Real-time State Estimation with Whole-Body Multi-Contact Dynamics: A modified UKF Approach

Kendall Lowrey[1], Jeremy Dao[2], and Emanuel Todorov[1]

Abstract— **We present a real-time state estimator applicable to whole-body dynamics in contact-rich behaviors. Our estimator is based on the Unscented Kalman Filter (UKF), with modifications that proved essential in the presence of strong contact non-linearities combined with unavoidable model errors. Instead of the standard UKF weighting scheme, we use uniform weighting which is less susceptible to samples that violate unilateral constraints. We also develop a rich model of process noise including noise in system parameters, control noise, as well as a novel form of timing noise which makes the estimator more robust. The method is applied to an enhanced Darwin robot in walking as well as pseudo-walking while lying on the floor. Estimation accuracy is quantified via leave-one-out cross-validation, as well as comparisons to ground-truth motion capture data which is not used by the estimator. A full update takes around 7 msec on a laptop processor. Furthermore we perform the computationally-expensive prediction step (involving 210 forward dynamics evaluations in the MuJoCo simulator) while waiting for sensor data, and then apply the correction step as soon as sensor data arrive. This correction only takes 0.5 msec. Thus the estimator adds minimal latency to closed-loop control, even though it handles the whole-body robot dynamics directly, without resorting to any of the modeling shortcuts used in the past.**

I. INTRODUCTION

Accurate state estimation is a prerequisite for developing robust and efficient feedback controllers for modern robots. Such robots have high-dimensional dynamics that cannot be modeled exactly, and are subject to strong non-linearities (particularly due to contact phenomena) which make estimation challenging. Our goal is to develop a general estimator that works in real-time and yields results that can be safely fed into a whole-body feedback controller.

The starting point of our work (as well as almost every other approach to estimation) is recursive Bayesian inference. However, with the exception of a few special cases, we can not represent the true Bayesian posterior. Thus, we must resort to some approximation, seeking a favorable trade-off between approximation accuracy and computational efficiency. The two most popular approximations are the sample approximation used in particle filters, and the multivariate Gaussian approximation used in Extended Kalman Filters (EKF) and Unscented Kalman Filters (UKF). Particle filters have the theoretical advantage that in the limit of infinitely many samples, they converge to the true posterior, but the number of samples required for high dimensional models many samples proves difficult for real-time computation.

Gaussian approximations on the other hand can become very inaccurate and cause filter divergence. The UKF is known to alleviate this problem compared to the EKF, however it comes without any stability or optimality guarantees for general nonlinear systems, and can diverge in practice. Filter divergence if often caused by over-confidence, which in turn reflects insufficient uncertainty in the dynamics or sensor models. Indeed we have found that the traditional UKF in our setting usually diverges. In order to make the filter stable and accurate, we introduce two modifications. First, we weight the samples computed by the UKF uniformly, instead of using the standard weighting schemes derived from considerations of higher-order accuracy. Uniform weighting increases robustness to samples that violate contact constraints and require the filter to postulate very large external force in order to explain them. Second, we develop a stochastic dynamics model with several noise sources: system parameter uncertainty obtained from system identification; timing noise that models the fact that sensor measurements accumulate during the time step; and control noise that models the fact that our actuator model is not perfect and there is some backlash. We show empirically that including all three noise sources results in more accurate estimation than including any two, any one, or none of them.

Another important concern is real-time performance. Even though the UKF is much faster than a particle filter, it still requires $2N + 1$ evaluations of the system dynamics where N is the state space dimensionality. Even a relatively simple humanoid such as the Darwin robot we use here has 26 DOFs (6 of them are unactuated torso DOFs) resulting in $N = 52$. Thus we need 105 dynamics evaluations per update, or in other words, our simulator has to run 100 times faster than realtime. We achieve this partly through parallel processing. The remaining speedup reported here is due to the efficiency of the MuJoCo simulator we use, and the way we warm-start the iterative contact solvers (which are found in all modern physics simulators, and are usually the computational bottleneck). Our estimator is able to handle the whole-body dynamics of the robot, and perform an update in about 7 msec on a laptop processor. Importantly, we hide most of that latency in the time when the computer is waiting for the next sensor data to arrive. This is possible because the UKF has the usual predictor-corrector form, and the predictor step does not require the new sensor data.

This work was funded by the US National Science Foundation.

[1]Department of Computer Science and Engineering, University of Washington, and Roboti LLC. {klowrey,todorov}@cs.washington.edu

[2]Department of Applied Mathematics, University of Washington. jdao913@uw.edu

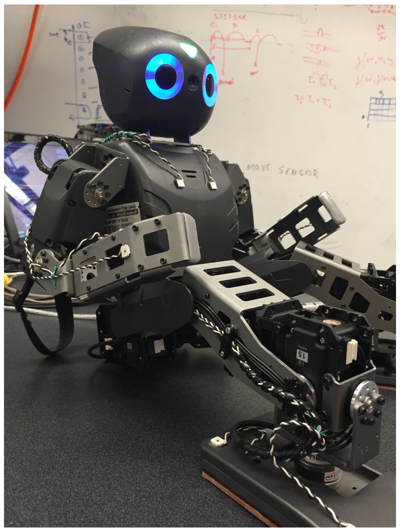

Fig. 1. A picture of the Darwin-OP robot used in experiments.

II. RELATED WORK

Although estimation is utilized in almost every robotic platform, performance and operating limitations frequently lead to compromised accuracy. This is usually the case when the dimensionality of the state starts increasing, or the dynamical system is difficult to model, such as during frequent contacts. We limit this section to the discussion of applied estimation techniques to humanoid robots instead of estimation in general.

A. Simplified Models

Frequently simplified models are used. Humanoids are often approximated as an inverted pendulum model that shifts its base during footsteps [1] [2]. This technique was originally developed to simplify the planning of walking, but similarly extends to estimation. Some model simplification makes the inherent assumption that the robot's movements are quasi-static in its manipulation of the environment, which is a common strategy [3] [4] [5]. This of course drops information about more complicated dynamic actions and motions. Additional information may be provided to the estimator through contact location [6]. As contacts are a critical phenomena of an under-actuated robot, planning trajectories featuring contact and providing this contact information allows an estimator to perform better, but requires prior planning.

B. Split Estimation

Another technique is to split the estimate between root orientation, using a high quality inertial measurement unit, and the joint state of positions and velocities [7] [8] [9][10]. This results in information loss between the joint configuration and the root, and may also be used in addition to a reduced model. This type of split estimation allows for increased drift as the root has less information to integrate regarding the joint's contacts with the world; this is usually overcome with vision and LIDAR systems providing accurate root position and orientation [11]. In a similar way, [12] is able to track a high degree of freedom human hand in a cluttered environment utilizing information dense vision and contact physics. However, these are expensive solutions to the problem, with LIDAR systems being relatively cost-prohibitive. These systems also rely on extremely information dense sensors, a prime example being [13] which runs at 1000Hz.

Many of the above techniques use optimization as the workhorse underlying the estimation. This may provide an accurate measurement, but may require additional data to increase accuracy, such as in a Moving Horizon framework [14]. This limits performance as now multiple optimization minimizations are needed; in non-linear systems this is often non-trivial. Prior work performing fixed lag estimation in this way meant that estimation updates had high latency after acquiring new sensor data, and sensor data may even be lost due to the computation time [15].

C. Non-linear Dynamics

While the Kalman Filter framework works exactly for linear systems, under-actuated robotics in the presence of contacts make for highly non-linear state transitions. Each contact is a discontinuous constraint upon some configurations of the state vector. These may be so non-linear that an Extended Kalman framework may be insufficient to describe state transitions: linearization near a contact may not capture the physical phenomena of an actual contact. This may be alleviated by 'softening' the contacts to allow for action at a distance, but this reduces the realism of the simulated robotic system and may reduce estimation accuracy. Prior work [15] required the use of softened contacts to remain differentiable, which allowed for incorrectly estimated sliding in some configurations. [16] uses contact switching to deal with contacts, but this method requires dynamically changing the model.

III. OUR METHOD

We define the problem as follows: we wish to estimate a robotic system's state $x_k = [q_k; v_k]$ where q_k and v_k are the generalized configuration (positions) and velocities of the degrees of freedom of the system at some time k. We utilize an Unscented Kalman Filter (UKF) to propagate sample points through the non-linear state transition function. We briefly describe our UKF as follows. We wish to estimate our state and state covariance matrix at some time k given our previous state at time $k-1$, which we denote as $\hat{x}_k$ and P_{ks}.

We generate a set of $2N+1$ sigma points–which we denote as χ_i for the ith point–where N is the size of our state vector x_{k-1}. Our sigma points are the set of points perturbed from the original as below; we may drop the $k-1$ term for clarity.

$$\begin{aligned}
\chi_0 &= \hat{x}_{k-1} \\
\chi_{i=1:N} &= \hat{x}_{k-1} + (\sqrt{(N+\lambda)P_{k-1}})_i \\
\chi_{i=N+1:2N} &= \hat{x}_{k-1} - (\sqrt{(N+\lambda)P_{k-1}})_{i-N}
\end{aligned}$$

where

$$(\sqrt{(N+\lambda)P_k})_i$$

is the ith column of the matrix square root (we use a Cholesky Decomposition), and λ is a constant. We then propagate each χ through the state transition function f which accepts the sigma point and controls u to find our state at time $k+1$.

$$\chi_{i,k} = f(\chi_{i,k-1}, u_k)$$

The first step in a Kalman process is the prediction. In a UKF, we sum each sigma point χ_i to create our *apriori* state $x_{k|k-1}$ and covariance $P_{k|k-1}$ estimate:

$$\hat{x}_{k|k-1} = \sum_{i=0}^{2N} W_i \chi_i$$
$$P_{k|k-1} = \sum_{i=0}^{2N} W_i [\chi_i - \hat{x}][\chi_i - \hat{x}]^T$$

where W_i are the chosen weights applied to each sigma point.

Similarly the vector of sensors measurements $\gamma_{i,k}$ can also be gathered through this Unscented transform process where the observation function h is used instead of the state transition function f.

$$\gamma_{i,k} = h(\chi_i)$$

These predicted sensor measurements and sensor covariance are also combined with a weighted sum, in addition to the cross covariance matrix.

$$\hat{z}_k = \sum_{i=0}^{2N} W_i \gamma_i$$
$$P_{z_k,z_k} = \sum_{i=0}^{2N} W_i [\gamma_i - \hat{z}_k][\gamma_i - \hat{z}_k]^T$$
$$P_{x_k,z_k} = \sum_{i=0}^{2N} W_i [\chi_i - \hat{x}_{k|k-1}][\gamma_i - \hat{z}_k]^T$$

These values along with the observed sensor data z_k are used to calculate the Kalman gain matrix K_k and subsequently calculate the *a posteriori* state and covariance estimate.

$$K_k = P_{x_k,z_k} P_{z_k,z_k}^{-1}$$
$$\hat{x}_k = \hat{x}_{k|k-1} + K_k(z_k - \hat{z}_k)$$
$$P_k = P_{k|k-1} - K_k P_{z_k,z_k} K_k^T$$

A. UKF Weights

While there are many techniques to choose the weights W_i used in the Unscented transform, we eventually settled upon even weighting as the most stable for our use case. Traditionally, any weights may be chosen so long as $\sum W_i = 1$, including negative weights in many use cases [17]. However, we found that the use of weights less than 0 or greater than 1 exacerbate unrealistic forces induced by the discontinuous state constraints that a robot experiences through contact. Said another way, if a particular sigma point χ_i is in violation of the contact constraints, there is no particular rule to prevent it from inducing bad data into the prediction and correction of the UKF except by having weights $0 < W_i < 1$ where $W_i = 1/N$.

We attempted to preserve the traditional weighting scheme by adjusting those sigma points known to be in violation of contacts by using an adaptive line search style re-weighting of the perturbance $(\sqrt{(N+\lambda)P_k})_i$ whilst also adjusting the particular weights W_i according to [18], but this proved to not be as successful as the even weighting scheme.

B. Modeling Error & Process Noise

Modeling error is an ever present difficulty in accurately predicting robotics. Simple errors can manifest themselves as constant biases in the system's state transition function $x_k = f(x_{k-1}, u_k) + \epsilon$ such as Gaussian noise; these can be readily reduced in standard system identification procedures. More systematic errors, especially those involving modeling the physical phenomena of contact can dramatically affect state transitions with even minor changes.

Additionally, the real world also suffers from the affliction of process noise, which is any difficult-to-model effects that affect the state vector. While known sources of process noise can be estimated along-side the state in an augmented UKF [19], complicated physical systems such as under-actuated robotics may not be able to quantify this relationship. Thus we have two sources of discrepancy between our modelled state transition function: the process noise and modeling error. To accurately estimate a robot's state we need to overcome these sources of error.

We may represent an ideal state transition function as f_{ideal}, which is a function of the state and controls that is dependent on our model. We can rewrite it as such:

$$x_k = f_{ideal}(x_{k-1}, u_k) = f_{model}(x_{k-1}, u_k) + q_k(x_{k-1}, u_k)$$

where f_{model} is our known, modeled state transition and q_k is our unknown process noise function that may be time varying. We first make the observation that forces applied along the degrees of freedom of the robot can mimic the process noise or overcome the difference between real and modelled state transitions, similar to the changes induced by q_k. Other sources of error could be changed through the modification of the underlying model governing f_{model}, if only we knew better model parameters.

These two ideas coupled with the UKF's sample points gives us the opportunity to sample from these sources of noise:

$$\chi_{i,k} = f(\chi_{i,k-1+N_k}, u_k + n_u, M + n_m)$$

where n_u and n_m are control noise and model noise respectively.

The parameters that we add noise to are the control vector, model mass parameters, and the time index as zero mean Gaussian noise. The time aspect is inspired by the knowledge that our sensor readings are not instantaneously time synced across all sensors coming from hardware. As such the variance of the Gaussian noise is the variance of the dt (delta timestep) of the collected sensor data. The model mass noise parameter was determined from a system identification procedure of measuring the robot mass in different configurations to discover any measurement discrepancies. This mass noise is distributed randomly among the different body components of the model whilst still normalizing to the expected mass of the robot (2.97 kg). Mass, of course, is an underlying parameter that effects all aspects of the state transition, and from prior work [20], we found it to be critical in generating robust trajectories. Finally, varying the controls is most analogous to changing the degree of freedom forces to mimic process noise, but also helps to overcome the difficulty of modeling backlash and deadbands found in geared, position controlled motor systems.

TABLE I
APPROXIMATING PROCESS NOISE

Noise Source	Standard Dev.
Time	0.00096 seconds
Mass	0.012 kg
Control	1e-4 radians

In the end, these noise sources serve two functions. One is to approximate the stochastic process noise and overcome the modeling error inherent in real world robotics, and secondly, to encourage our UKF formulation to develop appropriately descriptive state covariance matrices. Although these parameters could be included in the x_k state vector of positions and velocities as additional state terms, sampling from a distribution for the noise does not increase computational load (as we do not have additional sigma points) as well as encourage the state covariance matrix to richly describe the state variability at time k. If all the samples in a simulator integrate forward in time with the same deterministic state transition function, the covariance matrix hardly provides useful information. While process noise is frequently sampled and added to each sigma point's state vector, this approach leads to constraint violations which can contribute additional inaccuracies to the estimate.

It should be noted that adding this kind of modelling noise is not apparent in estimation techniques that are not sampling based. Control and time noise are able to be included in frameworks such as a Moving Horizon Estimator or EKF, but it is less obvious to include mass noise when your framework depends on state and control linearization to calculate Jacobian matrices.

C. Sensor Noise

Another critical detail of any estimator is quantifying the sensor noise. As highly accurate sensor hardware is prohibitively expensive, we overcome sensor noise through a combination of system identification and sensor weighting. Gathering statistics on sensor noise such as mean and variance is standard procedure for any robotics application. Using these values, we have a baseline for the diagonal of the sensor covariance matrix P_{z_k,z_k}. This, in effect, tells the estimator how much to trust the sensor readings from the robot hardware, and is a set of tunable values acting as sensor weights. Some values need to be adjusted due to large dynamic range of the sensed values during operation, as well as the sensitivity of the estimator to some sensors. For example, force and torque sensors can capture highly discontinuous events that can cause an estimate to over-correct, while a gyroscope is an intrinsic sensor centered at zero.

TABLE II
ASSUMED SENSOR NOISE VARIANCE. WE FIRST MEASURED THE SENSOR VARIANCE FROM COLLECTED DATA, AND ADJUSTED THESE VALUES EMPIRICALLY TO BENEFIT ESTIMATOR PERFORMANCE.

Accelerometer	1e-4
Gyroscope	1e-6
Force	1e-3
Torque	1e-2
Joint Position	1e-7
Joint Velocity	1e-5

IV. PRACTICAL CONSIDERATIONS

Estimators need to provide data accurately and quickly to allow for successful robot operation of dynamic controllers. As such, we utilize the UKF over other means due to the performance benefits of a specific, controlled sample set – unlike a Particle Filter – while maintaining accuracy in dynamic situations such as making and breaking contact. In the following section we describe a few methods used to achieve real-time performance, which we define here as completing all estimator computations within the time between sensor updates.

V. ROBOTIC PLATFORM

TABLE III
ESTIMATOR TIMING

Machine	Prediction Time (ms)	Correction Time (ms)
Desktop	5.2 ± 1.46	0.57 ± 0.18
Laptop	6.74 ± 1.22	0.66 ± 0.21

Our simulation environment is based on the MuJoCo physics engine, which allows for fast evaluation of dynamical systems even with contact [21]. While MuJoCo is nominally designed to allow for fast finite differencing, the UKF style of estimator is computationally similar to a finite difference procedure. Our UKF needs to evaluate $2N+1$ sample points centered on one centroid point through forward dynamics. As such the accelerations calculated for the centroid point can be used to warm-start the dynamics evaluations for the rest of the sigma points, reducing computational overhead through data reuse.

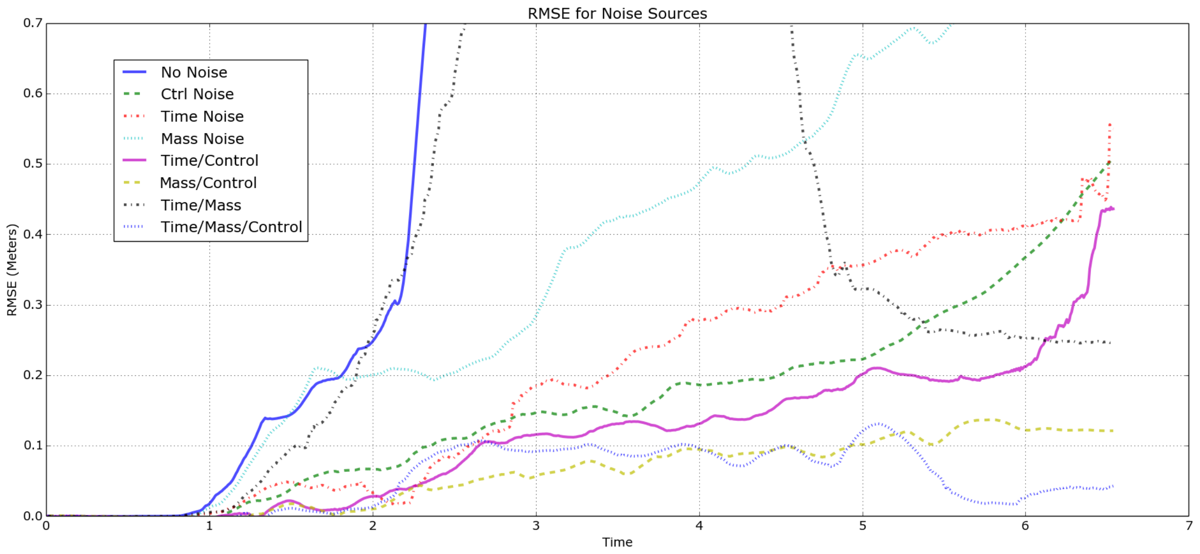

Fig. 2. Shown is the Root Mean Squared Error for a walking trajectory with the estimator utilizing different sources of noise. Lines that leave the plot area usually have experienced an estimation divergence. The Phasespace markers used to calculate the RMSE were located on the Darwin-OP torso to isolate any drift, as limb deviations could contribute to high RMSE without causing drift. It should be noted that the estimation performed here did not use the Phasespace data in the estimation loop; drift would be nearly inevitable without some similar position based sensor.

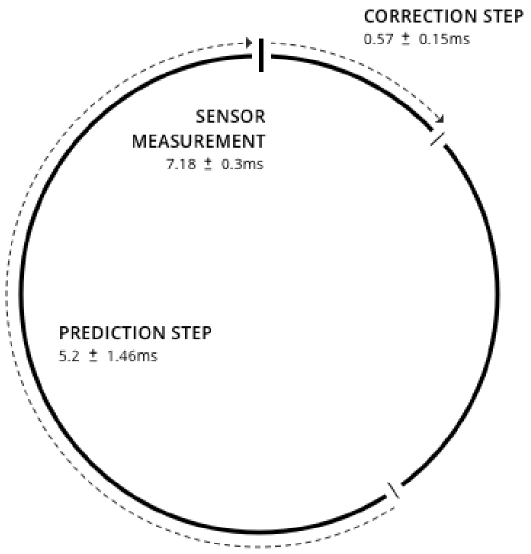

Fig. 3. Our estimation loop consists of a prediction phase and correction phase that must happen within the sensor measurement loop. As the sensing happens consistently, we can perform our prediction step before receiving sensor data (where normally we would use the measured time delta to integrate in our state transition), which leaves the Kalman correction step to happen after the sensors have provided data. As our process loop happens to be fast enough, we have time to evaluate from a feedback controller based on the calculated estimate between the correction and prediction steps.

Further performance improvements arise from full utilization of modern CPU architectures. As each sigma point is evaluated independently from the others, our computations are highly parallelizable which we take advantage of through multithreading the dynamics evaluation. Finally, because our robotic platform's sensor updates have consistent timings, we opt to complete the prediction step of the Kalman Filter *before* the arrival of new sensor data, allowing a new state estimate to be completed in less than 1 millisecond after new data. This is intended to benefit future dynamical control efforts by being able to utilize our estimated state as soon as possible after new sensor data is received.

The Darwin-OP humanoid biped produced by Robotis Ltd. is a 26 degree-of-freedom robot – 3 root positions, 3 root orientations, and 20 actuated joints. Each actuator is a MX-28 servo motor with position measurement of 12-bit resolution over 2π radians that is position controlled. Prior work utilizing this platform led us to encounter a number of technical difficulties in using the platform as is, as such we modified the original hardware to increase sensing capability and compute performance as follows.

The original on-board low-power x86 CPU was removed to make way for a higher performance inertial measurement unit (IMU), the PhidgetSpatial High Resolution, which connected to an installed USB hub alongside the original Darwin-OP motor sub-controller board. Additionally, the original force resistive sensing feet were replaced with two ATI Nano-25 6-axis force/torque (F/T) sensors. These F/T sensors need to be connected to large signal amplification boxes off the robot; as such their cables were routed to the back of the Darwin-OP torso and bundled with the USB cables necessary to communicate with the IMU and sub-controller. These amplification boxes were then routed to a analog to digital signal converter which also communicates over USB. Finally, 16 Phasespace Impulse IR tracking markers were installed on the torso and limbs of the robot to serve as ground truth tracking for estimator verification. These markers needed to be powered by a large wireless pod which was also off the robot platform.

As the motors, IMU, and F/T sensors all communicate over USB, we have the option to link our modified Darwin-OP platform to any computer system we would like. For our purposes we developed and tested our estimator on both a desktop with an Intel Core-i7 5930k CPU as well as a laptop containing a Core-i7 4710HQ processor. While our estimator's evaluation proved quick enough on the laptop, we also tested on the larger desktop processor to gauge additional performance improvements. These details are discussed in section IV.

VI. EXPERIMENTS

We tested our estimator on data collected from our robotic platform through an open-loop, fast, ZMP-style walking (software provided by the manufacturer) across a flat surface as well as the same walking process while the robot was lying face down. The walking takes approximately three steps every two seconds. The face down test was to examine reduced sensor usage – as the robot was not standing on its contact sensors – as well as increasing the contact surface. As an estimator attempts to find an accurate state configuration given sensor data, we stressed our estimator with these kinds of contact phenomena during dynamic motion.

A good estimator would have good sensor accuracy while also avoiding drifting of the robot. Accuracy is examined with Leave-One-Out tests that regenerate specific sensor values when those sensors are left out of the estimation process.

We validated our drifting tests by using the Phasespace markers as ground truth (and not in the estimator) after appropriately filtering out poor marker readings as not all markers were visible all the time. We attempt to validate two things. First, our estimator should not have a high root mean squared error between its predicted Phasespace markers and the actual measured marker data, where the error at time k is the average Euclidean distance for the set of used markers.

$$RMSE_k = \frac{1}{N_{markers}} \sum_{markers} \|z_k - \hat{z}_k\|$$

To show our estimator also works over longer periods of time as well, we perform the same RMSE test with our process noise model and including the Phasespace markers in the estimator. Without the markers in the estimator, any estimator would be likely to drift over time.

Where here the z_k terms are only the marker measurements instead of the full sensor vector. We calculate this $RMSE$ for different configurations of model noise to show the combined effects. Secondly, we attempt to validate our model noise parameters by scaling its values and calculating an average $RMSE$ for the entire trajectory to show how changing these model noise terms affects the drift amount.

As we did not utilize the external position tracking of the Phasespace markers in our estimator, we initialized the estimator with our simulated models data after the simulation was configured with the same initial control vector of the actual robot. This meant that the estimator initialization began close to the actual true starting position – we assume the initialization of the robot hardware begins near the origin of the simulator's coordinate system in a robot centric fashion. Without any root position and orientation tracking system, divergence would be otherwise inevitable as the root of the robot is not directly observable.

VII. RESULTS

Our estimation technique utilizing modeling noise successfully tracks our robot through dynamical motions with numerous contacts. Importantly, it does so while keeping drift low even while walking across a flat surface without external position sensing used for the estimate. We had implemented a basic EKF for comparison, but the filter diverged almost instantly due to heavy non-linearities from contacts. Thus, for brevity we only include data from our main UKF in this paper.

Figure 2 shows the effects of different configurations of model noise; the best performing configuration was with time, mass, and control noise all utilized for both the walking trajectory and fallen position. This intuitively makes sense as allowing for more noise in the estimator gives it more possible states to correct for. The plot shows one particular walking sequence of eight steps over five seconds with a drift of about 4 cm; the total distance travelled was 35.2 cm. Additionally, some estimator configurations with less noise would actually diverge. This is especially apparent in the configuration without contributing noise: the robot falls over in a configuration the Kalman update cannot correct for, leading to an over-correction and blow-up. However, it is important to note that the estimator does perform very well when given all sources of noise. In a fallen position, there are many contacts occurring, with even more dynamic motion and contacts coming from the walking sequence. It is a great example of the estimator's strong ability to deal with multiple contacts.

TABLE IV

AVG. RMSE WITH MISSING SENSORS. WE FIND THAT OUR ESTIMATOR IMPROVES FROM USING ALL AVAILABLE SENSORS (STILL NOT INCLUDING THE PHASESPACE MARKERS).

Sensor	Avg. RMSE (Meters)
All	0.0407
No Accl	0.0493
No Gyro	7.76e+14
No Force	0.0434
No Torque	0.0809
No J.Pos	2.86e+9
No J.Vel	0.4215

Figure 4 shows the estimator's attempts to predict missing sensor values. The ability to predict these are necessary to demonstrate the physical consistency of the estimated state. This shows which sensors are critically necessary for accurate estimation. For example, the gyroscope is poorly estimated when left out causing large errors, but predicted almost exactly when provided to the estimator. This is because the gyroscope is a low variance sensor, and can

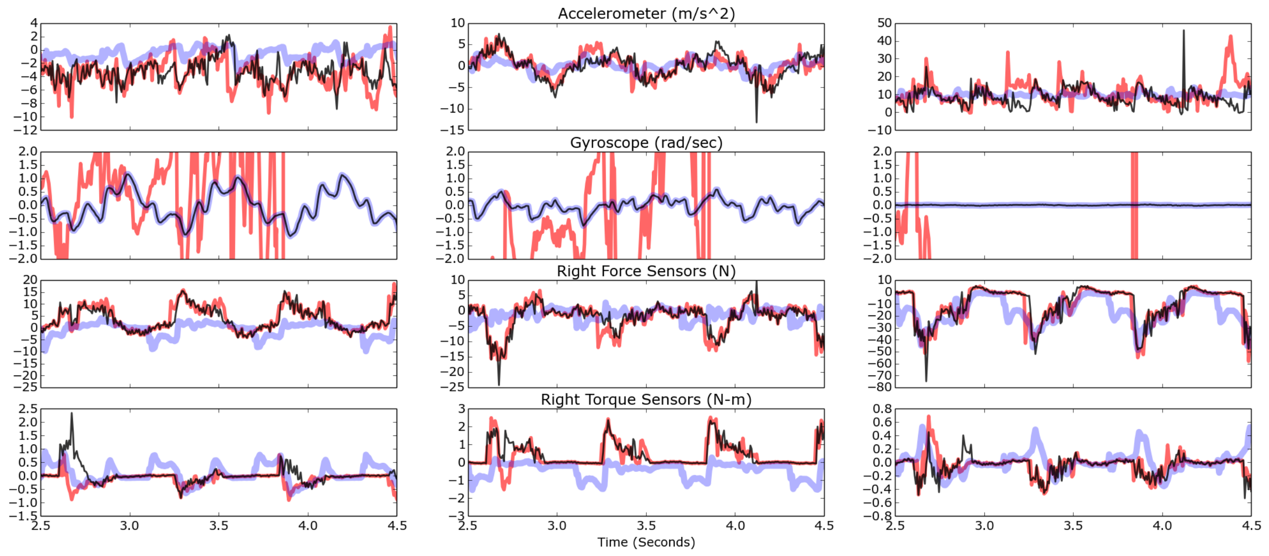

Fig. 4. We show the results of a Leave-One-Out (LOO) test where the walking robot was estimating its state without the use of the indicated sensor, and we plot the prediction for the missing sensor. The thick blue line is the recorded sensor value from the hardware, while the red line is the value predicted when the sensor was disabled during estimation. For reference, the black line is the predicted sensor value with full sensors. This LOO test shows a correspondence between the sensor weighting and how closely it is matched.

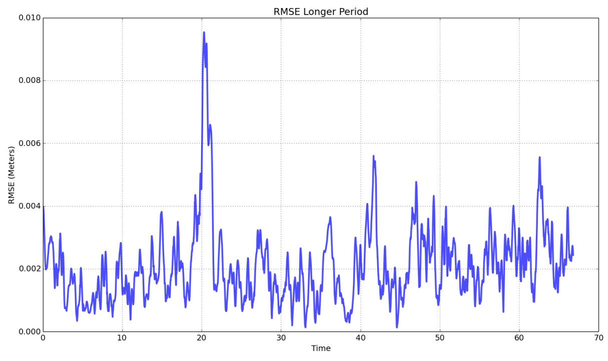

Fig. 5. As drift is likely without high fidelity positional sensors, such as LIDAR, we include this example of our UKF tracking over a minute including the Phasespace sensors in the estimator. RMS error of this size can most likely be attributed to Phasespace calibration and system identification errors.

thus be trusted more. Further insights are to be had when looking at the F/T sensors. The matching is poor, but not off base when the sensor is left out, especially along the X, Y axes. The Z axis values match more closely as they are more critical to drift-free forward walking and standing; these terms match more closely. The discrepancy of the X, Y axes can be partially explained by the sensitivity of modelling errors regarding these particular sensors. Small errors in contact geometry, hardness, or frictional coefficients may greatly affect these predicted values. By performing the Leave-One-Out test on all sensors, we gain insight on to how well they can be trusted, and thus how large the variance should be in the estimator. We also show how the noise parameters used by the estimator can also hurt the performance in figure 6 and table IV. The plot shows how scaling the sources of noise can actually increase RMSE values when too small or too large. Too little noise restricts the estimator's possible states, preventing it from making accurate corrections. Too much noise will throw off the estimator, causing it to diverge. The table calculates the average RMSE when specific sensors are not used in the estimation; there is divergence when the joint position and gyroscope sensors are not used.

VIII. CONCLUSIONS AND FUTURE WORK

We demonstrate our full body modified UKF estimator on a 26 DOF humanoid robotic in dynamic tasks with contacts. We are able to achieve low drift and accurate state and sensor reproduction with real-time performance characteristics. This was achieved without needing to 'soften' the rigid body contacts or splitting the estimate into two for root and joints, and without a reduced model. We were also able use only kinematic sensors at a reasonable sampling rate.

Further improvements could still be made. Though our attempt at handling estimate constraint violations (see section III-A) resulted in inaccurate estimates, adjusting the UKF's sigma points to avoid un-realistic violations could improve the estimate. Instead of the line-search style approach, one idea is to project the sigma points from the previous timestep towards their new positions; this may allow some of the benefits of a MHE to occur.

Additionally, as our estimator works in real-time, future work should use this UKF estimator for dynamic feedback controls instead of the open-loop ZMP style walking we used here. More dynamic behavior may stress the estimator in its current configuration more; additional sources of modeling noise, including geometry, inertia orientations, and actuator parameters may be needed to improve its performance.

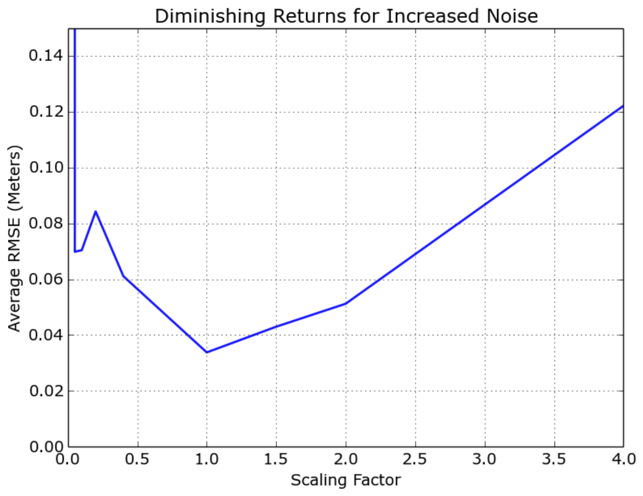

Fig. 6. Here we show how our noise parameters contribute to estimator performance. Too little or too much added noise from time, control, and mass can cause a reduction in performance – in this case more drift. The RMSE values are averaged along the time of the trajectory to give us a measure of the mean deviation for the entire run.

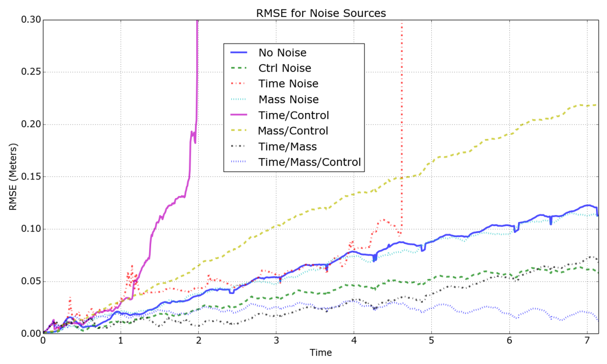

Fig. 7. RMSE plot of when the robot was started in a fallen position. The walking sequence was still applied to induce dynamic motion and additional contacts. The effects of our estimator are not as apparent in this plot, although the configuration with all sources of noise still performs the best. This may be due to the fact that the robot does not traverse any distance, thus creating less chance for drift. It should be noted that some configurations of noise lead to blow-up in this fallen position; the extra contacts of the robot laying on the ground may have allowed estimator to go unstable from the noise inducing increased constraint violations.

REFERENCES

[1] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Yokoi, and H. Hirukawa, "Biped walking pattern generation by a simple three-dimensional inverted pendulum model," *Advanced Robotics*, vol. 17, no. 2, pp. 131–147, 2003.

[2] T. Sugihara, Y. Nakamura, and H. Inoue, "Real-time humanoid motion generation through zmp manipulation based on inverted pendulum control," in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, vol. 2. IEEE, 2002, pp. 1404–1409.

[3] A. Bicchi and V. Kumar, "Robotic grasping and contact: a review," in *IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA '00*, vol. 1, 2000, pp. 348–353 vol.1.

[4] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on Control Systems Technology*, vol. 18, no. 2, pp. 267–278, Mar. 2010.

[5] A. Okamura, N. Smaby, and M. Cutkosky, "An overview of dexterous manipulation," in *IEEE International Conference on Robotics and Automation, 2000. Proceedings. ICRA '00*, vol. 1, 2000, pp. 255–262 vol.1.

[6] 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*, vol. 17, pp. 17–24, 2013.

[7] S. Kuindersma, R. Deits, M. Fallon, A. Valenzuela, H. Dai, F. Permenter, T. Koolen, P. Marion, and R. Tedrake, "Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, 2016.

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

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

[10] M. Benallegue, A. Mifsud, and F. Lamiraux, "Fusion of force-torque sensors, inertial measurements units and proprioception for a humanoid kinematics-dynamics observation," in *2015 IEEE-RAS International Conference on Humanoid Robots*, Nov. 2015.

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

[12] N. Kyriazis and A. Argyros, "Physically plausible 3d scene tracking: The single actor hypothesis," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 9–16.

[13] M. Falln, M. Antone, N. Roy, and S. Teller, "Drift-free humanoid state estimation fusing kinematic, inertial and lidar sensing," in *2014 14th IEEE-RAS International Conference on Humanoid Robots*, Nov. 2014.

[14] E. L. Haseltine and J. B. Rawlings, "Critical evaluation of extended kalman filtering and moving-horizon estimation," *Industrial & engineering chemistry research*, vol. 44, no. 8, pp. 2451–2460, 2005.

[15] K. Lowrey, S. Kolev, Y. Tassa, T. Erez, and E. Todorov, "Physically-consistent sensor fusion in contact-rich behaviors," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 1656–1662.

[16] N. Rotella, M. Bloesch, L. Righetti, and S. Schaal, "State estimation for a humanoid robot," in *2014 14th IEEE/RJS International Conference on Intelligent Robots and Systems*, Sept 2014.

[17] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[18] P. Vachhani, S. Narasimhan, and R. Rengaswamy, "Robust and reliable estimation via unscented recursive nonlinear dynamic data reconciliation," *Journal of process control*, vol. 16, no. 10, pp. 1075–1086, 2006.

[19] Y. Wu, D. Hu, M. Wu, and X. Hu, "Unscented kalman filtering for additive noise case: Augmented versus nonaugmented," *IEEE Signal Processing Letters*, vol. 12, no. 5, pp. 357–360, May 2006.

[20] K. Mordatch, I. anbd Lowrey and E. Todorov, "Ensemble-cio: Full-body dynamic motion planning that transfers to physical humanoids," in *2015 IEEE/RJS International Conference on Intelligent Robots and Systems*, Sept 2015.

[21] E. Todorov, T. Erez, and Y. Tassa, "Mujoco: A physics engine for model-based control," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2012, pp. 5026–5033.