, asymmetric or enlarged.

D. Perturbations

Beside biased model errors, we test the robustness against delayed communication and sensor noises as well. These scenarios are shown in Fig.5-C together with nominal limit-cycles. We add Gaussian noise to IMU orientation angles and joint positions sensors. At maximum tolerable conditions, the standard deviation (std) of IMU noises could be around 3^o while the std of joint sensors could be 1^o. Note that joint errors accumulated in the kinematic chain of the leg can affect the end-effector tracking performance severely. Also since IMU angles determine the whole orientation of the robot, a small wrong rotation can induce a large motion on the feet. Stronger perturbations lead to self collision of feet or result in taking very large steps which are impossible regarding the friction polyhedrals. In practice, a Kalman filter could be used to filter out these noises. The robot also tolerates delayed communications up to $10ms$. Although it takes large steps in different directions and the motion looks less symmetric and periodic, but it is still able to maintain the dynamic stability.

E. Uneven terrain

Our final test validates the performance over an uneven terrain, i.e. random external geometric errors. We assume flat-ground when planning future footsteps and forming arc trajectories for the swing foot. However they are not often valid since the swing foot might touch the ground earlier or later than the expected moment. The intrinsic compliance of the first layer improves the stability in the sense that minimal bouncing happens at these moments. However we are interested to know the effect of such perturbations in footstep planning and cyclic behavior of the robot. The result of our test is shown in Fig.5-D where terrain variations are around $\pm 5cm$. Although the robot slips at some point due to weakness of the contact modeling in our simulator, it can recover by taking proper steps. Our responsive footstep planner enables

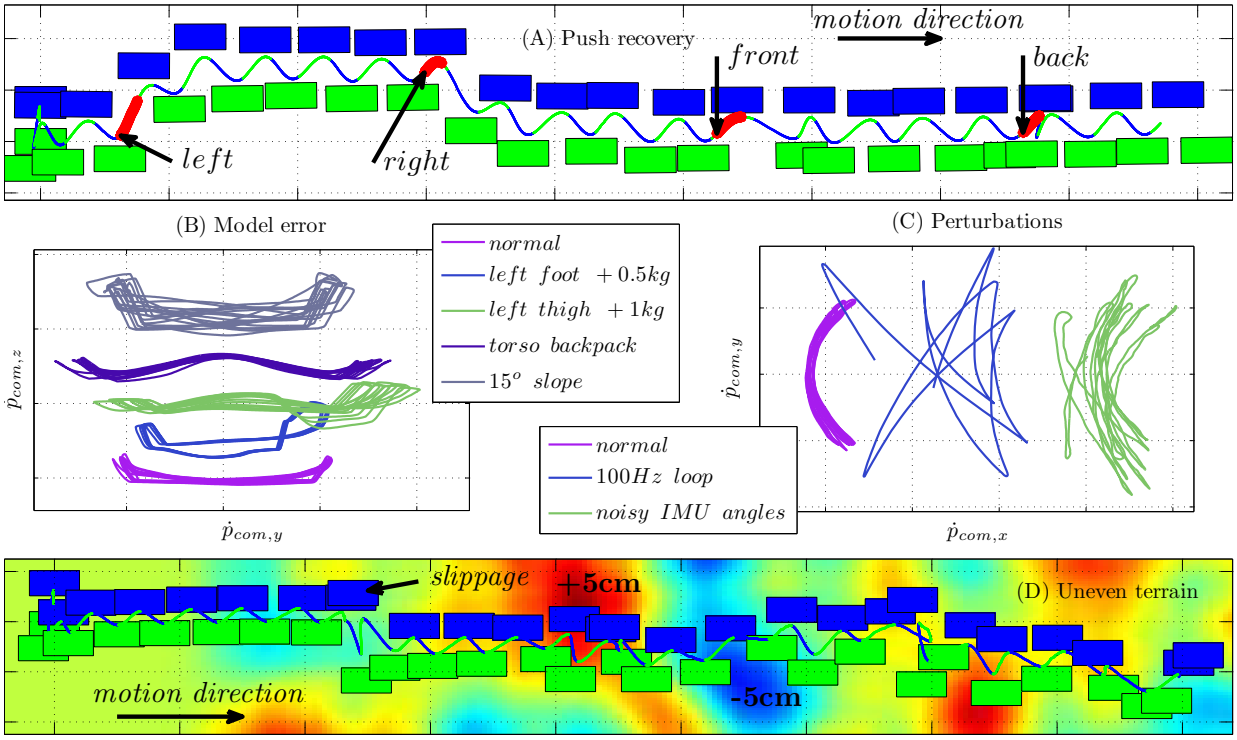

Fig. 5. (A) Push recovery: The robot is subject to $10N$ external pushes applied on its torso to various directions while walking at $0.2m/s$. Each push lasts $0.3s$ and then, the robot recovers by taking proper steps. (B) Model Errors: Extra weight is added to various parts of the robot. In the "torso backpack" scenario, we add $1kg$ to the torso and shift its CoM $-1cm$ back. In the slope scenario, the robot gradually takes larger steps on steeper slopes until it falls. 15^o is the maximum tolerable degree. (C) Perturbations: In noise scenario, a white Gaussian noise of std=3^o is added to the IMU orientation angles as well as a noise of std=1^o to the joint position sensors. The robot also tolerates $10ms$ of delay in communication link, while still being stable. (D) Uneven terrain: Here, terrain variations are $\pm 5cm$ and the robot still walks robustly. Larger variations could also be tolerated, but our simulator has problems in simulating contacts on complex meshes.

the robot to recover from fast perturbations like slippage and external pushes which can happen in the real world as well.

VI. CONCLUSION

In this paper, we presented a hierarchical controller able to perform walking over a wide range of forward and steering speeds. Our controller is based on a dynamics model of the robot without the need to any off-line optimization. Different levels translate the problem first from the joint torques to Cartesian accelerations, then to footstep positions and then only to forward and steering desired speeds, given by a user. The specific formulation of the first layer makes the robot compliant and considers most of the physical constraints on the robot including torque limits, frictions and CoP. Such flexible low level controller plays the key role in being compliant and robust, when the robot is exposed to various kinds of perturbations that make its interaction with the environment unpredictable. The second layer produces smooth Cartesian trajectories for the feet and CoM and tracks them with simple PD controllers. In the third layer, we simplify the robot by a LIPM which allows us to predict future motion of the robot in closed form over multiple steps. We can then plan the next desired step which captures the energy of the robot, while realizing the desired speed determined by the user.

One of the main difference between our method and a large group of works is that we react to perturbations by taking proper footsteps rather than modulating ankle torques. These torques can only help if the CoP is not violated which is limited due to small size of the feet. However one can take larger steps and use the available friction to guide the CoM. This requires planning over hybrid states, i.e. switching of the left and right feet. Possible improvement for our method could be combining its footstep planning with CoP control policies in our second layer where we feed the open-loop LIPM accelerations to the CoM. Another future work is adding constraints to the footstep planner to avoid self collisions as well which makes the problem non-convex. Furthermore, one can consider exact-foot placement which needs back-propagation of constraints over footsteps. Our simple formulation is a flexible basis for adding various constraints that might not be realizable with other non-linear future planing methods. Evident by various kinds of robustness tests and minimal parameter tunings, we can suggest the method for walking of a wide range of torque controlled bipedal robots. The accompanying movie shows more scenarios where the robot performs walking and steering at the same time, as well as movies of the scenarios presented here.

REFERENCES

[1] Farhad Aghili. A unified approach for inverse and direct dynamics of constrained multibody systems based on linear projection operator: applications to control and simulation. *Robotics, IEEE Transactions on*, 21(5):834–849, 2005.

[2] Youngjin Choi, Bum-Jae You, and Sang-Rok 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*, volume 2, pages 1966–1971. IEEE, 2004.

[3] Johannes Englsberger, Christian Ott, Maximo A Roa, A Albu-Schaffer, and Gerhard Hirzinger. Bipedal walking control based on capture point dynamics. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4420–4427. IEEE, 2011.

[4] Salman Faraji, Christopher Atkeson, Pouya Soha, and Auke Ijspeert. Versatile and Robust 3D Walking with a Simulated Humanoid Robot (Atlas): a Model Predictive Control Approach. In *Robotics and Automation (ICRA), IEEE International Conference on*, 2014.

[5] Siyuan Feng, X Xinjilefu, Weiwei Huang, and Christopher G Atkeson. 3D Walking Based on Online Optimization.

[6] Alexander Herzog, Ludovic Righetti, Felix Grimminger, Peter Pastor, and Stefan Schaal. Momentum-based Balance Control for Torque-controlled Humanoids. *arXiv preprint arXiv:1305.2042*, 2013.

[7] Shuuji Kajita and Kazuo 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*, pages 1405–1411. IEEE, 1991.

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

[9] Twan Koolen, Tomas De Boer, John Rebula, Ambarish Goswami, and Jerry 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*, 31(9):1094–1113, 2012.

[10] Mayuresh V Kothare, Venkataramanan Balakrishnan, and Manfred Morari. Robust constrained model predictive control using linear matrix inequalities. *Automatica*, 32 (10):1361–1379, 1996.

[11] Scott Kuindersma, Frank Permenter, and Russ Tedrake. An Efficiently Solvable Quadratic Program for Stabilizing Dynamic Locomotion. *arXiv preprint arXiv:1311.1839*, 2013.

[12] Jacob Mattingley and Stephen Boyd. CVXGEN: a code generator for embedded convex optimization. *Optimization and Engineering*, 13(1):1–27, 2012.

[13] Michael Mistry and Ludovic Righetti. Operational space control of constrained and underactuated systems. *Robotics: Science and systems VII*, pages 225–232, 2012.

[14] Igor Mordatch, Martin De Lasa, and Aaron Hertzmann. Robust physics-based locomotion using low-dimensional planning. *ACM Transactions on Graphics (TOG)*, 29(4): 71, 2010.

[15] Katsuhiko Ogata and Yanjuan Yang. Modern control engineering. 1970.

[16] Michael Posa and Russ Tedrake. Direct trajectory optimization of rigid body dynamical systems through contact. In *Algorithmic Foundations of Robotics X*, pages 527–542. Springer, 2013.

[17] 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*, 26(2):129–143, February 2001.

[18] Jerry Pratt, Twan Koolen, Tomas De Boer, John Rebula, Sebastien Cotton, John Carff, Matthew Johnson, and Peter Neuhaus. Capturability-based analysis and control of legged locomotion, Part 2: Application to M2V2, a lower-body humanoid. *The International Journal of Robotics Research*, 31(10):1117–1133, 2012.

[19] Ludovic Righetti, Jonas Buchli, Michael Mistry, Mrinal Kalakrishnan, and Stefan Schaal. Optimal distribution of contact forces with inverse-dynamics control. *The International Journal of Robotics Research*, 32(3):280–298, 2013.

[20] Philippe Sardain and Guy 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*, 34(5):630–637, 2004.

[21] Emmanouil Spyrakos-Papastavridis, Gustavo A Medrano-Cerda, Nikos G Tsagarakis, Jian S Dai, and Darwin G Caldwell. A push recovery strategy for a passively compliant humanoid robot using decentralized LQR controllers. In *Mechatronics (ICM), 2013 IEEE International Conference on*, pages 464–470. IEEE, 2013.

[22] Benjamin J Stephens and Christopher G Atkeson. Dynamic balance force control for compliant humanoid robots. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pages 1248–1255. IEEE, 2010.

[23] Benjamin J Stephens and Christopher G Atkeson. Push recovery by stepping for humanoid robots with force controlled joints. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 52–59. IEEE, 2010.

[24] Eric C Whitman. *Coordination of Multiple Dynamic Programming Policies for Control of Bipedal Walking*. PhD thesis, Carnegie Mellon University, 2013.

[25] Eric C Whitman and Christopher G Atkeson. Control of instantaneously coupled systems applied to humanoid walking. In *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*, pages 210–217. IEEE, 2010.