State Estimation for legged robots using proprioceptive sensors

Thesis by

Naman Gupta

CMU-RI-TR-24-994

In Partial Fulfillment of the Requirements for the
Degree of
Masters of Science in Robotics

Thesis Committee:
Howie Choset (Co-advisor)
Matthew Travers (Co-advisor)
Michael Kaess
Ishani Chatterjee

To my parents and grandparents

Abstract

Mobile robots need good estimates of their state to perform closed-loop control in structured and unstructured environments. A number of existing algorithms rely on data fusion from multiple sensors to compute these estimates. This work focuses on state estimation using sensors which only measure information (acceleration, motor speed, joint angles) internal to the robot – proprioceptive sensors – since measurements of external features (light intensities, distance measurements, sound amplitude) may not always be reliable. Wheeled robots conventionally use IMUs and motor encoders for robust proprioceptive odometry. Legged robots, however, interact with their environment through intermittent foot-ground contacts which introduces additional noise in the IMU and joint encoder measurements making this problem challenging. We implement an Extended Kalman Filter (EKF) based state estimator which uses foot-ground contact information to counteract noisy sensor measurements from the IMU and motor encoders. This method has been implemented on simulation based quadruped and on an actual hexapod system.

Table of Contents

List of Illustrations

List of Tables

Nomenclature

$\overline{x}$.	a variable with an overline is only a prediction and not a full estimate.
X.	a variable in uppercase and bold is a Matrix .
x.	a variable in lowercase and bold is a vector .
COM.	Center of Mass.
EKF.	Extended Kalman Filter.
IMU.	Inertial Measurement Unit.
UKF.	Unscented Kalman Filter.

Chapter 1

Introduction

Robotic systems are used for carrying out a different variety of tasks in different kinds of environments. In order to carry out these tasks successfully and efficiently, one needs to control them using feedback about the state of the robot. The state of the robot has to be estimated either with respect to a feature in the environment or with respect to their current position. This thesis specifically focuses on the latter method which is also called odometry. The odometry method implemented in the paper does not assume anything about the number of legs, the type of gait or the kind of terrain. The results are shown on two kinds of platforms, quadruped in simulation and on actual hexapod system.

Generally, the different methods of state estimation of the robots are based on their structure and the environment they operate in. The structure of the robot usually includes, (i) kinematics: the design of the robot (links and joints), (ii) dynamics: the expected motion of the robot under an influence of external or internal force or torque. The environment of the robot can greatly influence the motion of the robot by introducing an external force or torque on the system. The inaccuracies in determining the exact structure and environmental disturbances lead to inaccuracies in the prediction of the motion of the robot. In other words, the expected motion of the robot is not exactly equal to the actual motion of the robot.

For instance, a manipulator arm might struggle picking up objects of varying shapes, sizes and materials in an open loop control because of inaccuracies in the robot dynamics model and controller. Therefore, we need to close the loop using the feedback from the sensors. Sensors tend to be noisy and some of them do not give the full information about the robot motion. Hence, robot motion will have to be estimated by fusing multiple sensors in such a way that not only the full state is observable but also noise of each sensor is corrected by other sensors. The interactions of robots with the environment also need to be estimated to infer how well the robot is performing a task.

The state of the robot comprises of knowledge of robot motion and of features in the environment which, if known, fully describe the robot's motion over time. State estimation is generally done by combining the dynamic model of the robot with a measurement model. The dynamic model of the robot gives an estimate of the expected state and that expected state is corrected using the measurement from different kinds of sensors on the robot which gives additional information about one or more state quantities of the robot. The sensors on the robot can be broadly divided into two sets:

1. Introceptive/Proprioceptive Sensors: Measure information from within the robot's structure.For instance, accelerometer, gyroscope, wheel odometer.
2. Exteroceptive Sensors: Measure information from outside. For instance, camera, time of flight transmitter/receiver, Global Positioning System (GPS).

Although more sensors enrich the information content recovered about the state of the robot but this comes at a greater post processing computation and payload cost. In a way, state estimation is about making the best of the sensors the robot has. The challenge of estimating the state of the robot can vary depending on the model of the robot, the sensors used and the amount and type of noise associated with them. It is also important to note that certain sensors cannot describe the state of the robot completely and hence, observability analysis needs to be done to conclude which states are observable, if any.

Legged robots ranging from a hopper robot to a multi-legged system like all other robotic systems need a state estimation method too. The reason why state estimation is challenging on such robots are the following:

1. The dynamics are highly non-linear and there are inaccuracies in calculating a dynamic model mathematically.
2. The state space of such a robot is large because it includes not only the pose of the robot's chassis but also the pose of the joints of its feet.
3. The system is interacting with the environment via multiple intermittent ground contacts and impacts which makes sensors more noisy.

To solve these problems, today legged robots use a variety of sensors ranging from IMUs, contact sensors, time of flight sensors and cameras to do both the prediction

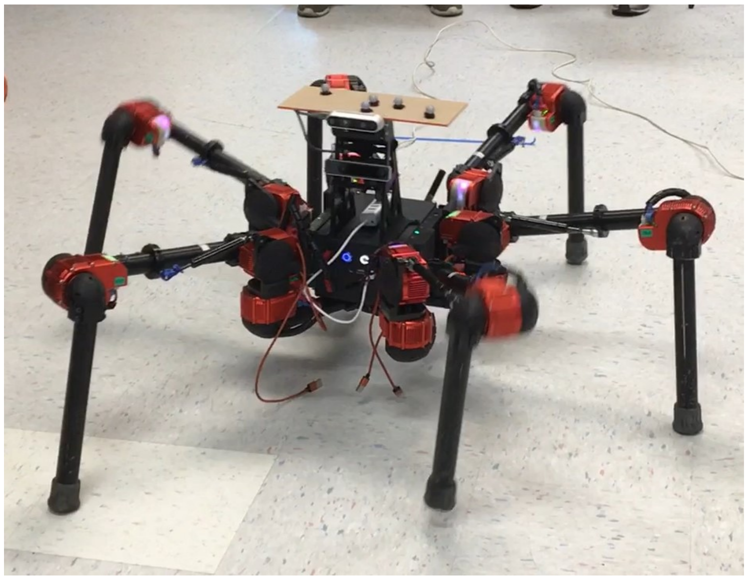


Figure 1.1: **Hebi Hexapod robot:** The robot built by Hebi Robotics using X-modules which work on the principle of series elastic actuators.

and correction step. Such a method is referred to as sensor fusion which predicts and corrects the state of the robot using different sensors. Typically, prediction is done using the dynamics model of the robot and correction is done using observations given by a sensor. Since the dynamics of a legged robot is highly non-linear and linearization of the dynamics is required for EKF, an easier dynamics model is adopted by using an IMU for prediction. The joint encoders are used for correction to perform the full state estimation.

The IMU measures the linear acceleration and angular velocity which are integrated over time to predict the position, velocity and orientation of the robot. The joint encoders are used to calculate the feet positions (only the ones which are in contact with the ground) and update the position and orientation of the robot. Contacts for the legs are calculated using joint torques which give a good enough estimate removing the need for having physical contact sensors on the feet.

The papers ([1],[2],[3],[4]) which make use of only proprioceptive sensors show that with the use of sensors and the state space used in this paper, the absolute position and yaw angle are unobservable. Even though they are unobservable, the error between estimated and ground truth in absolute position and yaw angle seemed to reduce the drift under limited slippage, as compared to only using an IMU for estimation.

Contributions A state estimator that fuses the information from an IMU and kinematics has been adopted and tested on a HEBI hexapod (6-legged) robot (as shown in figure 1.1) and the results have been closely verified with the ground truth. The ground truth was collected from an external motion tracker system and measurements were taken from the on-board IMUs. Before the algorithm was adopted on hardware, the experiments were done on a 4-legged robot in simulation where ground truth and measurements were taken from Matlab-Simulink. This state estimator uses only proprioceptive sensors. There are no assumptions made regarding the gait and number of robot legs and it is immune to little foot slippage caused either due to the gait or terrain.

The rest of the thesis report is divided in the following sections, (i) Related work followed by (ii) the the details of the model of the robot on which the experiments were carried out. Further, (iii) the state estimation algorithm is described theoretically with the pseudo-code and finally, (iv) Results are discussed both quantitatively and intuitively.

Chapter 2

Background and Related work

State Estimation is a probabilistic approach to robotics which intends to estimate the state of the robot in terms of probability distributions. Such an estimated state is also referred to as the belief state of the robot. Most common probability distributions used for state estimators are Gaussian Distribution and are estimated using conditional probabilities over the observed variables referred to as measurements.

2.1 A primer on probability and state estimation

A belief state is posterior probability over state variables conditioned on the available data. The available data is typically of the past control inputs $u_{1:t}$ and the past observations acquired from the sensors, $z_{1:t-1}$. Mathematically a belief over a state variable x_t can be represented as follows:

$$\overline{bel(x_t)} = p(x_t|z_{1:t-1}, u_{1:t})$$

Since this uses only past information and not current observation, this is referred to as "prediction" in the context of probabilistic filtering. It is important to note that although we have control input at the time t, we don't have the observations at time t because we are predicting the state of the robot at time t. The observations will be recorded once the robot moves and this observation will be used to correct the prediction which makes up the next step of probabilistic filtering called the "correction or measurement update".

The common algorithm used for probabilistic filtering is a Bayes filter. It is a recursive method used to calculate beliefs at time step t from beliefs calculated at previous time step t-1. In order to account for the previous beliefs, we need to multiply the probability of previous belief and the probability where it would end up with the new control input. This prediction will then be corrected using the observation, z_t by multiplying the above prediction with probability that the new observation may be observed. This results in the following equation:

$$bel(x_t) = \eta p(z_t|x_t) \underbrace{\int p(x_t|u_t, x_{t-1}) bel(x_{t-1}) dx}_{\overline{bel(x_t)}} \tag{2.1}$$

The product of two probabilities might not lead to a probability, that is, might not integrate to one. Thus, we introduce a normalization constant, η to make sure that the integration of the beliefs over all states x_t equal to one. It should be noted that we assume Markov property of the state and therefore only consider the previous state, current control input, and current observation to predict and update the current state.

There are variety of filters which are derived from Bayes filter with different set of assumptions on the basis of process model, measurement model and noise models. However, the goal of these filters is the same which is to minimize the effect of noise on our state estimation. In order to solve this problem, a prior for the noise of each sensor is assumed. Since it is not exactly known which type of distribution is noise associated with, one or more of the filtering methods might be applicable to a particular system. Based on the assumptions and computational complexity, a method is chosen for a particular robot. Few of the common Bayes filter based state estimation methods are summarized in the Table 1.1.

The Extended Kalman Filter works on the basis of linearization of non-linear process dynamics and non-linear measurement dynamics. Unscented Kalman Filter does not linearize the models and thus non-linearity of the model is captured by sampling over the probability distribution. Particle filter which is based on Monte Carlo method does not assume anything about the probabilistic distribution of the belief state.

2.2 State Estimation for Legged Robots

The Extended Kalman Filter (EKF), Unscented Kalman Filter (UKF) and Particle Filter satisfy the assumptions for performing state estimation on legged robots. However, particle filter is very computationally intensive due to the non-linearity of dynamics and high dimensional state space of such robots. A particle filter might need large number of samples because of the number of moving parts in the system and constant interactions with the environment which are difficult to model. EKF and UKF are thus, a better fit for such a problem.

Method	Process and Measurement Model	Noise Distribution	Computational Complexity
Kalman Filter	linear	Gaussian	$O(\mathrm{n}^2 + k^{2.8})$
Extended Kalman Filter	non-linear	Gaussian	$O(\mathrm{n}^2 + k^{2.8})+$ $O(f(x)) + O(h(x))$
Unscented Kalman Filter	non-linear	non-Gaussian	$O(\mathrm{L}^2(n^2 + k^{2.8})+$ $O(f(x)) + O(h(x))$
Particle Filter	non-linear	non-Gaussian	$O(M \log M)$

Table 2.1: Different types of Bayes filter with their assumptions and computational complexities for each time step of running these algorithms(n is the dimension of x_t, k is the dimension of z_t, L is the number of weights and M is the number of particles sampled.)

Although there has been work done using simplified dynamics model of the legged robots (like Spring Loaded Inverted Pendulum (SLIP) [5] and Linear Inverted Pendulum Model (LIPM) [6][7]), there hasn't been many papers on state estimation of legged systems for number of legs greater than two using such a dynamics model.

A good state estimator for a legged robot should be robust to different kinds of terrain, gaits, locomotion speed and number of legs. This is ensured by skipping the dynamics model which includes such parameters and instead using IMU as a process dynamics model. It should also take into account the fact that the motions are periodic over an interval and that the system is interacting with the environment via multiple intermittent ground contacts.

Most of the research groups which work on state estimation for legged robots using proprioceptive sensors make use of contact/pressure sensors. One of the common approaches of combining forward kinematics and contact information is called foothold matching. Foothold matching is a method to estimate the motion of the body with respect to the feet which are continously in contact with the ground over the concerned time steps. It is calculated using the feet positions with respect to the body frame which can easily be derived using forward kinematics of the legs.

Gassmann, Zacharias, Zollner, *et al.* [8] extended the work on foothold matching by introducing probabilistic weights in order to consider the reliability of the ground contact constraints. This work was done using the ground reaction force which was calculated from motor current/torque measurements and without use of any external contact sensors. One of the earliest works on state estimation using forward kinematics was published by Roston and Krotkov [9] on their Ambler hexapod. This work was extended in [10] where it was assumed that the robot is in contact with three of its six feet at all times and that the terrain was completely flat. In [11], same research group fused the leg based odometry obtained from kinematics with data from an IMU which is found to be useful in handling tripod gait running. Görner and Stelzer [12] have shown that roll and pitch angles can also be estimated if joint torque measurements are available. The angular estimates obtained this way can be fused with the ego-motion estimates obtained from foothold matching and can be combined with inertial data to globally localize a robot [13].

A fundamental limitation with using foothold matching is that a minimum of three contact points are required. In order to include more dynamic gaits where this assumption might not be valid, data-driven approaches have been used. Reinstein and Hoffman [14] presented a data-driven approach using joint encoders, pressure sensors and an on-board IMU. The limitation of this approach is the trade-off between the amount of training data required and generalization to various locomotion patterns and terrains.

A significant amount of work has also been done on state estimation by including additional sensor modalities and using filter based methods. IMU measurements have been used in the prediction model for a Kalman filter [15] based sensor fusion method ([13] [16] [17]). This has been shown to be efficient and accurate as it keeps the filter size small, avoids using of dynamics model and offer the possibility for online bias estimation. Assuming some knowledge about the terrain, Chitta et al. [18] developed a pose estimator based on particle filter which fuses IMU and kinematics of the leg to get the odometry.

A fundamental limitation of all the above work is the assumption that the contact points are stationary. Measuring slippage can help avoid noise and corrupted measurements. In filtering setups, the stochastic nature of the approach can be used to evaluate the probability of observing a given measurement. In [1], the authors demonstrate a way to apply this to legged robotic state estimation in order to reject the kinematic measurements of slipping feet by thresholding the mahalanobis distance of the innovation (difference between prediction and measurement). If available, the consideration of further sensor readings such as force sensors can also be helpful for slip detection [19].

2.3 Approach Presented in this Thesis

In the presented approach, an Extended Kalman Filter has been implemented which fuses IMU, leg-ground contacts and leg kinematics. This approach has been adopted from [2].

A general algorithm of an EKF is shown in Algorithm 1. An EKF is a type of a recursive Bayes filter which requires previous belief state of the robot and measurements and control at time t. Using the previous belief state and input, a state is predicted. Its covariance is calculated using process Jacobian (F_t) found by linearizing $f(\mu_t, u_t)$ at the predicted state,$\overline{\mu_t}$ and control input,u_t. This predicted belief is then corrected by a measurement residual,y_t which is the error between measured observations and predicted observations. This measurement residual is then scaled by a gain matrix, called Kalman Gain K_t to get the corrected belief mean,μ_t. Covariance of this corrected state, Σ_t is found with the help of measurement Jacobian (H_t) and predicted covariance,$\overline{\Sigma_t}$ as shown in the last step of the algorithm. This algorithm is then called recursively till the last time step.

Algorithm 1 Generic Extended Kalman Filter Algorithm

```
Require: $\mu_{t-1}, \Sigma_{t-1}, u_t, z_t$
  $\overline{\mu_t} \leftarrow f(\mu_{t-1}, u_t)$                          // Predicted State estimate
  $\overline{\Sigma_t} \leftarrow F_t \Sigma_{t-1} F_t^T + Q_t$              // Predicted Covariance estimate
  $y_t = z_t - h(\overline{\mu_t})$                                   // Measurement Residual
  $K_t \leftarrow \overline{\Sigma_t} H_t^T (H_t \overline{\Sigma_t} H_t^T + R_t)^{-1}$   // Kalman Gain
  $\mu_t \leftarrow \overline{\mu_t} + K_t y_t$                            // Updated State estimate
  $\Sigma_t \leftarrow (I - K_t H_t) \overline{\Sigma_t}$                      // Updated Covariance estimate
  return $\mu_t, \Sigma_t$
```

This algorithm is modified for legged systems such that IMU is used for prediction of the state and, contacts and forward kinematics are used for correcting the prediction. Using an IMU for prediction is analogous to a dynamics model of a car where acceleration and steering information is used as a process dynamics.

State Representation

The state of the robot at time t consists of a mean and covariance. The mean of the state is represented as follows:

$$\mu_t := (r_t^w, v_t^w, q_t^{bw}, p_{it}^w, b_{ft}^b, b_{wt}^b)$$

where, r_t^w is the COM position in the world frame, v_t^w is the velocity of the COM in the world frame, q_t^{bw} is the rotation from body frame to world frame, p_{it}^w are feet positions in the world frame and b_{ft}^b and b_{wt}^b are biases for accelerometer and gyroscope respectively represented in the body frame, all at time t. Thus the total size of the state is $(16 + 3N)$ where N is the number of legs of the robot.

A quaternion representation is used in order to avoid the singularity whenever we add a change in orientation, either measured (prediction step) or calculated (update step). However in order to linearize the process dynamics, quaternion representation is replaced by the axis-angle representation. Therefore, the covariance of the state is represented by a positive semi-definite matrix of size, one dimension lesser than the state, $(15 + 3N, 15 + 3N)$.

Sensor Devices

1. **IMU:** The IMU meaures the acceleration and angular velocity in the sensors frame which are transformed to the body frame to get a_t^b and ω_t^b which are used directly in the algorithm as inputs.

2. **Joint Encoders:** Joint encoders provide the angular position of all joints. Based on the kinematics derived from the robot, the location of each foot can be estimated in the base frame of the leg. This can be then transformed to be represented in the body frame of the robot.

Prediction Step

The belief state of the robot is predicted using only IMU data. The accelerometer gives us the linear acceleration and gyroscope gives the angular velocities both in the IMU frame. This is then transformed to body frame using the appropriate rotation and translation. The acceleration is integrated twice to get the position and integrated once to get the velocity. Similarly, the angular velocity is integrated once and converted to quaternion (using a function denoted by ζ) to get the orientation of the COM in the quaternion representation. The following equations are the discrete process dynamics which are used for the prediction of the state.

$$\text{Position:} \quad \overline{r_{t+1}} = r_{t+1} + \Delta t v_t + \frac{\Delta t^2}{2}(R_{wb}a_t + g) \tag{2.2}$$

$$\text{Velocity:} \quad \overline{v_{t+1}} = v_t + \Delta t(R_{wb}a_t + g) \tag{2.3}$$

$$\text{Quaternion:} \quad \overline{q_{t+1}} = \zeta(\Delta t\omega) \otimes q_t \tag{2.4}$$

$$\text{Feet Position:} \quad \overline{p_{i,t+1}} = p_{i,t} \qquad \forall i = 1, ...N \tag{2.5}$$

$$\text{Accelerometer Bias:} \quad \overline{b_{a,t+1}} = b_{a,t} \tag{2.6}$$

$$\text{Gyroscope Bias:} \quad \overline{b_{\omega,t+1}} = b_{\omega,t} \tag{2.7}$$

Using the error dynamics model, the process Jacobian can be evaluated which can be used to predict the covariance of the state by the following equation:

$$P_{k+1}^{-} = F_k P_k^{+} F_k^T + Q_k$$

Measurement Step

Using a forward kinematics model the location of the point feet can be computed with respect to the robot body. Assuming that feet in contact with the ground remain stationary over a part of the gait cycle, they can be matched between successive time steps and thereby used to calculate the incremental motion. Since encoders are less noisy than IMU, forward kinematics is used for correction of the predicted state. The measurement residual, for this application, is the difference between actual feet positions and their predicted values ($h(\overline{\mu_t})$) which is given by the following equations:

$$\text{Measured feet positions in the body frame, } s = fwd\_Kin(\alpha) \tag{2.8}$$

$$\text{Predicted feet positions in the body frame, } \overline{s} = \overline{R_{bw}}(\overline{p_{i,k}} - \overline{r_k}) \tag{2.9}$$

$$\text{Measurement Residual, } y = s - \overline{s} \tag{2.10}$$

Where, α is the joint angles, $\overline{R_{bw}}$ is predicted rotation matrix, $\overline{p_{i,k}}$ is predicted feet positions and $\overline{r_k}$ is predicted robot's COM position.

The measurement Jacobian are calculated by considering the error states and removing higher order terms as shown in [2]. Using the measurement residual and measurement Jacobian, correction in the state is calculated which is added back to the predicted value to get the resultant state.

Observability Analysis

This approach does not assume anything about the gait, robot's dynamics and kinematics and terrain which means this algorithm can be used for any type of gait and robot as long as it has an on-board IMU and joint encoders on the legs. Through an observability analysis, it is shown that apart from absolute position and yaw angle of the robot, all other states are fully observed. The pitch and roll angles become fully observable when the robot regains ground contact and thus, errors reduce.

Chapter 3

Modelling and Implementation

The state estimation algorithm is shown to be working on two sets of robotic platform, (i) Simulation: on a quadruped using Matlab-Simulink, and (ii) Hardware: Hebi X-series Hexapod. The following sections explain in detail about their kinematics, dynamics, controls and sensors of the robots.

3.1 Simulation

A 4-legged robot was simulated using Matlab-Simulink as shown in Figure 3.1. The robot's main body or the chassis is 60 cm long, 30 cm wide and 15 cm thick. It weighs 16.2 kg and each foot weighs 838 grams making the total mass of the body to be 19.5 kg. Each leg of the robot has 3 degree of freedoms and 4 components namely, base, shoulder, elbow and feet. The physics engine used to simulate foot-ground contacts and impacts is the SimMechanics Contact Library. There is just enough friction and damping on the ground in order to have a balance between inelastic and elastic collision to simulate a little slippage.

The robot's gait is similar to an animal gait where diagonally opposite legs move together. The controller is designed such that the feet follows a cubic spline while moving until the feet hits the ground. It can also be referred to as an alternating gait. The trajectory on which experiments are carried out is of a walking sequence in a straight line for over a minute in which it covers a distance of around 20 meters.

The ground truth and measurements are acquired from a Simulink block called "Body Transform Sensor" which calculates the distance of the frame attached to the body with respect to the frame assigned as the world frame. Thus the ground truth pose follows exactly the motions of the robot. The acceleration and angular velocity is also given by the Body Transform Sensor which simulates an on-board IMU sensor.

The controller runs at a frequency of 400 Hz and the sensors give data at a frequency of 10 KHz. The experiments were done offline by running the state estimator at the frequency of both, 1 KHz and 10 KHz.

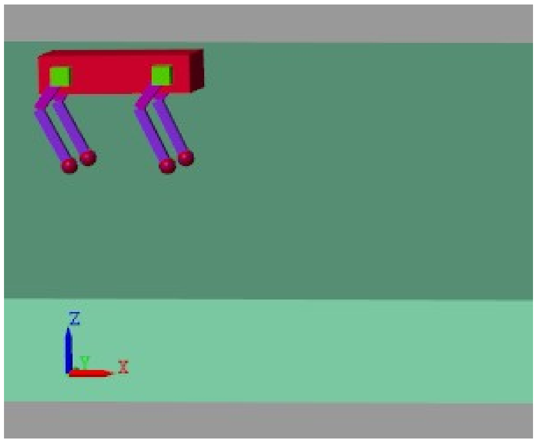


Figure 3.1: **Model of the Simulated Robot:** This is a snapshot of the simulink model of the 4-legged robot spawned in Matlab.

3.2 Hardware

A remote-controlled hexapod robotics kit commercially sold by HEBI Robotics (shown in Figure **??**) has also been used for experiments. This robot is called X-monster because of the X-series actuators/modules used for all of the six legs. The entire robot weighs 21 kg with a total length and width of around 1.1 m. The legs are made of 3 links namely, base, shoulder and elbow. The robot walks at a speed of 0.13m/sec following a tripod gait. The robot moves with very little slip due to the rubber feet and gas springs attached to every leg.

X-series actuators have sensors that enable simultaneous control of position, velocity, and torque as well as three axis inertial measurement. It is important to note that there is no IMU on the chassis itself. Hence, we simulated an IMU on the center of the mass by first, doing a SE(3) transformation and integrating the measurements from the 6 IMUs on the first (or "base") joint of the legs which do not move as they are attached to the rigid chassis through a fixed link. We measure position and torque from all the X-series modules to do forward kinematics and contact estimation. Since, it follows a tripod gait, it is safe to assume that there will always be atleast 3 legs which will be in contact with the ground. We use this assumption in our state estimator.

The ground truth data is collected using a motion capture system. Three markers are put on the robot's chassis such that the centroid of the three markers lie around the geometric center of the mass. These markers are tracked to get the entire pose of the robot.

The robot's feedback handler which acquires the sensor data is running at 1KHz and the ground truth data is received at 120 Hz which is limited by the tracking sensors frequency. The state estimator also runs at 1KHz.

3.3 Implementation details

This section explains the details of how the approach mentioned in section 2.3 has been implemented on the two platforms, simulation and hardware. To simplify the notation, we denote the World frame as W, Body frame as B, IMU frame as I, Shoulder of the robot as S and Feet of the robot as F. These frames and points of reference are shown in Figure 3.3. The following subsections explains the flow of algorithm pictorially shown in Figure 3.2.

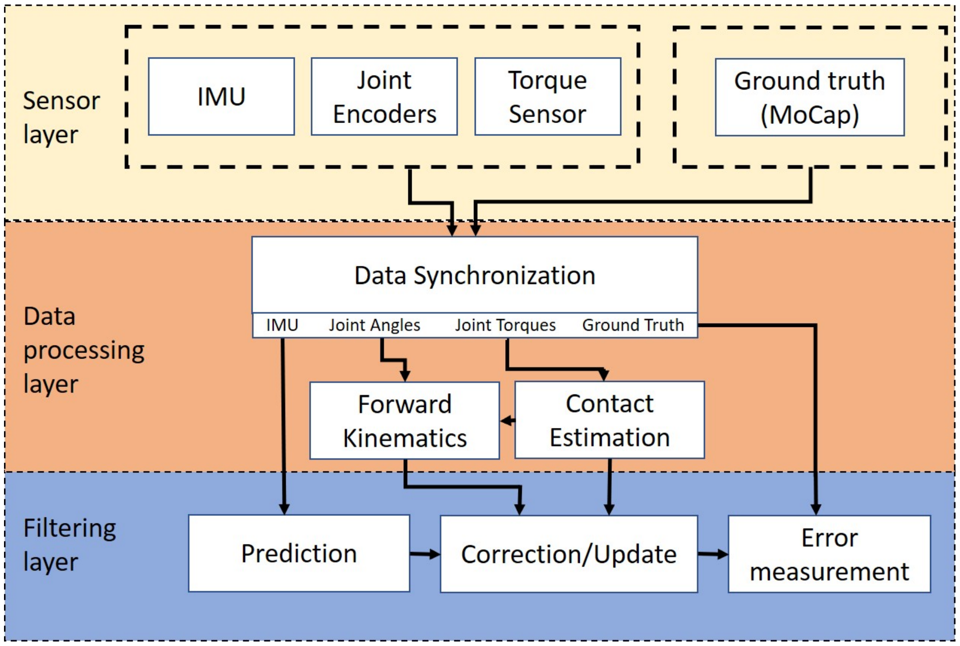


Figure 3.2: **Flow of the algorithm:** This is a flow chart of the algorithm dividing the implementation into three layers, data acquisition, data pre-processing and filtering.

Sensor Layer

In simulation, the integration of linear acceleration and angular velocity are straight forward because IMU frame and body frame are assumed to be the same. This case is different for the the real platform, the hexapod as there doesn't exist any IMU on the chassis of the robot. There are IMUs in each of the Hebi X-series modules. The base modules are attached to the chassis by a rigid link. The IMUs in the base modules of all the 6 legs are used to emulate an IMU in the body frame by transforming them to the COM and then averaging them. The equations used are as follows:

$$a_t^b = R_{bs} a_t^s - \omega_t^b \times (\omega_t^b \times r_{bs}) \tag{3.1}$$

$$\omega_t^b = R_{bs} \omega_t^s \tag{3.2}$$

Both the equations transform the measured quantities from the sensor frame to base frame using a rotation matrix calculated based on the kinematics of the robot. The second term in equation 3.1 includes angular velocity and its cross product with the translation offset from COM to the sensor. This is required to capture the effect of angular velocity of the sensor on the linear acceleration of the COM. Once the linear acceleration and angular velocity of all the sensors are in the body frame, mean of these readings are taken to get the acceleration and angular velocity of the COM.

Data Pre-processing Layer

The data for ground truth is received at a different rate than the sensor feedback data. Also, since they are run through two different programs, a data synchronization step is needed. This is done by storing the Eastern Standard Time (EST) in milliseconds.

Once, the data is synchronized, we use the joint angles to get the forward kinematics of the legs which are in contact with the ground. In the simulation, Matlab-Simulink which uses the SimMechanics Contacts Library [1] gives the contact measurements directly. However, that is not true on the hexapod and hence, we measure the joint torques and calculate the ground reaction force using the following equations:

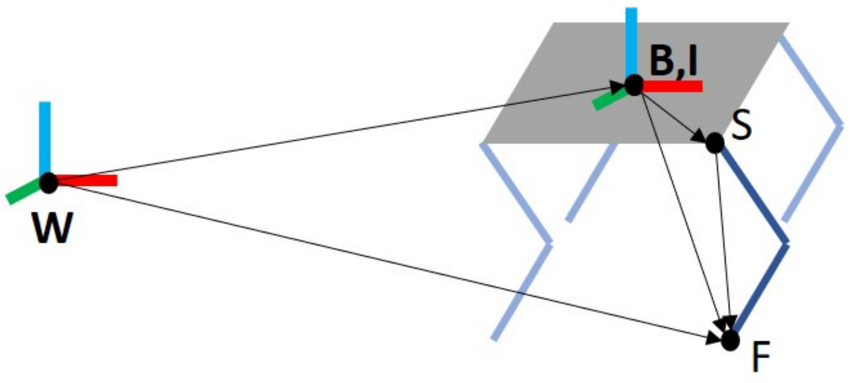
$$F_S = (J_S^T)^{-1} \tau \tag{3.3}$$

$$F_B = \overline{R_{BS}} F_S \tag{3.4}$$

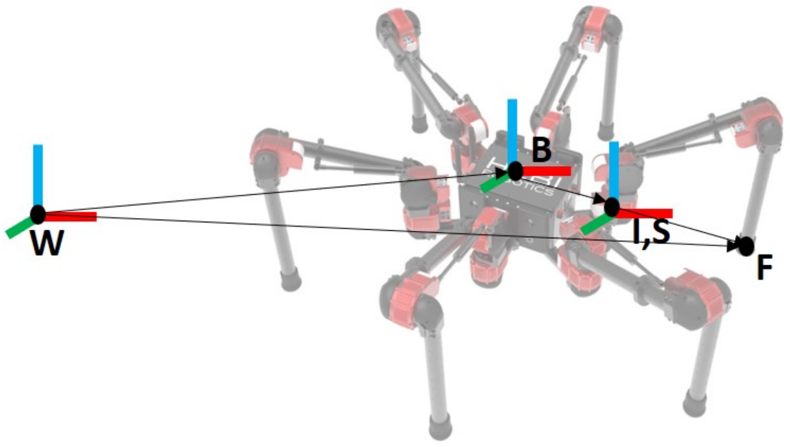
$$c_i = (F_{B_z} <= 0) \tag{3.5}$$

where F_S is the force exerted by the feet on the ground in the spatial (world) frame, J_S is the leg Jacobian (only translation) in the spatial frame, τ is the measured torques on the motors of the leg, F_B is the force exerted by the feet on the ground in the body

[1] Steve Miller (2019). Simscape Multibody Contact Forces Library (https://www.mathworks.com/matlabcentral/fileexchange/47417-simscape-multibody-contact-forces-library), MATLAB Central File Exchange. Retrieved July 18, 2019.



(a) Quadruped model



(b) Hebi hexapod Model

Figure 3.3: **Understanding frames and points of references for both the platforms, simulation and hardware.**

frame, $\overline{R_B S}$ is the predicted rotation to go from body frame to the spatial frame, c_i is a boolean where 1 is in contact and 0 not in contact and F_{Bz} is the z-component of F_B. The dimensions of F_S is 3×1 because we only consider the translation part of the Jacobian (3×3).

Information Fusion or Filtering Layer

The filtering algorithm as implemented is shown in the Algorithm 2. The prediction step is carried out using only the IMU measurements in the body frame as per the equations 2.2-2.7. The update step is only carried out for the feet in contact with the ground so the measurement residual is of size, $(3C, 1)$, where C is number of feet in contact multiplied by the degree of freedom (= 3) for each foot.

Algorithm 2 EKF for legged robots using IMU, Contacts and Forward Kinematics

```
Require: $\mu_{t-1}, \Sigma_{t-1}, a_t$: linear acceleration, $\omega_t$: angular velocity, $\alpha_t$: joint angles,
    $c_t$: boolean vector representing legs in contact
  Prediction Step:
      $\overline{\mu_t} \leftarrow f(\mu_{t-1}, a_t, \omega_t)$
      $\overline{\Sigma_t} \leftarrow F_t \Sigma_{t-1} F_t^T + Q_t$
  Update Step:
      $s_t^b \leftarrow leg\_kin(\alpha_t)$ + offset of shoulder from base
      if new contacts detected then
        $p_i^w \leftarrow r_t^w + s_t^b$
      end if
      for all legs in contact do
        $y_t \leftarrow s_t^b - h(\overline{\mu_t})$
        $K_t \leftarrow \overline{\Sigma_t} H_t^T (H_t \overline{\Sigma_t} H_t^T + R_t)^{-1}$
        $\mu_t \leftarrow \overline{\mu_t} + K_t y_t$
        $\Sigma_t \leftarrow (I - K_t H_t) \overline{\Sigma_t}$
      end for
  return $\mu_t, \Sigma_t$
```

Chapter 4

Results and Discussion

The comparison between the ground truth and filter results are shown in Figure 4.1 and 4.2. The simulated robot walks at a speed 0.33 m/s whereas the real robot is walking at a speed of 0.06 m/s in the x-direction. The y direction is considered to be parallel to ground and perpendicular to x and the z direction points outward from the ground. The simulation was carried out with realistic noise levels in the measurements given by the Simulink body transform sensor. The results on hardware were carried out on multiple trajectories and the noise parameters were chosen such that they are same for multiple experiments. The plots 4.1 show the results for the simulation which is a trajectory of the robot going forward for 10m in the x direction for 30 seconds. The drift is as low as 7% in the position estimate in x direction and further lesser drift in y and z direction even though the acceleration in z is large due to the impacts. The velocity estimates although not exactly accurate do not show drift. The orientation estimates are accurate with no drift.

The plots 4.2 demonstrate the results for the hardware platform which covers a distance of about 2 meters in 35 seconds. The drift is again limited to 8% in the position estimate in x direction with little to no drift in y and z. Similar to the simulation results, the velocity estimates are noisy but do not experience drift and orientation estimates are very accurate with no drift.

These results validate the observabiity analysis presented in the paper [2] that position and yaw angle are unobserved whereas the velocity, roll and pitch angles are observed. Intuitively, the reason that position and yaw are unobservable is because neither IMUs nor joint angles can estimate the position and yaw angle without drift. However, we do get a sense of absolute zero velocity when the robot is stationary which is estimated using contacts. Roll and pitch angles can also be estimated exactly using forward kinematics (in the body frame) on legs in contact with the ground in consecutive time steps.

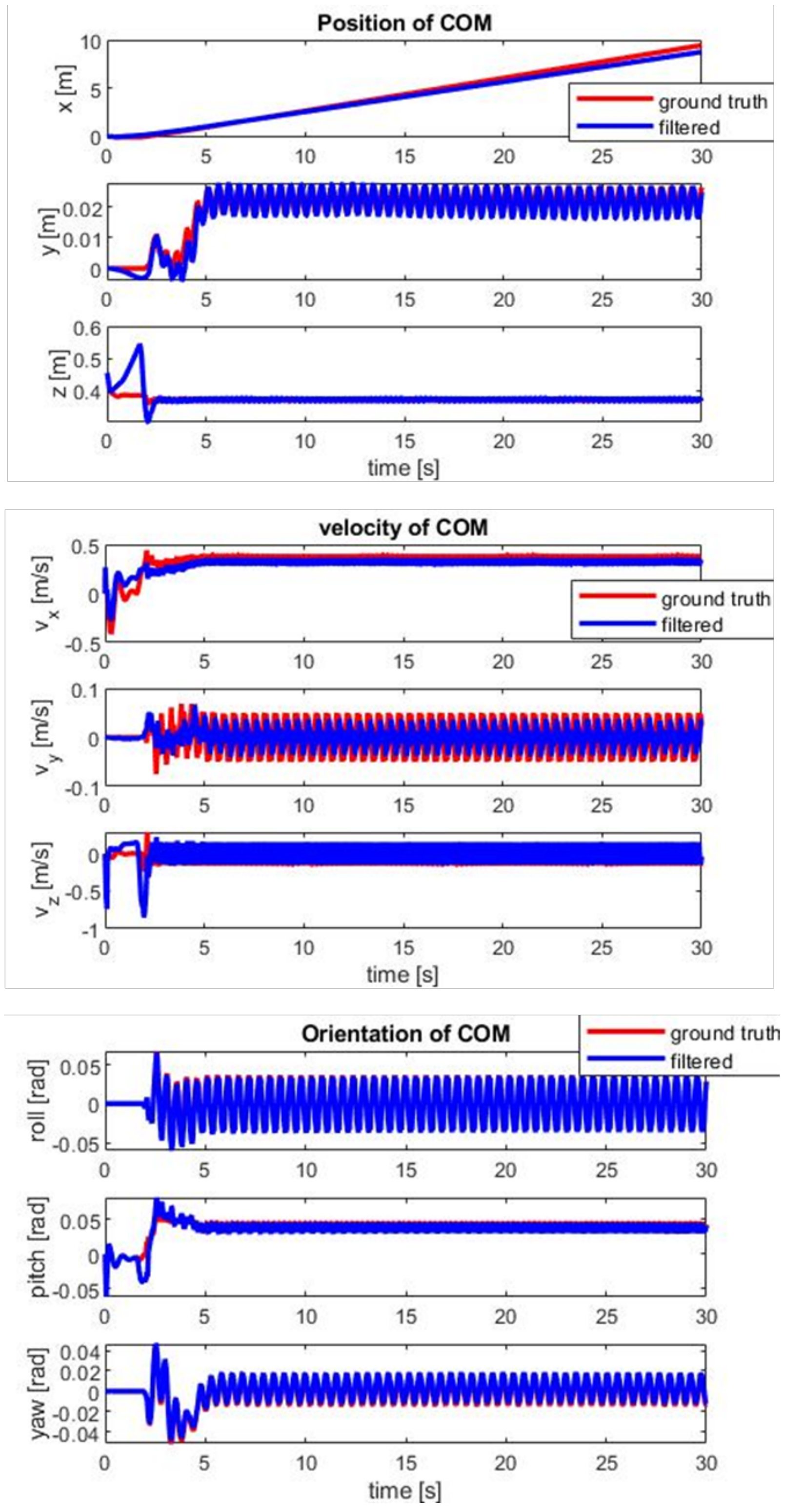


Figure 4.1: **Resulting plots of the state estimator on Simulation :** It shows a comparison of estimated states and their respective ground truth data.

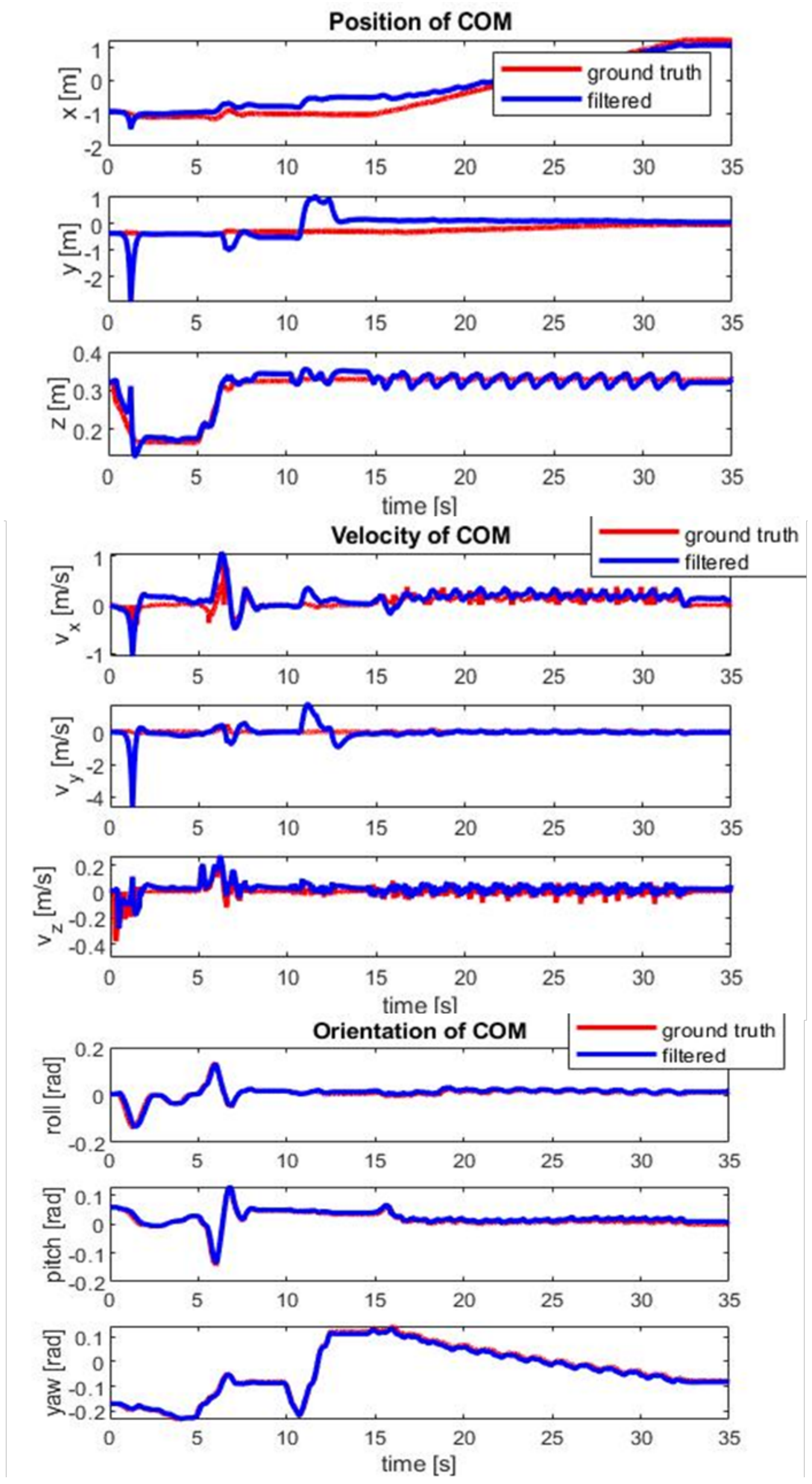


Figure 4.2: **Resulting plots of the state estimator on Hardware :** It shows a comparison of estimated states and their respective ground truth data.

Chapter 5

Conclusion and Future Work

A reliable state estimation using only proprioceptive sensors was implemented and tested on an actual hexapod system without the need for restrictive assumptions such as flat ground or pre-defined gait pattern. The results obtained are observed to be consistent with the theoretical understanding of the algorithm. A closed form linearization of process and measurement dynamics to obtain the respective Jacobians speeds up the EKF algorithm. The state estimation algorithm performs better with higher frequency measurements and high frequency estimation because linearization is more accurate then. For the same reasons, this performs better on slower gaits with limited slippage.

This work can be directly extended to terrain analysis by using the position of the robot and feet estimated. This work can also be used for building control strategies that could help the robot to follow a desired trajectory. There can be significant improvements made for stability and traversability using both better control strategies and terrrain analysis. This work also can be extended by measuring slippage of the feet and including that information in the state estimator.

Bibliography

[1] M. Bloesch, C. Gehring, P. Fankhauser, M. Hutter, M. A. Hoepflinger, and R. Siegwart, “State estimation for legged robots on unstable and slippery terrain,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2013, pp. 6058–6064.

[2] 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.

[3] R. Hartley, M. G. Jadidi, J. W. Grizzle, and R. M. Eustice, “Contact-aided invariant extended kalman filtering for legged robot state estimation,” *ArXiv preprint arXiv:1805.10410*, 2018.

[4] R. Hartley, J. Mangelson, L. Gan, M. G. Jadidi, J. M. Walls, R. M. Eustice, and J. W. Grizzle, “Legged robot state-estimation through combined forward kinematic and preintegrated contact factors,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 1–8.

[5] O. Gur and U. Saranli, “Model-based proprioceptive state estimation for springmass running,” *Robotics: Science and Systems VII*, pp. 105–112, 2012.

[6] C. G. Atkeson *et al.*, “State estimation of a walking humanoid robot,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2012, pp. 3693–3699.

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

[8] B. Gassmann, F. Zacharias, J. M. Zollner, and R. Dillmann, “Localization of walking robots,” in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, IEEE, 2005, pp. 1471–1476.

[9] G. P. Roston and E. P. Krotkov, “Dead reckoning navigation for walking robots,” CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, Tech. Rep., 1991.

[10] P.-C. Lin, H. Komsuoglu, and D. E. Koditschek, “A leg configuration measurement system for full-body pose estimates in a hexapod robot,” *IEEE Transactions on robotics*, vol. 21, no. 3, pp. 411–422, 2005.

[11] ——, “Sensor data fusion for body state estimation in a hexapod robot with dynamical gaits,” *IEEE Transactions on Robotics*, vol. 22, no. 5, pp. 932–943, 2006.

[12] M. Görner and A. Stelzer, “A leg proprioception based 6 dof odometry for statically stable walking robots,” *Autonomous Robots*, vol. 34, no. 4, pp. 311–326, 2013.

[13] A. Chilian, H. Hirschmüller, and M. Görner, “Multisensor data fusion for robust pose estimation of a six-legged walking robot,” in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2011, pp. 2497–2504.

[14] M. Reinstein and M. Hoffmann, “Dead reckoning in a dynamic quadruped robot based on multimodal proprioceptive sensory information,” *IEEE Transactions on Robotics*, vol. 29, no. 2, pp. 563–571, 2012.

[15] R. E. Kalman, “A new approach to linear filtering and prediction problems,” *Journal of basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[16] 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.

[17] J. Ma, M. Bajracharya, S. Susca, L. Matthies, and M. Malchano, “Real-time pose estimation of a dynamic quadruped in gps-denied environments for 24-hour operation,” *The International Journal of Robotics Research*, vol. 35, no. 6, pp. 631–653, 2016.

[18] S. Chitta, P. Vemaza, R. Geykhman, and D. D. Lee, “Proprioceptive localilzatilon for a quadrupedal robot on known terrain,” in *Proceedings 2007 IEEE International Conference on Robotics and Automation*, IEEE, 2007, pp. 4582–4587.

[19] K. Kaneko, F. Kanehiro, S. Kajita, M. Morisawa, K. Fujiwara, K. Harada, and H. Hirukawa, “Slip observer for walking on a low friction floor,” in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2005, pp. 634–640.