Center of Mass Estimator for Humanoids and its Application in Modelling Error Compensation, Fall Detection and Prevention

X Xinjilefu, Siyuan Feng, and Christopher G. Atkeson

Abstract— **We introduce a center of mass estimator, and its application in full body inverse dynamics control, fall detection and fall prevention for humanoid robots. We use the Linear Inverted Pendulum Model dynamics with an offset to predict the center of mass motion. This offset can be interpreted as a modelling error on the center of mass position, or an external force exerted on the center of mass of the robot, or a combination of both. The center of mass estimator is implemented on our Atlas humanoid robot. During the DARPA Robotics Challenge Finals, it successfully saved our robot from falling on two occasions, and made us the only competitive team without a fall or human physical intervention among all humanoid teams.**

I. Introduction

In order to achieve reliable estimation of humanoid robot Center of Mass (CoM) motion, we need a model that explains how the CoM moves under external forces. The Linear Inverted Pendulum Model (LIPM) model is a natural choice given its simplicity [1]. We introduce an additional offset term into the LIPM dynamics. This offset can be interpreted as a modelling error on the CoM position, or an external force exerted on the CoM of the robot, or a combination of both. This paper introduces the CoM estimator, and its application in inverse dynamics, fall detection, and fall prevention for humanoid robots. The estimator itself is similar to [2], but the implementation and application are different.

The estimation accuracy of a model-based state estimator relies on the model itself. For the Atlas robot, Boston Dynamics provides both the kinematic and the dynamic parameters. The kinematic model is useful for manipulation tasks and foot placement during walking. The dynamic model is used by the controller to generate desired joint torques. Both kinematic and dynamic models are used to estimate the full-body states of the robot [3][4].

A. Kinematic Modeling Error

The kinematic model is rooted at the pelvis. We experimented on the hand end effector to determine the arm kinematic error. We put Atlas in double support, and controlled the hand to follow a predefined trajectory. The trajectory was a concatenation of manipulation task trajectories, because we care about the kinematic model accuracy in those situations. We tracked the motion of the hand, feet, and pelvis using an OptiTrack motion capture system with 8 cameras, and recorded the marker data and robot data simultaneously at 100Hz. We found that the hand end effector had on average an error of about 1cm relative to the pelvis. The error from the motion capture system is negligible.

We also tested the leg kinematic error during double support without the motion capture system, because we know that both feet are not moving in our test. We control the pelvis of the robot to follow a spiral trajectory in the coronal plane. The average error for the distance between the two feet is within 1cm.

There are several factors contributing to the end effectors' kinematic error. The joint position data we collected was pre-joint transmission (the sensors are on the actuator, not the joint axis). It is computed based on the linear actuator length measured by the Linear Variable Differential Transformers (LVDT) which is affected by the loading conditions. The sole pads beneath the robot's feet are deformable up to a few millimeters. There are also backlash, play and structural deformation on the robot. These effects all contribute to the kinematic error between the two feet.

B. Dynamics Modeling Error

The dynamic model has mass, moments of inertia and center of mass location of each link as its parameters. Dynamic parameters are generally not identified as accurately as the kinematic parameters. The error of these parameters is difficult to model individually because the dimension of the parameter space is high. We were not allowed to take the robot apart to measure the inertial parameters. It is hard to account for flexible elements such as oil filled hoses that cross joints in a rigid body model.

On the other hand, the dynamics of the center of mass of the robot can be modeled with a small number of parameters. A widely used control scheme for humanoids is two-level hierarchical control. The high level controller generates the center of mass motion using a simple dynamic model, such as the LIPM, and the low level controller tries to generate the full-body motion given the center of mass motion. It is useful for the controllers to know accurately the motion of the center of mass. Due to modeling error of the link parameters, the center of mass position and velocity computed from forward kinematics will have modeling error as well.

To observe the modelling error, we conduct a simple quasi-static experiment using our Atlas robot. The robot pelvis follows a spiral trajectory in double support. In an ideal situation, where there is neither modeling error in the dynamic model parameters nor forward kinematics, and the foot force/torque sensors are calibrated, the CoM should coincide with the Center of Pressure (CoP) in the horizontal plane when the robot is static. We observe there is a nearly

All the authors are with the Robotics Institute, Carnegie Mellon University, 5000 Forbes Ave, USA {xxinjile, sfeng, cga}@cs.cmu.edu

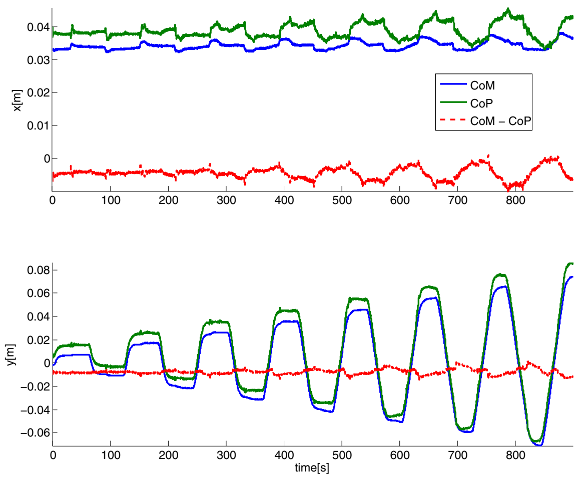

Fig. 1. Computed CoM, measured CoP and their difference in the horizontal ground plane.

constant offset between the CoM position computed from kinematics, and the CoP computed from foot force/torque sensors and kinematics. Fig. 1 shows the computed CoM, measured CoP and their differences in the horizontal plane.

We derive the CoM estimator in Section II, and describe its implementation in Section III. In Section IV we discuss the application of the CoM estimator in fall detection and prevention. Section V concludes the paper.

II. The Center of Mass Kalman Filter

We use the following parameterization of the discrete time linear Kalman filter,

$$x_{k+1} = f(x_k, u_k) = Ax_k + Bu_k \tag{1}$$
$$y_k = h(x_k) = Cx_k + Du_k \tag{2}$$

The LIPM is a simplified model to describe the dynamics of a humanoid robot. It assumes the CoM is at a constant height, therefore the total vertical force must be equal to the body weight at all times. The dynamic equation of LIPM in the horizontal direction is given by Eq. (3)

$$\ddot{x}_{com} = \omega^2(x_{com} - u_{cop}) \tag{3}$$

where $\omega = \sqrt{g/z_{com}}$ is the LIPM eigenfrequency. x_{com}, $\dot{x}_{com}$ and $\ddot{x}_{com}$ are the position, velocity and acceleration of the CoM in the horizontal plane respectively (both x and y components). u_{cop} is the position of the CoP, g is gravity, and z_{com} is the CoM height relative to the CoP. In Eq. (3), u_{cop} represents the total CoP.

The error of the LIPM can be modeled by an offset. If we assume the offset is changing relatively slowly, then its first and second order time derivatives are both close to zero. We formulate the CoM estimator with offset as a Kalman filter. It predicts the robot CoM position, velocity and offset in the horizontal plane. The state vector for the CoM estimator is given by $x = [x_{com}, \dot{x}_{com}, x_{offset}]^T$. We consider the CoP as an input to our state estimator. The process model uses the LIPM dynamics to predict the CoM acceleration. The LIPM dynamics with offset is given by Eq. (4)

$$\ddot{x}_{com} = \omega^2(x_{com} + x_{offset} - u_{cop}) \tag{4}$$

x_{offset} is the LIPM offset. The states are expressed in the world coordinates. z_{com} is computed every estimation time step by forward kinematics. The Kalman filter becomes time variant due to z_{com} being a parameter in the state matrix A in Eq. (5). The process model of the Kalman filter follows Eq. (1). It has the following parameters:

$$A = \begin{bmatrix} \mathbb{I} + \frac{1}{2}\omega^2\Delta t^2\mathbb{I} & \Delta t\mathbb{I} & \frac{1}{2}\omega^2\Delta t^2\mathbb{I} \\ \omega^2\Delta t\mathbb{I} & \mathbb{I} & \omega^2\Delta t\mathbb{I} \\ 0 & 0 & \mathbb{I} \end{bmatrix},$$
$$B = \begin{bmatrix} -\frac{1}{2}\omega^2\Delta t^2\mathbb{I} \\ -\omega^2\Delta t\mathbb{I} \\ 0 \end{bmatrix} \tag{5}$$

where Δt is the time step, $\mathbb{I}$ is a 2×2 identity matrix.

The observation model provides predicted measurement from states. We choose x_{com} and $\ddot{x}_{com}$ as predicted measurements. The measurement equation can be written as

$$y_k = \begin{bmatrix} x_{com} \\ \ddot{x}_{com} \end{bmatrix}$$
$$= \begin{bmatrix} \mathbb{I} & 0 & 0 \\ \omega^2\mathbb{I} & 0 & \omega^2\mathbb{I} \end{bmatrix} \begin{bmatrix} x_{com} \\ \dot{x}_{com} \\ x_{offset} \end{bmatrix} + \begin{bmatrix} 0 \\ -\omega^2\mathbb{I} \end{bmatrix} u_{cop} \tag{6}$$

where the output matrix C and feedthrough matrix D in Eq. (2) are given by

$$C = \begin{bmatrix} \mathbb{I} & 0 & 0 \\ \omega^2\mathbb{I} & 0 & \omega^2\mathbb{I} \end{bmatrix}, D = \begin{bmatrix} 0 \\ -\omega^2\mathbb{I} \end{bmatrix} \tag{7}$$

The measurement depends on sensors available to the robot. We use sensed joint position with forward kinematics to compute CoM position. The root variables are estimated using the base state estimator introduced in [3]. For walking, the position error on the base is below 2% (20cm total error over a course of 10.12m, 26steps). Our odometry performance is comparable to that reported in [5]: "over the course of the 25 steps our position drifted in XYZ by (0.2, 0.0, 0.1) meters".

Ideally, 6-axis force/torque sensors under the robot feet provide information about external forces, which can be used to calculate the CoM acceleration in the horizontal plane. However, using the 3-axis (F_z, M_x and M_y) foot force/torque sensors on the Atlas, we can only compute the CoP in the foot frames. Alternatively, we approximate the CoM acceleration by transforming the acceleration measured by the Inertial Measurement Unit (IMU) mounted on the pelvis to the location of the CoM. This transformation assumes that the CoM is on the same rigid body (pelvis) as the IMU. The transformation is given by:

$$a_{com} = a_{imu} + \omega_{imu} \times (\omega_{imu} \times r) + \dot{\omega}_{imu} \times r \tag{8}$$

where a_{com} is the approximated CoM acceleration. a_{imu} and ω_{imu} are IMU measured acceleration and angular velocity expressed in the world frame. r is the vector from IMU to CoM computed using forward kinematics. $\dot{\omega}_{imu}$ is the angular acceleration of the IMU. It can be computed by finite differentiating ω_{imu}. We assume a constant angular velocity model, so this term is zero. The x and y component of a_{com} are used as measurement.

III. Implementation and Application

We implement the CoM estimator on the Atlas robot. The filter noise parameters are summarized in Table I. The measurement noise parameters for the acceleration are different for walking and for manipulation. Because the walking motion is more dynamic than manipulation, we filter the acceleration less in walking.

TABLE I

CoM Kalman filter noise parameters.

	x_{com}	$\dot{x}_{com}$	x_{offset}
Q	$1e^{-10}$	$1e^{-4}$	$1e^{-10}$

Walking	x_{com}	$\ddot{x}_{com}$
R	$1e^{-6}$	$1e^{-3}$

Manipulation	x_{com}	$\ddot{x}_{com}$
R	$1e^{-6}$	$1e^{-2}$

*Q and R are process and measurement noise covariances.

The estimated offset plays several roles in the control of the robot. The first application is to translate it into an external force acting on the robot CoM, and to apply the force while solving the inverse dynamics using quadratic programming (QP) [6]. During early development, the offset was also used to bias the target CoM location while solving Inverse Kinematics (IK) for walking [7], IK is no longer used in our walking controller. The second application of the offset is to detect and prevent robot falling. In this case, the offset is interpreted as a position offset of the robot CoM, and then used to compute a corrected capture point. The details will be discussed in Section. IV.

A. Kinematic Modelling Error Compensation

Due to pre-transmission joint position sensing and backlash, play and compliance in the mechanism, the forward kinematics model for Atlas is not very accurate. For stationary single support stance, there can be up to 3cm offsets between the model CoM from forward kinematics and measured CoP. A simple linear torque dependant heuristic to reduce the effects of joint compliance is proposed by [8], which is also employed in our controller for the leg joints.

In addition to the pre-transmission joint position sensing, the shoulder and elbow joints are also equipped with incremental encoders that measure position changes. The encoders provide more accurate joint position measurement because they are after the transmission. They can only be used after calibration. Our calibration procedure is to drive all the joints to the known mechanical joint limits and fit parameters to the offsets and scale factors. A very similar procedure provided by Boston Dynamics is used to calibrate the electric forearms.

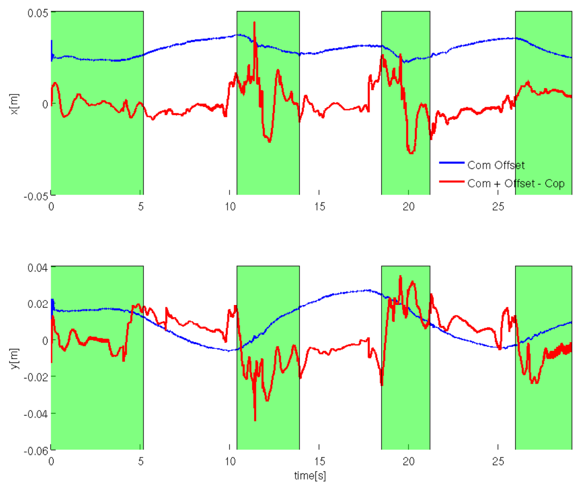

Fig. 2. Plot of the estimated CoM offset, and the difference between CoM and CoP after applying CoM offset during a static walk. The shaded green region is double support, the unshaded region is single support.

B. Dynamic Modelling Error Compensation

The estimated CoM offset x_{offset} can be interpreted as a virtual force $f_{ext} = m\omega^2 x_{offset}$ applied at the CoM of the robot. Given the equation of motion of the full body floating base dynamics of a humanoid robot,

$$M(q)\ddot{q} + h(q,\dot{q}) = S\tau + J_c^T(q)f \tag{9}$$

where M is the mass matrix, h is the nonlinear vector of gravity, Coriolis and centrifugal forces, J_c is the contact Jacobian, f is the contract force, S is a selection matrix, and τ is the actuator torque. We add the estimated virtual force f_{ext} to the right-hand side of Eq. (9) pre-multiplied by the CoM Jacobian transpose J_{com}^T,

$$M(q)\ddot{q} + h(q,\dot{q}) = S\tau + J_c^T(q)f + J_{com}^T(q)f_{ext} \tag{10}$$

so that the modelling error can be handled properly by the inverse dynamics.

We demonstrate the effectiveness of our modelling error compensation approach in a forward walking experiment. In Fig. 2, we show the estimated CoM offset, and the difference between CoM and CoP considering the CoM offset. If the robot is under a static configuration, the difference between corrected CoM and CoP should be zero. When the CoM is accelerating, LIPM dynamics explains the difference. We observe near zero difference in single support, and large differences during double support when the robot is shifting its weight distribution across its supporting feet.

Fig. 3 is a plot of the CoM velocity from the Kalman filter, and from forward kinematics. The CoM velocity from forward kinematics has noise injected from the measured IMU angular velocity, and occasional spikes from the left shoulder and wrist pitch joints. The estimated CoM velocity has a lower noise level compared to that computed from forward kinematics.

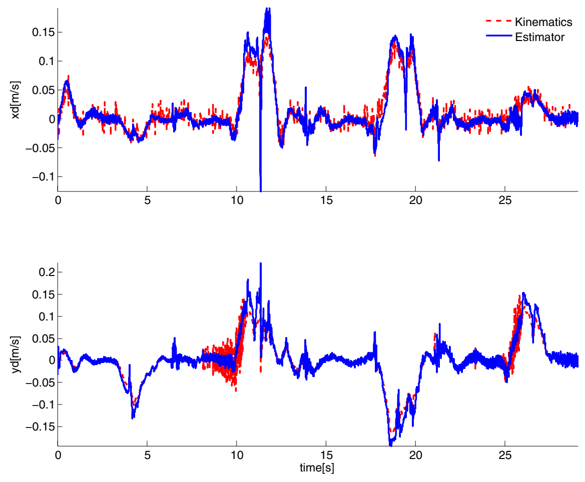

Fig. 3. Plot of the estimated CoM velocity, from the CoM offset estimator, and from full body kinematics starting at the root.

IV. Fall Detection and Prevention

Compared to wheeled and multilegged robots, humanoids usually have a higher CoM and a smaller support region. This means a higher risk of hardware damage when falling and a smaller stability margin. In a real world scenario where humanoid robots have to interact with the environment without a safety belay system, fall prevention and recovery behaviors are necessary. A good example is the DARPA Robotics Challenge (DRC), where robots have to perform disaster response related tasks involving physical contact with the environment. What we observed during the contest was that most humanoids fell down. We will discuss our implementation of fall detection and prevention algorithms using the CoM estimator.

A. Background

Fujiwara's group in JAIST has worked on fall control of humanoid robots [9][10][11][12][13][14]. Their work focused on minimizing impact during fall. Another line of work deals with fall detection. The general idea is to detect abnormality in sensor measurements or computed quantities such as CoM and CoP. The methods differ in what features to use and how they are parameterized, whether the model is built online or offline, and what algorithms are used to detect abnormality. Sensor measurements are compared with sensor models in humanoid walking to determine the dynamic instability of the gait. The sensor model is built offline in [15], and a heuristic is used online in [16]. Ogata et al. detect abnormality through discriminant analysis and learning [17][18], similarly, pattern classification is used in [19]. Yun et al. introduced the concept of "Fall trigger boundary" which encloses a region in the robot's feature space where the balance controller is able to stabilize the robot [20].

B. Capture Point

The Capture Point (CP), introduced in [21], is the point on the ground where a humanoid must step to in order to come to a complete stop. If the humanoid dynamics is approximated using the LIPM, the CP has a very simple expression as a linear combination of the CoM position and velocity in the horizontal plane. We can derive the CP by analysing the LIPM orbital energy [21], or by solving the LIPM ODE Eq. (3) and posing boundary conditions [22]. We briefly derive the CP expression by solving Eq. (3). The CoM has the following solution

$$x_{com}(t) = \frac{1}{2}\left[x_{com}(0) - u_{cop} + \frac{\dot{x}_{com}(0)}{\omega}\right]e^{\omega t} + \frac{1}{2}\left[x_{com}(0) - u_{cop} - \frac{\dot{x}_{com}(0)}{\omega}\right]e^{-\omega t} + u_{cop} \tag{11}$$

The CP x_{cp} is defined as the point where the CoM comes to a complete stop, which can be written as

$$x_{cp} = \lim_{t\to\infty} x_{com}(t) = \lim_{t\to\infty} u_{cop} \tag{12}$$

In order for Eq. (12) to be finite, the divergent component of Eq. (11) $e^{\omega t}$ has to have zero coefficient, which translates to

$$x_{com}(0) + \frac{\dot{x}_{com}(0)}{\omega} = x_{cp} \tag{13}$$

Given that Eq. (13) has to be satisfied all the time, the time index can be dropped and the CP is defined as

$$x_{cp} = x_{com} + \frac{\dot{x}_{com}}{\omega} \tag{14}$$

Because the simple CP expression Eq. (14) is derived from the LIPM dynamics, it is only an approximation of the real CP when the robot motion can be explained by the LIPM dynamics. More complicated approximations such as the Angular Momentum Pendulum Model lead to more complicated expressions for the CP. Refer to [21] for a detailed discussion. We use the simplest CP on the Atlas robot because the LIPM is a good enough approximation for tasks our robot performs.

C. Fall detection and prevention

Our fall detection algorithm is based on whether the CP is within the robot's polygon of support. The CoM estimator is used to compute the CP, and the key of our algorithm is to use a "Corrected Capture Point (CCP)", where the LIPM offset is used as a CoM position offset. The CCP $\hat{x}_{cp}$ is defined by

$$\hat{x}_{cp} = x_{com} + \frac{\dot{x}_{com}}{\omega} + x_{offset} \tag{15}$$

The $\hat{x}_{cp}$ is readily available since every quantity in Eq. (15) is a state estimated by the CoM filter. $\hat{x}_{cp}$ can handle modelling error and relatively small dynamics forces, such as drag from the tether. It is especially helpful in estimating the unplanned quasi-static external forces applied at unknown locations on the robot, which happens quite often during manipulation tasks. We are making the assumption that any

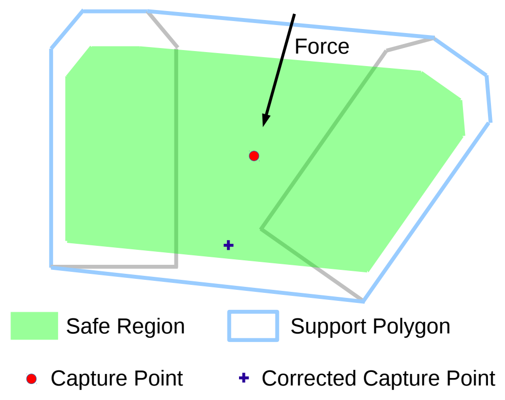

Fig. 4. For the manipulation controller, the robot is in double support and pushed back by an external force. The modelled CP is maintained at the center of support polygon by the controller. The CCP is pushed back and close to the boundary of the safe region. The controller will freeze the robot as soon as the CCP is outside of the safe region. In our implementation, the safe region is the convex hull of the shrunk foot corners, where the shrunk foot has the following dimension: 4.5cm to the front and side, 5cm to the back of the physical foot.

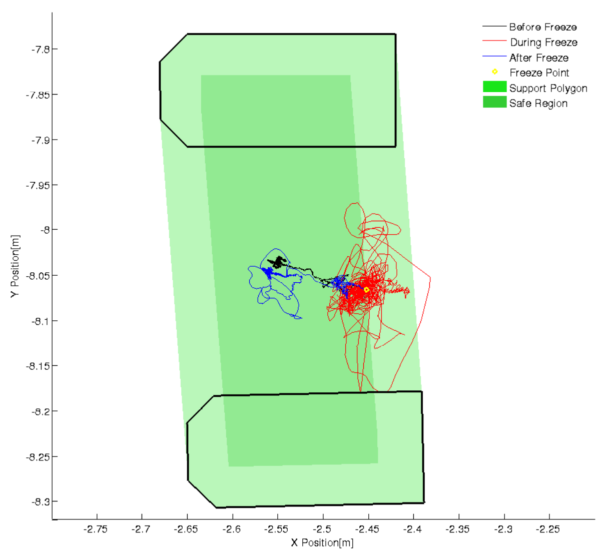

Fig. 5. The CCP motion during the drill task on the second day of the DRC Finals. The black trace started from the center of support polygon and moved towards the back, corresponding to the robot pushing itself backwards. Once the CCP exceeds the boundary of the safe region (the yellow circle), the arm motion is stopped and the robot went into freeze mode. The red trace showed the corrected CP motion during the freeze. It went outside of support polygon briefly and had oscillations. The blue trace was when the operator unfroze the robot and it released the drill.

external force is caused by the robot's action, and that is why we freeze/stop, to stop the force from increasing, moving the CP outside the polygon of support, and forcing a stepping response on fall.

Consider the following example: the robot is standing still and pushing on the wall slowly with one hand until it tips over, see Fig. 4. Even though the modelled CP x_{cp} is kept at the center of the support polygon by the balance controller, the CCP $\hat{x}_{cp}$ is pushed back and out of the support polygon once the robot is tipped over. Without estimating the LIPM offset, the modelled CP can not predict falling.

Even though we assume the total external force applied to the robot is slow and quasi-static, so that x_{offset} has zero derivative which is consistent with Eq. (5), the CoM estimator formulation is not limited by this assumption. By increasing Q (the expected process noise) on the x_{offset}, and decreasing R (expected measurement noise) on the CoM acceleration, we can essentially tune how fast the x_{offset} is estimating the total external force. The adaptive Kalman filter is a good candidate to handle different situations automatically.

The implementation of the fall detection and prevention algorithm on the robot is controller dependent. There are two full body controllers implemented on the robot, one for manipulation and one for walking. Details of these controllers can be found in [6].

1) Manipulation: For the manipulation controller, the robot is always assumed to be in double support, and the support polygon is computed by finding the convex hull of the foot corner points using Andrew's monotone chain 2D convex hull algorithm [23]. The foot corner points are computed using forward kinematics. To prevent the robot from falling during manipulation, we require the CCP to lie within a subset of the support polygon called the safe region (Fig. 4). The moment the CCP escapes the safe region, a freeze signal is sent to the manipulation controller, and it clears all running joint trajectories and freezes the robot at the current pose (the balance controller is still running).

During our second run in the DRC finals, the right electric forearm mechanically failed when the cutting motion was initiated during the drill task. The uncontrolled forearm wedged the drill into the drywall and pushed the robot backwards. The controller froze and saved the robot from falling. The operator was able to recover from an otherwise catastrophic scenario. Fig. 5 is a plot of the CCP in the foot ground plane.

The time plot in Fig. 6 shows candidate fall predictors during this event. We can eliminate some candidate fall predictors easily. The CoM (and a corrected CoM (not shown)) usually provide a fall warning too late, because the CoM velocity is not included. The CP does not include information about external forces. The CoP can warn of foot tipping, but it is less reliable about warning about robot falling, which is not the same thing as foot tipping in a force controlled robot or if there are non-foot contacts and external forces. In this plot, we see that the CoP moves away from the safe region during recovery, predicting that the robot is going to fall, while the CCP moves towards the interior of the safe region.

The fall detection algorithm for the walking controller is similar to that of the manipulation controller and will be discussed next.

2) Walking: Compared to the manipulation tasks where the robot is always in double support, the support polygon undergoes constant change during walking, and is determined by contact state. The contact state is computed by

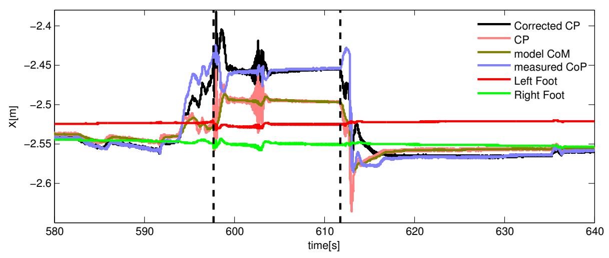

Fig. 6. Plots of candidate fall predictors in the fore/aft direction during the drill task. The black vertical dashed lines mark the freeze time and the start of manual recovery.

thresholding the 3-axis foot force/torque sensor readings. The strain gages on the robot are not perfect even after calibration. Often a nonzero value up to ±50N is measured on the swing foot. Sometimes a much bigger value can be observed during swing. This value can change from step to step (see Fig. 7), which makes robust contact state detection difficult. We implemented a simple Schmitt trigger for contact state detection. Instead of a single threshold, a Schmitt trigger has both low and high thresholds. If the foot is in contact and the force/torque sensor reading drops below the low threshold, the contact state of that foot switches to swing. If the foot is in swing and the sensed force is above the high threshold, the contact state changes to stance. We choose the low and high thresholds to be 80N and 100N respectively. The Schmitt trigger eliminates a lot of error compared to a single threshold. Even with the Schmitt trigger, the contact detection can go wrong sometimes due to bad sensor readings. We suspect the strain gage fluctuation is due to thermal effects, and a possible change in the strain gage bonding with the substrate.

For dynamic walking, the CP naturally goes beyond the supporting polygon during swing phase and is captured by the touchdown foot. One way to predict a fall is to use the swing foot time to touchdown and predict if the current CP is within the support polygon of possible or desired footholds. This approach is complicated because it depends on the controller. Our simpler solution is to use a heuristic that detects a fall only if the CP is outside of the support polygon for a continuous period of time. The time is set to 0.6 seconds after extensive testing. As soon as the fall is detected, there are several things the humanoid can do, such as using angular moment to maintain balance, or taking a step as presented in [22]. We have implemented a simple step recovery controller that works in single support where the robot puts down the swing foot right away given there is no self collision.

If none of the aforementioned approaches can stop the robot from falling down, then a safe fall behavior should be triggered. In testing, there happened to be a false positive due to false contact detection during one of our terrain task practice runs two days before the DRC Finals, so we turned off the fall detection in the walking controller in the final contest.

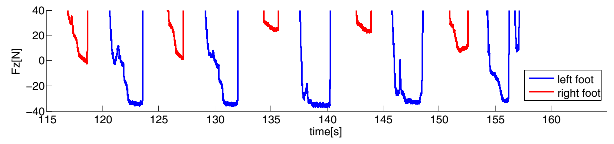

Fig. 7. The force/torque sensor readings during swing for several steps, the sensors are calibrated in the air before taking steps. Ideally the flat part should read near zero and equal for both feet, but we observe measurements ranging from -40N to 40N and they are not constant or repeatable.

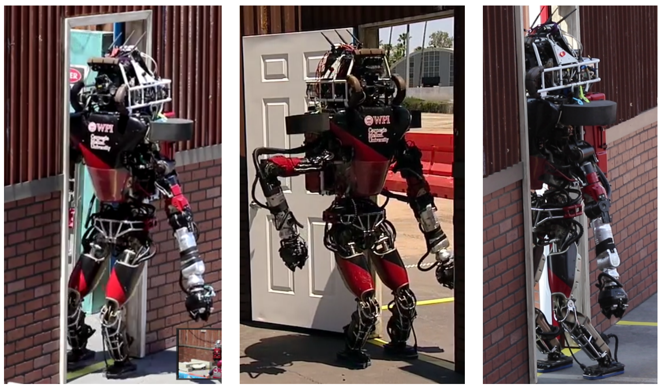

Fig. 8. Successful sidestepping through the door (left, middle) and the failure in the DRC rehearsal (right) in which the protective cage (black padding) for the head is against the white door frame.

On the DRC rehearsal day, the robot was caught on the door frame when sidestepping through (Fig 8). The walking controller detected that the CoP had not moved as expected, delayed liftoff and remained in double support, and stopped the current behavior for manual recovery. Data from this experiment is plotted in Fig 9.

V. Conclusions

In the paper, we introduced a CoM estimator for humanoid robots and its applications in modeling error compensation and fall detection and prevention. A better estimate of the CoM motion improves the controller performance and stability of the robot. Essentially, it uses the LIPM dynamics with variable height to approximate the dynamics of the CoM of the robot. The modeling error on each link is lumped as the CoM offset. It makes state estimation simple and robust, and saves us from tuning mass models of the robot. More importantly, it serves as an observer of unexpected external forces. This enables fall being detected and sometimes prevented. During DRC Finals, the CoM estimator successfully saved our robot from falling in two cases, and made us the only competitive team that attempted all tasks, and did not fall or require human physical assistance. Its limitations come from the limitation of the LIPM model, which does not take into account angular momentum.

Acknowledgment

This material is based upon work supported in part by the US National Science Foundation (ECCS-0824077, and

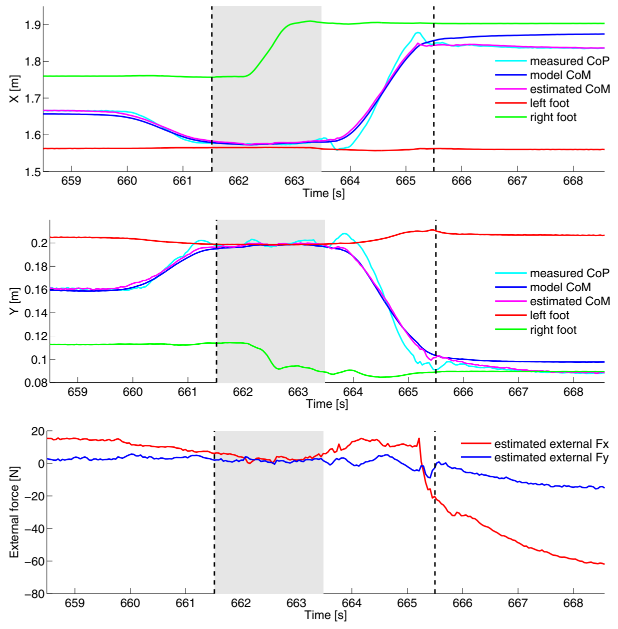

Fig. 9. Atlas was caught on the door frame when sidestepping through it during the rehearsal at the DRC Finals. The walking controller delayed liftoff and remained in double support when the CoM estimator detected a large offset. Single support phase is shown by the shaded area, and the black dashed lines indicate the planned liftoff time. The estimated CoM is the sum of the model CoM and the estimated CoM offset.

IIS-0964581) and the DARPA M3 and Robotics Challenge programs under DRC Contract No. HR0011-14-C-0011.

References

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

[2] B. J. Stephens, "State estimation for force-controlled humanoid balance using simple models in the presence of modeling error," in *Robotics and Automation (ICRA), IEEE International Conference on*. IEEE, 2011, pp. 3994–3999.

[3] X. Xinjilefu, S. Feng, W. Huang, and C. Atkeson, "Decoupled state estimation for humanoids using full-body dynamics," in *Robotics and Automation (ICRA), IEEE International Conference on*. IEEE, 2014, pp. 195–201.

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

[5] M. F. Fallon, M. Antone, N. Roy, and S. Teller, "Drift-free humanoid state estimation fusing kinematic, inertial and lidar sensing," in *Humanoid Robots (Humanoids), 14th IEEE-RAS International Conference on*. IEEE, 2014, pp. 112–119.

[6] S. Feng, X. Xinjilefu, C. Atkeson, and J. Kim, "Optimization based controller design and implementation for the Atlas robot in the DARPA Robotics Challenge Finals," in *Humanoid Robots (Humanoids), 15th IEEE-RAS International Conference on*. IEEE, 2015, pp. accepted.

[7] S. Feng, E. Whitman, X. Xinjilefu, and C. G. Atkeson, "Optimization based full body control for the Atlas robot," in *Humanoid Robots (Humanoids), 14th IEEE-RAS International Conference on*. IEEE, 2014, pp. 120–127.

[8] T. Koolen, S. Bertrand, G. Thomas, T. d. Boer, T. Wu, J. Smith, J. Englsberger, and J. Pratt, "Design of a momentum-based control framework and application to the humanoid robot atlas," *International Journal of Humanoid Robotics*, 2015.

[9] K. Fujiwara, F. Kanehiro, S. Kajita, K. Kaneko, K. Yokoi, and H. Hirukawa, "Ukemi: falling motion control to minimize damage to biped humanoid robot," in *Intelligent Robots and Systems (IROS). Proceedings IEEE/RSJ International Conference on*, IEEE, 2002, pp. 2521–2526.

[10] K. Fujiwara, F. Kanehiro, S. Kajita, K. Yokoi, H. Saito, K. Harada, K. Kaneko, and H. Hirukawa, "The first human-size humanoid that can fall over safely and stand-up again," in *Intelligent Robots and Systems (IROS). Proceedings IEEE/RSJ International Conference on*, vol. 2. IEEE, 2003, pp. 1920–1926.

[11] K. Fujiwara, F. Kanehiro, H. Saito, S. Kajita, K. Harada, and H. Hirukawa, "Falling motion control of a humanoid robot trained by virtual supplementary tests." in *Robotics and Automation (ICRA), IEEE International Conference on*. IEEE, 2004, pp. 1077–1082.

[12] K. Fujiwara, F. Kanehiro, S. Kajita, and H. Hirukawa, "Safe knee landing of a human-size humanoid robot while falling forward," in *Intelligent Robots and Systems (IROS). Proceedings. IEEE/RSJ International Conference on*, vol. 1. IEEE, 2004, pp. 503–508.

[13] K. Fujiwara, S. Kajita, K. Harada, K. Kaneko, M. Morisawa, F. Kanehiro, S. Nakaoka, and H. Hirukawa, "Towards an optimal falling motion for a humanoid robot," in *Humanoid Robots, 6th IEEE-RAS International Conference on*. IEEE, 2006, pp. 524–529.

[14] ——, "An optimal planning of falling motions of a humanoid robot," in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*. IEEE, 2007, pp. 456–462.

[15] R. Renner and S. Behnke, "Instability detection and fall avoidance for a humanoid using attitude sensors and reflexes," in *Intelligent Robots and Systems (IROS), IEEE/RSJ International Conference on*. IEEE, 2006, pp. 2967–2973.

[16] J. Ruiz-del Solar, J. Moya, and I. Parra-Tsunekawa, "Fall detection and management in biped humanoid robots," in *Robotics and Automation (ICRA), IEEE International Conference on*. IEEE, 2010, pp. 3323–3328.

[17] K. Ogata, K. Terada, and Y. Kuniyoshi, "Falling motion control for humanoid robots while walking," in *Humanoid Robots, 7th IEEE-RAS International Conference on*. IEEE, 2007, pp. 306–311.

[18] ——, "Real-time selection and generation of fall damage reduction actions for humanoid robots," in *Humanoid Robots, 8th IEEE-RAS International Conference on*. IEEE, 2008.

[19] O. Höhn and W. Gerth, "Probabilistic balance monitoring for bipedal robots," *The International Journal of Robotics Research*, vol. 28, no. 2, pp. 245–256, 2009.

[20] S.-k. Yun, A. Goswami, and Y. Sakagami, "Safe fall: Humanoid robot fall direction change through intelligent stepping and inertia shaping," in *Robotics and Automation (ICRA), IEEE International Conference on*. IEEE, 2009, pp. 781–787.

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

[22] O. E. Ramos, N. Mansard, and P. Soueres, "Whole-body motion integrating the capture point in the operational space inverse dynamics control," in *Humanoid Robots (Humanoids), 14th IEEE-RAS International Conference on*. IEEE, 2014, pp. 707–712.

[23] A. M. Andrew, "Another efficient algorithm for convex hulls in two dimensions," *Information Processing Letters*, vol. 9, no. 5, pp. 216–219, 1979.