Practical considerations in using inverse dynamics on a humanoid robot: torque tracking, sensor fusion and Cartesian control laws

Salman Faraji*, Luca Colasanto* and Auke Jan Ijspeert*

***Abstract*—Although considering dynamics in the control of humanoid robots can improve tracking and compliance in agile tasks, it requires local and global states of the system, precise torque control and proper modeling. In this paper we discuss practical issues to implement inverse dynamics on a torque controlled robot. By modeling electrical actuators off-line, inverting such model and estimating the friction on-line, a high bandwidth torque controller is implemented. In addition, a cascade of optimization problems to fuse all the sensory data coming from IMU, joint encoders and contact force sensors estimate the robot's global state robustly. Our estimation builds the kinematic chain of the legs from the center of pressure which is more robust in case of slight slippage, tilting or rolling of the feet. Thanks to precise and fast torque control, robust state estimation and optimization-based whole body inverse dynamics, the real robot can keep balance with very small stiffness and damping in Cartesian space. It can also recover from strong pushes and perform dexterous tasks. The highly compliant and stable performance is based on pure torque control, without any joint damping or position/velocity tracking.**

I. INTRODUCTION

In legged robotics, kinematic-based algorithms on position-controlled robots have dominated the state of the art for a long time. Even though the formulations for considering dynamics of the robot were already developed in earlier works, it was computationally expensive to consider them for typical complex humanoid robots. But nowadays, with the appearance of advanced computational units, performing dynamic calculations on-line is becoming affordable. Furthermore, humanoids are gradually starting to interact with humans and getting out of the laboratories which require great compliance and robustness. Therefore, recently there are many torque-controlled robots being built and the research is focusing on control methods that provide robustness, compliance and preciseness. The wish to perform more complex and dynamical tasks has motivated researchers to think of task-space motions to reduce complexity of the high-level planning problems. Therefore, a precise controller that converts Cartesian tasks to actuator inputs is crucial in the loop to ensure precision and feasibility of the tasks regarding physical constraints.

Although in position-controlled robots, contact force readings have provided a partial observation of the dynamics of the robot, in torque-controlled robots we additionally have sensing and control over the torque in all individual joints. Respecting inequality constraints such as torque and contact friction limits as well as keeping the Center of Pressure (CoP) inside each support polygon could not be easily addressed in classical formulations like [1]. Stephens and Atkeson [2] formulated the problem into a two staged optimization. At first they optimized contact forces and moments with respect to the Center of Mass (CoM) acceleration and the total momentum. Contacts' friction and CoP limitations were also considered in this stage. Then they found joint torques using more conventional pseudo-inverse methods like [1]. Herzog [3] followed similar approach with a hierarchical formulation that executed the desired tasks with different priorities. In our recent works [4], [5] we combined all the stages together in a single optimization process. Given desired Cartesian accelerations for feet, hands, CoM and the torso (tasks), we find joint torques by optimizing all the unknown variables in a single quadratic programming problem subject to various equality and inequality constraints. We have successfully tested this layer combined with a footstep planner to generate walking on the Atlas [4] and Coman [5] robots in simulation. Inverse dynamics helps making the robot compliant, precise and robust against various sources of noise and errors.

In this paper, we discuss how inverse dynamics is interfaced with our real Coman robot [6] to perform balancing and other Cartesian tasks. While this article solves practical problems for a specific robot, Coman, we think that the methods presented here could help improving whole-body torque control on other robots such as Sarcos [3] and Atlas [7]. The control chain consists of robot's state estimation, Cartesian task planner, inverse dynamics and actuator torque control. In our previous simulations [4], [5], position encoding and torque tracking performance were ideal. The robot's global state was computed with simple kinematics and the output torques were directly applied to the joints. However on the real robot, joint encoders have low resolution, control delay is considerable and torque tracking is slow, implemented by a PI loop. The IMU is broadcasting data with slower rates and due to magnetic interferences, the yaw angle we get from IMU is unreliable. Our goal in the present work is to propose estimation and actuation methods that improve the interface between the inverse dynamics layer and the hardware.

The contribution of the present work is twofold. In torque tracking, we propose a friction observer that acts like a feedback term in conjunction with inverse of the motor's model as feed-forward term. Such architecture improves bandwidth, latency and also transparency of the joint when commanding

*Biorobotics Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland

978-1-4799-9994-1/15/$31.00

zero torque. In other relevant works, fast current PID loops [8] and torque PID loops [8]–[11] compensate disturbances, sometimes only considering internal actuator dynamics [9] or only the friction observation [10]. The proposed architecture however combines the two on a voltage-controlled motor with a novel observer design. By modeling the motor resistance and back electromagnetic effects [12], we find a feed-forward term that essentially replaces the additional current loop [8].

In state estimation, a two-stage quadratic programming method is proposed. We fuse IMU data, leg kinematics and contact forces together in order to estimate the inertial-frame position and velocity of the robot along with the yaw angle. In case of slight rolling/tilting of the feet which happens frequently in very dynamic tasks, the Center of Pressure (CoP) lies on the border. Although the CoP moves all the time, we assume that the foot's surface at the CoP has no relative translational motion with respect to the ground. Therefore at each instance of time, the chain of the leg can start from the CoP point in the inertial frame. Such general concept improves the precision and stability while we do not make any restricting assumption on foot/terrain orientation like [13]. Our second contribution compared to relevant works is to estimate the CoP in each contact and the global state together. Our formulation is similar to Kalman filtering with fixed covariances, however other methods proposed in [7], [13]–[15] linearize the model to update the covariance matrices for statistical optimality. Our method is computationally faster, but requires proper off-line tuning. Moreover, we assume that leg kinematics (positions and velocities) are perfect like [13], [14] while Xinjilefu in [7], [15] puts such assumption on positions only and uses the full model of the robot to filter velocities. We consider such improvement for future work as the case is different due to existence of series springs in Coman.

The paper starts with describing our torque controllers. We continue in the third section by formulating a hierarchy of optimization problems that fuse the sensory data to estimate the global state of the robot. In the fourth section we introduce our inverse dynamics formulation and Cartesian controllers suited for dexterous demonstrations discussed in the results section.

II. Torque Tracking

The baseline PI torque controller in Coman does not have the bandwidth required for our tasks. Therefore we identify motor parameters off-line and invert the resulting model to find a feed-forward voltage. We also need to estimate the friction which is considerable in Coman due to high gear ratios. In literature, friction identification and compensation is studied broadly in position/velocity/torque controlled robots [16]–[18]. There are various statical or dynamical friction models used in different robots or experimental setups. In a position/velocity control paradigm, since the desired direction of motion and its velocity are known, it is easy to compensate the Coulomb friction. Such compensation improves the position tracking and reduces the effort made by feedbacks [16]. More complex models such as LuGre [19] are also identified and studied in [20] by exploiting a set of observers to estimate the parameters on-line.

In torque controlled motors however, less complicated models are considered such as hyperbolic tangent [21] or Coulomb-viscous [18]. In general, since the desired direction of motion is unknown in a torque controller, it is not easy to compensate the Coulomb friction. Therefore other works in torque controlled joints either observe the friction with some dynamics and latency [10] or make a fast PID loop to compensate the torque error [8], [9].

In Coman, there are two encoders before and after the spring and a torque sensor before the spring. Due to low resolution of post-spring encoders and therefore modeling difficulty, we assume negligible transient drive in springs. In this section we describe our procedure to identify the motor's electromagnetic properties like [16]–[18] and then we formulate an inverse control law which generates proper actuator input (voltage) to realize the desired torque in the output (Fig.1).

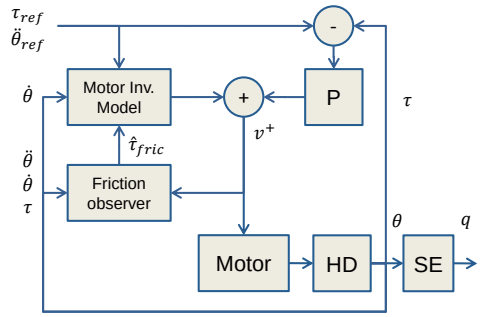

Fig. 1: A typical joint in Coman consists of an electrical motor, a Harmonic Drive (HD) and an in-series spring (SE). After identification, we model the relation between motor voltage v and the torque τ after the harmonic drive. Such model is used to estimate the friction and produce feed-forward voltage added to a simple proportional feedback (P).

A. Model identification

To identify motor properties, we track two sets of reference trajectories (sinusoids and trapezoids to explore motor dynamics and friction properties respectively) with a simple PD controller of low gains that produces the motor voltage. After recording velocities (by finite differentiation) and output torques, we setup a least square optimization problem which fits the following model to the signals:

$$\begin{aligned} \frac{v - K_e N \dot{\theta}}{R_m} k_t N &= \tau_{fric} + J_m N^2 \ddot{\theta} + \tau = \tau_{mot} \qquad (1) \\ \tau_{fric} &= \tau_{co}\ sign(\dot{\theta}) + \tau_v \dot{\theta} \end{aligned}$$

Where v is the voltage of the motor, θ is the joint angle before the spring, τ is the load torque and τ_{fric} corresponds to all friction torques. The constant k_e is back EMF coefficient, k_t is torque constant, N is harmonic drive ratio, R_m is motor resistance and J_m is total inertia of the motor and the harmonic drive. We use a simple combination of Coulomb (τ_{co}) and viscous (τ_v) frictions inspired from the type of data we observe. Adding stiction or Stribeck effects or using more

advanced dynamic models like Dahl or LuGre [19] turned out having no considerable improvement in our setup.

The goal of our identification is to adjust motor parameters k_t, k_e, R_m, J_m we already have from the data-sheet and identify friction coefficients τ_{co} and τ_v. Since Coman has joint torque sensors, we can perform identification for each joint individually unlike [18]. We assume equal torque and back EMF constants and also N is known from the data-sheet. Fig.2 shows a typical trained model, plotted as a friction-velocity profile. The red curve shows our friction model while the black curve corresponds to the characteristic of friction estimated from the measured torque and the motor model. In practice, to explore the whole range of output torques and high currents, we apply different external loads to the joint during the experiment. According to the data-sheet, the motors in Coman have very low magnetic reluctance and therefore, we observed negligible improvement by adding reluctance term during the identification process.

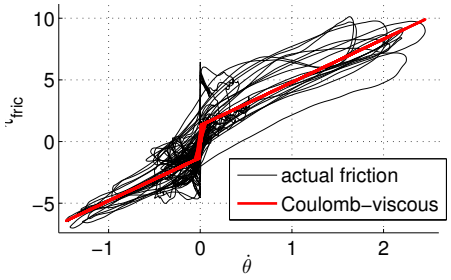

Fig. 2: The friction in shoulder roll joint and the approximate coulomb-viscous curve. Note that the actual curve is obtained via estimating motor parameters.

B. Control law

After optimizing the model (1) for each joint individually, we invert it to find the feed-forward voltage terms v^+ to be applied at the next time-step:

$$v^+ = k_e N\dot{\theta} + \frac{R_m}{k_t N}(\hat{\tau}_{fric} + J_m\ddot{\theta}_{ref} + \tau_{ref}) \tag{2}$$

Where $\hat{\tau}_{fric}$ stands for the estimated friction. Although the optimized set of parameters better describe the data, the difference is still considerable in Fig.2. Unlike using the trained parameters [21], we try to estimate the friction based on the torque measurements. knowing the voltage applied in the previous time-step v^- and the resulting motor velocity $\dot{\theta}$, motor acceleration $\ddot{\theta}$ and the output torque τ_{out}, we can estimate the friction by:

$$\begin{aligned} v^- &= k_e N\dot{\theta} + \frac{R_m}{k_t N}(\hat{\tau}_{fric} + J_m\ddot{\theta} + \tau) \\ \hat{\tau}_{fric} &= \hat{\tau}_{co} + \tau_v\dot{\theta} \end{aligned} \tag{3}$$

Where the viscous friction is assumed to follow the model. Subtracting (3) from (2) yields:

$$v^+ - v^- = \frac{R_m}{k_t N}(J_m(\ddot{\theta}_{ref} - \ddot{\theta}) + (\tau_{ref} - \tau)) \tag{4}$$

Which basically means the voltage of the motor is integrating the torque and acceleration errors, ensuring convergence to zero. Finally we add a simple proportional gain to correct the frequency response of the system (Fig.1). In practice we multiply the estimated Coulomb friction $\hat{\tau}_{co}$ by a factor close to one to make the system more stable against un-modeled effects. The controller can show remarkable zero-torque transparency and track very fast torque profiles precisely. It can also be easily applied on other robots which have torque sensing capabilities, unless a model is fitted to the actuators. The performance will be further demonstrated in section V and compared to the momentum based estimation proposed in [10]. In next section, we describe another crucial component required for agile motions which is the global state estimation.

III. Robot State Estimation

Using inverse dynamics requires precise and robust estimation of the robot's state in the global coordinates frame (i.e. 6 DoF of the pelvis and their derivatives). These values should be indeed consistent with contact constraints. One can easily assume that the geometric center of the foot is fixed and use this constraint to determine the global base (pelvis) position [5]. The IMU mounted on the pelvis can also be used to determine orientations. However this simplistic method on the real robot results in slippage when the CoP goes to the borders like Fig.3. Small motions on the foot easily result in a considerable perturbation on the whole inverse dynamics. A common method to cancel out such vibrations is to add an additional term on the reference torque to damp joint velocities. However, these unwanted forces sometimes cause more slippage and unacceptable tracking. Therefore we are interested in estimating the robot's global state robustly and precisely in order to avoid joint damping and consequently reach faster motions.

We assume that the feet have no relative translational motion at the CoP point, but they might have relative rotational motion. Such assumption makes velocity estimation more precise. However other methods either consider fixed foot position [7], [13]–[15] or at most, add fixed foot orientation constraint [13]. Possible slippage, tilting or rolling is considered as disturbance of the process in the aforementioned methods. However, the fact that the reference foot position/orientation can slowly change and adapt through Kalman filtering of [13], [14] is not yet implemented in the present work.

Coman is equipped with a precise Microstrain IMU on the pelvis which computes orientations, angular velocities and linear accelerations. Coman also carries 6-axis force-torque sensors on each foot. We do not have considerable backlash on joints and we differentiate high resolution pre-spring encoders to obtain joint velocities, assuming stiff springs. In fact post-spring encoders are used to build kinematics, but the signal does not have enough resolution for differentiation. Similar works exist in the literature which perform steady state Kalman filtering [15], [22], considering full-body dynamics to estimate joint velocities. In this work however we only consider estimation of the global states and leave the joint states and springs for future work.

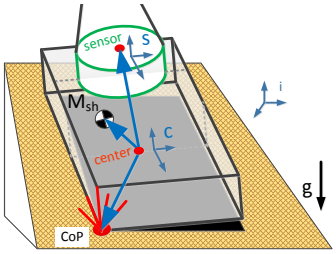

Fig. 3: An arbitrary posture of the foot over an inclined surface. It is very usual on the real robot that the inverse dynamics algorithm uses boundaries of the support region to provide balance. This perturbs the state estimation in case of tilting or rolling of the foot. In this figure, the frame $\{\mathbf{c}\}$ refers to the geometric center of the bottom surface, $\{\mathbf{s}\}$ refers to the frame attached to the 6D force sensor and $\{\mathbf{i}\}$ refers to the inertial frame. Note that M_{sh} corresponds to the total mass under the sensor.

We setup a staged problem to estimate base position $\mathbf{x_b^i}$, velocity $\dot{\mathbf{x}}_\mathbf{b}^\mathbf{i}$ and yaw angle ϕ_z. The superscript $\{\mathbf{i}\}$ indicates that the variable is expressed in the inertial frame. Inputs are joint positions $\mathbf{q_j}$, joint velocities $\dot{\theta}_\mathbf{j}$, base frame angular rate $\boldsymbol{\omega}^\mathbf{b}$, base linear acceleration $\ddot{\mathbf{x}}_\mathbf{imu}$, base pitch ϕ_y and roll ϕ_x angles, contact forces $\mathbf{f_s}$ and moments $\mathbf{m_s}$ which are expressed in the sensor frames $\{\mathbf{s}\}$. We rely on the internal filtering of the IMU to cancel biases and temperature effects. The first stage of our method estimates the base velocity and CoP in each contact while the second stage estimates the base position and yaw angle.

A. Stage 1: base velocity estimation

In this stage we use the base linear acceleration and contact forces to determine the CoP in the feet as well as the base velocity in the local frame. For hand tips we actually assume point contact with fixed CoP, though the robot has small spheres that can slightly roll. Assuming small rotational motions, we get the static contact moment equilibrium from [23] for the feet and transfer it from the inertial frame $\{\mathbf{i}\}$ to the contact's frame $\{\mathbf{c}\}$ (Fig.3):

$$\begin{aligned}\mathbf{m_{CoP}} &= \mathbf{S}(\mathbf{f_s}+\mathbf{f_{sh}})\mathbf{x_{CoP}}+\mathbf{S}(\mathbf{f_s})\mathbf{x_s}+\mathbf{m_s}+\mathbf{S}(\mathbf{f_{sh}})\mathbf{x_{sh}}\\ \mathbf{f_{sh}} &= \mathrm{M_{sh}}\mathbf{R_c^i}^{-1}\mathbf{g}\end{aligned} \tag{5}$$

Where $\mathrm{M_{sh}}$ refers to the weight of the parts under the force sensor, $\mathbf{x_{sh}}$ refers to their center of mass, $\mathbf{g}$ is gravity, $\mathbf{x_s}$ denotes sensor frame location, $\{\mathbf{s}\}$ is skew symmetric matrix and $\mathbf{f}$ and $\mathbf{m}$ correspond to forces and moments respectively. The variable $\mathbf{x_{CoP}}$ is the CoP relative position inside the contact polygon. Note that all $\mathbf{x}$, $\mathbf{m}$ and $\mathbf{f}$ variables are expressed in $\{\mathbf{c}\}$ frame. In our formulations, $\mathbf{R_a^b}$ is generally the rotation matrix of the frame $\{\mathbf{a}\}$ expressed in $\mathbf{b}$. We use superscripts in our notations to show the reference frame. In (5), we want to be free of the global rotation whereas it appears in $\mathbf{f_{sh}}$. However since the gravity $\mathbf{g}$ is along the z axis, any yaw rotation does not change $\mathbf{g}$. Therefore we only need IMU roll and pitch angles in each time-step.

Now assuming that the two surfaces have no relative translational motion at the CoP point, we can relate the translational velocity of the contact point (which is zero) to the base velocity using kinematic relations:

$$-\dot{\mathbf{y}} = (\mathbf{R_c^b}\mathbf{S}(\boldsymbol{\omega}^\mathbf{c})+\mathbf{S}(\boldsymbol{\omega}^\mathbf{b})\mathbf{R_c^b})\mathbf{x_{CoP}}+\dot{\mathbf{x}}_\mathbf{c}^\mathbf{b}+\mathbf{S}(\boldsymbol{\omega}^\mathbf{b})\mathbf{x_c^b} \tag{6}$$

Where we have defined $\dot{\mathbf{y}} = \mathbf{R_b^i}^{-1}\dot{\mathbf{x}}_\mathbf{b}$, the superscript $\mathbf{b}$ refers to the base (pelvis) frame, $\boldsymbol{\omega}^\mathbf{b}$ and $\boldsymbol{\omega}^\mathbf{c}$ are local angular velocities of frames $\{\mathbf{b}\}$ and $\mathbf{c}$, $\mathbf{x_c^b}$ refers to foot's center-point position (Fig.3) and $\mathbf{x_{CoP}}$ is the same variable in (5), expressed in $\{\mathbf{c}\}$ frame. One can also estimate the new velocity of the base by:

$$\dot{\mathbf{y}}_\mathbf{ref} = \dot{\mathbf{y}}(t-\triangle t)+\ddot{\mathbf{y}}_{imu}\triangle t \tag{7}$$

Where $\triangle t$ is the duration of a time-step. We can now combine (5), (6) and (7) and form a constrained quadratic optimization as:

$$\begin{aligned}\min_{\dot{\mathbf{y}},\mathbf{x_{CoP}}} &\sum \mathbf{V}_{\mathbf{Q}_\delta}(\delta)+\sum \mathbf{V}_{\mathbf{Q}_{\mathbf{m_{CoP}}}}(\mathbf{m_{CoP}})+\mathbf{V}_{\mathbf{Q_y}}(\dot{\mathbf{y}}-\dot{\mathbf{y}}_\mathbf{ref})\\ &\mathbf{m_{CoP}} = \mathbf{A}\mathbf{x_{CoP}}+\mathbf{B}\\ &-\dot{\mathbf{y}}+\delta = \mathbf{C}\mathbf{x_{CoP}}+\mathbf{D}\\ &\mathbf{x_{CoP}} \in \text{support polygon surface}\end{aligned} \tag{8}$$

Here we define $\mathbf{V_Q}(\psi) = \psi^\mathbf{T}\mathbf{Q}\psi$ and each active contact (either foot or hand) appears in the optimization with its own constraints. We have compacted (5) and (6) and avoided to index different contacts for simplicity.

If we consider $\mathbf{Q}$ matrices to be the inverse of error's covariance matrix for each equation, the optimization in (8) becomes equivalent to Kalman filtering, but with inequality constraints. However here we assume fixed covariances and in fact we tune them to get the desired performance. We possibly lose optimality in terms of statistical properties, but avoid large calculations of optimal Kalman gains and still get the desired performance. Regarding CoP calculations, the costs for tangential elements of $\mathbf{m_{CoP}^c}$ and $\mathbf{m_{CoP}^{ci}}$ are set to large values while the cost for z components is set to zero. Adding slack variables δ to (6) means filtering kinematic data which is crucially needed due to encoder limitations in Coman. (8) in fact filters all the sensory data together with minimal setup where the internal process model is assumed to be a floating IMU. In future work we would consider whole body dynamics and integrate the full model like [7] in order to estimate the joint velocities as well.

B. Stage 2: base position and yaw angle estimation

So far we have determined the base linear velocity and we already know the angular rate from the IMU, both expressed in the base frame. In this stage we are going to solve another optimization problem to find the yaw angle ϕ_z and the position $\mathbf{x_b^i}$ of the base, considering the CoP found in the previous stage and available roll and pitch angles from IMU. We express the orientation of the base by:

$$\mathbf{R_b^i} = \mathbf{R}(\triangle\phi_z)\mathbf{R}(\phi_z(t-\triangle t))\mathbf{R}_{\phi_x\phi_y} = \mathbf{R}(\triangle\phi_z)\mathbf{R}' \tag{9}$$

Where $\mathbf{R}_{\phi_\mathbf{x}\phi_\mathbf{y}}$ denotes the rotation matrix of the pitch and roll angles we get from the IMU and $\mathbf{R}(\triangle\phi_\mathbf{z})$ is the delta

rotation matrix around z axis. Now we can write the CoP in the inertial frame as:

$$\mathbf{x^i_{CoP,ref}} = \mathbf{x^i_c}(0) + \mathbf{R^i_c}(0)\mathbf{x^c_{CoP}} = \mathbf{x^i_b} + \mathbf{R}(\triangle\phi_\mathbf{z})\mathbf{R}'\mathbf{x^b_{CoP}} \quad (10)$$

Where $\mathbf{x^i_c}(0)$ and $\mathbf{R^i_c}(0)$ represent the initial foot center frame. The variable $\mathbf{x^i_{CoP,ref}}$ is therefore the reference center position plus the relative displacement of the CoP at the current time-step. We can also use the velocities and accelerations to approximate the $\mathbf{x^i_b}$ as:

$$\mathbf{x^i_b}(\mathrm{t}) = \mathbf{x^i_b}(t-\triangle t) + \mathbf{R}(\triangle\phi_z)\mathbf{R}'(\dot{\mathbf{y}}\triangle t + \ddot{\mathbf{y}}_{imu}\frac{\triangle t^2}{2}) \quad (11)$$

Defining $\triangle\phi_{z,ref} = \boldsymbol{\omega}^\mathbf{b}_z\triangle t$ where $\boldsymbol{\omega}^\mathbf{b}_z$ is the IMU angular rate, we can combine (11) and (10) in an optimization problem:

$$\begin{aligned}\min_{\mathbf{x^i_b},\triangle\phi_z} \sum Q_\delta\boldsymbol{\delta}^2 + Q_\phi(1-cos(\triangle\phi_{z,ref}-\triangle\phi_z))\\ \mathbf{x^i_b} + \boldsymbol{\delta} = \mathbf{R}(\triangle\phi_z)\mathbf{A} + \mathbf{B}\end{aligned} \quad (12)$$

(11) is always present in the optimization (in compact form) while different active contacts can participate with their own constraints and costs, again in a similar constraint form. We avoid indexing for the sake of simplicity. The optimization in this stage is in fact nonlinear and we solve it via Matlab's symbolic engine by setting the derivatives of the objective function to zero. It turns out that if we choose equal costs in each diagonal cost matrix, we can find $\mathbf{x^i_b}$ linearly depending on sine and cosine of $\triangle\phi_z$. Therefore one can easily replace it in the equations and find multiple solutions for $\triangle\phi_z$ where we choose the closest one to zero and thus calculate $\mathbf{x^i_b}$ as well. Therefore the full state of the pelvis can be calculated with these two stages which take 0.2ms on a modern computer.

This method can easily disable different contacts by setting their costs to zero. Using inequality constraints, we limit the CoP and ensure stability of the algorithm in case of slight tilting or rolling of the foot. Introduction of slack variables enable us to filter kinematics and contact forces together in a minimal setup and improve the robustness against noise. In future work we will find a policy to re-estimate the reference CoP position in (10) like [14] and update it during locomotion/slippage. Note that the proposed method can be easily applied on most of the humanoid robots with IMU, joint encoders and contact force sensors. In addition to torque tracking and state estimation, there are few remarks on task controllers and parameter tuning of inverse dynamics layer that we explain in next section.

IV. Controllers

In this section we briefly introduce the control algorithm we use to perform the tasks described in section V. This controller has three sub-layers itself: inverse dynamics, Cartesian controllers and trajectory generation.

A. Inverse dynamics layer

We use the general inverse dynamics framework of our previous work [5]. In brief, given the Cartesian accelerations $\ddot{\mathbf{x}}_\mathbf{e}$ (translational and rotational), we use a quadratic program as formulated in (13) to minimize joint accelerations $\ddot{\mathbf{q}}$, joint torques $\boldsymbol{\tau}$ and contact forces $\boldsymbol{\lambda}$ under various physical constraints.

$$\begin{aligned}\min_{\ddot{\mathbf{q}},\boldsymbol{\lambda},\boldsymbol{\tau},\boldsymbol{\sigma}} \mathbf{V}_{\mathbf{Q}_q}(\ddot{\mathbf{q}}) + \mathbf{V}_{\mathbf{Q}_\lambda}(\boldsymbol{\lambda}) + \mathbf{V}_{\mathbf{Q}_\tau}(\boldsymbol{\tau}) + \mathbf{V}_{\mathbf{Q}_\sigma}(\boldsymbol{\sigma})\\ \mathbf{M}\ddot{\mathbf{q}} + \mathbf{h} = \boldsymbol{\tau} + \mathbf{J}^\mathrm{T}_\mathbf{c}\boldsymbol{\lambda}\\ \boldsymbol{\sigma} + \ddot{\mathbf{x}}_\mathbf{e} = \mathbf{J_e}\ddot{\mathbf{q}} + \dot{\mathbf{J}}_\mathbf{e}\dot{\mathbf{q}}\\ \mathbf{A}\begin{bmatrix}\boldsymbol{\tau}^\mathrm{T} & \boldsymbol{\lambda}^\mathrm{T}\end{bmatrix}^\mathrm{T} \leq \mathbf{B}\end{aligned} \quad (13)$$

Where the matrices $\mathbf{Q_i}$ are diagonal quadratic costs and the variables $\boldsymbol{\sigma}$ induce a soft constraint on Cartesian tasks. $\mathbf{M}$ is the mass matrix, $\mathbf{h}$ represents all gravitational, centripetal and Coriolis forces, $\mathbf{J_c}$ is the Jacobian of contact points and $\mathbf{J_e}$ is the Jacobian of end-effectors (contacts, CoM and torso orientation). The matrices $\mathbf{A},\mathbf{B}$ represent physical inequality constraints such as torque limits, friction polyhedrals and CoP limitations. In practice, we perform stiff position control on the joints that are not included in the motion. The implementation of (13) in CVXGEN [24] enables us to remove sparsities from the mass matrix and Jacobians which enhance the performance up to 4 times. We reach 0.7ms to 1.1ms on a modern computer depending on the number of contacts involved (2 and 4 respectively).

Our choice of cost gains are uniform diagonal matrices $\mathbf{Q}_q = 10^{-2}$, $\mathbf{Q}_\lambda = 10^{-2}$, $\mathbf{Q}_\tau = 10$, $\mathbf{Q}_\sigma = 10^4$ for floating and $\mathbf{Q}_\sigma = 10^7$ for contacting points. These costs provide stable performance on the real robot and ensure preciseness of the desired tasks. Fixed contacts have more importance than floating tasks. We penalize joint torques more than contact forces and accelerations to provide smoother torque profiles, but indeed floating tasks have higher priority than torques. Adding soft constraints on the Cartesian tasks is beneficial specially when CoM falls outside the support polygon. If for any reason, the upper layer provides infeasible accelerations, the soft constraints ensure satisfaction of physical inequality constraints. For instance, in case of keeping balance and being pushed extremely so that the CoM falls outside the support polygon, the robot still keeps full contact at the feet without tilting or rolling. Therefore we reject invalid Cartesian accelerations by sacrificing the precision in normal conditions unlike the prioritized hierarchy of [3]. Note that the proposed method can be easily applied on torque controlled robots unless other control layers are present as well as a full model of the robot.

B. Cartesian control layer

Our task space formulation requires inertial-frame accelerations for the hands, feet, CoM and torso orientation. For Cartesian translational tasks $\ddot{\mathbf{x}}^\mathbf{i}_\mathbf{ref}$, we use a simple PD law as:

$$\ddot{\mathbf{x}}^\mathbf{i}_\mathbf{e} = \ddot{\mathbf{x}}^\mathbf{i}_\mathbf{ref} + \mathbf{K}(\mathbf{x}^\mathbf{i}_\mathbf{ref} - \mathbf{x}^\mathbf{i}_\mathbf{e}) + \mathbf{D}(\dot{\mathbf{x}}^\mathbf{i}_\mathbf{ref} - \dot{\mathbf{x}}^\mathbf{i}_\mathbf{e}) \quad (14)$$

For angular tasks, the formulation we use is inspired from the work in [25], but including accelerations. Imagine the frame $\{\mathbf{e}\}$ is attached to an end-effector with orientation $\mathbf{R}^{\mathbf{i}}_{\mathbf{e}}$, angular velocity $\boldsymbol{\omega}^{\mathbf{i}}_{\mathbf{e}}$ and angular acceleration $\boldsymbol{\alpha}^{\mathbf{i}}_{\mathbf{e}}$ expressed in the inertial frame $\{\mathbf{i}\}$. The desired target orientation, angular velocity and accelerations are $\mathbf{R}^{\mathbf{i}}_{\mathbf{ref}}$, $\boldsymbol{\omega}^{\mathbf{i}}_{\mathbf{ref}}$ and $\boldsymbol{\alpha}^{\mathbf{i}}_{\mathbf{ref}}$. Viewed from the desired reference frame, we want to minimize the error observed in this frame. Therefore we define the PD law as:

$$\boldsymbol{\alpha}^{\mathbf{ref}}_{\mathbf{e}} = -\mathbf{K}\boldsymbol{\theta}^{\mathbf{ref}}_{\mathbf{e}} - \mathbf{D}\boldsymbol{\omega}^{\mathbf{ref}}_{\mathbf{e}} \tag{15}$$

Where $\boldsymbol{\theta}^{\mathbf{ref}}_{\mathbf{e}}$ is the angles coming from $\boldsymbol{R}^{\mathbf{ref}}_{\mathbf{e}}$. Using standard kinematics, the angular acceleration of the frame $\{\mathbf{e}\}$ in the inertial frame is then calculated by:

$$\boldsymbol{\alpha}^{\mathbf{i}}_{\mathbf{e}} = \mathbf{R}^{\mathbf{i}}_{\mathbf{ref}}[\mathbf{S}(\boldsymbol{\omega}^{\mathbf{ref}} + 2\boldsymbol{\omega}^{\mathbf{ref}}_{\mathbf{e}})\boldsymbol{\omega}^{\mathbf{ref}}_{\mathbf{e}} + \boldsymbol{\alpha}^{\mathbf{ref}}_{\mathbf{e}}] + \boldsymbol{\alpha}^{\mathbf{i}}_{\mathbf{ref}} \tag{16}$$

Here in fact the Coriolis motion is considered when converting accelerations between different frames. Replacing Cartesian PD controllers by more advanced policies like MPC in future works can improve the tracking performance.

C. Cartesian trajectory generation layer

In experiments discussed in next section, we produce various agile motions that require proper trajectory generation from an initial to a final configuration. We use smooth exponential functions to generate 3-times differentiable trajectories as well as Quaternion SLERP functions used in [25], derived once more to obtain the desired accelerations. We skip the details for the purpose of conciseness.

V. RESULTS AND DISCUSSIONS

In previous sections we discussed the three-layer controller. A robust and fast method in each of them is crucial for performing agile tasks discussed in this section.

A. Torque tracking

First of all, we characterize the performance of torque tracking block shown in Fig.4(a,b,c). A high level PID controller together with inverse dynamics layer that produce feed-forward torques is used to track sinusoidal trajectories on single joints. We do not choose large-amplitude signals to avoid reaching current limits since a considerable portion of the generated torque is used to accelerate the rotor. The double derivative of the desired trajectory $\ddot{\theta}_{ref}$ is also given to the torque tracker to compensate the torque required for the rotor.

Fig.4(a) shows the Bode diagram of force tracking transfer functions. One can see the poor performance of the basic initial PI controller while using feed-forward and feedback terms improve the transfer function. The remaining phase lag (which is around 15ms) is due to the fact that torque tracking is now implemented on an external PC with delayed communication. In future we transfer it to individual motor controller boards to reduce the delay considerably. Fig.4(b) corresponds to the momentum based friction observer proposed in [10]. Even-though its tracking is satisfactory for low torques, when the user holds the joint, the controller shows considerable delay in tracking higher torque profiles which comes from the filtering nature of this observer. However in Fig.4(c), our proposed estimator shows lower latency in torque tracking.

The advantage of our method over PID controllers [8] or disturbance observers [10] is that our controller compensates internal dynamics of the actuator while those methods try to resolve these issues with faster loop frequencies. Also with such estimation of the friction from the previous time-step, there is no need to close the torque loop with high PID gains.

B. State estimation and control

Next, we will demonstrate the performance of our state estimation and inverse dynamics formulation over some multi-joint tasks. The performance is quantified for two tasks of rotation around vertical (z) and lateral (y) axes while other challenging scenarios are demonstrated in the accompanied video. We frequently use the simple kinematics based filter in previous works [5] which starts the chain from the center of the feet and does not use the IMU data.

In scenario 1, the robot genuflects quickly around the hip. Fig.4(d) shows the desired and actual trajectories of the torso pitch angle, showing a fast and stable motion. In fact this demonstration is challenging due to limited support polygons. The remaining steady state error is due to the fact that we use low PD gains in the Cartesian space and our CAD model does not match the real robot perfectly. However the robot is compliant which can be observed in the accompanied video. One can easily integrate the pitch error to converge in steady state, however we want to ensure that most of the control policy is generated by feed-forward terms which indicates proper modeling of the system.

In scenario 2, we have depicted the performance of our estimator in Fig.4(e) where the robot rotates around the vertical axis at 1Hz. The base yaw angle is plotted using the previous simple kinematics approach versus our new staged optimizer. In simulations, the basic kinematic approach is stable enough to perform all the tasks. But on the real robot, we have considerable noise that is rejected successfully by the proposed staged filtering.

In scenario 3, we have repeated the same motion at 0.5Hz (Fig.4(f)), but this time replacing the new state estimator in the loop by the simple one to observe the resulting performance. In scenario 4, using back the new state estimator, we have added damping to all the joints in torque tracking level in order to see if the vibration could be canceled. Such unwanted damping improves stability, but similar to scenario 3, it spoils the tracking performance. Comparing Fig.4(e) and Fig.4(f), one can conclude that tracking is worse in fast motions, but we have less vibrations because the joints change direction more rapidly and physically, there is less time for the joint to build up static friction (refer to [19]). Note that all the 12 degrees of freedom in the lower body are active, having small start/stop motions in most. Overcoming the Coulomb friction is still not perfect in torque tracking layer which results in such vibrations. Further improvement could be possibly reducing the previously mentioned delay in the torque tracking loop.

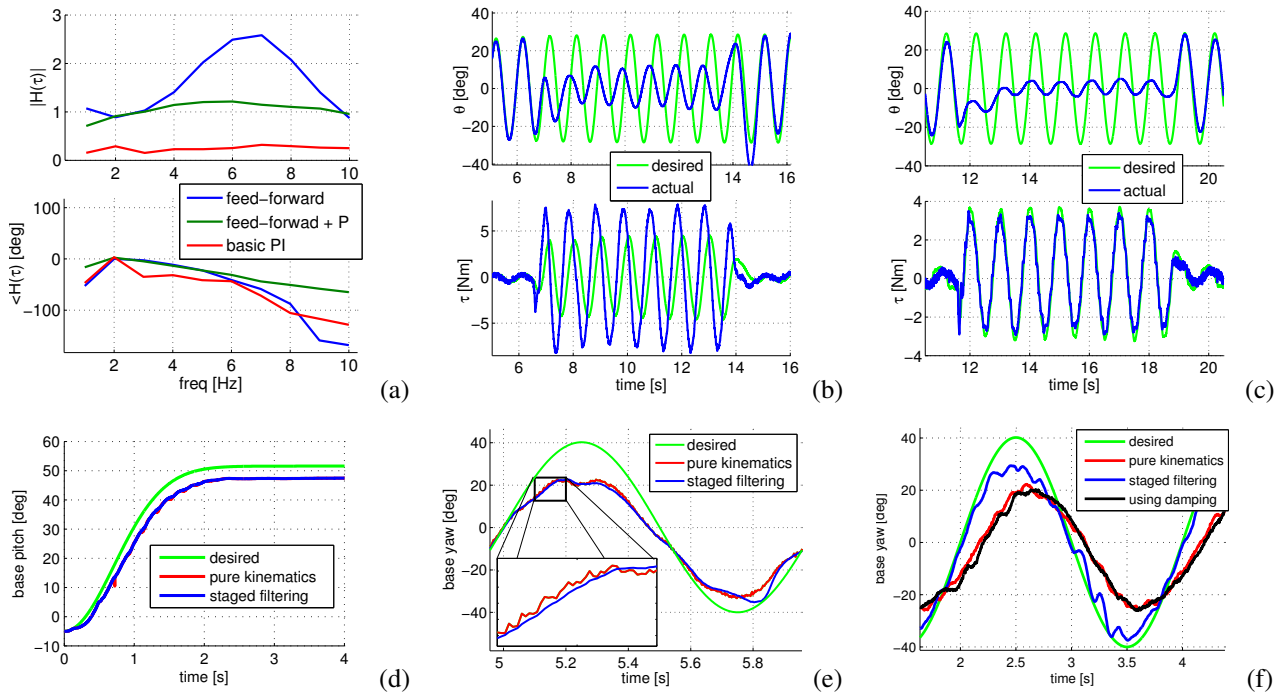

Fig. 4: **(a)** The Bode diagram of force tracking transfer function for basic PI, feed-forward only and looped controllers. Note that the feed-forward term already closes an integrating loop in its friction observer. We close a second loop by a simple P gain to flatten the bode diagram.
(b) Tracking performance of the estimator proposed in [10] at 1Hz in the presence of an external holding force. The controller shows considerable delay in higher torques and overshoots when the external force is removed.
(c) Tracking performance of our estimator at 1Hz in the presence of an external holding force. The controller successfully follows the desired torque (before the spring). Note that the PID gains of the high level loop (over position) are small so that the user holding the joint can almost stop it. In (b) and (c) graphs, reference position trajectories are the same, though desired torque profiles are different due to the high level PID controller.
(d) Scenario 1: rapid genuflection of Fig.6(f). The full demonstration is shown in the accompanied video where the robot complies with external pushes.
(e) Scenario 2: rotation around the vertical axis at 1Hz. The main source of noise is in fact the precision of post-spring encoders. Our new filtering method successfully rejects noise and spikes and provides a smooth estimation compared to the previous kinematic method used in [5].
(f) Scenario 3,4: repeating the same vertical rotation scenario at 0.5Hz, but in control we use once the novel state estimation and once the basic kinematics method. We also repeat the same task with the novel state estimator, but adding some joint damping in torque tracking level.

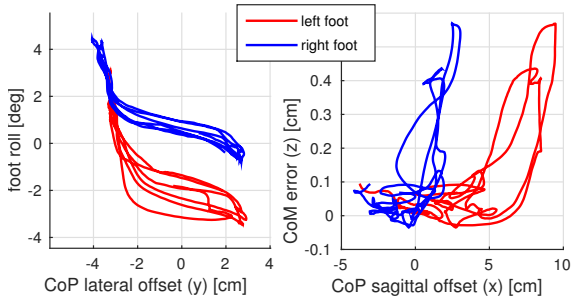

Fig. 5: Left: When the CoP goes to the borders, there is a slight tilting or rolling where the geometric center point of the foot is not fixed in the inertial frame anymore. Right: demonstrates the difference between estimated CoM vertical position by our staged filtering and the basic kinematics based filter.

In order to characterize the effectiveness of starting the chain from the CoP, we have plotted few more signals in Fig.5 corresponding to scenario 2. As shown in the accompanied video, the feet slightly tilt and slip on the ground. This indicates that the fast motion of this scenario is on the margin of physical constraints. Such whole body rotation at 1Hz is challenging for a human as well. Since there is a thin flexible silicon layer at the bottom of each foot in Coman, one can expect slight tilting or rolling if the CoP goes to the borders as shown in Fig.5 left. Now the assumption of fixed center-foot position which is used by the basic kinematics estimator of [5] is not true anymore. Therefore, the two new and old estimation algorithms can systematically differ during such dynamical tasks (Fig.5 right). Our new filtering however starts the chain from the CoP and efficiently uses the kinematic constraint while in similar works [7], [13], this information might be ignored by considering large covariances.

Finally, we have demonstrated the full body compliance of the robot over different tasks. In fact the main goal of modeling the motor, performing torque control and using inverse dynamics is to find proper feed-forward terms and reduce the effect of feedbacks. This approach decreases the stiffness of the robot and makes it compliant. Fig.6 shows few snapshots of extreme compliance capabilities without violating physical constraints like contact frictions. Full demonstrations are shown in the accompanied video.

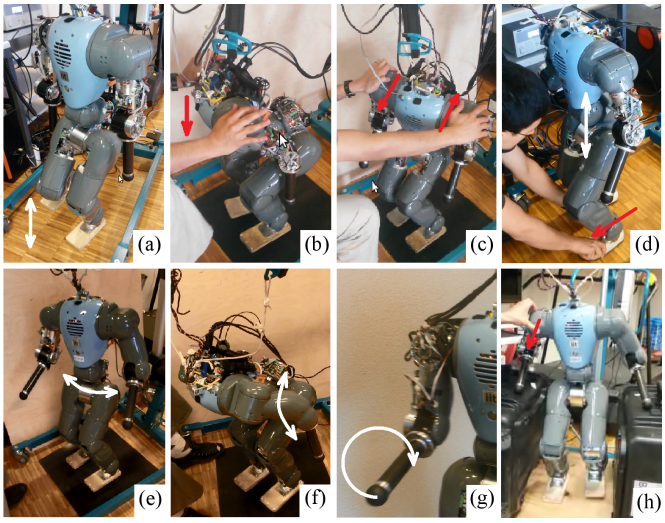

Fig. 6: Different scenarios using improved torque control and whole body inverse dynamics. White arrows show robot's motion while red arrows show external push. **(a)** balance on single foot, **(b,c)** withstanding extreme pushes downward and rotational, **(d)** squatting motion while the user slightly perturbs the foot location, **(e)** rotation around vertical axis, **(f)** genuflection around y axis, **(g)** circular motion with hands, **(h)** full body balance on three contacts while withstanding an external push. Please refer to the accompanied video to see each demonstration.

VI. CONCLUSION

In the present paper, we have shown the gap between our previous work in simulations and challenges we faced on the real robot. We have decomposed the controller into three estimation, control and actuation layers and discussed our strategies to improve the performance. The first novelty of this work lies in the combination of a new friction observer and motor inverse dynamics to improve the bandwidth and precision of the torque tracking. The second novelty of this work lies in the staged optimization problems that act like a Kalman filter to find base position and orientation from IMU, kinematics and contact force sensor data. We build the kinematic chain starting from the CoP and effectively handle very dynamic motions where the feet might slightly tilt or roll. The third novelty finally refers to the fast demonstrations and extreme compliance of the robot which is superior compared to similar approaches [2], [3], [13]. Future work would be including the modeling of springs to improve torque tracking and state estimation, improving communication delay and adapting reference foot location on-line during state estimation.

VII. ACKNOWLEDGMENTS

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

REFERENCES

[1] M. Mistry and L. Righetti, "Operational space control of constrained and underactuated systems," *Robotics: Science and systems VII*, pp. 225–232, 2012.

[2] 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*, pp. 1248–1255.

[3] A. Herzog, L. Righetti, F. Grimminger, P. Pastor, and S. Schaal, "Balancing experiments on a torque-controlled humanoid with hierarchical inverse dynamics," in *IEEE International Conference on Intelligent Robotics Systems*, 2014.

[4] S. Faraji, P. Soha, and A. 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] S. Faraji, S. Pouya, and A. Ijspeert, "Robust and agile 3d biped walking with steering capability using a footstep predictive approach," in *Robotics Science and Systems (RSS)*, 2014.

[6] N. G. Tsagarakis, S. Morfey, G. M. Cerda, L. Zhibin, and D. G. Caldwell, "Compliant humanoid coman: Optimal joint stiffness tuning for modal frequency control," in *Robotics and Automation (ICRA), 2013 IEEE International Conference on*. IEEE, 2013, pp. 673–678.

[7] X. Xinjilefu, S. Feng, and C. G. Atkeson, "Dynamic state estimation using quadratic programming," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 989–994.

[8] M. Hutter, C. D. Remy, M. A. Hoepflinger, and R. Siegwart, "High compliant series elastic actuation for the robotic leg scarleth," in *Proc. of the International Conference on Climbing and Walking Robots (CLAWAR)*, no. EPFL-CONF-175826, 2011.

[9] T. Boaventura, C. Semini, J. Buchli, M. Frigerio, M. Focchi, and D. G. Caldwell, "Dynamic torque control of a hydraulic quadruped robot," in *Robotics and Automation (ICRA), 2012 IEEE International Conference on*. IEEE, 2012, pp. 1889–1894.

[10] L. Le Tien, A. Albu-Schaeffer, A. De Luca, and G. Hirzinger, "Friction observer and compensation for control of robots with joint torque measurement," in *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*. IEEE, 2008, pp. 3789–3795.

[11] M. Mosadeghzad, G. A. Medrano-Cerda, N. Tsagarakis, and D. G. Caldwell, "Impedance control with inner pi torque loop: Disturbance attenuation and impedance emulation," in *Robotics and Biomimetics (ROBIO), International Conference on*. IEEE, 2013, pp. 1497–1502.

[12] P. Pillay and R. Krishnan, "Modeling, simulation, and analysis of permanent-magnet motor drives. ii. the brushless dc motor drive," *Industry Applications, IEEE Transactions on*, vol. 25, no. 2, pp. 274–279, 1989.

[13] N. Rotella, M. Bloesch, L. Righetti, and S. Schaal, "State estimation for a humanoid robot," in *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*. IEEE, 2014, pp. 952–958.

[14] M. Bloesch, M. Hutter, M. A. Hoepflinger, S. Leutenegger, C. Gehring, C. D. Remy, and R. Siegwart, "State estimation for legged robots-consistent fusion of leg kinematics and IMU," *Robotics*, p. 17, 2013.

[15] X. Xinjilefu, S. Feng, W. Huang, and C. Atkeson, "Decoupled state estimation for humanoids using full-body dynamics," in *IEEE Intl. Conf. on Robotics and Automation (ICRA), Hong Kong, China*, 2014.

[16] C. T. Johnson and R. D. Lorenz, "Experimental identification of friction and its compensation in precise, position controlled mechanisms," *Industry Applications, IEEE Transactions on*, vol. 28, no. 6, pp. 1392–1398, 1992.

[17] R. Kelly, J. Llamas, and R. Campa, "A measurement procedure for viscous and coulomb friction," *Instrumentation and Measurement, IEEE Transactions on*, vol. 49, no. 4, pp. 857–861, 2000.

[18] S. Traversaro, A. Del Prete, R. Muradore, L. Natale, and F. Nori, "Inertial parameter identification including friction and motor dynamics," *arXiv preprint arXiv:1410.4410*, 2014.

[19] H. Olsson, K. J. Åström, C. Canudas de Wit, M. Gäfvert, and P. Lischinsky, "Friction models and friction compensation," *European journal of control*, vol. 4, no. 3, pp. 176–195, 1998.

[20] Y. Tan and I. Kanellakopoulos, "Adaptive nonlinear friction compensation with parametric uncertainties," in *American Control Conference, 1999. Proceedings of the 1999*, vol. 4. IEEE, 1999, pp. 2511–2515.

[21] S.-m. Hur, S.-K. Kim, Y. Oh, and S.-R. Oh, "Joint torque servo of a high friction robot manipulator based on time-delay control with feedforward friction compensation," in *RO-MAN, 2012 IEEE*, pp. 37–42.

[22] M. F. Fallon, M. Antone, N. Roy, and S. Teller, "Drift-free humanoid state estimation fusing kinematic, inertial and lidar sensing," *Humanoid Robots (Humanoids)*, 2014.

[23] L. Sentis, J. Park, and O. Khatib, "Compliant control of multicontact and center-of-mass behaviors in humanoid robots," *Robotics, IEEE Transactions on*, vol. 26, no. 3, pp. 483–501, 2010.

[24] J. Mattingley and S. Boyd, "CVXGEN: a code generator for embedded convex optimization," *Optimization and Engineering*, vol. 13, no. 1, pp. 1–27, 2012.
[25] B. J. Bacon, "Quaternion-based control architecture for determining controllability/maneuverability limits," in *AIAA Guidance, Navigation, and Control Conference*, 2012.