

Master's Thesis  
MSC-054



# State Estimation and Feedback Control of Legged Robots

Zustandsschätzung und Feedback-Steuerung von Robotern  
mit Beinen

by  
Shahram Khorshidi

Supervisors: Prof. Dr.-Ing. Robert Seifried (TUHH)  
Prof. Dr. Norbert Hoffmann (TUHH)

Advisors: Prof. Dr. Ludovic Righetti (MPI-IS)  
Dr.-Ing. Majid Khadiv (MPI-IS)  
M.Sc. Daniel-André Dücker (TUHH)

Hamburg University of Technology (TUHH)  
Institute of Mechanics and Ocean Engineering  
Prof. Dr.-Ing. R. Seifried

Hamburg, September 2022



MAX PLANCK INSTITUTE  
FOR INTELLIGENT SYSTEMS





## Acknowledgements

I would like to express my profound gratitude to my primary supervisor, Dr. Majid Khadiv, Research Scientist at Max Planck Institute for Intelligent Systems, Tübingen, who guided me throughout this project. Special thanks to Prof. Ludovic Righetti, Max Planck Research Group Leader for Movement Generation and Control Group, for giving me the opportunity to be a member of this amazing team, which provided me insights into my future goals and career in robotics. Additionally, I wish to acknowledge the help and support that was provided by all my colleagues in this group.

Finally, I would like to thank Prof. Robert Seifried, Prof Norbert Hoffmann, and M.Sc. Daniel-André Dücker, who offered guidance and support to me in this project, for giving me the opportunity to work together, and extend our relationship.



# Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
1.1	Legged Robots Locomotion . . . . .	4
1.2	Open Robot Dynamic Initiative . . . . .	5
1.3	Research Objectives . . . . .	6
1.4	Contributions . . . . .	7
1.5	Overview . . . . .	8
<b>2</b>	<b>Literature Review</b>	<b>9</b>
2.1	Background . . . . .	9
2.2	State Estimation . . . . .	10
<b>3</b>	<b>State Estimation Framework</b>	<b>13</b>
3.1	Mathematical Prerequisites . . . . .	13
3.2	Kalman Filter . . . . .	16
3.3	Extended Kalman Filter . . . . .	17
3.4	Mathematical Derivation of EKF . . . . .	18
3.5	Contact Detection . . . . .	24
3.6	Implementation . . . . .	25
<b>4</b>	<b>Controllers</b>	<b>27</b>
4.1	MPC Controller . . . . .	27
4.2	Observer-Based Controller . . . . .	30

<b>5</b>	<b>Results and Discussion</b>	<b>33</b>
5.1	Experimental Setup . . . . .	33
5.2	Experimental Results . . . . .	34
5.2.1	Moving the Base while Balancing . . . . .	35
5.2.2	Trotting Gait . . . . .	38
5.2.3	Jumping Motion . . . . .	41
5.2.4	Trotting Gait with EKF . . . . .	43
5.3	Discussion . . . . .	43
<b>6</b>	<b>Conclusion</b>	<b>47</b>
6.1	Summary . . . . .	47
6.2	Future Work . . . . .	48
	<b>Bibliography</b>	<b>49</b>
	<b>Appendix</b>	<b>53</b>
A.1	Contents Archive . . . . .	53

## Abstract

Legged robots require knowledge of the position, velocity and orientation of the base in order to maintain stability and execute controlling schemes. We present a state estimation framework for legged robots locomotion based on Extended Kalman Filtering (EKF). The core idea of this approach is to exploit the information from the kinematics constraints given by the feet in contact with the ground at each interval, and then fuse it with the sensory data from Inertial Measurement Unit (IMU) on the robot, therefore the whole task of control is executed by the information provided with onboard sensory devices. Consequently, we will design the estimator (EKF) based on the formulation of the underlying stochastic model. As shown in previous works, such a filter is observable for the linear velocity vector, and roll and pitch angles of the base, which are the crucial elements in order to use the estimated states for the feedback to the whole body controllers in legged robots locomotion. To this end, the estimator is implemented and tested with different types of motion on the quadruped robot Solo12, and the correctness and reliability of such a framework is shown. Finally, the observer-based controller is successfully tested in a feedback scheme for the whole body control of the robot without using any external sensory device.



# Chapter 1

## Introduction

In this work we address the state estimation problem for legged robot, which is one of the main challenges in legged robot locomotion. In order to develop and have efficient motion planning schemes with different controllers for a free floating base robot, accurate estimates of the robot's states are needed. Some states, such as joint angles are directly accessible through encoders (and also joint velocities, by taking the differentiation from encoders' measurements), while other states, such as the robot's base position, velocity and orientation require additional sensor.

We show that by fusing different sensory data with an estimation framework, we can have an accurate estimation of the robot's base states. The advantage of this framework comparing to the alternative approaches is simplicity and being computationally efficient.

In our work, we only rely on using common proprioceptive sensors and knowledge of leg kinematics. The key idea is to extract information from the kinematic constraints given through the intermittent contacts with the ground and to fuse this information with inertial measurements coming from the onboard sensors. The estimation framework accommodates contact switching and makes no assumptions about gait or terrain, making it applicable on any quadruped or biped platform.

In order to develop and implement the estimation algorithm, we first briefly mention the importance of this study for legged robot locomotion, and present the research objectives, and contributions of this work.

## 1.1 Legged Robots Locomotion

Legged robotics is rapidly transitioning from research laboratories into the real world, as demonstrated by the recent introduction of several commercial quadruped platforms. They have the potential and promising applications in a wide range of usage such as logistics, delivery industry, aid in search and rescue and assistants in different fields.

In the recent times, there are commercial platforms, such as Spot [Spot] from Boston Dynamics that are capable of walking on different terrains. They are currently used primarily for indoor mapping and construction site applications. Another equally popular robot is ANYmal from Anybotics [FankhauserHutten18], a Switzerland-based company. They focus both on the commercial application and research work. They maintain close ties with ETH Zurich to contribute extensively to the community. Fig. 1.1 shows these quadruped robots along with one of the first legged robots designed and developed in 1970's by McGhee and his group at Ohio State University [SchneiderSchmucker06].



Figure 1.1: From left to right: Boston Dynamics Spot, Anybotics ANYmal, and OSU Hexapod.

To be truly useful, legged robots must be able to reliably and rapidly navigate across rough terrain and be stable in the presence of disturbances, such as slips or pushes. They must also be able to perceive and manipulate the environment whilst avoiding collisions with obstacles and people. In order to develop and have efficient motion planning schemes and robust feedback controllers for these robots to handle these tasks with different gaiting patterns on diverse terrain, accurate estimates of the robot's states are needed. The robot's state is used to plan and track body trajectories, to balance and recover from external disturbances, and to map the environment and navigate through it. Since the base of legged robots is a free flyer with 6 degrees of freedom (DoF) in space, and they make intermittent contact with the ground, this contact switching poses a challenge for obtaining robot's state.

Mobile robots are equipped with different sensors, and they are categorized in two main types, “Exteroceptive”, and “Proprioceptive”. In order to differentiate between these two categories of sensors, it is first necessary to define the general characteristics that are important for their use in mobile robotics. All sensors are characterized by; an acquisition frequency (response time), a measurement resolution, and a noise on the measured physical quantity (defined by its repeatability). In addition, the measurement principle defines whether the sensor performs an absolute measurement (always with respect to a known reference frame) or a relative one. Proprioceptive sensors such as inertial measurement unit (IMU), and encoders measure the state of the robot itself (acceleration, joint position or velocity, etc.), while exteroceptive sensors such as cameras or GPS units, measure the state of the environment (mapping, temperature, etc.).

While wheeled robots are assumed to remain in contact with the ground at all times, legged robot locomotion inherently involves intermittent contacts. This makes stability a main concern as well as complicates odometry-based estimation approaches. The main control challenge for Legged robots is that they require knowledge of the full 6 DOF placement of the base. As mentioned, legged robot locomotion is inherently unstable, and it forces the tasks of control and localization to depend on internal (proprioceptive) sensing. Therefore, the problem of state estimation for legged robots is fundamentally different than that of estimation for wheeled robots (which is generally a localization task).

Most legged robots are equipped with an inertial measurement unit (IMU) that can measure linear acceleration and angular velocity, although with noise and bias disturbances. Additionally the leg kinematics are available from intermittent contacts of the robot locomotion through the encoders. Therefore, because of nonlinear dynamics of these robots, nonlinear observers are typically utilized to fuse these data to estimate the robot’s states.

The advantage of this approach is that the whole localization task and estimation process is relying on internal (proprioceptive) sensing. Furthermore, additional sensory data such as visual and LiDAR can be integrated later to the observer in order to correct the position and yaw angle drift and also to build maps of the environment for the navigation task.

## 1.2 Open Robot Dynamic Initiative

It is often difficult to test advanced control and learning algorithms for legged robots without significant hardware development efforts or maintenance costs. Present-day hardware is often mechanically complex and costly, different robot systems are hard to compare to one another, and many systems are not com-

mercially available. To support rapid and broad progress in academic research, the project “open robot dynamic initiative” (ODRI) was started by movement generation and control group. This group is led by Prof. Ludovic Righetti in the autonomous motion department at Max Planck Institute for Intelligent Systems. Their main research topics are focused on understanding the fundamental principles for robot locomotion and manipulation that will endow robots with the robustness and adaptability necessary to efficiently and autonomously act in an unknown and changing environment.

With the help of modular design, several open source robotics platforms under ODRI project have been designed and developed, such as biped (Bolt), manipulator (Tri-finger), and quadruped (Solo12) as shown in Fig. 1.2.



Figure 1.2: From left to right: Solo, Bolt, and Tri-finger Robots from ODRI.

Solo12 quadruped Fig. 1.3 [GrimmingerEtAl20], is a torque-controlled modular robot architecture for legged locomotion research. This is a new research robot, developed as an open-source project in collaboration among Max Planck Institute for Intelligent System (MPI-IS), Laboratory for Analysis and Architecture of Systems (LASS), and NYU Tandon School of Engineering. All designs are open-sourced; including mechanical drawings, electronic circuits, and control software. The dog-resembling, torque-controlled quadruped is capable of very dynamic movements. It is made entirely of 3D printed parts and off-the-shelf components, which makes it an easy to replicate platform ideal for fundamental research in legged locomotion and robotic education.

### 1.3 Research Objectives

In order to control a free floating base legged robot platform, we need accurate estimates of the robot’s states; namely robot configuration ( $\mathbf{q} \in SE(3) \times \mathbb{R}^n$ ), and robot generalized velocity ( $\mathbf{v} \in \mathbb{R}^{n+6}$ ), which  $n$  is the number of degrees of freedom on the robot.

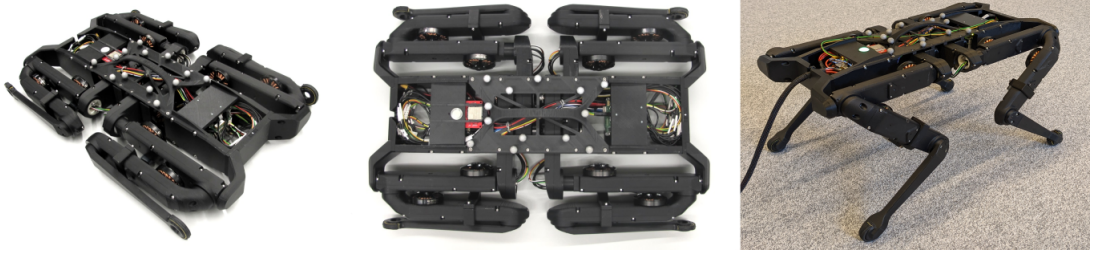


Figure 1.3: Solo12 quadruped robot with 12 DoF in the revolute joints.

In this work, we will focus on Solo12 robot with  $n = 12$ , (12 DoF in revolute joints), and 6 DoF in the free floating base. The robot is equipped with encoders at each joint and an onboard inertial measurement unit (IMU). The joints positions and joints velocities are accessible at each time step through the encoders, and the angular velocity of the base is measured by the gyroscope of the IMU. Furthermore, since we are using torque control robot, the joint torques are estimated from motors current at each joint. In order to obtain the base position, orientation and linear velocity (called robot’s base states in this work), we use an external sensory data as motion capturing system.

This setup imposes some challenges for our robot locomotion, that is to say the range of robot’s motion is limited by the motion capturing system around the robot. Therefore, we wish to design and develop an alternative way to obtain robot’s base state without using any external sensing device.

Accordingly, the goal of this work is to develop a “state estimation framework”, entirely based on proprioceptive sensory data (onboard sensing devices on the robot), in order to “track the pose  $SE(3)$ , and the linear velocity of an articulated floating base robot”.

## 1.4 Contributions

The main contributions of this work are as follows:

- We implement a state estimation framework based on the fusion of IMU data and leg kinematics as two sources of proprioceptive sensory data on robot.
- We design and implement the EKF in simulation and on the real hardware, based on the formulation of underlying stochastic process and measurement models.

- We present a way of contact detection for our quadruped, to determine which leg is in contact with the ground at each time step in order to use the kinematics information of that leg in the measurement model of our estimator.
- We perform an extensive set of experiments on Solo12 with different motions, in order to evaluate the performance and reliability of the proposed estimation framework.
- We use the designed EKF with a nonlinear feedback MPC scheme to show the advantage of this framework in legged robot locomotion, which eliminates any external sensing and all robot's states are accessible through onboard sensing devices.

## 1.5 Overview

The rest of this work is organized as follow. Chapter 2 provides a brief introduction to the related literature on the state estimation for legged robots, and discusses different approaches in this field, and the advantage of the proposed estimation method over alternative approaches. Chapter 3 explains the state estimation framework chosen in this study in details with the mathematical derivation, and presents the implementation in simulation and on the real robot. Chapter 4 mentions the different types of controllers, which are used in the experiments, and their structures. Finally, in Chapter 5, we present and discuss the results obtained from an extensive set of experiments with different gaits on the real quadruped robot Solo12, followed by challenges faced in this estimation framework, and avenues for related future research in Chapter 6.

# Chapter 2

## Literature Review

In this chapter we briefly have a look at the history of state estimation in legged robots locomotion, and review the related works and notable techniques in this field. Furthermore, we elaborate on the relevance of these literature to our work, and finally we reason about why we choose the proposed state estimation framework in this project.

### 2.1 Background

Legged robots are a subclass of mobile robots that locomote through direct and switching contact with the environment. Legged robot locomotion inherently involves intermittent contacts, and this makes stability a main concern. One of the main control challenge for Legged robots is that they require knowledge of the full 6 DoF placement of the base. These robots typically contain proprioceptive sensors, such as IMU, joint encoders, and force/torque and contact sensors. In addition, some legged robots also have access to exteroceptive sensors, namely cameras and LiDARs. As with all mobile robots, state estimation for legged robots is critical for mapping, planning, designing feedback controllers, and developing general autonomy.

The majority of legged robot estimation works to date have focused on base state estimation. This problem is particularly difficult for floating base systems, as the base pose cannot be measured directly. One of the most widely used choices for base state estimation is the Extended Kalman Filter (EKF) [BloeschEtAl13a], [RotellaEtAl14], [FallónEtAl14]. Although, there have been recent attempts to use more advanced approaches based on the use of factor graphs [FourmyEtAl21], and invariant Kalman Filters [HartleyEtAl20], the EKF framework is commonly used due to its compromise between simplicity, efficiency, and performance.

For base state estimation, most works have fused Inertial Measurement Unit (IMU) data with leg odometry to estimate the base position and velocity as well as its orientation [BloeschEtAl13b], [RotellaEtAl14]. Furthermore, to compensate for the drift of the unobservable base position while simultaneously mapping the environment, sensor modalities like LiDARs and cameras are often added.

## 2.2 State Estimation

As described in [CamurriEtAl20], the literature on the state estimation for legged robots can be classified according to several criteria: the type of sensors used (proprioceptive, exteroceptive, or both); the output frequency (at control rate, e.g., 400 Hz or camera/LiDAR rate, e.g., 10 Hz); state definition (placement, velocity, joint states, contact points, etc.); the presence of loop closures (odometry vs. SLAM); the degree of marginalization of past states (from filtering to full batch optimization). Finally, if there is fusion of proprioceptive and exteroceptive signals, this can be performed in a loosely or tightly coupled manner.

The related work can be divided into three main categories:

- Proprioceptive state estimation, which includes filtering methods to fuse only the high-frequency signals, such as IMU and kinematics.
- Multi-sensor filtering, which covers filtering methods with proprioceptive and exteroceptive sensor fusion.
- Multi-sensor smoothing, which typically involves fusion of visual odometry (VO), IMU, and kinematics in a tightly coupled manner within probabilistic graphical model frameworks, such as factor graphs [CamurriEtAl20].

[RostonKrotkov92] presented one of the earliest navigation system which extracts information from leg kinematics. By matching the foot positions between two consecutive time steps they compute the incremental motion of the main body. Further, they introduce a slip detection method which relies on the invariance of the distance between feet that are in contact with the ground.

One of the first approach exploiting information given by the leg kinematics was implemented by [LinKomsuogluKoditschek05] on a hexapod robot. Assuming that the robot is in contact with three of its six feet at all times and assuming completely flat terrain, they implemented a leg-based odometer. Their method requires the robots to follow a tripod gait and is affected by drift. The same group fused the leg-based odometer with data from an IMU and thus is able to handle tripod running.

Using the assumption of knowing the precise relief of the terrain, [ChittaEtAl07] implemented a pose estimator based on a particle filter. It fuses leg kinematics and IMU in order to globally localize a robot.

Given the above assessment of previous works, it is conjectured that a suitable filter for general legged robot state estimation should:

- Make no assumptions about the gait or terrain.
- Be easily adapted to any legged robot platform.
- Use as little computational resources as possible.

[BloeschEtAl13a] were the first to introduce a novel EKF for state estimation on the quadruped starETH which fused leg odometry and IMU data to estimate the full pose of the robot without making these assumptions (the EKF-based state estimator did not depend on a specific type of gait or number of legs). By choosing an appropriate state vector, they broke down the estimation problem to the proper formulation of a few simple measurement equations. Further, it was shown that as long as at least one foot is in contact with the ground then all states other than absolute position and yaw (neither of which matters for stability) are observable.

As an extension to this study, the same group presented an estimation framework in [BloeschEtAl13b] based on unscented Kalman filtering. While following a similar overall approach, in the sense that accurate estimates of the full body pose were obtained by fusing information from an on-board IMU and kinematic measurements, the latter approach extended and improved different aspects of the previous methodology. By deriving velocity constraints from the feet that are in contact with the ground, simple measurement equations were obtained which reduced the size of the state and which were more suitable for slippage detection. Since angular velocity from the IMU appeared on both the inertial process model and the measurement update, the authors proposed the use of the Unscented Kalman Filter (UKF) instead of an EKF to better handle the correlation between the joint and gyroscope noises. Further, a robot-centric formulation of the state space was chosen in order to appropriately partition the filter states and avoid problems with unobservable states.

When foot sensors are unavailable, the contact feet are detected by putting thresholds on the Ground Reaction Forces (GRF), which are estimated from the joint torques. [CamurriEtAl17] proposed a method that evaluates GRF discontinuities to discard invalid leg odometry velocity measurements on the quadruped robot.

In this work, we prefer to utilize a Kalman-based filtering in order to fuse IMU data with leg kinematics, for several reasons. First, comparing to more sophisti-

cated methods, Kalman-based filtering framework is simpler and computationally less expensive. Second, our quadruped is equipped with an onboard accurate IMU device, with small noise and bias terms, and by the integration of the dynamics of the robot by the measured linear acceleration and angular velocity, we can have a good process model with small drifts in time. Additionally, we have an almost precise kinematics of the joints for our robot which makes it a reliable measurement model for the base linear velocity. Furthermore, in order to reduce weight and increase agility, our quadruped robot is not equipped with force sensors on the end effectors, but contact forces from ground can be estimated with the current measurement of the motors, which can be used as a model for contact detection in any gaiting pattern.

## Chapter 3

# State Estimation Framework

Filtering methods involve estimating the robot’s current state using the set of all measurements up to the current time. With a linear process and measurement model and white Gaussian noises, the Kalman Filter provides an estimate of the state vector “ $\mathbf{x}$ ” along with a corresponding covariance matrix “ $\mathbf{P}$ ”, which specifies the uncertainty of the estimate. The filter involves two steps: prediction and update. In the prediction step, the estimates are propagated through the system dynamics in order to produce a priori estimates and covariance matrix, and in the update step, the estimates are updated using a measurement to produce a posteriori estimates and covariance matrix. However, the standard Kalman Filter is applicable only for state estimation in linear systems. Most practical robots have nonlinear system dynamics and mainly sensory data are also nonlinear; therefore, for these cases an Extended Kalman Filter (EKF) can be employed, which utilizes Taylor series expansions to linearize the process and measurement models around the current state estimate. Although, the convergence and optimality are no longer guaranteed for such a filter, it is mostly advantageous because of its performance and simplicity, and being computationally efficient. In our case the process model is dynamics of the robot’s base, and we use the base linear velocity measured from kinematics of the set of legs in contact with ground as the measurement model. Since our robot has a free floating joint in the base (with 6 DoF), we use quaternion in order to represent the orientation of the base and to avoid singularities in other representations such as Euler-angles.

### 3.1 Mathematical Prerequisites

The number of degrees of freedom (DoF) of a robot is the smallest number of real-valued coordinates needed to represent its configuration [LynchPark17]. In order to specify the position and orientation of a rigid body in three-dimensional

physical space, a minimum of six numbers are needed. In this section we give an overview of a systematic way to describe a rigid body's position and orientation which relies on attaching a reference frame to the body. The configuration of this frame with respect to a fixed reference frame is then represented as a  $4 \times 4$  matrix. This matrix is an example of an implicit representation of the C-space (configuration space), the actual six-dimensional space of rigid-body configurations is obtained by applying ten constraints to the 16-dimensional space of  $4 \times 4$  real matrices.

### Rigid Body Rotation

In  $\mathbb{R}^3$ , the rotation group  $SO(3)$  is the group of rotations around the origin under the operation of composition. Rotations are linear transformations that preserve vector length and relative vector orientation. Its importance in robotics is that it represents rotations of rigid bodies in 3D space: a “rigid motion” requires precisely that distances, angles and relative orientations within a rigid body be preserved upon motion. In contrast, if norms, angles or relative orientations are not kept upon motion, the body could not be considered rigid. The special orthogonal group  $SO(3)$ , or the group of rotation matrices, is the set of all  $3 \times 3$  real matrices  $\mathbf{R} \in \mathbb{R}^{3 \times 3}$  that satisfy

$$\mathbf{R}^T \mathbf{R} = \mathbf{I} \quad (3.1)$$

$$\det(\mathbf{R}) = 1. \quad (3.2)$$

We can define the rotation operator

$$\begin{aligned} r : \mathbb{R}^3 &\rightarrow \mathbb{R}^3 \\ \mathbf{v} &\mapsto r(\mathbf{v}) \end{aligned} \quad (3.3)$$

acting on vector  $\mathbf{v} \in \mathbb{R}^3$  as a linear operator; since it is defined from the scalar and vector products, which are linear. Therefore it can be represented by a matrix  $\mathbf{R} \in \mathbb{R}^{3 \times 3}$ , which produces rotations to vectors through the matrix product

$$r(\mathbf{v}) = \mathbf{R}\mathbf{v}. \quad (3.4)$$

### The Rotation Group and the Quaternion

In this section, we would like to mention the connection between quaternions and rotation matrices as representations of the rotation group  $SO(3)$ . The set of unit quaternions forms a group under the operation of multiplication. This group is topologically a 3-sphere, that is the 3-dimensional surface of the unit sphere of  $\mathbb{R}^4$ , and is commonly noted as  $S^3$ .

Similar to Eq. (3.1), the corresponding two properties of quaternions as rotation operators are

$$\begin{aligned} ||\mathbf{q}||^2 &= 1 \\ \mathbf{q}^* \otimes \mathbf{q} &= \mathbf{q} \otimes \mathbf{q}^* = 1 \end{aligned} \quad (3.5)$$

where  $\mathbf{q}^*$  stands for the conjugate of the quaternion.

The equivalent formula to Eq. (3.4) can be written as quaternion rotation action

$$r(\mathbf{v}) = \mathbf{q} \otimes \mathbf{v} \otimes \mathbf{q}^*. \quad (3.6)$$

For the sake of completeness, we also mention the corresponding ordinary differential equations (ODE) for rotation matrix and quaternion, which later are needed for the derivation of EKF. For detailed discussion and derivation, one can refer to [Sola17].

$$\dot{\mathbf{R}} = \mathbf{R}[\boldsymbol{\omega}]_{\times} \quad (3.7)$$

$$\dot{\mathbf{q}} = \frac{1}{2} \mathbf{q} \otimes \boldsymbol{\omega} = \frac{1}{2} \boldsymbol{\Omega}_R(\boldsymbol{\omega}) \mathbf{q} \quad (3.8)$$

Where  $[\cdot]_{\times}$  stands for skew-symmetric operator, and  $\boldsymbol{\Omega}_R(\boldsymbol{\omega}) \in \mathbb{R}^{4 \times 4}$  is the right-quaternion-product matrix for pure quaternion  $\boldsymbol{\Omega} = [0, \omega_x, \omega_y, \omega_z]^T$

$$[\boldsymbol{\omega}]_{\times} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \quad (3.9)$$

$$\boldsymbol{\Omega}_R(\boldsymbol{\omega}) = \begin{bmatrix} 0 & -\boldsymbol{\omega}^T \\ \boldsymbol{\omega} & -[\boldsymbol{\omega}]_{\times} \end{bmatrix}. \quad (3.10)$$

### Rigid Body Motion

We now consider representations for the combined orientation and position of a rigid body. A natural choice would be to use a rotation matrix  $\mathbf{R} \in SO(3)$  to represent the orientation of the body frame  $B$  in the fixed frame  $W$  and a vector  $\mathbf{p} \in \mathbb{R}^3$  to represent the origin of  $B$  in  $W$ . Rather than identifying  $\mathbf{R}$  and  $\mathbf{p}$  separately, they are packed into a single matrix as “The special Euclidean group  $SE(3)$ ”.

The special Euclidean group  $SE(3)$ , also known as the group of rigid-body motions or homogeneous transformation matrices in  $\mathbb{R}^3$ , is the set of all  $4 \times 4$  real matrices  $\mathbf{T}$  of the form

$${}^W\mathbf{T}_{WB} = \begin{bmatrix} {}^W\mathbf{R}_{WB} & {}^W\mathbf{p}_{WB} \\ 0 & 1 \end{bmatrix} \quad (3.11)$$

where  $\mathbf{R} \in SO(3)$ , and  $\mathbf{p} \in \mathbb{R}^3$  are respectively a rotation matrix and a column vector.

There are three major uses for a transformation matrix  $\mathbf{T}$ :

- To represent the configuration (position and orientation) of a rigid body.
- To change the reference frame in which a vector or frame is represented.
- To displace a vector or frame.

## 3.2 Kalman Filter

Kalman filtering [Kalman60] is based on linear dynamical systems discretized in the time domain. We consider the linear discrete dynamic system as described in Eq. (3.12), where both prediction  $\mathbf{F}_k$  and measurement  $\mathbf{H}_k$  models are corrupted by zero-mean Gaussian process noise  $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$ , and measurement noise  $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$  respectively

$$\begin{aligned}\mathbf{x}_k &= \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k \\ \mathbf{y}_k &= \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k.\end{aligned}\tag{3.12}$$

The goal of the Kalman Filter is to estimate the mean  $\boldsymbol{\mu}$  and covariance matrix  $\mathbf{P}$  (which specifies the uncertainty of the estimate) of the Gaussian distribution over the state  $\mathbf{x}_k$ , given the previous state  $\mathbf{x}_{k-1}$ , the current control input  $\mathbf{u}_k$ , and the current measurement  $\mathbf{y}_k$ .

The filter involves two steps: propagating the estimates through the system in the prediction step to produce “*a priori*” estimates  $\boldsymbol{\mu}_k^-$  and  $\mathbf{P}_k^-$  (indicated by minus superscript), and then updating the estimates using a measurement in the update step to produce “*a posteriori*” estimates  $\boldsymbol{\mu}_k^+$  and  $\mathbf{P}_k^+$  (indicated by plus superscript).

The Kalman filter is a recursive estimator. This means that only the estimated state from the previous time step and the current measurement are needed to compute the estimate for the current state.

### Prediction Step

First we calculate the predicted “*a priori*” state estimate and estimate covariance matrix

$$\begin{aligned}\boldsymbol{\mu}_{k+1}^- &= \mathbf{F}_k \boldsymbol{\mu}_k^+ + \mathbf{B}_k \mathbf{u}_k \\ \mathbf{P}_{k+1}^- &= \mathbf{F}_k \mathbf{P}_k^+ \mathbf{F}_k^T + \mathbf{Q}_k.\end{aligned}\tag{3.13}$$

### Update Step

Then the updated “a posteriori” state estimate and estimate covariance matrix are computed as follows

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_{k+1}^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_{k+1}^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \\ \mathbf{P}_{k+1}^+ &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k+1}^- \\ \boldsymbol{\mu}_{k+1}^+ &= \boldsymbol{\mu}_{k+1}^- + \mathbf{K}_k (\mathbf{y}_k - \mathbf{H}_k \boldsymbol{\mu}_{k+1}^-) \end{aligned} \quad (3.14)$$

where  $\mathbf{K}_k$  is the optimal Kalman gain.

## 3.3 Extended Kalman Filter

Suppose we have the discrete-time nonlinear system as shown in the Eq. (3.15). The state is first predicted using the non-linear discrete transition function  $\mathbf{f}(\cdot)$  (prediction model) and then corrected by the observation function  $\mathbf{h}(\cdot)$  (measurement model). The functions are corrupted by zero-mean Gaussian process noise  $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}_k)$  and measurement noise  $\mathbf{v}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_k)$

$$\begin{aligned} \mathbf{x}_k &= \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{u}_{k-1}, \mathbf{w}_k) \\ \mathbf{y}_k &= \mathbf{h}(\mathbf{x}_k, \mathbf{v}_k). \end{aligned} \quad (3.15)$$

We can simply linearize these models around the current estimate at  $(t_k)$ , in order to find the prediction and measurement Jacobians as follow

$$\mathbf{F}_k = \left( \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right)_{\boldsymbol{\mu}_k^-}, \quad \mathbf{H}_k = \left( \frac{\partial \mathbf{h}}{\partial \mathbf{x}} \right)_{\boldsymbol{\mu}_k^-}. \quad (3.16)$$

Then by having the process and measurement Jacobians, the propagation of the error covariance matrix and calculation of the Kalman gain would be the same as the Kalman filter. Finally, we use the nonlinear models to find the mean of the state in the prediction and update step.

For the “prediction step” we have

$$\begin{aligned} \boldsymbol{\mu}_{k+1}^- &= \mathbf{f}(\boldsymbol{\mu}_k^+, \mathbf{u}_k, 0) \\ \mathbf{P}_{k+1}^- &= \mathbf{F}_k \mathbf{P}_k^+ \mathbf{F}_k^T + \mathbf{Q}_k \end{aligned} \quad (3.17)$$

and the “update step” is as follow

$$\begin{aligned} \mathbf{K}_k &= \mathbf{P}_{k+1}^- \mathbf{H}_k^T (\mathbf{H}_k \mathbf{P}_{k+1}^- \mathbf{H}_k^T + \mathbf{R}_k)^{-1} \\ \mathbf{P}_{k+1}^+ &= (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k+1}^- \\ \boldsymbol{\mu}_{k+1}^+ &= \boldsymbol{\mu}_{k+1}^- + \mathbf{K}_k (\mathbf{y}_k - \mathbf{h}(\boldsymbol{\mu}_{k+1}^-, 0)). \end{aligned} \quad (3.18)$$

This method is known as the “Extended Kalman Filter” (EKF). Although optimality and convergence are no longer guaranteed, this is a common approach for nonlinear estimation, and we choose the EKF over alternative frameworks such as the Unscented Kalman Filter [BloeschEtAl13b], for its simplicity and low computational cost.

### 3.4 Mathematical Derivation of EKF

We would like to track the placement (position and orientation), and the linear velocity of an articulated floating base robot with two or more legs and equipped with an onboard IMU, joint sensing of position and torque. We adopt the estimation approach for extended Kalman filtering framework as presented in [CamurriEtAl20]. In this work, we focus on the Solo12 quadruped, but this approach is general and can be applied on the robots of the same type with differences only in the link lengths and sensor locations.

#### Frames Definitions

In Fig. 3.1 we illustrate the reference frames relevant to our estimation problem based on [CamurriEtAl20]. The inertial frame  $W$  and the base frame  $B$  are rigidly attached to the ground and the robot’s floating base, respectively. The IMU frame  $I$  at the IMU sensor origin is also rigidly attached to the floating base. The relative placements (translation and orientation) of these frames are known by design, and they can be retrieved with respective transformation matrices. One or more temporal contact frame(s)  $K$  are created when a foot comes into contact with the ground.

Reference frame conventions for a typical quadruped:

- The world frame  $W$  is fixed to the earth.
- The base frame  $B$  is rigidly attached to the robot’s base.
- The IMU frame  $I$  is rigidly attached to the robot’s chassis.
- The contact frame  $K$ , is perpendicular to the ground.

#### Notations

In the remainder of this section, we adopt the following conventions; the robot position  $\mathbf{p} = {}^W\mathbf{p}_B \in \mathbb{R}^3$  and orientation  $\mathbf{q} = {}^W\mathbf{q}_B \in S^3$  (with the corresponding

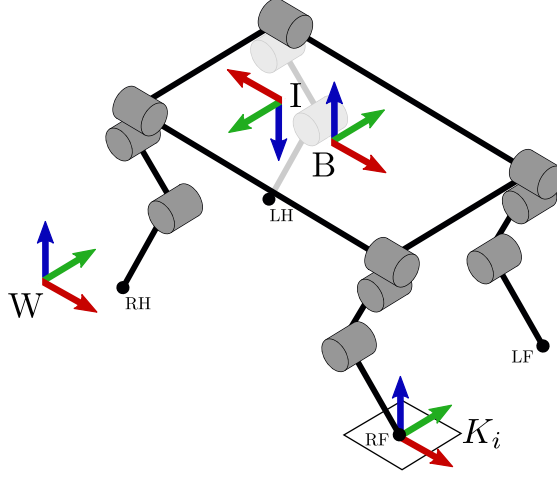


Figure 3.1: Reference frame conventions for a typical quadrupe.

rotation matrix  $\mathbf{R} = {}^W\mathbf{R}_B \in SO(3)$ ) are from world to base and expressed in world coordinates; the robot linear and angular velocities are respectively  $\mathbf{v} = {}^B\mathbf{v}_{WB}$ ,  $\boldsymbol{\omega} = {}^B\boldsymbol{\omega}_{WB}$ , and  $\mathbf{v}, \boldsymbol{\omega} \in \mathbb{R}^3$  from world to base and expressed in base coordinate; the IMU accelerometer and gyroscope biases are respectively  ${}^I\mathbf{b}^a, {}^I\mathbf{b}^\omega \in \mathbb{R}^3$  expressed in IMU coordinate. A time-dependent vector quantity  $\mathbf{a}$  computed at time  $t_k$  is shortened as  $\mathbf{a}_k = \mathbf{a}(t_k)$ .

### State Definition

The robot state is defined as the vector concatenating position, orientation, linear velocity, and IMU biases. The angular velocity does not appear, as it is assumed to be directly measured by the IMU once the bias is properly compensated. The state at time  $t_k$  is

$$\mathbf{x}_k = [\mathbf{p}_k, \mathbf{v}_k, \mathbf{q}_k, \boldsymbol{\omega}_k, \mathbf{b}_k^a, \mathbf{b}_k^\omega]. \quad (3.19)$$

### Process Model

The acceleration and angular velocity of the floating base are sensed by the IMU at high frequencies in the range 0.4–1 kHz. They are affected by bias and zero-mean Gaussian noise. By considering the offset between the base and IMU frame  ${}^B\mathbf{r}_{IB}$ , and the angular acceleration of the base (it can be computed by a simple finite differentiation from angular velocity); we can express the IMU sensed values

in the base frame

$$\mathbf{u}_t = \begin{bmatrix} \boldsymbol{\omega}_t \\ \mathbf{a}_t \end{bmatrix} = \begin{bmatrix} {}^B \mathbf{R}_I (\tilde{\boldsymbol{\omega}} - \mathbf{b}^\omega - \boldsymbol{\eta}^\omega) \\ {}^B \mathbf{R}_I (\tilde{\mathbf{a}} - \mathbf{b}^a - \boldsymbol{\eta}^a) + \mathbf{a}_{\alpha, \omega} \end{bmatrix} \quad (3.20)$$

where  ${}^B \mathbf{R}_I$  is the rotational part of the rigid transform between the IMU and base frames, and  $\mathbf{a}_{\alpha, \omega}$  is the effect of angular acceleration ( $\boldsymbol{\alpha}$ ), and angular velocity ( $\boldsymbol{\omega}$ ) of the floating base and is calculated as follow

$$\mathbf{a}_{\alpha, \omega} = (\boldsymbol{\alpha} \times {}^B \mathbf{r}_{IB}) + \boldsymbol{\omega} \times (\boldsymbol{\omega} \times {}^B \mathbf{r}_{IB}). \quad (3.21)$$

By taking time derivative from the states, we can find the non-linear dynamics. For the base position we have

$$\dot{\mathbf{p}} = {}^W \mathbf{v}_{WB} = \mathbf{R} \mathbf{v}. \quad (3.22)$$

By taking time derivative from Eq. (3.22), we can find the ordinary differential equation for the base linear velocity

$$\begin{aligned} {}^W \dot{\mathbf{v}}_{WB} &= \dot{\mathbf{R}} \mathbf{v} + \mathbf{R} \dot{\mathbf{v}} \rightarrow \\ \mathbf{R} \dot{\mathbf{v}} &= -\dot{\mathbf{R}} \mathbf{v} + {}^W \dot{\mathbf{v}}_{WB} \rightarrow \\ \mathbf{R} \dot{\mathbf{v}} &= -\mathbf{R}[\boldsymbol{\omega}]_{\times} \mathbf{v} + {}^W \dot{\mathbf{v}}_{WB} \rightarrow \\ \dot{\mathbf{v}} &= \mathbf{R}^T \left( -\mathbf{R}[\boldsymbol{\omega}]_{\times} \mathbf{v} + ({}^W \mathbf{a}_{WB} + \mathbf{g}) \right) \rightarrow \\ \dot{\mathbf{v}} &= -[\boldsymbol{\omega}]_{\times} \mathbf{v} + \mathbf{R}^T \mathbf{g} + {}^B \mathbf{a}_{WB}. \end{aligned} \quad (3.23)$$

The dynamics for the rotation part comes from Eq. (3.8), and we consider random walk noise for the biases. Finally, the quantities from Eq. (3.20) are used as inputs to the nonlinear continuous process model in the EKF

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\mathbf{p}} \\ \dot{\mathbf{v}} \\ \dot{\mathbf{q}} \\ \dot{\mathbf{b}}^a \\ \dot{\mathbf{b}}^\omega \end{bmatrix} = \begin{bmatrix} \mathbf{R} \mathbf{v} \\ -[\boldsymbol{\omega}_t]_{\times} \mathbf{v} + \mathbf{R}^T \mathbf{g} + \mathbf{a}_t \\ \frac{1}{2} \boldsymbol{\Omega}_R(\boldsymbol{\omega}_t) \mathbf{q} \\ \boldsymbol{\eta}_b^a \\ \boldsymbol{\eta}_b^\omega \end{bmatrix}. \quad (3.24)$$

Since Eq. (3.24) is continuous and nonlinear, it must be discretized and linearized for the implementation purposes. Following the EKF framework outlined in section 3.3, this requires two systems of equations: (1) a discrete, nonlinear system for prediction of the mean of the state and measurement, and (2) a discrete, linear system for propagation of the state covariance matrix through the prediction model and for computation of the Kalman gain in the update step.

For the “prediction step”, given Eq. (3.24), we can predict the mean of the state,  $\boldsymbol{\mu}_k$ , by simple first order Euler integration over the period  $\Delta t = t_k - t_{k-1}$

$$\boldsymbol{\mu}_{k+1}^- = \begin{bmatrix} \mathbf{p}_k \\ \mathbf{v}_k \\ 0 \\ \mathbf{b}_k^a \\ \mathbf{b}_k^\omega \end{bmatrix} + \begin{bmatrix} (\mathbf{R}_k \mathbf{v}_k) \Delta t \\ (-[\boldsymbol{\omega}_k]_\times \mathbf{v}_k + \mathbf{R}_k^T \mathbf{g} + \mathbf{a}_k) \Delta t \\ \mathbf{R}_k \exp([\boldsymbol{\omega}_k \Delta t]_\times) \\ 0 \\ 0 \end{bmatrix}. \quad (3.25)$$

Note that the attitude is integrated separately using the exponential map between the Lie group of rotations and its Lie algebra at the identity.

Recall that discretized, linearized dynamics are required in order to propagate the error estimate covariance matrix. We adopt the approach in [RotellaEtAl14], and we first linearize and then discretize to get the continuous prediction Jacobian  $\mathbf{F}_c$ , and noise Jacobian  $\mathbf{L}_c$

$$\mathbf{F}_c = \begin{bmatrix} 0 & \mathbf{R} & -\mathbf{R}[\mathbf{v}]_\times & 0 & 0 \\ 0 & -[\boldsymbol{\omega}]_\times & [\mathbf{R}^T \mathbf{g}] & -\mathbf{I} & -[\mathbf{v}]_\times \\ 0 & 0 & -[\boldsymbol{\omega}]_\times & 0 & -\mathbf{I} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{F}_c \in \mathbb{R}^{15 \times 15} \quad (3.26)$$

$$\mathbf{L}_c = \begin{bmatrix} 0 & 0 & 0 & 0 \\ -\mathbf{I} & -[\mathbf{v}]_\times & 0 & 0 \\ 0 & -\mathbf{I} & 0 & 0 \\ 0 & 0 & \mathbf{I} & 0 \\ 0 & 0 & 0 & \mathbf{I} \end{bmatrix}, \quad \mathbf{L}_c \in \mathbb{R}^{15 \times 12}. \quad (3.27)$$

It is assumed for simplicity that the covariance matrix of each process noise vector is diagonal with equal entries. The continuous process noise covariance matrix is then

$$\mathbf{Q}_c = \text{diag}(\mathbf{Q}_a, \mathbf{Q}_\omega, \mathbf{Q}_{b_a}, \mathbf{Q}_{b_\omega}). \quad (3.28)$$

The covariance matrices,  $\mathbf{Q}_a, \mathbf{Q}_\omega, \mathbf{Q}_{b_a}, \mathbf{Q}_{b_\omega}$  are obtained from [Sola17] by integrating the covariances of  $\mathbf{a}_n, \boldsymbol{\omega}_n, \mathbf{a}_n^b, \boldsymbol{\omega}_n^b$ , over the step time  $\Delta t$

$$\mathbf{Q}_a = \sigma_{a_n}^2 \Delta t^2 \mathbf{I} \quad [\text{m}^2/\text{s}^2] \quad (3.29)$$

$$\mathbf{Q}_\omega = \sigma_{\omega_n}^2 \Delta t^2 \mathbf{I} \quad [\text{rad}^2] \quad (3.30)$$

$$\mathbf{Q}_{b_a} = \sigma_{a_n^b}^2 \Delta t \mathbf{I} \quad [\text{m}^2/\text{s}^4] \quad (3.31)$$

$$\mathbf{Q}_{b_\omega} = \sigma_{\omega_n^b}^2 \Delta t \mathbf{I} \quad [\text{rad}^2/\text{s}^4] \quad (3.32)$$

where the corresponding covariances ( $\sigma$ ) are to be determined from the information in the IMU data sheet, table 3.1.

Table 3.1: IMU sensor outputs.

	Accelerometer	Gyroscope	Magnetometer
Measurement range	$\pm 8 \text{ g}$	$300^\circ/\text{sec}$	$\pm 8 \text{ Gauss}$
Non-linearity	$\pm 0.02\% \text{ fs}$	$\pm 0.02\% \text{ fs}$	$\pm 0.3\% \text{ fs}$
Resolution	$< 0.1 \text{ mg}$	$< 0.003^\circ/\text{sec}$	–
Bias instability	$\pm 0.04 \text{ mg}$	$8^\circ/\text{hr}$	–
Initial bias error	$\pm 0.002 \text{ g}$	$\pm 0.04^\circ/\text{sec}$	$\pm 0.003 \text{ Gauss}$
Noise density	$20 \mu\text{g}/\sqrt{\text{Hz}}$	–	$400 \mu\text{Gauss}/\sqrt{\text{Hz}}$

Assuming a zero-order hold on inputs over the interval  $\Delta t = t_k - t_{k-1}$ , the discretized prediction Jacobian and discretized error covariance matrix are given by [Van Loan78]

$$\mathbf{F}_k = \exp(\mathbf{F}_c \Delta t) \quad (3.33)$$

$$\mathbf{Q}_{k-1} = \int_{t_{k-1}}^{t_k} \exp(\mathbf{F}_c(t_k - \tau)) \mathbf{L}_c \mathbf{Q}_c \mathbf{L}_c^T \exp(\mathbf{F}_c^T(t_k - \tau)) d\tau. \quad (3.34)$$

In practice, these expressions are often truncated at first order for simplicity to yield [RotellaEtAl14]

$$\mathbf{F}_k \approx \mathbf{I} + \mathbf{F}_c \Delta t \quad (3.35)$$

$$\mathbf{Q}_k \approx (\mathbf{F}_k \mathbf{L}_c \mathbf{Q}_c \mathbf{L}_c^T \mathbf{F}_k^T) \Delta t. \quad (3.36)$$

## Measurement Model

The linear velocity of the base is computed using leg odometry. When a foot is in contact with the ground, as shown in the Fig. 3.2 by simple vector addition we have

$${}^W \mathbf{p}_{WB} + {}^W \mathbf{R}_B \left( {}^B \mathbf{p}_{BK_i}(\alpha) \right) = {}^W \mathbf{r}_{WK_i} \quad (3.37)$$

where  ${}^B \mathbf{p}_{BK_i}(\alpha)$  is the position of the foot  $i$  relative to the base expressed in the base frame, which is a function of the joints angles of the corresponding leg. By taking time derivative from Eq. (3.37), and given the joint positions and velocities we can compute, using forward kinematics, the velocity and position of each end effector in the base frame. By assuming that the foot remains stationary while is in contact with the ground ( ${}^W \dot{\mathbf{r}}_{WK_i} = 0$ ), the measurement model can be written as follow [CamurriEtAl20]

$${}^W \dot{\mathbf{r}}_{WK_i} = {}^B \mathbf{v}_{WB} + {}^B \mathbf{v}_{BK_i} + {}^B \boldsymbol{\omega}_{WB} \times {}^B \mathbf{p}_{BK_i} \quad (3.38)$$

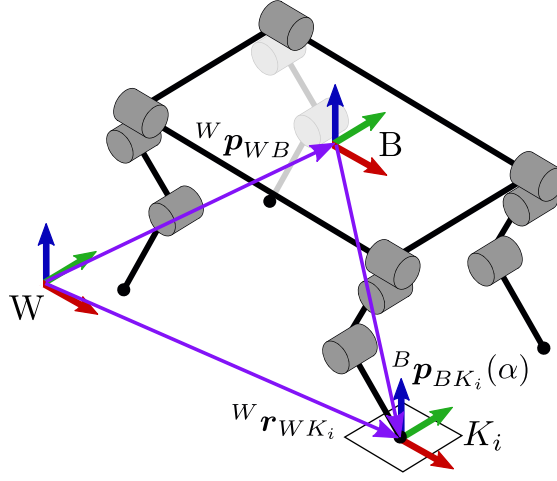


Figure 3.2: Measurement model.

where the angular velocity is measured directly by the IMU once the gyroscope bias was properly compensated. After determining the set of feet in contact with the ground (explained in details in the next section), the robot's base linear velocity can be measured from each foot in contact. By having the joint positions and velocities sensed from the encoders  $\mathbf{q}$ ,  $\dot{\mathbf{q}}$  and considering their additive noise  $\boldsymbol{\eta}^q$ ,  $\boldsymbol{\eta}^v$  the measurement model can be rewritten as follow

$$\tilde{\mathbf{v}}_k = -\mathbf{J}(\tilde{\mathbf{q}}_k - \boldsymbol{\eta}^q)(\tilde{\dot{\mathbf{q}}}_k - \boldsymbol{\eta}^v) - \boldsymbol{\omega} \times f_k(\tilde{\mathbf{q}} - \boldsymbol{\eta}^q). \quad (3.39)$$

We collect all the effects of noise into one additive term  $\boldsymbol{\eta}^v$

$$\tilde{\mathbf{v}}_k = -\mathbf{J}(\tilde{\mathbf{q}}_k)\tilde{\dot{\mathbf{q}}}_k - \boldsymbol{\omega} \times f_k(\tilde{\mathbf{q}}) + \boldsymbol{\eta}^v. \quad (3.40)$$

First, we compute  $\tilde{\mathbf{v}}_k$ , the measured base linear velocity, from forward kinematics of each foot in contact. Then by linearizing the measurement model Eq. (3.38) we can find the continuous measurement Jacobian

$$\mathbf{H}_c = \begin{bmatrix} 0 & \mathbf{I} & 0 & 0 & [{}^B\mathbf{S}_{BF_1}]_{\times} \\ 0 & \mathbf{I} & 0 & 0 & [{}^B\mathbf{S}_{BF_2}]_{\times} \\ 0 & \mathbf{I} & 0 & 0 & [{}^B\mathbf{S}_{BF_2}]_{\times} \\ 0 & \mathbf{I} & 0 & 0 & [{}^B\mathbf{S}_{BF_2}]_{\times} \end{bmatrix}, \quad \mathbf{H}_c \in \mathbb{R}^{12 \times 15} \quad (3.41)$$

which  ${}^B\mathbf{S}_{BF_i}$  corresponds to the position of the foot  $i$  (in contact with the ground) relative to the base frame, expressed in the base frame. Note that, if the foot  $i$  is not in contact with the ground, the corresponding row in the measurement Jacobian would be zero. The measurement model has algebraic dynamics, therefore

the measurement Jacobian and the measurement noise covariance matrices are the same for continuous and discrete-time systems

$$\mathbf{H}_k = \mathbf{H}_c \quad (3.42)$$

$$\mathbf{R}_k = \mathbf{R}_c. \quad (3.43)$$

The measurement noise covariance matrix  $\mathbf{R}_c$  should be determined experimentally in order to have an optimal performance, and it can be seen as a tuning knob in the EKF which handles the uncertainty in the measurement model, and it is also one way to incorporate the uncertainty in contact detection and slippage of the foot in contact in our estimation framework.

Eventually, by having  $\mathbf{H}_k$ , and  $\mathbf{R}_k$  we apply the update step of the EKF as described in Eq. (3.18) to get the “a posteriori” estimates.

### 3.5 Contact Detection

Quadruped robots are typically equipped with high-precision joint encoders from which low-noise joint velocity measurements can be derived. However, achieving accurate contact estimation is a major challenge since quadrupeds are not typically equipped with direct contact sensors, as they easily break during operation.

Quadruped robot feet are usually approximated to be point-like and then assumed to exert only pure forces onto the ground. By assuming that the robot base remains flat during contact transitions, we can consider an equal distribution of the robot total weight over the feet in contact with the ground. We use a Schmitt trigger to implement a robust hysteresis on the contact decision. we choose upper and lower bounds as two thresholds for contact classification based on measured force for each end effector. If the norm of the force at each end effector is higher than the upper threshold, we consider the foot as in contact with the ground, and if the norm is below the lower threshold, the end effector is no longer in contact. This allows some hysteresis in the contact detection which helps reject outliers due to high joint acceleration when the foot leaves the ground. We compute the end effector force norm using the joint torque as follows

$$\begin{aligned} \mathbf{F}_i &= (\mathbf{S}_i \mathbf{J}_i^T)^{-1} \mathbf{S}_i \boldsymbol{\tau} \\ F_i &= \|\mathbf{F}_i\| \end{aligned} \quad (3.44)$$

where  $\mathbf{S}_i$  is the selection matrix for the joints of leg  $i$ .

### 3.6 Implementation

In this section we present the overview of the implemented EKF, and briefly mention the simulation environment (used for debugging and testing), real hardware and related open source software.

#### Software Structure

The schematic of EKF is demonstrated in Fig. 3.3.

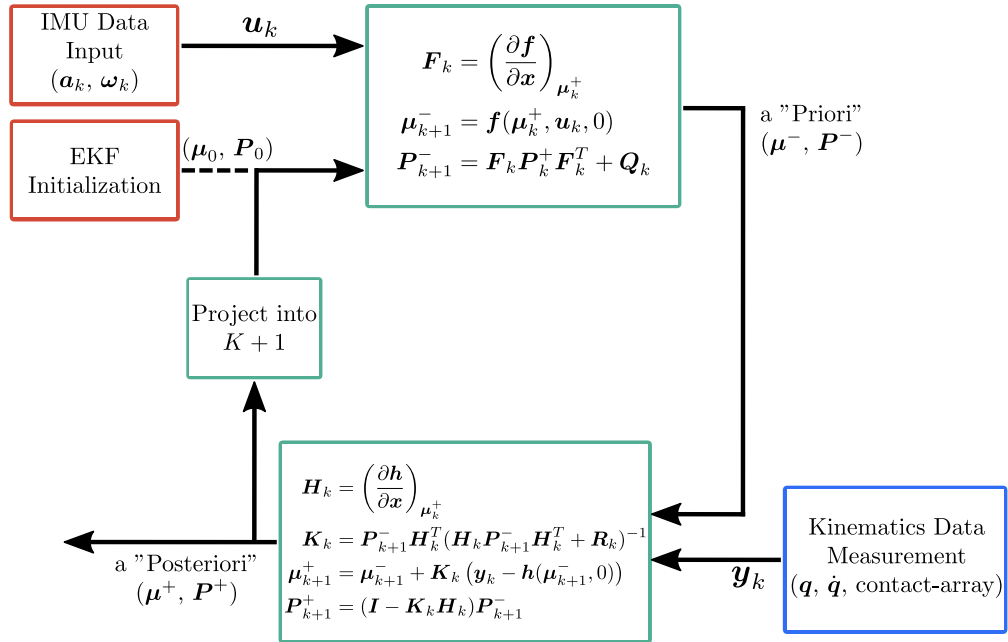


Figure 3.3: A summary of the extended Kalman filter.

#### Software Packages

We are using PINOCCHIO [CarpentierEtAl21] for computing the kinematics and dynamics of a robot model. PINOCCHIO instantiates the state-of-the-art “Rigid Body Algorithms” for poly-articulated systems based on revisited [Featherstone08] algorithms. PINOCCHIO is a C++ modeling library with PYTHON bindings.

### Simulation

The physics simulator used for this project is PYBULLET [CoumansBai20]. We use simulation results to validate the estimation framework before implementing on the real hardware.

### Real Hardware

On the real hardware, the framework is designed using Robot Operating System (ROS2) as its middle-ware to communicate with robot and perception services. ROS provides different set of programs, tools and libraries for the development of robotic software application. It has been open sourced, and a huge community constantly contributes to the development of different support packages. The complex framework, is simplified by providing tools for hardware abstraction, management, driver tools and visualizer to easy work on different applications. Moreover, the software architecture and tool kits for the overall framework are available open source in GitHub repository of [Machines in Motion Laboratory].

# Chapter 4

## Controllers

In this chapter we briefly present the “Nonlinear Model Predictive Control Framework” which is developed by [MeduriEtAl22] for whole body motion planning and implemented for Solo12.

### 4.1 MPC Controller

The rigid body dynamics of any robot can be described as follows

$$\mathbf{M}(\mathbf{q})\dot{\mathbf{v}} + \mathbf{N}(\mathbf{q}, \mathbf{v}) = \mathbf{S}^T \boldsymbol{\tau} + \sum_{j=1}^N \mathbf{J}_j^T \boldsymbol{\lambda}_j \quad (4.1)$$

where  $\mathbf{q}$  represents the generalized joint configuration of the robot, and  $\mathbf{v}$  is the generalized velocity vector.  $\mathbf{M}(\mathbf{q})$  is the mass matrix for the given robot configuration,  $\mathbf{N}(\mathbf{q}, \mathbf{v})$  is coriolis matrix which consist of the nonlinear effects of the dynamics,  $\boldsymbol{\tau}$  is the vector of joint torques,  $\mathbf{S}$  is the selection matrix that defines the under actuation of the robot,  $\mathbf{J}_j$  are the end effector jacobians and  $\boldsymbol{\lambda} = [\mathbf{f}, \boldsymbol{\tau}]$  is the vector of forces and torques applied at each end effector.

In the case of a floating base robot, the dynamics can be split into two parts, the actuated and unactuated part

$$\mathbf{M}_u(\mathbf{q})\dot{\mathbf{v}} + \mathbf{N}_u(\mathbf{q}, \mathbf{v}) = \sum_{j=1}^N \mathbf{J}_{u,j}^T \boldsymbol{\lambda}_j \quad (4.2)$$

$$\mathbf{M}(\mathbf{q})_a \dot{\mathbf{v}} + \mathbf{N}_a(\mathbf{q}, \mathbf{v}) = \boldsymbol{\tau} + \sum_{j=1}^N \mathbf{J}_{a,j}^T \boldsymbol{\lambda}_j \quad (4.3)$$

where the subscript “a”, and “u” correspond to “actuated” and “unactuated” dynamics respectively. The unactuated dynamics is equivalent to the Newton-

Euler equations of the center of mass (CoM) for the robot's base

$$\begin{bmatrix} \dot{\ell} \\ \dot{\mathbf{k}} \end{bmatrix} = \begin{bmatrix} m\mathbf{g} + \sum_{j=1}^N n_j \mathbf{f}_j \\ \sum_{j=1}^N n_j \left( (\mathbf{r}_j - \mathbf{c}) \times \mathbf{f}_j + \boldsymbol{\tau}_j \right) \end{bmatrix} \quad (4.4)$$

where  $\ell$ ,  $\mathbf{k}$  are the linear and angular momentum,  $m$  is the robot mass,  $\mathbf{g}$  is the gravity vector,  $\mathbf{c}$  represents the center of mass location,  $n_j$  is a binary integer that describes whether the end effector  $j$  is in contact,  $\mathbf{f}_j$ ,  $\boldsymbol{\tau}_j$ ,  $\mathbf{r}_j$  are the end effector force, torque and location respectively.

Splitting the dynamics enables multi-contact motion generation by only considering the unactuated dynamics or centroidal dynamics of the robot. Subsequently, a feasible whole-body trajectory can then be determined based on the centroidal plan and desired whole body tasks, provided there is sufficient torque authority. This is an attractive approach since it breaks the original highly nonlinear optimization problem into two simpler sub-problems. A desired motion plan using the centroidal dynamics can be generated by solving the following discrete optimal control problem (OCP)

$$\begin{aligned} \min_{\mathbf{c}, \dot{\mathbf{c}}, \mathbf{k}, \mathbf{f}, \boldsymbol{\tau}} \cdot & \sum_{t=0}^{T-1} \phi_r^t(\mathbf{c}_t, \dot{\mathbf{c}}_t, \mathbf{k}_t, \mathbf{f}_t, \boldsymbol{\tau}_t) + \phi^T(\mathbf{c}_T, \dot{\mathbf{c}}_T, \mathbf{k}_T, \mathbf{f}_T, \boldsymbol{\tau}_T) \\ \text{s.t. } & \mathbf{c}_{t+1} = \mathbf{c}_t + \dot{\mathbf{c}}_t \Delta t \\ & \dot{\mathbf{c}}_{t+1} = \dot{\mathbf{c}}_t + \sum_{j=1}^N \left( (n_t^j \frac{\mathbf{f}_t^j}{m} + \mathbf{g}) \Delta t \right) \\ & \mathbf{k}_{t+1} = \mathbf{k}_t + \sum_{j=1}^N \left( n_t^j ((\mathbf{r}_t^j - \mathbf{c}_t) \times \mathbf{f}_t^j + \boldsymbol{\tau}_t^j) \Delta t \right) \\ & \forall_{t,j} \sqrt{(\mathbf{f}_{t,x}^j)^2 + (\mathbf{f}_{t,y}^j)^2} \leq \mu \mathbf{f}_{t,z}^j \\ & \forall_{t,j} \mathbf{f}_{t,z}^j \geq 0 \\ & \forall_{t,j} \mathbf{r}_t^j \in \boldsymbol{\Psi} \\ & \forall \mathbf{c}_t \in \boldsymbol{\Omega} \\ & \mathbf{c}_0, \dot{\mathbf{c}}_0 = \mathbf{c}_{init}, \dot{\mathbf{c}}_{init} \end{aligned} \quad (4.5)$$

where  $\phi_r^t$  is the running cost, and  $\phi^T$  is the terminal cost,  $\Delta t$  is the time discretization,  $\mu$  is the coefficient friction,  $\boldsymbol{\Psi}$  is the set of all allowed stepping locations,  $\boldsymbol{\Omega}$  are the kinematic constraints,  $\mathbf{c}_{init}$ ,  $\dot{\mathbf{c}}_{init}$  are the initial conditions of the robot center of mass (COM).

The optimal joint trajectory is generated by solving an inverse kinematics (IK) problem which tracks the optimal centroidal momentum obtained from the previous step using the centroidal momentum matrix, along with additional full body

tasks. The generated momentum trajectory from the inverse kinematics problem is then used as a soft constraint in the centroidal OCP to obtain a refined centroidal trajectory and forces. This process is iterated until the two sub-problems converge. Finally, inverse dynamics (based on within the Recursive Newton Euler Algorithm (RNEA)) [Featherstone08] is used to map state trajectories and contact forces to actuated joint torques

$$\boldsymbol{\tau}_{RNEA} = \mathbf{M}_a(\mathbf{q})\dot{\mathbf{v}} + \mathbf{N}_a(\mathbf{q}, \mathbf{v}) - \sum_{j=1}^N \mathbf{J}_{a,j}^T \boldsymbol{\lambda}_j. \quad (4.6)$$

An overview of the entire framework is shown in Fig. 4.1.

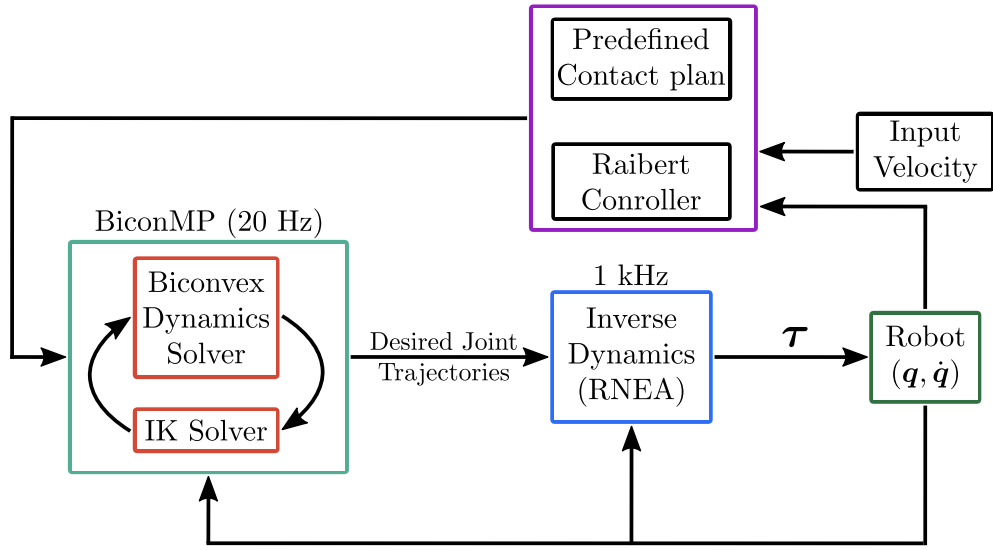


Figure 4.1: A bird's eye view of the nonlinear MPC framework [MeduriEtAl22].

Given the current states of the robot  $\mathbf{q}_{init}$ ,  $\mathbf{v}_{init}$ ,  $\dot{\mathbf{v}}_{init}$ , desired gait, planning horizon and velocity, a contact plan is either generated and adapted using the Raibert controller or pre-defined without contact adaptation for general motions. The BiConMP framework takes the input states and computes the optimal end effector forces, joint positions, joint velocities and joint acceleration trajectories for the entire horizon. Given the desired joint trajectories and contact forces, we use Eq. (4.6) along with a low joint impedance around the desired states to compute the desired torques at 1 kHz

$$\boldsymbol{\tau}_i = \boldsymbol{\tau}_{RNEA,i} + \mathbf{K}_p(\mathbf{q}_{d,i} - \mathbf{q}_{a,i}) + \mathbf{K}_d(\mathbf{v}_{d,i} - \mathbf{v}_{d,i}) \quad (4.7)$$

where subscripts “d” and “a” stand for “desired” and “actual” respectively. The desired torques is then sent to the robot which are tracked on board at 10 kHz. The entire full body planning loop is replanned at 20Hz(50ms) to update optimal trajectories.

## 4.2 Observer-Based Controller

As discussed in section 4.1, one important assumption in the MPC controller framework is that all the states of the robot,  $\mathbf{q}$  and  $\dot{\mathbf{q}}$ , are available for the control task. The joint positions and joint velocities are measured directly by the encoders, and for now the base placement,  $\mathbf{q}_b \in SE(3)$  (position and orientation), and base linear velocity,  $\mathbf{v}_b \in \mathbb{R}^3$ , are measured by motion capturing system. As mentioned before, the goal of EKF estimator is to have an accurate estimate of the robot base state completely based on proprioceptive sensors.

We wish to design an observer-based controller. In this framework, first the states are estimated by EKF, then they are used by the controller in order to generate the trajectories for given states and control input. One important aspect of such a structure, is observability of the states. This problem consists of identifying if a set of available measurements is enough to be able to estimate the state of the system. Since in our estimation problem, both the transition and observation functions in Eq. (3.15) are nonlinear, we need to perform a “nonlinear observability analysis”. Analyzing the observability characteristics of the underlying nonlinear system reveals the theoretical limitations of state estimation and can validate the employed approach. A detailed analysis of observability, when the measurement model is the base linear velocity measured by the forward kinematics of the leg, is given in [BloeschEtAl13b]. We briefly mention two essential points of this analysis. First, with the proposed EKF, the four dimensional manifold composed of robot’s base position and yaw angle (rotation around gravity vector  $\mathbf{g}$ ) is always unobservable. The second point is that as long as one foot is in contact the base linear velocity and robot’s base roll and pitch angles are always observable.

The unobservability of the base position, and base yaw angle in our proposed EKF framework imposes some challenges for the control task. In order to deal with this issue, we would like to distinguish between two types of controllers in legged robot locomotion, “low-level” and “high-level” control.

The low-level controller is responsible for motion planning and trajectory generation of the floating base. This controller is an essential part for the task of locomotion. On the other hand, the high-level controllers are implemented on top in order to tackle the different tasks such as localization and navigation in the environment.

For the task of locomotion, we need the robot general configuration (including base position, orientation and joint positions), and robot velocity (including base linear, angular velocity and joint velocities). We already mentioned that the joint positions and velocities are sensed through encoders, and angular velocity is measured by the IMU. For the observer-based controller the only challenge is the

unobservability of base position and yaw angle. But for the low-level controller, and for motions on flat terrain we only need the relative position of the base to the ground, and the  $x, y$  position and yaw angle are needed for high-level tasks. Therefore, we will see later that by some modification in the proposed EKF, we can use the estimated states in a feedback scheme with MPC controller in order to generate the trotting motion on the robot without using any exteroceptive sensing, such as Vicon or mounted cameras on the robot.

In Fig. 4.2, an overview of the observer-based controller is shown, which is later tested on Solo12 robot with trotting motion.

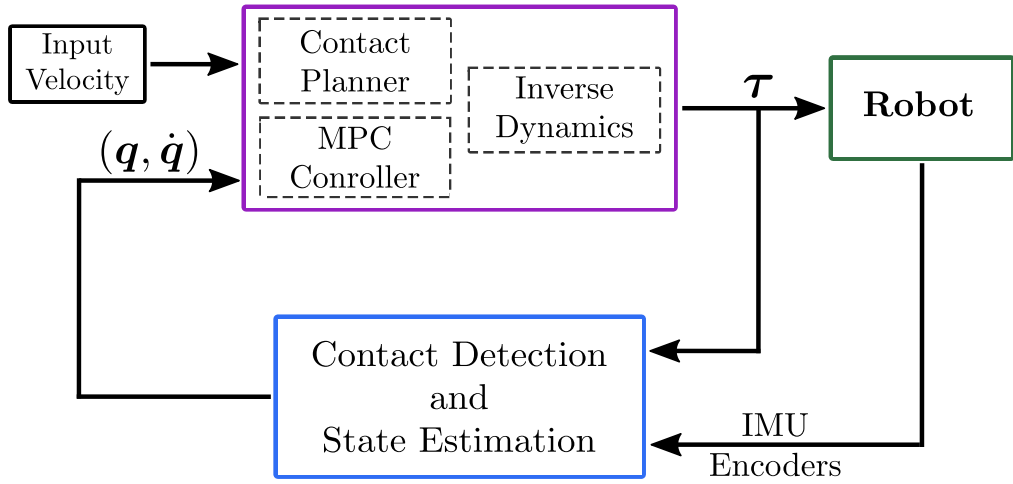


Figure 4.2: An overview of the observer-based controller.



# Chapter 5

## Results and Discussion

In this chapter, first we introduce the experimental setup used for this project, then we present the experimental results of performing estimation of the base states for a variety of different gaits on the quadruped Solo12. We compare the estimated states by the EKF estimator with the states measured by motion capturing system (Vicon), as the ground truth.

### 5.1 Experimental Setup

This section describes components and hardware which are used as the experimental setup (as shown in Fig. 5.1) in this project. For more details, one can refer to [Grimminger21].

#### Solo12 Quadruped

The Solo12 quadruped robot, with the total weight of 2.5 kg, consists of:

- A body module and 4 identical 3DoF legs.
- A wire connection for power supply and Ethernet communication.
- Electronics including: master board, micro drivers, brushless actuator module, inertia measurement unit (IMU).
- Vicon object with reflective markers.



Figure 5.1: Experimental setup.

### Motion Capture System - Vicon

Motion capture is a technology to track the motion of an object or a fixed marker. Its high frequency and accurate information allow closing control loop reliably for high dynamic motions. The frequency of the capture system is 1 kHz with max latency of 15 ms (including communication between Vicon workstation to destination PC via network). The motion capture system used for this project is Vicon V-16 as shown in Fig. 5.1. It provides the base position and orientation of the robot with accuracy up to  $63 \pm 5 \mu\text{m}$ . The system consists of:

- 6 motion capturing cameras around the robot.
- Motion capturing software.
- Vicon object with markers on the robot.

## 5.2 Experimental Results

As mentioned in section 1.1, the final goal of developing legged robot platforms is to deploy them in different environment, in such a way that they can reliably locomote across rough terrain and be stable in the presence of disturbances. For this reason, we inspect our estimation framework with different basic motions,

necessary for having robust quadruped locomotion. These motions are mainly, maintaining balance while all feet are in contact and the base of the robot is rotating, trotting forward and backward in order to be able to navigate in the environment, and finally jumping forward and backward in case that the robot needs to jump over some obstacles on the ground.

We analyze our estimation framework with four different motions:

1. Moving the base while balancing.
2. Trotting gait (using diagonal legs is pairs).
3. Jumping motion.
4. Trotting gait with EKF.

For the first motion, a trajectory is generated using the trajectory optimization framework, and a whole-body controller is used to track the trajectory while satisfying friction cone constraints. For the other motions, we are using the MPC and observer-based controllers introduced in chapter 4. For the first three motions, we use the motion capturing system to measure the base states and we feed it in our MPC controller.

We investigate the performance and limitations of EKF framework in each motion, by comparing the graph for the estimated states (position, orientation, and linear velocity of the base) with the ground truth (measurement from Vicon). Finally in the forth scenario we use the proposed observer-based controller introduced in section 4.2 with some modification in order to close the control loop around the estimated states for trotting motion. We show that by considering the robot's base remaining at the same height during this motion, we can use the EKF estimator in an observe-based controller to control the robot locomotion without using any measurement from motion capturing system.

### 5.2.1 Moving the Base while Balancing

The first scenario we consider, is a motion in which we move the base without any contact switching Fig. 5.2. The main goal of this test is to evaluate the quality of the estimated states in isolation from uncertainties introduced by contact switching, and as well as our contact detection framework as described in the section 3.5.

The estimated states are shown in Fig. 5.3 for the base position, in Fig. 5.4 for the base linear velocity, and in Fig. 5.5 for the base orientation. As expected, we have an accurate estimation for the observable states, linear velocity, roll and

pitch angles of the base. Root mean square error (RMSE) values for these states are given in table 5.1. Considering the unobservability of the position and yaw angle of the base, we can explain the drift seen over the time for these states.



Figure 5.2: Snapshots of the first motion scenario.

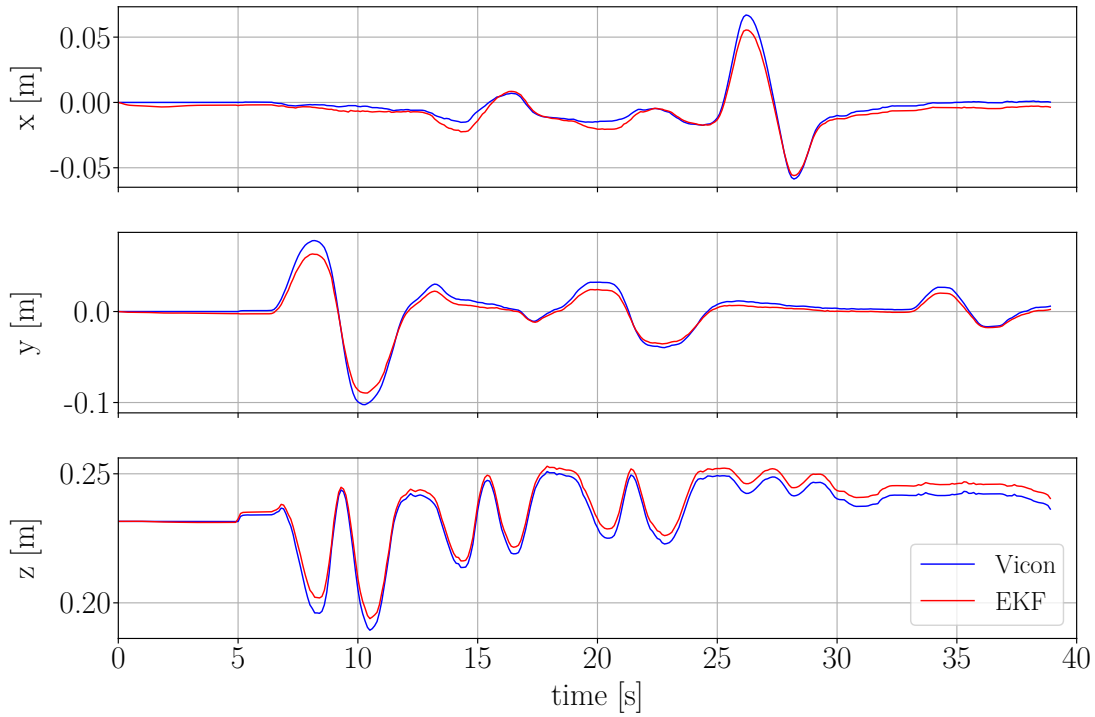


Figure 5.3: Motion #1: Estimation of the base position.

Table 5.1: Motion #1: RMSE values.

	Linear velocity (m/s)			Rotation (rad)	
	$V_x$	$V_y$	$V_z$	Roll	Pitch
RMSE	0.0163	0.0147	0.0170	0.0021	0.0104

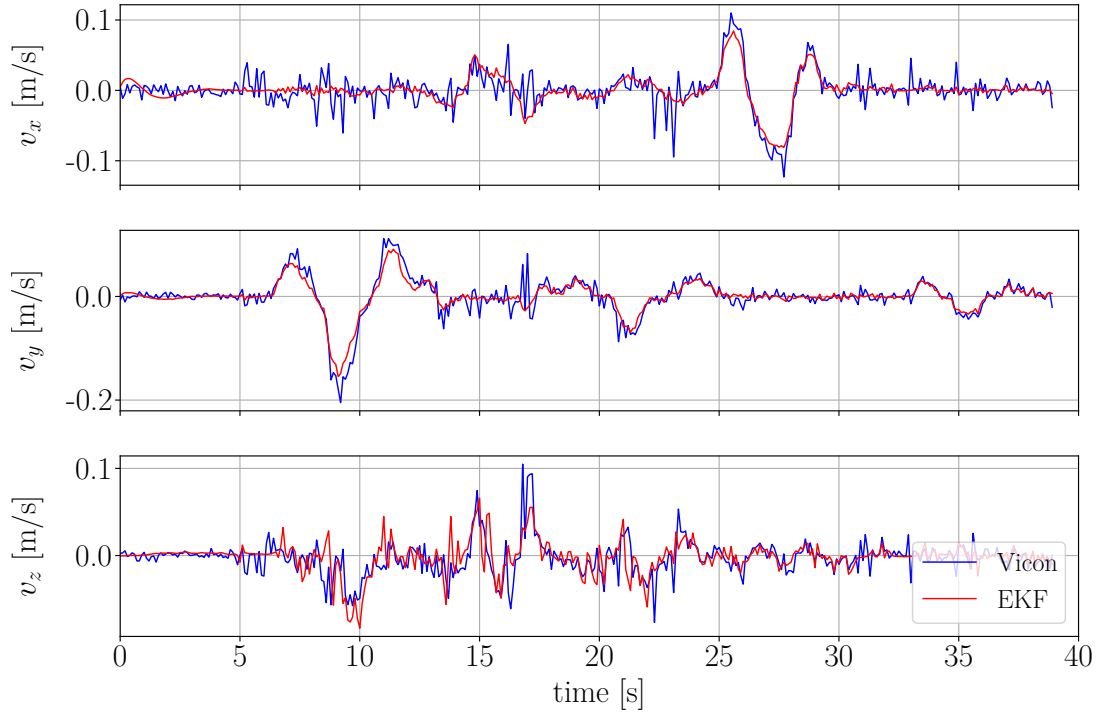


Figure 5.4: Motion #1: Estimation of the base linear velocity.

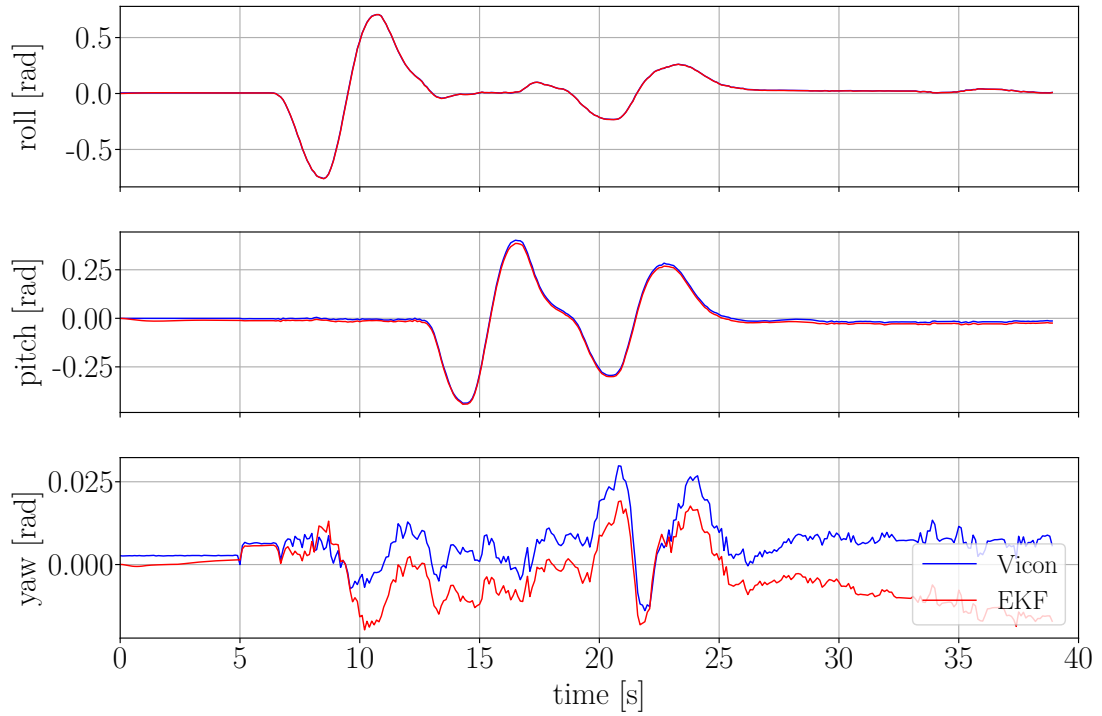


Figure 5.5: Motion #1: Estimation of the base orientation.

### 5.2.2 Trotting Gait

In the second scenario, we consider a more complex motion involving intermittent contact switching, Fig. 5.6. In this test a desired velocity in the x-y direction, by using a joystick, is given to the nonlinear MPC controller as an input command; and we move the robot a few meters around for one minute.

Although we have discontinuities caused by the contact switching in this motion, the magnitude of these discontinuities for the estimated linear velocity Fig. 5.8, roll and pitch angles Fig. 5.9, of the base are comparable to the Vicon measurements, indicating the accurate estimation of the observable states with small RMSE values as shown in table 5.2.



Figure 5.6: Snapshots of the second motion scenario (trotting).

The drift in the position Fig. 5.7, and yaw angle Fig. 5.9, are explainable by the unobservability of these states. Notable sources for the drift are the inaccurate leg kinematics, the fault-prone contact detection and the noise introduced in the estimation framework at each contact switching.

Contact detection is performed, based on Eq. (3.44), by projecting the joint torques of each leg through the jacobian into the space of end effector forces, and applying a force threshold to determine whether each foot is in contact. The estimated end effector force for the front-left (FL) foot along with the detected contact for this foot are shown in Fig. 5.10. In order to avoid outliers in the estimated contact, we use higher force thresholds (red and green lines in the graph) for contact detection in trotting motion, which starts after 5 sec, comparing to the case when the robot is stationary (from 0 to 5 sec). In the contact estimation the value 1 indicates that the the foot is in contact with the ground.

Table 5.2: Motion #2: RMSE values.

	Linear velocity (m/s)			Rotation (rad)	
	$V_x$	$V_y$	$V_z$	Roll	Pitch
RMSE	0.0456	0.0266	0.0557	0.0067	0.0071

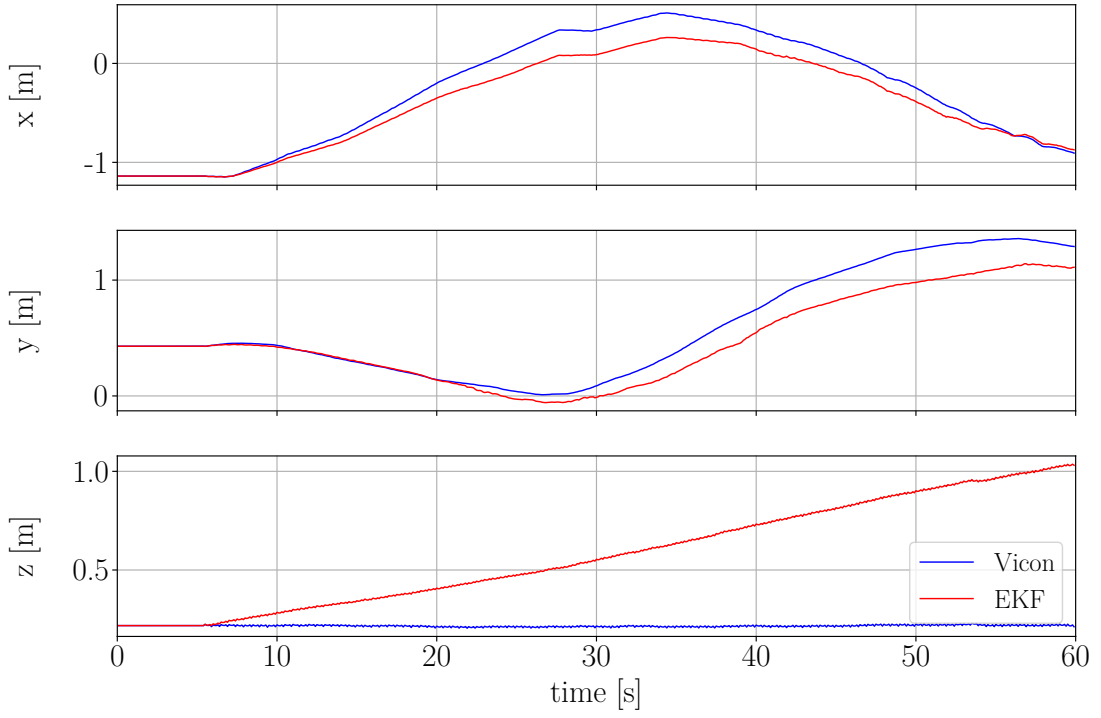


Figure 5.7: Motion #2: Estimation of the base position.

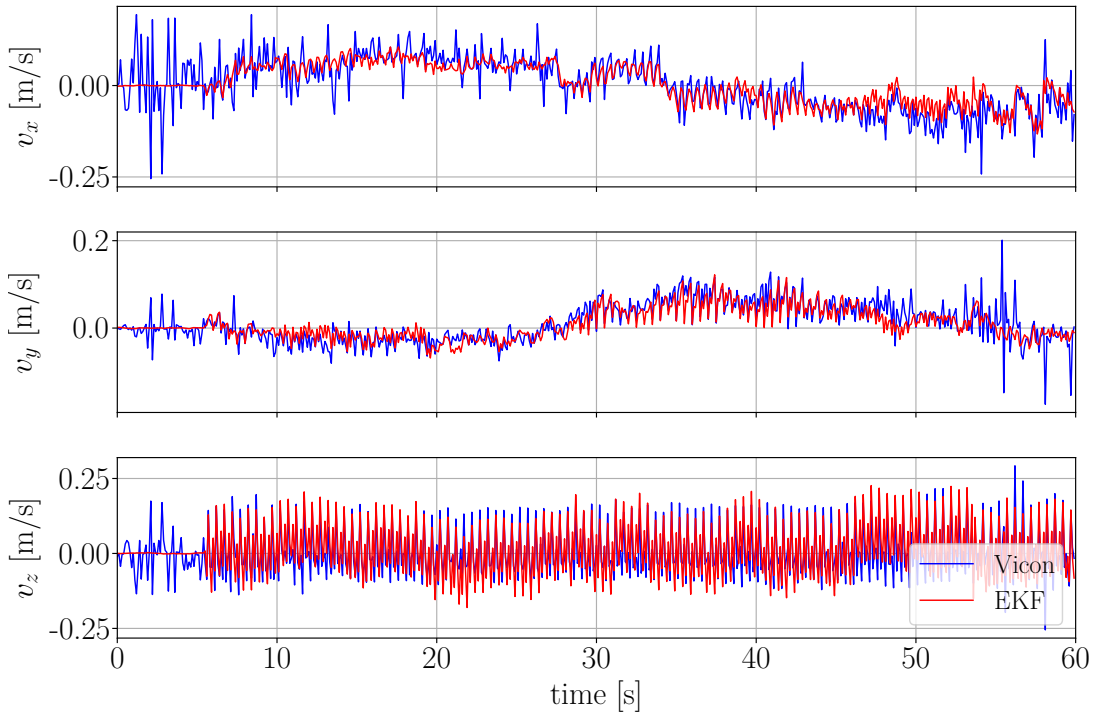


Figure 5.8: Motion #2: Estimation of the base linear velocity.

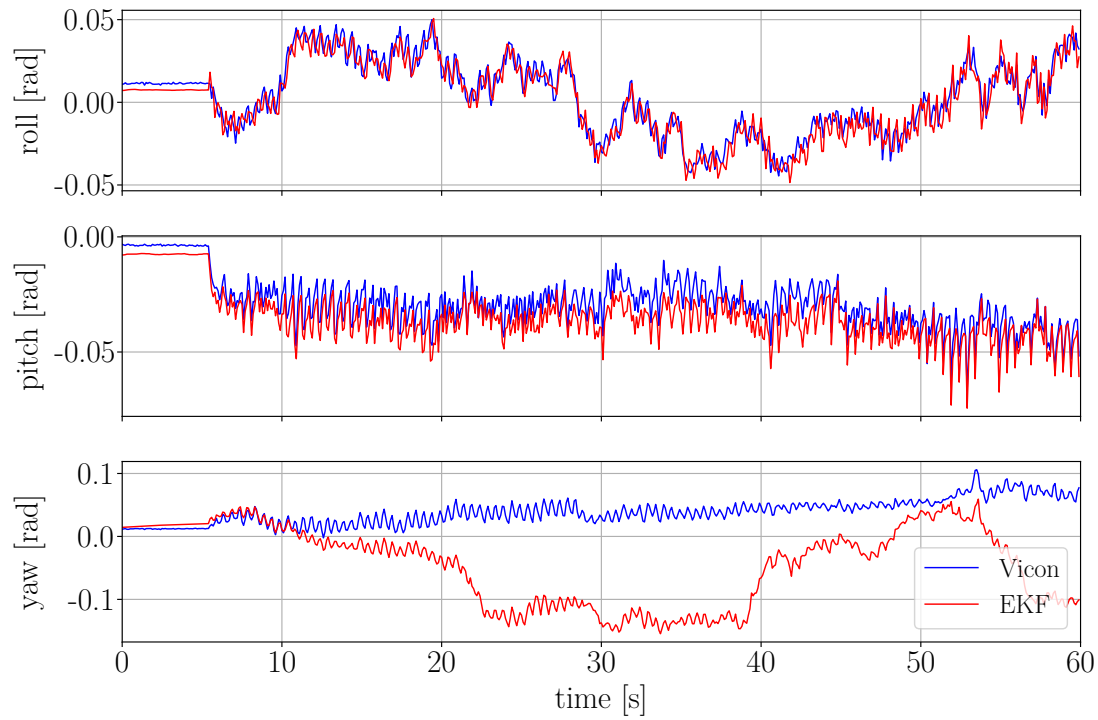


Figure 5.9: Motion #2: Estimation of the base orientation.

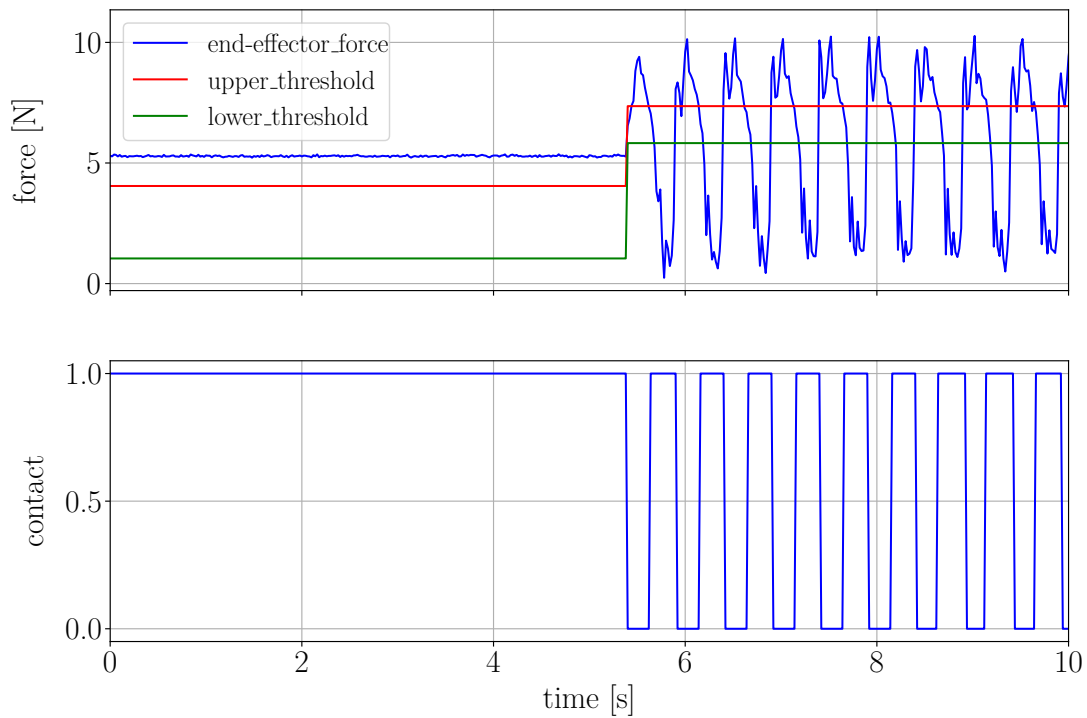


Figure 5.10: Motion #2: Force and contact estimation for front left end effector.

### 5.2.3 Jumping Motion

The third motion we consider, is a jump with commanded velocity to move the robot in x-y direction, Fig. 5.11. This motion is especially challenging, as there is considerable impact during landing that could be problematic if a direct force/torque sensor at the end-effector were used for contact detection. However, since we are instead using torque measurements in the proposed estimator, we benefit from the fact that Solo12's structural and drive system damping filters out the effect of the impact from measured torque considerably. As demonstrated by the results for observable states, shown in Fig. 5.13 and Fig. 5.14, our proposed estimator is performing accurate with small RMSE values in table 5.3, and is not affected by the impact during landing. The drift in the position Fig. 5.12, is more affected by the noise introduced to the estimation due to impact.

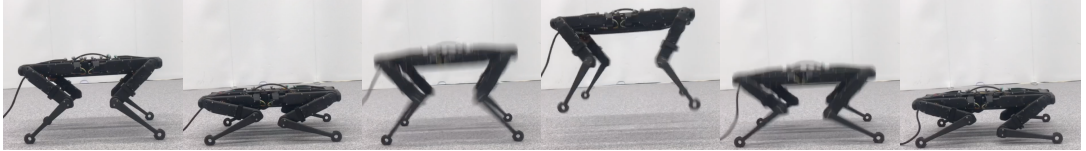


Figure 5.11: Snapshots of the third motion scenario (jump).

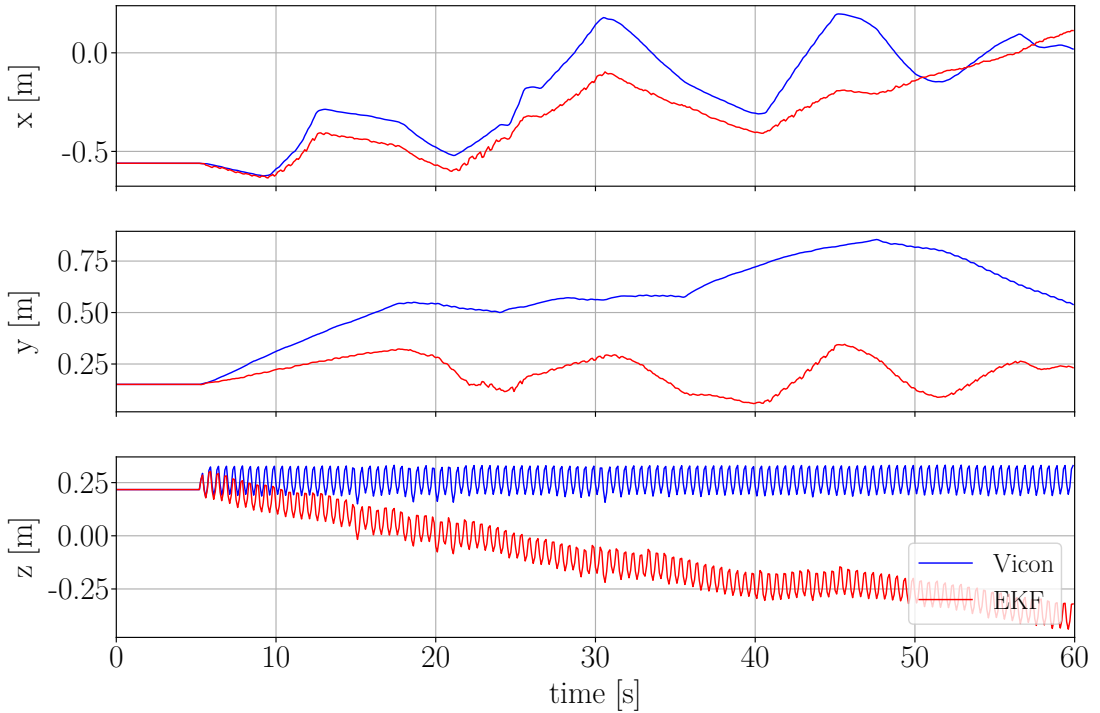


Figure 5.12: Motion #3: Estimation of the base position.

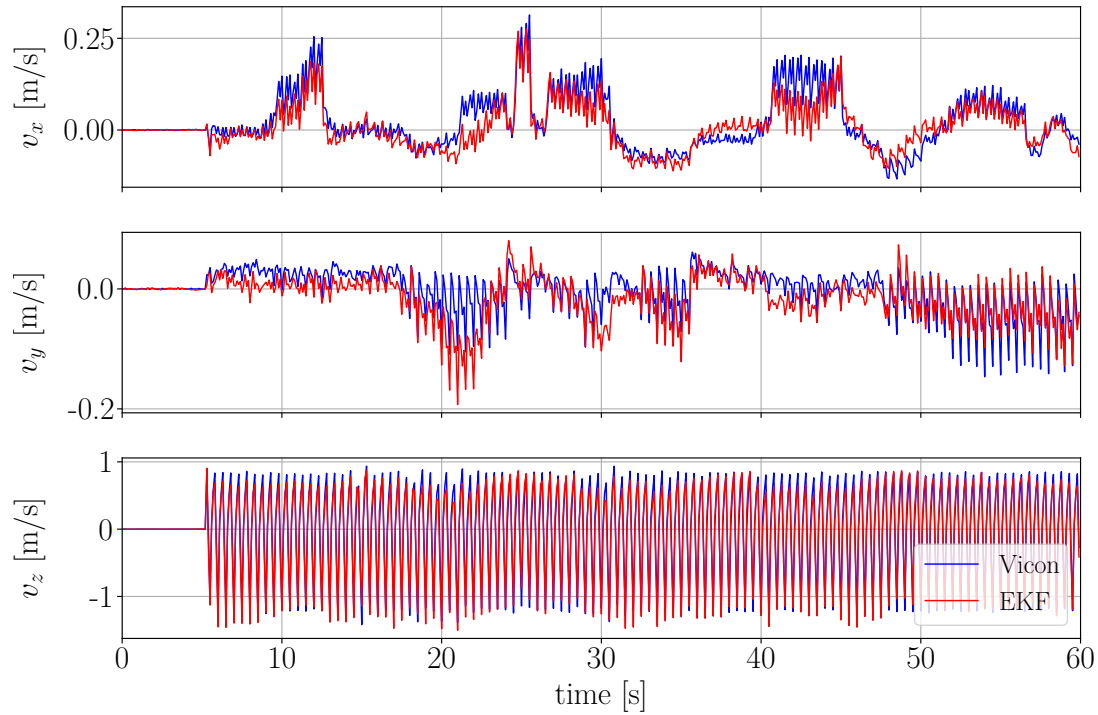


Figure 5.13: Motion #3: Estimation of the base linear velocity.

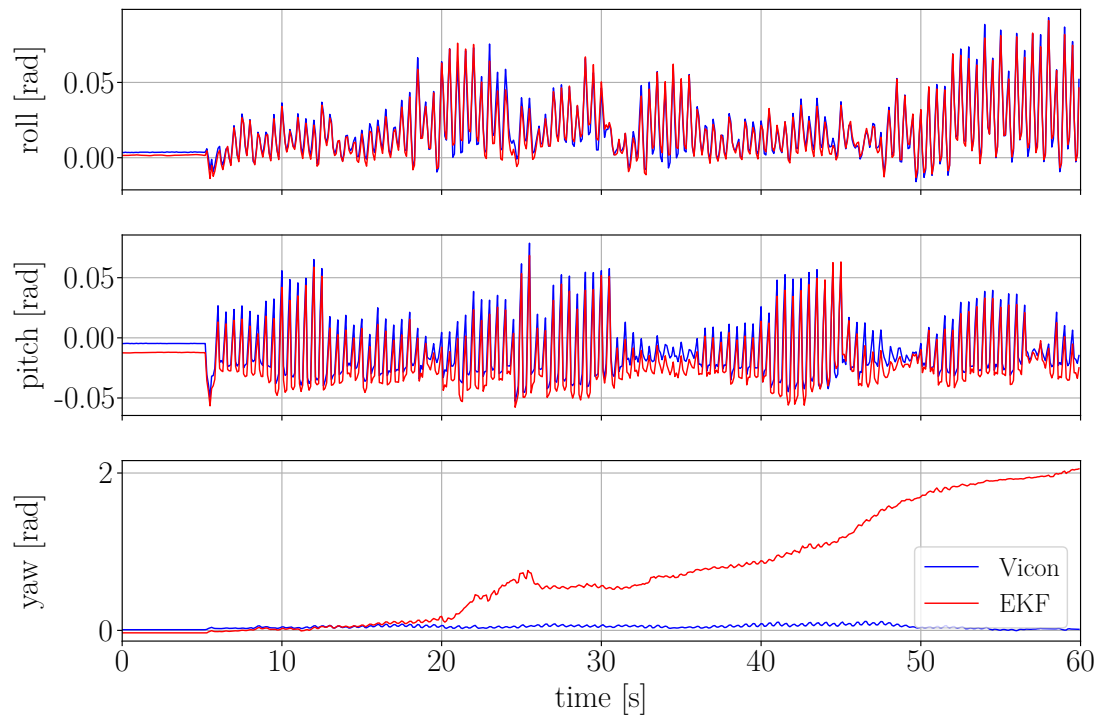


Figure 5.14: Motion #3: Estimation of the base orientation.

Table 5.3: Motion #3: RMSE values.

	Linear velocity (m/s)			Rotation (rad)	
	$V_x$	$V_y$	$V_z$	Roll	Pitch
RMSE	0.0370	0.0272	0.1125	0.0036	0.0094

### 5.2.4 Trotting Gait with EKF

As mentioned in section 4.2, the unobservability of the base position and yaw angle in the EKF is not affecting the robot's locomotion at low level control. Therefore, we only need relative position of the base with respect to end effectors in order to control the whole body motion planning of the robot. On the other hand, the global position (with respect to a fixed world frame), and the yaw angle of the base are needed for the task of localization, navigation and controlling robot's motion at high-level.

Considering significant drift in the z-direction of the base position estimated by the EKF, we correct this value by a constant. We assume that the robot's base remains flat and at the same height during the motion. Therefore, we substitute the value of the base position in z-direction estimated by the EKF by  $p_z = 0.22$  [cm].

By this modification, we rerun the trotting gait with the observer-based controller, and as expected the low-level control is performing same as before handling the task of locomotion perfectly. The results for the base position Fig. 5.15, linear velocity Fig. 5.16, and orientation Fig. 5.17 are consistent with the results we had in trotting motion with Vicon cameras.

Table 5.4: Motion #4: RMSE values.

	Linear velocity (m/s)			Rotation (rad)	
	$V_x$	$V_y$	$V_z$	Roll	Pitch
RMSE	0.0264	0.0230	0.0439	0.0071	0.0094

## 5.3 Discussion

Based on the experiments performed in section 5.2, we can summarize the results as follow:

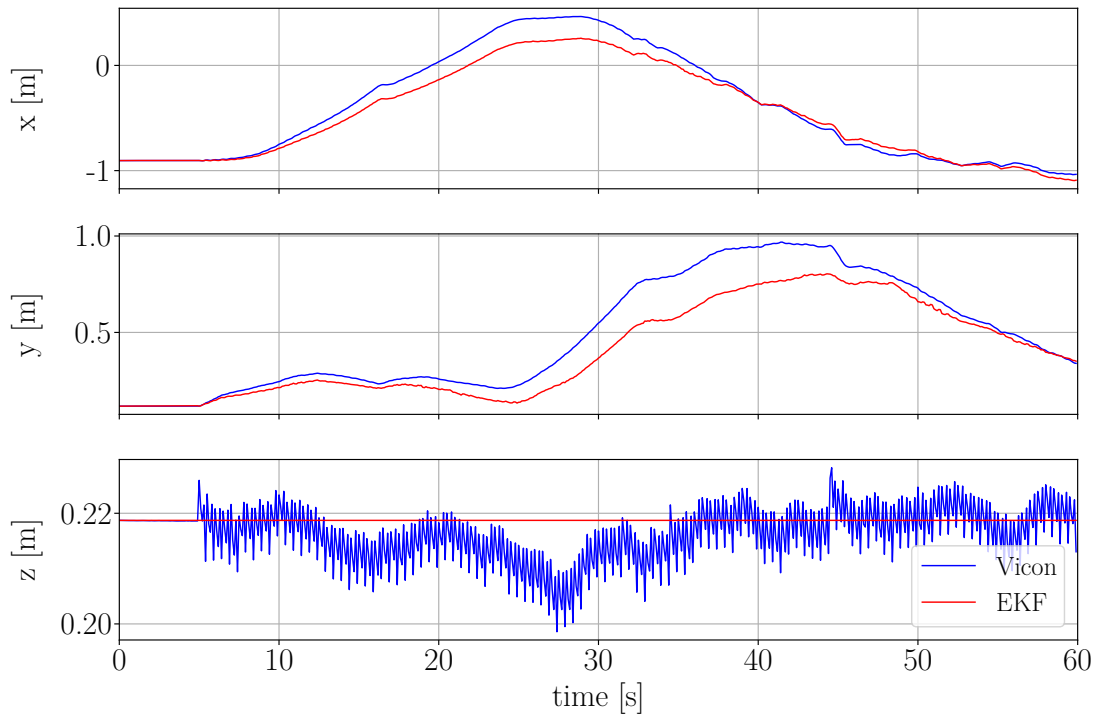


Figure 5.15: Motion #4: Estimation of the base position.

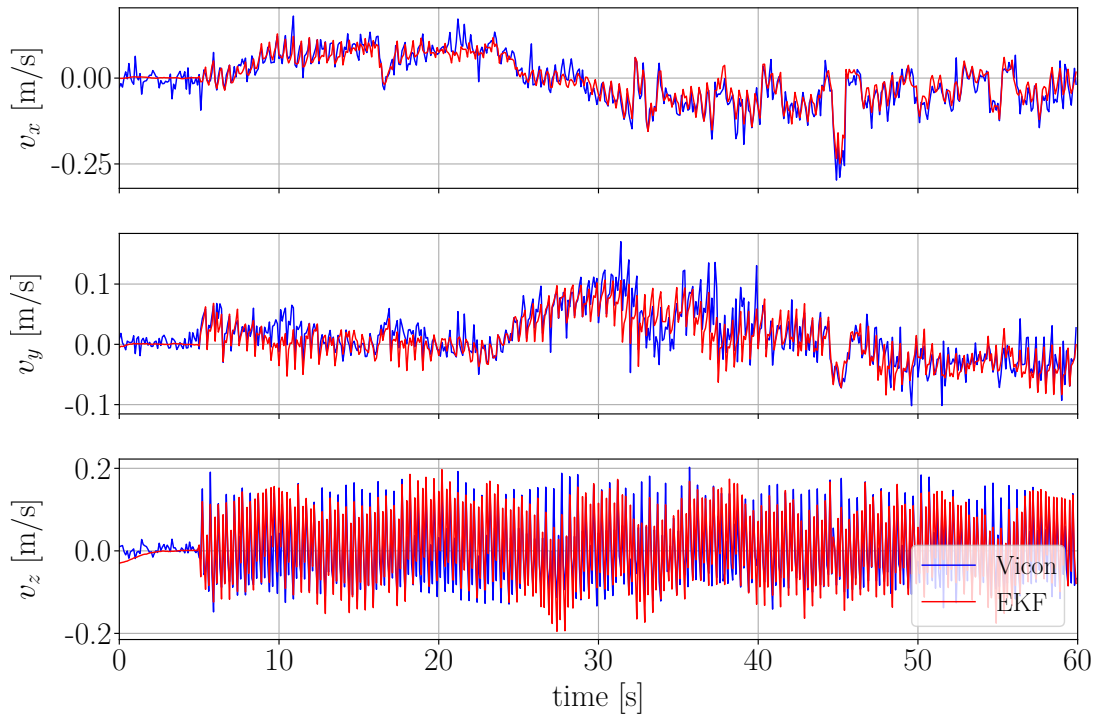


Figure 5.16: Motion #4: Estimation of the base linear velocity.

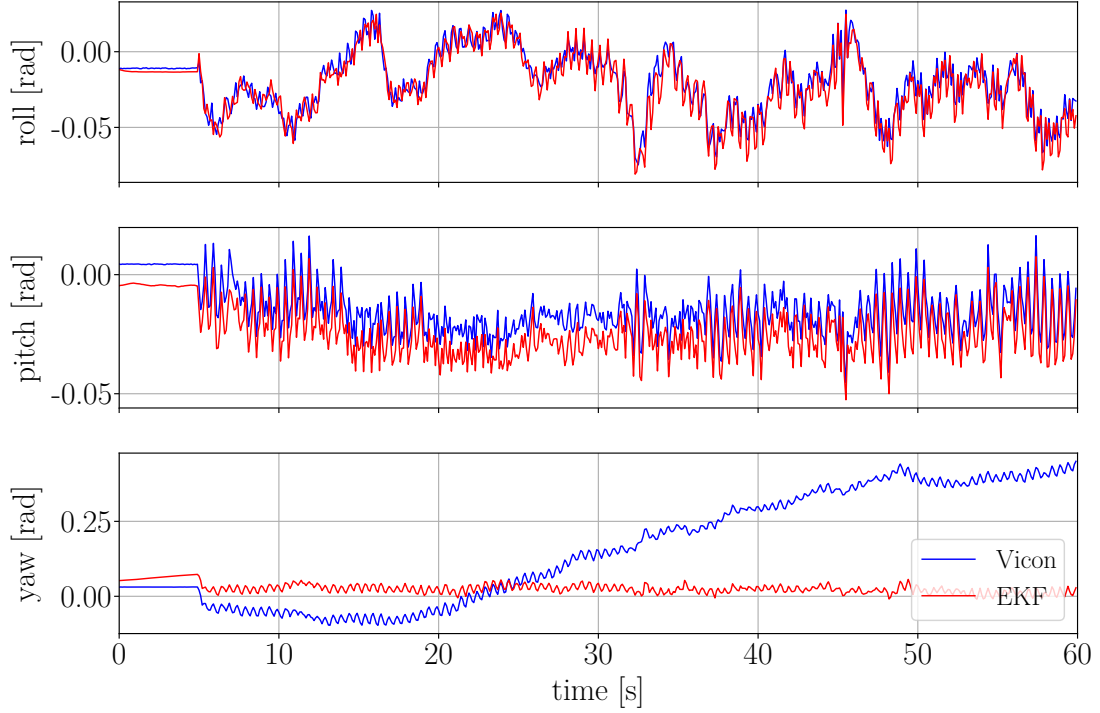


Figure 5.17: Motion #4: Estimation of the base orientation.

- In all motions, the EKF estimation for the observable states (linear velocity, roll and pitch angles of the base), is accurate enough for the task of local stabilization of force controlled legged robot, and this can be verified by the small RMSE values.
- The unobservable states (base position and yaw angle) estimated by the EKF are drifting over the time, and this divergence is noticeable for more dynamic motion. This can be explained by the fact that in highly dynamic motions the intermittent contact switching of the legs introduces noise to the estimation framework, which affects the performance of the EKF.
- Since the global position and yaw angle of the base are only important for the task of navigation and localization, with some modification the EKF can be successfully used in an observer-based controller in order to handle the motion planning of the robot in trotting gait.



# Chapter 6

## Conclusion

### 6.1 Summary

In this research work, we proposed a state estimation framework based on Extended Kalman Filtering to fuse inertial measurement unit (IMU) data with leg odometry (kinematics of a robot's leg when it is in contact with the ground), in order to estimate the position, orientation, and linear velocity of a free floating base with 6 DoF.

First, we investigated the crucial need for such a framework for legged robots locomotion, and the main reason was the inherent instability of legged robots due to intermittent contact with the ground. In order to have feedback controllers capable of planning robust and efficient motions, the knowledge of robots state at each control step is needed. Some of these states are measured directly with onboard sensors on the robot such as encoders and IMU, but the robot's base placement and linear velocity need external sensing device, or as an alternative method we can utilize efficient sensory data fusion in order to obtain the base states computationally, and completely based on proprioceptive sensing.

Then, we examined the related works on state estimation and elaborated on why the proposed estimation framework was suitable for our research project. In the next step, the mathematical formulation of the state estimator was derived, and the results were implemented on the real hardware.

Finally, we evaluated the performance of the resulting estimator for different motions on the quadruped robot Solo12. The results of these experiments showed that the estimated observable states from our approach are accurate and perfectly matching the ground truth data. Furthermore, we showed that our estimator can be used in a feedback scheme in an observer-based controller with robust behavior

for trotting motion, which eliminated the need for any external sensing in order to control the robot locomotion.

## 6.2 Future Work

As shown in section 4.2, the base position and yaw angle are unobservable with the proposed estimation framework. In the future work, we are interested in mounting a stereo camera on the robot, in order to compensate the drift for unobservable states in the EKF. The visual information from the camera will be integrated in the EKF as another measurement model. The advantage of this estimator is that the task of local stabilization and global navigation is done simultaneously.

All of the above works were based on the assumption of a stationary point of contact. This assumption is violated every time there are slippages or deformations of the leg and/or the ground. Contact detection methods can help to reject sporadic slippage or deformation events. However, when these occur regularly, they need to be modeled. As an alternative, learning-based methods can be applied to contact classification problem, in order to make the contact estimation for each leg more robust and accurate.

# Bibliography

- [BloeschEtAl13a] Bloesch, M.; et al.: State estimation for legged robots: Consistent fusion of leg kinematics and imu. *Book of Robotics: Science and Systems VIII*, pp. 17–24, 2013.
- [BloeschEtAl13b] Bloesch, M.; et al.: State estimation for legged robots on unstable and slippery terrain. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 6058–6064, 2013.
- [CamurriEtAl17] Camurri, M.; et al.: Probabilistic contact estimation and impact detection for state estimation of quadruped robots. *Journal of IEEE Robotics and Automation Letters*, Vol. 2, No. 2, pp. 1023–1030, 2017.
- [CamurriEtAl20] Camurri, M.; Ramezani, M.; Nobili, S.; Fallon, M.: Pronto: A multi-sensor state estimator for legged robots in real-world scenarios. *Frontiers in Robotics and AI*, Vol. 7, 2020.
- [CarpentierEtAl21] Carpentier, J.; Valenza, F.; Mansard, N.; et al.: Pinocchio: fast forward and inverse dynamics for poly-articulated systems. <https://stack-of-tasks.github.io/pinocchio>, 2015–2021.
- [ChittaEtAl07] Chitta, S.; Vemaza, P.; Geykhman, R.; Lee, D.D.: Proprioceptive localization for a quadrupedal robot on known terrain. *IEEE International Conference on Robotics and Automation*, pp. 4582–4587, 2007.
- [CoumansBai20] Coumans, E.; Bai, Y.: Pybullet, a python module for physics simulation for games, robotics and machine learning. <http://pybullet.org>, 2016–2020.
- [FallónEtAl14] Fallón, M.F.; Antone, M.; Roy, N.; Teller, S.: Drift-free humanoid state estimation fusing kinematic, inertial and lidar sensing. *IEEE-RAS International Conference on Humanoid Robots*, pp. 112–119, 2014.
- [FankhauserHutten18] Fankhauser, P.; Hutten, M.: AnyMal: A unique quadruped robot conquering harsh environments. *Research Features*, Vol. 126, p. 54–57, 2018.

- [Featherstone08] Featherstone, R.: Rigid Body Dynamics Algorithms. Springer, 2008.
- [FourmyEtAl21] Fourmy, M.; Flayols, T.; Léziart, P.A.; Mansard, N.; Solà, J.: Contact forces preintegration for estimation in legged robotics using factor graphs. IEEE International Conference on Robotics and Automation (ICRA), pp. 1372–1378, 2021.
- [Grimminger21] Grimminger, F.: Open robot actuator hardware - quadruped Solo12 with 12DoF. [https://github.com/open-dynamic-robot-initiative/open\\_robot\\_actuator\\_hardware](https://github.com/open-dynamic-robot-initiative/open_robot_actuator_hardware), 2019–2021.
- [GrimmingerEtAl20] Grimminger, F.; et al.: An open torque-controlled modular robot architecture for legged locomotion research. IEEE Robotics and Automation Letters, Vol. 5, No. 2, pp. 3650–3657, 2020.
- [HartleyEtAl20] Hartley, R.; Ghaffari, M.; Eustice, R.M.; Grizzle, J.W.: Contact-aided invariant extended kalman filtering for robot state estimation. The International Journal of Robotics Research, Vol. 39, No. 4, pp. 402–430, 2020.
- [Machines in Motion Laboratory] Machines in motion laboratory. <https://github.com/machines-in-motion>.
- [Kalman60] Kalman, R.E.: A new approach to linear filtering and prediction problems. Journal of basic Engineering, Vol. 82, No. 1, pp. 35–45, 1960.
- [LinKomsuogluKoditschek05] Lin, P.; Komsuoglu, H.; Koditschek, D.: A leg configuration measurement system for full-body pose estimates in a hexapod robot. Journal of IEEE Transactions on Robotics, Vol. 21, pp. 411–422, 2005.
- [LynchPark17] Lynch, K.M.; Park, F.C.: Modern Robotics Mechanics Planning, and Control. Cambridge University Press, 2017.
- [MeduriEtAl22] Meduri, A.; et al.: BiConMP: A nonlinear model predictive control framework for whole body motion planning. arXiv:2201.07601v1, 2022.
- [RostonKrotkov92] Roston, G.P.; Krotkov, E.P.: Dead reckoning navigation for walking robots. IEEE/RSJ International Conference on Intelligent Robots and Systems, 1992.
- [RotellaEtAl14] Rotella, N.; Bloesch, M.; Righetti, L.; Schaal, S.: State estimation for a humanoid robot. IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 952–958, 2014.

- 
- [SchneiderSchmucker06] Schneider, A.; Schmucker, U.: Force Sensing for Multi-Legged Walking Robots: Theory and Experiments Part 1: Overview and Force Sensing. London: IntechOpen, 2006.
- [Sola17] Sola, J.: Quaternion kinematics for the error-state kalman filter, 2017.
- [Spot] Boston dynamics spot. <https://www.bostondynamics.com/products/spot>.
- [Van Loan78] Van Loan, C.: Computing integrals involving the matrix exponential. IEEE Transactions on Automatic Control, Vol. 23, No. 3, pp. 395–404, 1978.



# Appendix

## A.1 Contents Archive

There is a folder **MSC\_054\_Khorshidi/** in the archive. The main folder contains the entries

- **MSC\_054\_Khorshidi.pdf**: the pdf-file of the thesis MSC-054.
- **Data/**: a folder with all the relevant data, programs, scripts and simulation environments.
- **Latex/**: a folder with the \*.tex documents of the thesis MSC-054 written in Latex and all figures (also in \*.svg data format if available).
- **Presentation/**: a folder with the relevant data for the presentation including the presentation itself, figures and videos.



## **Erklärung**

Ich, Shahram Khorshidi (M.Sc. Mechatronics an der Technischen Universität Hamburg, Matrikelnummer 54893), versichere, dass ich die vorliegende Masterarbeit selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel verwendet habe. Die Arbeit wurde in dieser oder ähnlicher Form noch keiner Prüfungskommission vorgelegt.

---

Unterschrift

---

Datum