Online Center of Mass Estimation for a Humanoid Wheeled Inverted Pendulum Robot

Munzir Zafar*, Akash Patel*, Bogdan Vlahov, Nathaniel Glaser, Sergio Aguilera, Seth Hutchinson

***Abstract*— We present a novel application of robust control and online learning for the balancing of a n Degree of Freedom (DoF), Wheeled Inverted Pendulum (WIP) humanoid robot. Our technique condenses the inaccuracies of a mass model into a Center of Mass (CoM) error, balances despite this error, and uses online learning to update the mass model for a better CoM estimate. Using a simulated model of our robot, we meta-learn a set of excitory joint poses that makes our gradient descent algorithm quickly converge to an accurate CoM estimate. This simulated pipeline executes in a fully online fashion, using active disturbance rejection to address the mass errors that result from a steadily evolving mass model. Experiments were performed on a 19 DoF WIP, in which we manually acquired the data for the learned set of poses and show that the mass model produced by a gradient descent produces a CoM estimate that improves overall control and efficiency. This work contributes to a greater corpus of whole body control on the Golem Krang humanoid robot.**

I. Introduction

Combining the maneuverability of a two-wheeled mobile platform and the dexterity of robotic arms, humanoid Wheeled Inverted Pendulum (WIP) robots present novel challenges to the robotics research community. Humanoid robot stabilization is fundamental to keep the robot safe and for the robot to accomplish higher-level objectives. Furthermore, keeping a WIP, such as the one presented in Fig. 1, balanced is a fundamental task in which the controller needs to be constantly working and thus should be energy efficient [1]. Stabilization is usually accomplished through the control of a simplified two Degree of Freedom (DoF) model which summarizes the Center of Mass (CoM) of all the joints into one as shown in Fig 2. This simplification is usually done for both WIP humanoid robots [2]–[4], as well as for legged humanoids [5]–[7]. All frameworks presented to stabilize WIP robots consider that the mass and CoM for each of the joints is accurately known [8]–[10] to compute the simplified two DoF WIP model. However, the mass and real location of the CoM is difficult to obtain, as robot systems can be complex and they might change throughout time. The discrepancy in the parameters of the robot affects the controller's performance, diminishing the robot's dexterity and increasing the power consumption.

*Joint First Authors

Munzir Zafar, Akash Patel, Bogdan Vlahov, Nathaniel Glaser, Sergio Aguilera and Seth Hutchinson are with the Institute of Robotics and Intelligent Machines at the Georgia Institute of Technology, Atlanta, GA, 30332, USA. email: mzafar7@gatech.edu, apatel435@gatech.edu, bvlahov3@gatech.edu, nglaser@gatech.edu, sfaguile@gatech.edu, seth@gatech.edu

Fig. 1. Full Body of our WIP Humanoid.

Regarding these uncertainties in the model, one common control methodology uses the Modern Control Paradigm [11] which focuses on the modeling of the system as, $\ddot{y} = f(y, \dot{y}, w, t) + bu$, where y is the position output and w is an unknown input force. Once the system is modeled, it is approximated to a linear, time-invariant and disturbance-free model, to design a control law. This approach relies on the model approximation $\bar{f}(\dot{y}, y)$ to be "close enough" to the real model in the neighborhood of the operation point. In [11] and later in [3], Extended State Observers (ESOs) are used to estimate the modeled uncertainties and improve the control of the systems. The approach used collapses all the uncertainties and external forces under one element which is later eliminated through feedback control. From an online learning approach, commonly used models rely on the knowledge and accuracy of the CoM [12]–[14]. Very few have worked on model parameter estimation such as [15], [16], but focus more on the estimation of external parameters such as terrain coefficient or external forces than on the robot itself. Finally, recent research involving mobile manipulators has focused on the use of Active Disturbance Rejection Control (ADRC) [17]–[19] to control systems which use external uncertainties to conduct feedback control.

Our approach improves our model parameter estimation using the knowledge of the ESO through online learning. The goal of this framework is to create models that are improved upon by real-world systems and data. Given a model of our system, we want to improve the values of the parameters by measuring the disturbances of the system when it is not subject to external forces. Then, as the robot changes its joints position, we are able to update our parameters in an online fashion. To accomplish this task, we propose the following methodology. Given an initial estimation of the parameters of our model β_0, we use ADRC [11] to

estimate the error between the parameters estimated CoM and the real one for different joint configurations. This error is used to update our knowledge of the model parameters through gradient descent. We show that this methodology works, but it might take numerous positions to converge. Thus, we propose a meta-learning algorithm to find the poses which induce the largest gradient step for gradient descent. The main contributions of this works are: *i)* a novel use of the ESO and ADRC to estimate the error in the model parameters; *ii)* an online learning algorithm to update and improve our model parameters; *iii)* a meta-learning framework to improve the speed and accuracy of our learning algorithm; *iv)* and preliminary results on a real robot with 19 DoF that show the improvement of the system's performance.

This paper is organized as follows. Section II presents the WIP robot and the methodology, as well as discusses the learning, meta-learning, and ADRC techniques. Sections III and IV describe and present the different simulations and experiments. Finally, section V presents the conclusions of our work.

II. Methodology

The goal of the proposed approach is to improve the CoM estimate of a WIP Humanoid. A WIP Humanoid is a highly redundant manipulator mounted on a differential wheeled drive able to dynamically balance itself in an inverted pendulum configuration (Fig 2). A good estimate of the CoM is important for any approach to control dynamically balancing humanoids. This is because the balancing task requires the CoM's ground projection to always lie in the support polygon. The support polygon of a WIP is a rectangle on width equal to the distance between the wheels and a small length given by the compression of the wheels against the ground. This support polygon is very thin, hence is important to decreasing the room for errors in CoM estimates compared to, say, bipedal humanoids where support polygons are much larger.

Let us define frame 0 as the frame where the origin is located at the midpoint between the wheels with its x-axis always along the heading direction and z-axis always vertical. We are interested in the coordinates of the CoM of the body in this frame. Specifically, we want the x-coordinate of body CoM in this frame to be zero in order to balance the robot. Homogeneous coordinates of body CoM in frame 0 are given by

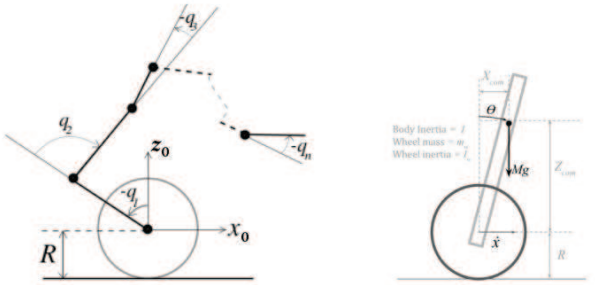
$$X_{com}(q) = \begin{bmatrix} x_{com}(q) \\ y_{com}(q) \\ z_{com}(q) \\ 1 \end{bmatrix} = \frac{\sum_i^L m_i X_i^0(q)}{\sum_i^L m_i} = \frac{\sum_i^L m_i T_i^0(q) X_i^i}{\sum_i^L m_i}$$

where we are interested in the x component of the CoM

$$x_{com}(q) = \phi(q)^\top \beta \tag{1}$$

and the variables are described in Table I.

β is the set of unknown parameters comprising mass and mass times CoM of individual links in the body. This



(a) (a) (b)

Fig. 2. Full Body of a typical WIP Humanoid with n links (a) 2D Simplified Model (b)

choice of parameters is such that the parameters appear linearly in the model. Treating CoM values independently from masses would make the model quadratic in parameters. Improving the estimate of the body's CoM entails improving our knowledge of β. One way to achieve this is to disassemble the robot into individual links and perform physical measurements for mass and CoM of each link. This is tedious and hence undesirable. However, the fact that the CoM model is linear in β allows us to use linear regression or gradient descent to improve our model parameter estimates. We choose gradient descent because of its ability to enforce physical consistency constraints. For one, it converges to β values in the neighborhood of the initial β which are more likely to be physically consistent as opposed to linear regression which might learn solutions that fit the data but are physically nonsensical. Secondly, constraints such as total body mass can be explicitly enforced in the learning process through the use of Lagrange multipliers. The details of this appear in Section II-A.

These learning techniques rely on our ability to collect data for poses q and corresponding values of outputs $x_{com}(q)$. The simplest way to collect this data is to make use of the fact that in the ideal case, $x_{com}(q) = 0$ when the robot is in a balanced state. Assuming that all joints in the body shown in Fig. 1b can be locked at a specific pose $\{ q_2, ..., q_L \}$, there exists a position for q_1 (the base link) that can balance the robot. We can collect data offline by manually moving q_1 such that the robot is in a balanced state. However, this is again tedious; performing the same job online would avoid this labor. To this end, we utilize ADRC [3] to balance

TABLE I

System variables.

Variable	Description
L	number of links in the body
q	$\begin{bmatrix} q_1 & ... & q_L \end{bmatrix}^\top$ position of all joints in the body
m_i	mass of link i
$X_i^0(q)$	is CoM of link i expressed in frame 0
X_i^i	$\begin{bmatrix} x_i & y_i & z_i & 1 \end{bmatrix}^\top$ local CoM of local frame i
$T_i^0(q)$	transformation from frame i to frame 0
β	$\begin{bmatrix} m_1 X_1^{1\top} & ... & m_L X_L^{L\top} \end{bmatrix}^\top \in \mathcal{R}^{4L}$
$\phi(q)$	$\begin{bmatrix} \phi_1(q) & ... & \phi_{4L}(q) \end{bmatrix}^\top$ feature vector of known geometric functions of q

the robot despite a bad estimate of body CoM, the details of which appear in Section II-C. One may ask: Why the need to improve the CoM model if there already exists a controller that is able to stabilize the robot despite a bad CoM estimate? The answer to this is twofold: Firstly, ADRC achieves balancing but is inefficient, i.e. it takes more time and aggressive control inputs to stabilize a bad estimate of CoM. Secondly, ADRC works only when controlling a single rigid link on wheels which is the case when body joints are locked. If however the joints are unlocked, more complex controllers are needed that rely on an accurate estimate of the CoM.

We have so far discussed how to obtain the value of $x_{com}(q)$ at any give pose q. It is important to determine what poses at which we should collect this data. This is because with a highly redundant system, the configuration space is too large and relying on arbitrary poses may make the learning process inefficient and time-consuming. We choose poses such that every next pose causes the largest average gradient descent step over a large set of randomly chosen erroneous β estimates. This is discussed in Section II-A.

A. Learning Algorithm

For the learning algorithm, we make use of gradient descent. The objective function to be minimized is determined based on the fact the x-component of CoM should be zero in a balanced pose. In order to make the cost function locally convex with respect to β, we aim to minimize the square of the x-component of CoM.

$$\begin{aligned} J(\beta) &= \tfrac{1}{2}\left[x_{com}(q;\beta)\right]^2 \\ &= \tfrac{1}{2}\beta^\top \phi(q)\phi(q)^\top \beta \end{aligned} \tag{2}$$

where we have made use of the definition of x_{com} in (1).

The gradient with respect to β will therefore be

$$\nabla_\beta J(\beta) = \phi(q)\phi(q)^\top \beta \tag{3}$$

The update step will be

$$\beta_{t+1} \leftarrow \beta_t - \eta \nabla_\beta J(\beta_t) \tag{4}$$

where η is the step-size, which is a hand-tuned parameter. We begin with an initial estimate of β. As data for the new balanced pose q is collected, we make use of the gradient update step in (4) to improve β estimates. This is repeated until $\phi(q)^\top \beta$ consistently drops below a threshold x_{tol} for a few iterations.

B. Meta-Learning Algorithm

We also deal with the problem of determining a training set of poses that makes the learning process efficient or less time-consuming. For robots with many Degrees of Freedom, the configuration space is huge and choosing an arbitrary set of training poses will likely make the learning inefficient. We determine this training set offline, only using the model in simulation, using the algorithm presented in Algorithm 1. The algorithm requires a large pool of randomly generated balanced and safe poses $\bar{q} \in \mathcal{R}^{n_{DOF} \times n_{poses}}$. A balanced pose is one where a "real" robot (*i.e.*, with β values we pretend to be real) is balanced. A safe pose is one where the robot does not collide with itself or the ground, and the joint values are within their physical limits. We precompute the numerical values of the feature vector $\phi(q)$ evaluated at each pose in $\bar{q}$ and store them in $\Phi \in \mathcal{R}^{dim(\beta) \times n_{poses}}$. The algorithm also requires a set of randomly generated erroneous β vectors: $\bar{\beta} \in \mathcal{R}^{dim(\beta) \times n_\beta}$. This is done by choosing values of β vectors that cause x_{com} estimate errors in estimating the "real" robot's CoM to be of the same order as is observed in the physical system. The key step in Algorithm 1 is step

Algorithm 1 Pose Filtering

Input: Set of randomly generated safe & balanced poses: $\bar{q} \in \mathbb{R}^{n_{DOF} \times n_{poses}}$,
Set of $\phi(q)$ evaluated at each given pose: $\Phi \in \mathbb{R}^{dim(\beta) \times n_{poses}}$,
Set of randomly generated erroneous βs: $\bar{\beta} \in \mathbb{R}^{dim(\beta) \times n_\beta}$

Output: Filtered set of poses: $\widetilde{q}$

1: **repeat**
2: $i^* \leftarrow \underset{i \in \{1,\ldots,n_{poses}\}}{\text{argmax}} \sum_k^{n_\beta} |\Phi_i^\top \beta_k|$
3: $\widetilde{q} \leftarrow [\widetilde{q} \quad \bar{q}_{i^*}]$
4: $\phi^* \leftarrow \Phi_{i^*}$
5: $\beta_k \leftarrow \beta_k - \eta\, \phi^* \phi^{*\top} \beta_k \quad \forall \quad k \in \{1, ..., n_\beta\}$
6: $\Phi \leftarrow \Phi \setminus \Phi_{i^*}$
7: $\bar{q} \leftarrow \bar{q} \setminus \bar{q}_{i^*}$
8: **until** $|\phi^{*\top} \beta_k| < x_{tol} \quad \forall \quad k \in \{1, ..., n_\beta\}$ for last few iterations
9: **return** $\widetilde{q}$

2 where the pose that causes the largest average error on all erroneous β's is chosen to be added to the filtered set of poses $\widetilde{q}$ which is the output of the algorithm. This pose is also used to perform gradient descent on all $\beta \in \bar{\beta}$ (step 5). We choose the pose that causes the largest prediction error over the updated set $\bar{\beta}$ in each iteration because it is the most informative for the learning process. The learning process stops when the prediction errors due to all $\beta \in \bar{\beta}$ consistently fall below some tolerance x_{tol} for a set number of iterations.

Even though the set of poses generated from meta-learning were acquired from different βs than that of the real robot, these poses generated a large error that then helped our entire $\bar{\beta}$ set to converge. If our robot's initial β is in or even close to the set $\bar{\beta}$, these poses should have a similar effect and cause it to converge.

C. Online Data Collection

We now discuss the problem of balancing the robot despite a bad estimate of body CoM to obtain data points for the learning process. Given that body joints are locked at the desired pose $\{ q_2, ..., q_L \}$, the robot is equivalent to a single rigid link on two wheels, to be balanced by manipulating the base link q_1 and the wheels. We utilize ADRC [3] for this purpose. This approach for balancing control of a

WIP Humanoid is originally intended to handle disturbances represented by a torque τ_D about the wheel axis. To see how this approach is applicable for our case, we can imagine a virtual robot that has β values equal to our current bad estimate and is experiencing a disturbance torque such that the effective CoM of the virtual system has shifted to the real CoM of the physical system. Thus the problem of controlling a robot with a bad CoM estimate is equivalent to one experiencing a disturbance torque about its wheel axle.

A brief explanation of the technique as it applies to our system is as follows. Linearizing the dynamics of WIP Humanoid with its joints locked at pose q in a 2 DoF system

$$\dot{X} = \frac{d}{dt}\begin{bmatrix} x & \dot{x} & \theta & \dot{\theta} \end{bmatrix}^\top = A(q)X + B(q)\tau_w \tag{5}$$

where

$$\begin{aligned} x, \dot{x} &= \text{position and heading speed of the robot} \\ \theta, \dot{\theta} &= \text{ang. position and speed of CoM about wheel axis} \\ \tau_w &= \text{sum of torques applied on both wheels} \\ B(q) &= \begin{bmatrix} 0 & 0 & b_x(q) & b_\theta(q) \end{bmatrix}^\top \end{aligned}$$

Note that A, b_x and b_θ are functions of parameters such as CoM distance from wheel axis and body inertia that are dependent on q. Applying LQR on this pose-dependent linearized system $(A(q), B(q))$ results in pose-dependent feedback gains

$$F(q) = \begin{bmatrix} F_x(q)^\top & F_\theta(q)^\top \end{bmatrix}^\top = \text{LQR}(A(q), B(q)) \tag{6}$$

Treating $\dot{x}$ and $\dot{\theta}$ dynamics as two independent subsystems by following [20], we can find the control inputs as

$$u_x = -F_x(q)^\top \begin{bmatrix} x & \dot{x} \end{bmatrix}^\top \quad u_\theta = -F_\theta(q)^\top \begin{bmatrix} \theta & \dot{\theta} \end{bmatrix}^\top$$

The standard feedback control setting for WIP systems has the control input defined by $\tau_w = u_x + u_\theta$. However, the key to perform active disturbance rejection is to estimate the numerical value of dynamic disturbances in the two subsystems, $\hat{f}_x$ and $\hat{f}_\theta$, due to the inaccurate CoM estimate and compensate for those disturbances using feedback linearization:

$$\tau_w = \left(u_x - \frac{\hat{f}_x}{b_x(q)}\right) + \left(u_\theta - \frac{\hat{f}_\theta}{b_\theta(q)}\right) \tag{7}$$

Here, $\hat{f}_x$ and $\hat{f}_\theta$ are estimating the dynamic disturbances f_x and f_θ in the subsystems appearing in state space representation of the dynamic model

$$\begin{aligned} \ddot{x} &= f_x(X, q, \tau_D, u_\theta) + b_x(q)u_x \\ \ddot{\theta} &= f_\theta(X, q, \tau_D, u_x) + b_\theta(q)u_\theta \end{aligned} \tag{8}$$

The estimates are found using Extended State Observers

$$\frac{d}{dt}\begin{bmatrix} \hat{\theta} \\ \hat{\dot{\theta}} \\ \hat{f}_\theta \end{bmatrix} = \begin{bmatrix} \hat{\dot{\theta}} + l_{\theta 1}(\theta - \hat{\theta}) \\ \hat{f}_\theta + l_{\theta 2}(\theta - \hat{\theta}) \\ l_{\theta 3}(\theta - \hat{\theta}) \end{bmatrix}, \frac{d}{dt}\begin{bmatrix} \hat{x} \\ \hat{\dot{x}} \\ \hat{f}_x \end{bmatrix} = \begin{bmatrix} \hat{\dot{x}} + l_{x1}(x - \hat{x}) \\ \hat{f}_x + l_{x2}(x - \hat{x}) \\ l_{x3}(x - \hat{x}) \end{bmatrix} \tag{9}$$

where the observer gains l_x and l_θ are designed using pole placement.

III. Simulation Results

We started experiments by simulating our pipeline; we first considered a WIP model with 7 DoF in Matlab and next a WIP with 19 DoF in the 3D Dynamic Animation and Robotics Toolkit (DART) [21]. The former served as a more tractable proof of concept that led into the latter, a more faithful representation of the robot that we will be using during the experiments. In both simulations, we provided the class methods for instantiating an L-link WIP model, updating their mass parameters β, approximating their dynamics, applying control, and visualizing the results. Simulation provided us with two key benefits over hardware: (1) it allowed us to rapidly spawn, control, and respawn our robot in a safe, realistic setting; and (2) it allowed us immediate access to parameters that were otherwise "unknowable", or difficult to obtain. For our system specifically, these parameters are the masses and Center of Masses for individual links, which are both numerous and inaccessible to measurements. To evaluate the performance of our algorithms we instantiated two full L-link WIP models – a ground truth model and an inaccurate model with an estimation of the parameters of the first robot. These two models served as placeholders for the arm's configuration and mass parameters. We then simplified these two models into their single link representation (Fig. 2 - right). In Matlab, using an ODE45 integration loop, we simulated the system dynamics from the ground truth model and then calculated the control signals based on the estimated simplified model. In the DART implementation, the dynamics were updated automatically by the simulator. We first started by tuning our ADRC's LQR gains to be able to control the estimated simplified model to the balance position of the ground truth model. During this process, we iteratively set both models to randomized joint angles on the configuration space. After tuning the controller and observer parameters for each joints configuration, the ADRC would balance the systems to its true balance position, i.e. for a given configuration $q_2, q_3, \ldots, q_L$, the ADRC would find the value of q_1 that balanced the system.

A. Gradient Descent Simulation

The offset given by the ADRC for the estimated model was used in a gradient descent algorithm to update our estimated model parameters. Starting with the Matlab simulation, the estimated model was subject to initial noise for the initial estimation of 20% from the real values of the parameters m_i, $m_i x_i$ and $m_i y_i$. Since each link had different properties (similar to our experimental robot), the noise perturbation differed; the first link has an approximated mass of $70kg$ which gives a noise around $14kg$, while the third link has a mass of $6kg$ which give us a noise around $1.2kg$. Using Eq. (4) we update our β for each iteration. A subset of the parameters of β are shown in Fig. 3.

It can be seen in Fig. 3 that our algorithm modifies the β vectors, reaching a local minimum. For some parameters (as m_1, $m_1 y_2$ or $m_3 y_3$) the estimated values converge to the real values, while for others (as m_2, m_3 or $m_1 y_1$) the values converge to a constant error. Even though we are finding a

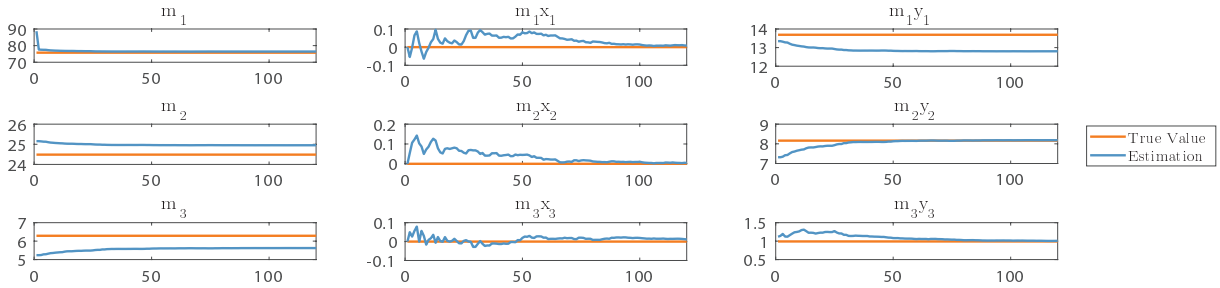


Fig. 3. Values of the different parameters of the estimated model over 120 different configurations. The red lines show the real value of the parameter for the real robot, while the blue lines show the learned weights.

local minima and not necessarily the correct values, we will show that our new estimate of β improves upon the initial values. After running different simulations, we notice that while the system always reaches a x_{CoM} error of zero, the weights converge to different values – giving the intuition that the system consists of several local minima.

This method has shown that the approach works in finding a better set of values than the ones we initially started with, but might not get to the global optimum (the real values). We think that this happens because of the nonlinearities of the system and because the β vector is not perfectly decoupled to the value of the masses.

B. Meta-learning for Gradient Descent Convergence

As described in section II-B, we simulated 20,000 poses over 500 erroneous βs, and got a set of 528 poses until the error was $2mm$. Without using the meta-learning algorithm this process takes over 5,000 poses. The result of our simulated learning curve is presented in Fig. 4. We tested for several initial erroneous βs which started with an x_{CoM} error of at least $2cm$ with a standard deviation of $0.5cm$. It can be seen that after 500 updates using the optimal poses, the mean error decreased to almost $0cm$, specifically the max β error decreased to $x_{tol} = 2mm$.

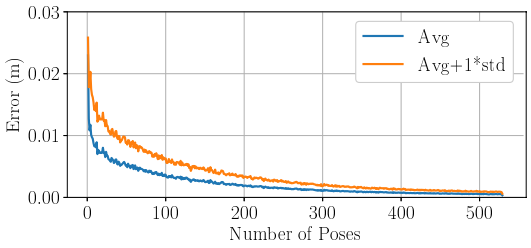


Fig. 4. Mean Error of several βs through the learning algorithm for the meta-learned best 500 poses.

IV. Experimental Results

For the robot that we are using, Golem Krang [22], determining its mass model link-by-link is intractable. Furthermore, the summarizing CoM described in II is difficult to obtain. Instead of extracting the full mass model or CoM estimates, we follow the procedure of other work [23], [24] to evaluate balancing performance. Where the authors analyze more readily observable phenomena, such as distance traveled, time spent stabilizing, and power consumption. In our

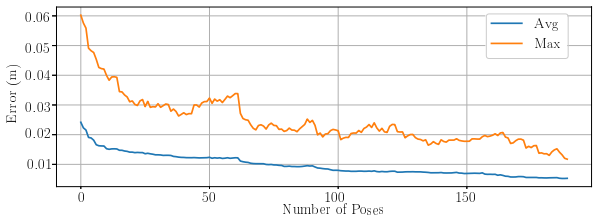


Fig. 5. Error in the parameters as we update the weights β for different random configurations.

case, these quantities were used to analyze whether or not subsequent refinements of an initial offset estimation (β_0) improves the stabilizing control. The physical experiments were separated in two parts: manual data collection and controller efficiency testing. For the first part, we collected data from our subset of pre-determined balanced poses – we manually positioned the robot in the first 236 poses acquired from the meta-learning algorithm in Section II-B and calculated the error between the real x_{com} and the estimation. We obtained this error by setting the robot to presumed balanced pose (which may not actually be balanced under our inaccurate β_0), and adjusted the base link angle q_1 until the system became balanced. We then separated this data into a training set of 190 poses and a testing set of 46 poses. Then, using the training set, we implemented gradient descent to obtain a series of betas going from $\beta_1, \beta_2, \ldots, \beta_{190}$. For each beta, we computed the errors produced by the remaining balanced poses in the testing dataset; the results are shown in Fig. 5. For β_0, we started with a mean error of $2.5cm$ in the x_{CoM} for the given 46 poses and a maximum error of $6cm$. With subsequent iterations, the mean error and the maximum error decreased. For β_{190} we achieved a mean error of $0.4cm$ with a maximum error of $1.2cm$ for any given pose in the testing set.

For the second part, we used five of our learned βs to balance the robot in a given pose. Specifically, we looked at the initial balancing action, which involves transitioning between a stable sitting position to an inverted pendulum position. For this action, the robot stands from three points of contact with the ground (two active wheels and a caster wheel). Then it rotates its wheels (at a speed which depends on its CoM estimate) to lift off the caster, and it finally balances as a two-wheeled WIP. The balancing experiments tested different β estimates to show how the overall control

improves during the transition and steady state of the robot. To investigate the connection between updated β vectors and controller performance, we show the results of testing β_{16}, β_{32}, β_{64}, β_{128} and β_{190}. Smaller βs are not shown, since the robot controller was not able to securely stabilize the system. Additionally, for each β_i we tested seven attempts to see the reproducibility of the results.

The instantaneous power consumption of the wheel motors during and after the transition to standing is shown in Fig. 6, and a summary of the control performance is presented in Table II. The instantaneous power was calculated by multiplying the torque and angular velocity of the wheels.

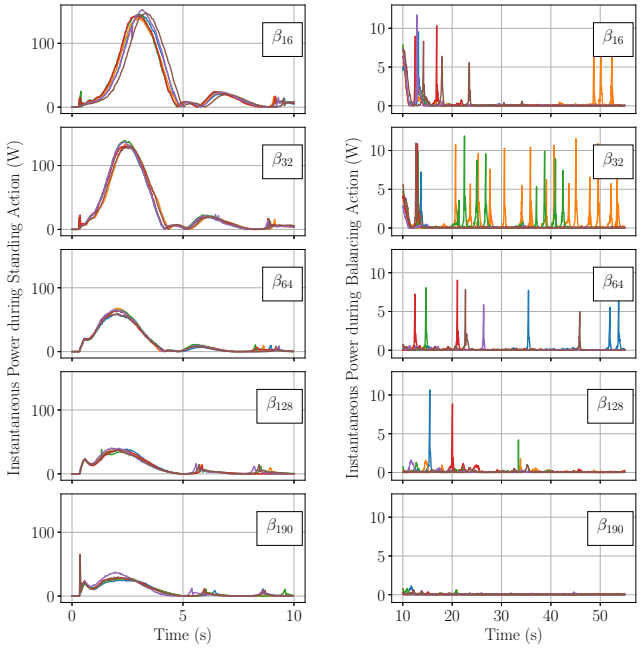


Fig. 6. Instantaneous power applied by the wheel motors. Each plot includes the results corresponding to 7 independent runs for different values of β (β_{16}, β_{32}, β_{64}, β_{128} and β_{190}). The left column summarizes the sitting-standing transition (the first 10 seconds of the experiment), and the right column summarizes the WIP balancing (the subsequent 10 to 60 seconds).

TABLE II

SUMMARY OF THE CONTROL PERFORMANCE UNDER DIFFERENT BETAS.

β	Max Pos. [m]	Resting Pos. [m]	Time until Resting [s]	Max Power [W]	Avg. Resting Power [mW]
β_{16}	4.70 ± 0.16	2.49 ± 0.03	11.5 ± 1.5	145 ± 4	7.83 ± 1.69
β_{32}	4.59 ± 0.11	2.67 ± 0.11	10.1 ± 1.1	133 ± 4	8.85 ± 7.09
β_{64}	3.59 ± 0.17	1.53 ± 0.05	7.59 ± 1.33	63.1 ± 3.3	2.95 ± 1.04
β_{128}	2.74 ± 0.07	1.13 ± 0.03	6.80 ± 1.20	41.3 ± 6.9	2.90 ± 0.60
β_{190}	$\mathbf{2.61 \pm 0.08}$	$\mathbf{1.08 \pm 0.03}$	$\mathbf{7.09 \pm 1.97}$	$\mathbf{34.5 \pm 13.2}$	$\mathbf{1.54 \pm 0.25}$

As shown in the left column of Fig. 6 and in Table II, the peak power consumption decreases with subsequent values of beta. As shown in the right column of the same figure, the number of balancing adjustments (spikes in power consumption) is similarly reduced. For the first β values, the system occasionally destabilized and readjusted, whereas the latest β_{190} value kept these adjustments and hence overall power consumption to a minimum.

Table II shows improvement in several quantities that characterize control performance: the initial overshoot position decreases by 44% between the β_{16} and β_{190} iterations; the resting position decreases by 57%; the time until resting decreases by 38%; the peak instantaneous power decreases by 76%; and the average power during steady state balancing decreases by 80%. Each of our performance metrics improves with more refined mass model parameters. Together, these trends support the claim that the CoM estimation procedure does improve balancing for a WIP.

V. CONCLUSION

We have shown that the proposed methodology improves the CoM estimate of a WIP Humanoid and that these improvements translate to improved controller performance. In simulation, using active disturbance rejection control, our robot successfully balances with an inaccurate prior mass model, collects new pose data at balanced positions, and learns from these poses to produce a more accurate CoM estimate. In hardware, we demonstrate that these refined estimates directly translate into improved controller performance. Together, our simulation and hardware results support the claim that our algorithm – a semi-automated, tractable procedure that refines the latent space mass model of a high dimensional system with few physically observable parameters – does improve overall balance. The algorithm was probed in simulation and verified physically on a 19 DoF WIP robot. Our future work will implement the fully automated estimation pipeline–active disturbance rejection control, balanced pose data collection, and online learning–in an entirely online fashion on the physical robot, where it will improve its parameter estimates through meta-learned poses.

REFERENCES

[1] A. A. Bature, S. Buyamin, M. N. Ahmad, and M. Muhammad, "A comparison of controllers for balancing two wheeled inverted pendulum robot," 2014.

[2] M. Zafar and H. I. Christensen, "Whole body control of a wheeled inverted pendulum humanoid," in *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov 2016, pp. 89–94.

[3] L. Canete and T. Takahashi, "Disturbance compensation in pushing, pulling, and lifting for load transporting control of a wheeled inverted pendulum type assistant robot using the extended state observer," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct 2012, pp. 5373–5380.

[4] T. Takei, R. Imamura, and S. Yuta, "Baggage transportation and navigation by a wheeled inverted pendulum mobile robot," *IEEE Transactions on Industrial Electronics*, vol. 56, no. 10, pp. 3985–3994, 2009.

[5] J. Carpentier, M. Benallegue, N. Mansard, and J. P. Laumond, "Center-of-mass estimation for a polyarticulated system in contact: A spectral approach," *IEEE Transactions on Robotics*, vol. 32, no. 4, pp. 810–822, Aug 2016.

[6] G. G. Muscolo, C. T. Recchiuto, C. Laschi, P. Dario, K. Hashimoto, and A. Takanishi, "A method for the calculation of the effective center of mass of humanoid robots," in *2011 11th IEEE-RAS International Conference on Humanoid Robots*, Oct 2011, pp. 371–376.

[7] M. Kudruss, M. Naveau, O. Stasse, N. Mansard, C. Kirches, P. Soueres, and K. Mombaur, "Optimal control for whole-body motion generation using center-of-mass dynamics for predefined multi-contact configurations," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Nov 2015, pp. 684–689.

[8] S. Kwon and Y. Oh, "Estimation of the center of mass of humanoid robot," in *2007 International Conference on Control, Automation and Systems*, Oct 2007, pp. 2705–2709.

[9] E. Sihite and T. Bewley, "Attitude estimation of a high-yaw-rate mobile inverted pendulum; comparison of extended kalman filtering, complementary filtering, and motion capture," in *2018 Annual American Control Conference (ACC)*, June 2018, pp. 5831–5836.

[10] A. Pajon, S. Caron, G. D. Magistri, S. Miossec, and A. Kheddar, "Walking on gravel with soft soles using linear inverted pendulum tracking and reaction force distribution," in *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, Nov 2017, pp. 432–437.

[11] Z. Gao, "Active disturbance rejection control: a paradigm shift in feedback control system design," in *2006 American Control Conference*, June 2006.

[12] D. Luo, Y. Wang, and X. Wu, "Online learning of com trajectory for humanoid robot locomotion," in *2012 IEEE International Conference on Mechatronics and Automation*, Aug 2012, pp. 1996–2001.

[13] Q. Chen, H. Cheng, C. Yue, R. Huang, and H. Guo, "Step length adaptation for walking assistance," in *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, Aug 2017, pp. 644–650.

[14] L. Yang, Z. Liu, and Y. Zhang, "Dynamic balance control of biped robot using optimized slfns," in *2016 Chinese Control and Decision Conference (CCDC)*, May 2016, pp. 5303–5307.

[15] T. Kim and H. J. Kim, "Path tracking control and identification of tire parameters using on-line model-based reinforcement learning," in *2016 16th International Conference on Control, Automation and Systems (ICCAS)*, Oct 2016, pp. 215–219.

[16] L. Jamone, B. Damas, and J. Santos-Victor, "Incremental learning of context-dependent dynamic internal models for robot control," in *2014 IEEE International Symposium on Intelligent Control (ISIC)*, Oct 2014, pp. 1336–1341.

[17] L. Jiang, H. Qiu, Z. Wu, and J. He, "Active disturbance rejection control based on adaptive differential evolution for two-wheeled self-balancing robot," in *2016 Chinese Control and Decision Conference (CCDC)*, May 2016, pp. 6761–6766.

[18] X. Ruan, X. Wang, X. Zhu, Z. Chen, and R. Sun, "Active disturbance rejection control of single wheel robot," in *Proceeding of the 11th World Congress on Intelligent Control and Automation*, June 2014, pp. 4105–4110.

[19] D. Wei, C. Ren, M. Zhang, X. Li, S. Ma, and C. Mu, "Position/force control of a holonomic-constrained mobile manipulator based on active disturbance rejection control," in *IECON 2017 - 43rd Annual Conference of the IEEE Industrial Electronics Society*, Oct 2017, pp. 6751–6756.

[20] R. Miklosovic and Z. Gao, "A dynamic decoupling method for controlling high performance turbofan engines," in *Proc. of the 16th IFAC World Congress*, vol. 16. Czech Republic, 2005, pp. 482–488.

[21] J. Lee, M. X. Grey, S. Ha, T. Kunz, S. Jain, Y. Ye, S. S. Srinivasa, M. Stilman, and C. K. Liu, "Dart: Dynamic animation and robotics toolkit," *Journal of Open Source Software*, vol. 3, no. 22, 2018.

[22] M. Stilman, J. Olson, and W. Gloss, "Golem krang: Dynamically stable humanoid robot for mobile manipulation," in *2010 IEEE International Conference on Robotics and Automation*, May 2010, pp. 3304–3309.

[23] A. A. Bature, S. Buyamin, M. N. Ahmad, and M. Muhammad, "A comparison of controllers for balancing two wheeled inverted pendulum robot," *International Journal of Mechanical & Mechatronics Engineering*, vol. 14, no. 3, pp. 62–68, 2014.

[24] A. Khosla, G. Leena, and M. Soni, "Performance evaluation of various control techniques for inverted pendulum," *Performance Evaluation*, vol. 3, no. 4, pp. 1096–1102, 2013.