Politecnico di Torino

Department of Control and Computer Engineering
Master of Science Degree in Mechatronic Engineering

MASTER THESIS

Kalman filter for contact detection and localization during shin collisions in a dynamic legged robot

Candidato:
Letizia Ariaudo

Thesis advisor:
Prof. Marcello Chiaberge

Research supervisor:
Dr. Claudio Semini
Dr. Geoff Fink
Dr. Victor Barasuol

Academic Year 2020-2021

Ai miei genitori, ad Aringa, ad Ilaria e alla mia bellissima famiglia.

Abstract

Nowadays, dynamic legged robots are one of the major fields of research in robotics thanks to their capability of movement in various and challenging scenarios. They are used in outdoor environments to help human and collaborate with them. They perform tasks in situations that are critical and even hazardous for human safety. The demanding work for researchers is to develop techniques to maintain a robust motion in difficult terrains that allow the robot to accomplish the desired tasks.
The robot's motion is performed using innovative methods based in trajectory planning, optimization, control and state estimation. Although several studies in these fields and optimal results achieved, shin collision can occur during the locomotion, causing the robot to get stuck and preventing, as a consequence, the achievement of the predetermined goals. One solution is to evaluate and estimate the contact point along the shin when the leg collides an object,in order to use this information as a feedback to stabilize the trunk controller and help the robot to overcome the obstacles.
This thesis extends the work about contact detection and localization during shin collisions, developed in a 2D approximation. The approach is based on a novel model performed in a 3D environment, considering both velocity and acceleration to detect in a more accurate way the contact point. The estimation of this point is performed using the Kalman Filter, one of the most common technique of filtering, in order to avoid noise corruption due to the usage of sensors and to determine the optimal output, following the trial and error procedure based mainly on the tuning of the parameters.
The results are validated both in simulation and on the Hydraulically actuated Quadruped robot (HyQ), including experiments with a pallet. This thesis is the outcome of a one year project performed at the Dynamic Legged Systems Lab (DLS) at IIT in Genova.

Contents

List of Figures

List of Tables

Acronyms

DLS	Dynamic Legged System
IIT	Istituto Italiano di Tecnologia
HyQ	Hydraulically actuated Quadruped robot
PDF	Probability Density Function
RV	Random Variable
FDI	Fault Detection and Isolation
CPF	Contact Particle Filter
DH	Denavit Hartenberg Convention
ISA	Integrated Smart Actuators
IMU	Inertial Measurement Unit
ROS	Robot Operating System
HAA	Hip Abduction-Adduction
HFE	Hip Flexion-Extension
KFE	Knee Flexion-Extension
EKF	Extended Kalman Filter
RCF	Reactive Controller Framework
VO	Visual Odometry

Chapter 1

Introduction

The first works about legged robots appeared in the late '60s with the contribution of Robert McGhee who started the study of quadruped and hexapod robots at the University of South California [50]. A significant change occured in the beginning of '80s due to the transition from quasi-static gaits to dynamic locomotion for all legged robots. Since that event, legged robots became one of the major field of interest outperforming wheeled and tracked systems in specific environments as uneven,unstructured and irregular terrains. More precisely, quadruped legged robots present high mobility in case of rough, damaged and hostile area wherefore they are mainly used to explore and work in these types of scenarios. For their characteristics they can substitute humans in rescue and discovery operations,providing informations about disaster and dangerous zones, thanks to the usage of sensors like LIDAR or cameras.
The difficulty in the motion of autonomous legged robot is to always maintain stability even in such difficult terrains and to keep going on in order to satisfy the desired tasks. To accomplish this, it is important to establish a close relationship between locomotion and perception. Therefore the robot has to be able to analyze the whole area and then elaborate the signals coming from itself and from the environment. For this purpose, the detection and localization of contact points, due to the presence of obstacles, can be very useful to improve the robustness of robot's motion.

1.1 Aim

The objective of this thesis is to propose an improvement on the detection and localization of shin collisions during the locomotion [10]. This is the result of a project fulfilled at the Dynamic Legged System (DLS) at Istituto Italiano di Tecnologia (IIT), in Genova.
The thesis is based on three important aspects:

- a novel model,based on the kinematics, for finding the collision during the motion of the robot. Starting from the work performed in 2D, this thesis present an extension in a 3D model.
- The usage of velocity-based model and acceleration-based model to be more precise and accurate in the evaluation and detection of the contact point along the shin. In this way, it is possible to switch from one model to the other, providing a better estimation of the contact point.

- a variant approach to estimate the shin collision with the contribution of the Kalman filter, in order to reduce and remove the noise given by the usage of sensors. This estimation is then used as a feedback for stabilizing the trunk controller of the robot.

As said before the starting point of this thesis is the model in 2D performed [10] with the purpose of localize a single contact point when the robot enters in collision with obstacle during the locomotion. Extending the model in a 3D environment has the advantage of using velocity and acceleration to localize the point in a more accurate way, building as a consequence two systems, the first based on the velocity and the second based on the acceleration with the purpose of choosing the best one to obtain a better perfomance.
Since this work is about the *estimation* of the contact point, the last part is related to the filtering of the signal, developed with the contribution of the Kalman Filter,one of the most common technique used for estimating a state.
Provide a correct estimation of the contact point is useful to help the robot to overcome a step or an obstacle, avoiding the slippage of the shin along the surface and improving as a consequence its motion in a difficult environment.

1.2 Methodologies

The obtained model was initially analyzed and run through simulations, using the 3D visualizer RVIZ and the software simulator Gazebo. To check if the results were consistent with the reality, the Signal Scope tool was used to study the real-time behaviour of the robot. Then all the datas collected from the simulations were developed on Matlab in order to improve the phase of filtering, providing, as a consequence, a desired output thanks to a manually tuned procedure. At the end of simulation-based validation, the experiments were performed using Hyq green robot, following the sanity check and guaranteeing the safety for all the operators.
Doing experiments the robot was able to detect the collision in Real-Time, providing the localization of the contact point along the shin.

1.3 Outline

The thesis is organized in the following fashion:
Chapter 2 introduces the concepts of state estimation and filtering,focusing on probability theory, modelling theory and Bayes filter.
Chapter 3 describes the target platform HyQ used in this dissertation.
Chapter 4 explains how to obtain the model for detecting the shin collision, starting from the kinematics. After finding the model, it will be filtered using the Kalman Filter to have an estimated output for the contact point during the locomotion.
Chapter 5 shows the obtained results both in simulation and on the robot.
Chapter 6 is a review of the concepts investigated in the previous chapters and it presents a section dedicated to the future works.

1.4 State of the Art

In the last 30 years, dynamic legged robots are mainly used instead of wheeled robots with the purpose of going through unstructured,uneven and rough terrains. Considering these types of environment and situation,it is very likely that shin collisions occur during the locomotion. A solution to avoid fall, slippage or the getting stuck of the robot is to detect and estimate the contact point along the shin, in order to use it as a feedback to stabilize the trunk controller.

This thesis has the aim to investigate and analyze the contact points along the shin of a quadruped legged robot during its motion, due to the presence of obstacles or undesired and unexpected objects that collide with it. First of all it is important to explain how the motion of the robot happens, its working principle and its limitations, for then understanding why it is necessary to detect the collisions as fast as possible, analyzing the advantages in terms of robot performance.

The robot locomotion is perfomed using innovative methods based on Trajectory Planning Optimization [39] for improving its motion, Control algorithms and foothold adaptation for guaranteeing robustness [34] and State Estimation [17] for detecting its pose in the environment. The motion of a quadruped legged robot is based on two main phases: i) the stance phase in which the foot is in contact with the terrain, ii) the swing phase in which the foot doesn't touch the ground and as a consequence no force is applied. Analyzing the locomotion of HyQ, it is able to perform different types of movements,even in difficult scenarios: trotting [9], crawling [59],climbing on two surfaces [26], bounding [37] and walking on soft terrains [24]. The combination between control algorithms and perceptions allows the robot to perform advanced locomotion skills. An example is the study of Focchi et al. [25] in which, on the basis of the forces evaluated at the feet, the trajectory is then generated accordingly in order to adjust the step and get over the obstacles. The locomotion of HyQ robot over challenging terrains was also analyzed in the work of Winkler et al. [59]. For their work, based on the force foothold adaptation, a map of the environment is pre-acquired with a depth sensor and then collected in an OctoMap data structure. After finding the map, the feasible foothold is chosen appropriately, optimizing the whole body trajectory, in order to perform the motion of the robot in challenging scenarios. To improve the robot motion in difficult terrains and increase its robustness, Bratta et al. [16] proposed a model based on trajectory optimization. Adding two constraints in the Single Rigid Body Dynamics-based trajectory optimizer, the robot is able to avoid leg collisions with obstacles.
All these works don't analyze the case in which a possible collision happens, due to the presence of an unexpected or undetected object. They are mainly based on the study of the optimal trajectory,followed by the robot, in order to avoid obstacles. However, during a locomotion, it is likely that the legs of the robot collide with barriers, rocks or steps especially in rough terrains. Moreover it is important to detect the collision, only using proprioceptive sensors, because: i) it simplifies the model increasing the efficency, ii) it considers the case in which the robot is blind, due to, for example, the presence of interference or malfunction in exteroceptive sensors. The detection of the collision can also be used as a feedback for improving the motion of the robot, allowing it to overcome the step. These are the reasons why the field of contact detection and localization is important for a better locomotion of the robot.

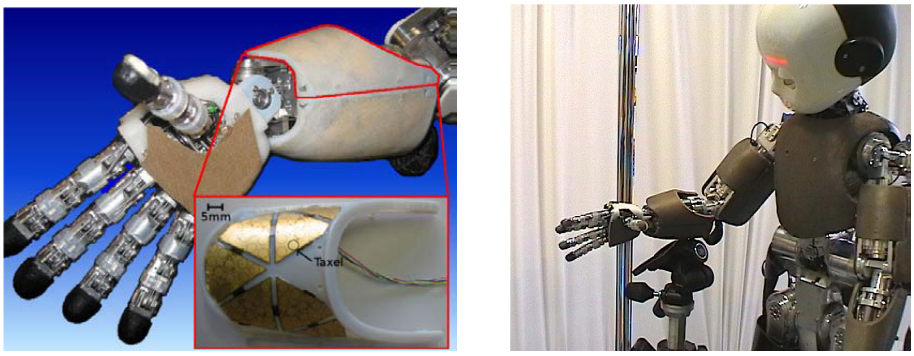
Several studies were performed about detection and estimation of contact points, using different methods. Before entering in the details, it is necessary to briefly describe the sensors used for the estimation. They can be of two types: *interoceptive*, sensors that are related to the internal state of the robot, such as accelerometer or gyroscope and *exteroceptive* like cameras

or time-of-flight transmitter/receiver that provide informations about the external environment. Interoceptive sensors are used to estimate velocity and accelerations, while exteroceptive sensors are mainly used to properly estimate the pose of the robot; the fusion of the two provides the best description and estimation of the state of the robot.

It is important to take into account that in the field of contact detection, in which it is necessary to detect the collision as early as possible, specific approaches are adopted to promptly act and obtain the estimation in the least possible time, in order to control the robot and take countermeasures to stabilize it.
The different methods performed to detect and localize the contact point,when the robot enters in collision with an obstacle, are analyzed in the following sections.

1.4.1 Force Torque Sensors and Tactile Sensor Skin

A F/T sensor is an electronic device used for monitoring, detecting, recording and regulating linear and rotational forces. As a contact sensor, it is specifically designed to interact with physical objects in its environment. This type of sensor,with the contribution of a distributed skin [36], can be used to detect the contact point, analyzing external forces that are applied in a specific area [54]. The tactile skin is composed by tiny pressure-sensitive elements, in communication each other through serial bus. They are able to detect the contact either is a light touch or a total-body weight,this is why they are mainly used to cover the surface of humanoid robot, involved for precision tasks in human-robot interaction. To measure the contact localization and



(a) mounted skin on the right Arm of iCub.

(b) iCub with mounted skin on torso, arm, fingertips.

Figure 1.1: iCub humanoid robot [22]- Italian Institute of Technology.

the contact forces, it is necessary to have two elements: i) external forces sensor set on the contact area, ii) geometric model of environment and robot.

One of the study about the combination of F/T sensors and tactile skin, is analyzed in the work of A. Del Prete et al. [22]. The accurate detection and localization of the contact point using these types of sensors is a demanding task. In fact, in order to do this exhaustively, it is important to take into account two considerations: i) the measured contact forces are corrupted by noise, and as a consequence, they are affected by errors, ii) the tracking of the contact forces may be not so accurate. A. Del Prete et al. demonstrate how the errors, during the estimation

of the contact point in the iCub robot, may affect also the contact forces. Moreover, in their work, it is also shown that adding a feedback term, corrupted by noise, is not necessarily useful to reduce the errors given by the estimation.

The usage of this type of sensors, for contact detection and localization, requires complexity of the platform, high costs and limited material choices, this is why most of the robots don't use this approach. Moreover, since they are mainly used for tasks concernig human-robot interaction, they are designed for handling small contact forces, a situation that doesn't happen during the locomotion of the robot.

1.4.2 Generalized Momenta Approach G/M

The dynamic model of a robot can be represented using two methods:

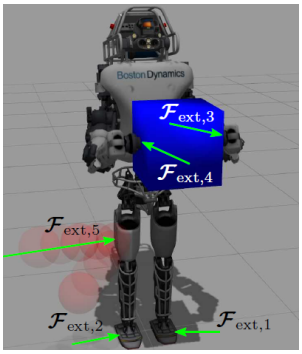
1. *Lagrangian formulation* in which the equations are obtained using *generalized coordinates* and *generalized forces*, in a systematic way not depending on the reference coordinate frame. The *Lagrangian* is:
$$\mathcal{L} = \mathcal{T} - \mathcal{U} \tag{1.1}$$
where $\mathcal{T}$ is the kinetic energy, while $\mathcal{U}$ is the potential energy. The *Lagrange equation* is expressed as :
$$\frac{d}{dt}\Big(\frac{\delta \mathcal{L}}{\delta \dot{q}}\Big)^T - \Big(\frac{\delta \mathcal{L}}{\delta q}\Big)^T = \xi \tag{1.2}$$

2. *Newton-Euler Formulation* in which the equations are obtained using a recursive algorithm, starting from the evaluation of forces and moments applied in the end-effector. The *translational* motion of the centre of mass can be represented as:
$$f_i - f_{i+1} + m_i g_0 = m_i \ddot{p}_{C_i} \tag{1.3}$$
The generalized forces are computed as:
$$\tau_i = \begin{cases} f_i^T z_{i-1} + k_{ri} I_{mi} \dot{\omega}_{mi}^T z_{mi} + F_{vi} \dot{d}_i + F_{si} sgn(\dot{d}_i) & \textit{prismatic joint} \\ \mu_i^T z_{i-1} + k_{ri} I_{mi} \dot{\omega}_{mi}^T z_{mi} + F_{vi} \dot{\theta}_i + F_{si} sgn(\dot{\theta}_i) & \textit{revolute joint} \end{cases} \tag{1.4}$$

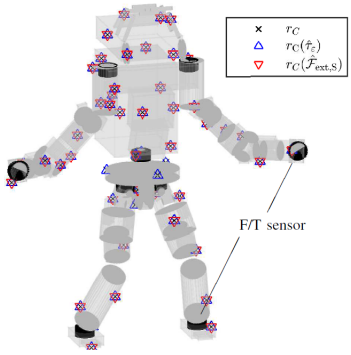
From the equations above, it is possible to express the *generalized momentum* as $p = M(q)\dot{q}$. The idea of this approach is to detect collisions between robot and obstacles without the usage of extra sensors [21]. For this purpose only proprioceptive sensors, as joint encoders, are used to identify *contact isolation*, comparing the computed torque with the real one,in order to deduce the action of external forces.

In the work of A. De Luca et al., the FDI technique [20] is applied to detect the collision at any point along the robot arm, without knowing neither the measurements of acceleration nor the inversion of the robot inertia matrix.The FDI method is based on the detection of *residuals* according to potential faults that can corrupt the system. These residuals are obtained using observers, comparing the filtered output with the measured one. In the end, all the informations about the contact detection and contact force intensity are contained inside the residual vector,useful for then design a hybrid force/motion controller that allows to have the interaction task which follows the detected collision.

The disadvantages of this approach are that:



(a) humanoid robot Atlas performing a manipulation task and entering in collision with an unwanted object on the knee.



(b) Distribution of three types of contact:i)true contact r_C, ii)estimated contact using G/M approach $r_C(\hat{\tau}_\xi)$, iii)estimated contact using F/T approach $r_C(\hat{F}_{ext,S})$.

Figure 1.2: Atlas robot [57] - Boston Dynamics.

1. the detection of the collision is not localized;it is possible to find the contact link instead of the contact point.

2. it is mainly applied to fixed-base robots. As a consequence this method is not useful for shin collisions occuring during the locomotion of the robot.

Vorndamme et al., to find a model also for floating-base robots, in particular humanoids robot, combined the G/M approach with F/T sensor-based technique [57].Their work is implemented on *Atlas* robot and it is based on the detection of the collision point,considering three types of possible contact: i) the feet via ground contact, in which the robot is in touch with its environment, ii) the hands contact during the execution of a specific task, iii) the unwanted contact due to a colliding object in the knee Fig. 1.2.
Despite the precision and accuracy of this proposed model, it was perfomed in simulation with a stationary robot, considering an applied external force as an unexpected collision.

1.4.3 Particle Filters and Proprioceptive Sensors

More complex is the model of Manuelli et al. [35], based on CPF, a contact particle filter in order to detect and localize the contact point.
The particle filter is a different implementation of Bayes filter, in which the posterior *bel(x_t)* is composed by a set of random state samples called *particles* [53], defined as:

$$\mathcal{X} := x_t^{[1]}, x_t^{[2]}, ..., x_t^{[M]} \tag{1.5}$$

In comparison with Gaussian, this representation is approximate and non-parametric, providing more freedom in the choice of the posterior distribution shape. At each particle is assigned a specific weight which corresponds to the probability of that particle will be sampled by the probability density function. The algorithm followed by the particle filter is:

Table 1.1: Recursive Algorithm of Particle Filter.

1: $\bar{\mathcal{X}}_t = \mathcal{X}_t = 0$
2: **for** m = 1 **to** m= M **do**
3: sample $x_t^{[m]} \sim p(x_t|u_t, x_{t-1}^{[m]})$
4: $w_t^{[m]} = p(z_t|x_t^{[m]})$
5: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + (x_t^{[m]}, w_t^{[m]})$
6: **end for**
7: **for** m = 1 **to** m= M **do**
8: draw i with probability $\propto w_t^{[m]}$
9: add $x_t^{[i]}$ to $\mathcal{X}_t$
10: **end for**
11: return $\mathcal{X}_t$

The starting point of their work is the usage of the generalized momentum observer, in order to provide the estimation of the external joint torque, as in the work of De Luca et al. [21]. In the approach of Manuelli et al. the contact localization problem is considered as a nonlinear optimization problem, furthermore, fixing the contact location r_c, it can be formulated as a convex optimization problem:

$$\begin{aligned} &min(\gamma - J_{rc}(q)^T F_c)^T(\gamma - J_{rc}(q)^T F_c) \\ &\text{subject to } F_c \in \mathcal{F}(r_c). \end{aligned} \tag{1.6}$$

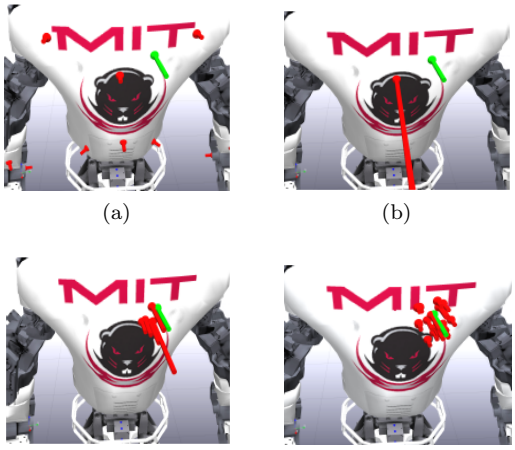
Considering the equation above,it is possible to use the particle filter, in which each particles corresponds to a specific contact location on the surface of the robot. This method can work with the contribution of three elements:

1. *measurement model*, obtained as a residual from the observer.
2. *motion model*, it corresponds to a zero velocity assumption when the link of the robot enters in contact with an obstacle.
3. Contact Particle Filter that is a combination of measurments and motion models. As in [29] it is necessary to specify a threshold for determining when an external contact happens.

Despite the high performance of this model Fig. 1.3, CPF approach presents relevant limitations:

1. Model Error. The main problem related to this technique is that the dynamic model should be very accurate to avoid errors in the measured states and as a consequence in the residuals.
2. Identifiability. If the contact situation is not *identifiable*, there exist multiple sets of contact with their associated contact forces, that produce the same observed residual.As a consequence this model doesn't work properly if there is more than one contact point.
3. Point Contacts. It is not considered the case in which multiple,simultaneous and continuous contacts happen (for example when the robot is sitting).
4. Filter Divergence. The filter can diverge if there is a bad estimation of the contact point.

Moreover this work was only performed in simulation and it was not considered the case in which the contact occurs during the motion of the robot.



(a) (b) (c) (d)

Figure 1.3: In the figure are shown four iterations using CPF approach - Red arrows represent particles while the green arrow is the real contact force [35] - MIT.

1.4.4 Contact localization using velocity constraints

Another approach to localize contacts and collisions is proposed in the recent work of Wang et al. [58], submitted at the 2020 IROS Conference.

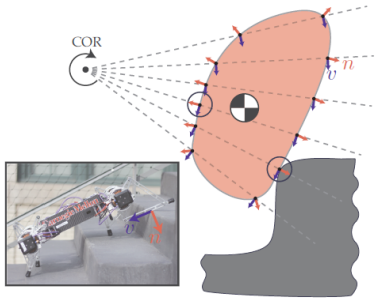


Figure 1.4: scheme of the velocity-based contact localization in the Minitaur quadruped robot [58]

Their method, implemented in 2D, is based on two assumptions: i) the contact velocity that is in direction normal to the surface of the link must be zero, as shown in Fig. 1.4 and ii)the contact velocity, just before the collision, must be positive. Their approach is simple because with only the usage of proprioceptive sensors, it is possible to detect the collision, providing a set of contact points, not only just one point, along the robot link. It can also be implemented easily, in order to obtain a single contact point, adding filtering or additional dynamic constraints such as acceleration. Moreover, they proposed extensions to their model to be more accurate in the evaluation of the contact point: i) **numerical method**, in which it is presented an algorithm to solve numerically the contact point Tab. 1.2, ii) **particle filter** in order to reduce

the uncertainties,considering the set of contact points obtained with the velocity-based method.

Table 1.2: Algorithm for velocity-based model [58].

$\bar{\mathcal{B}} \leftarrow$ discretized surface
$\mathcal{C} \leftarrow 0$
for $\mathbf{c} \in \bar{\mathcal{B}}$ **do**
    if $|\dot{c}_n(t)| < \xi$ **then**
      $\mathcal{C} = \mathcal{C} \cup \mathbf{c}$
    end
end
if $t = t_0$ **then**
    for $\mathbf{c} \in \bar{\mathcal{C}}$ **do**
      if $\dot{c}_n(t_0^-) < 0$ **then**
        $\mathcal{C} = \mathcal{C} \setminus c$
      end
    end
end

Chapter 2

Background Theory

2.1 State Estimation

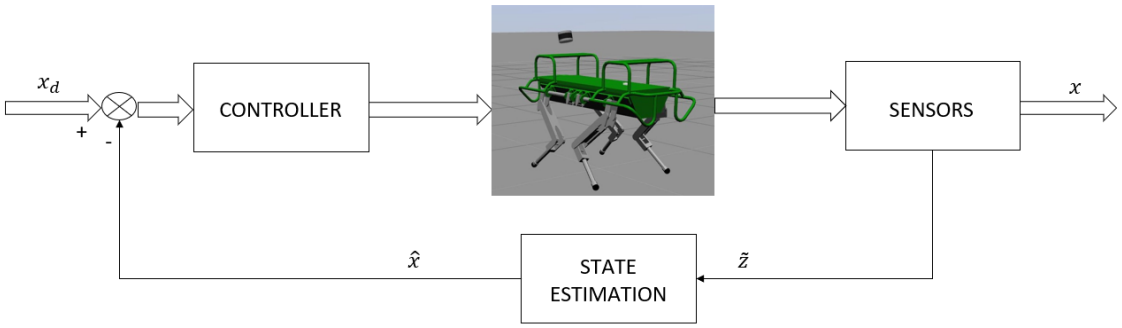


Figure 2.1: General block scheme of State Estimation with feedback

State estimation and control are the key elements of robot's motion. One method to perform State Estimation is the so called *probabilistic robotics*, a technique of estimating state and computing the *belief* from sensor data [53]. Since the usage of sensors, provides uncertainty in the measurements, the main purpose of State Estimation is to describe the *state* of the robot in the best possible way. In order to do that, probabilistic state estimation algorithms have the role to infer and estimate quantities that are not directly observable or measurable from sensor data. To properly understand the concepts of State Estimation and recursive estimation algorithm, it is important to do a brief introduction on probability theory and on robotics modelling theory.

2.1.1 Probability Theory

The main idea of probabilistic robotic is to use probability theory in order to represent explicitly uncertainty in robot perception and action.
Uncertainty in robotics is related to different factors:

1. **environments** - the robot surrounding world is dynamic and unpredictable.

2. **sensors** - the usage of sensors provides limitations and corrupts the measuremnts due to: i)resolution of sensor itself, ii)presence of noise.
3. **robots** - robot actuators may be inaccurate due to their quality of manufacturing.
4. **models** - building a model is a sort of simplification of the real scenario, as a consequence it gives rise to errors in the system.
5. **computation** - the algorithms used to describe and model the robot are an approximation of real-time system, providing as a consequence less accuracy in the results.

A solution to deal with it is to *estimate* the measurements coming from sensor data, following filtering algorithms based on probability theory. This is the reason why in this Chapter the basic concepts of probability are explained, in order to better understand later the working principle of filtering, analzyed in Chapter 2.2.

At the basis of Probabilistic Robotics there is the assumption that all the quantities can be consdered as random variables. Considering x as a random variable, $p(x)$ is the PDF of the random variable x over the interval $[a, b]$ [12]. $P(x)$ is a non-negative function that satisfies the axiom of total probability:

$$\int_a^b p(x)dx = 1 \tag{2.1}$$

Taking into account N-dimensional continuous variables as $p(x) = p(x_1, x_2, ..., x_N)$ where $x = (x_1, ..., x_N)$, in an interval $[ai, bi]$, the axiom of joint probability densities is:

$$\int_a^b p(x)dx = \int_{a_N}^{b_N} ... \int_{a_2}^{b_2} \int_{a_1}^{b_1} p(x1, x2, ...x_N)dx_1dx_2...dx_N = 1 \tag{2.2}$$

In probability theory it is important to show two main properties linked to PDF:

1. the mean μ

$$\mu = E[x] = \int xp(x)dx \tag{2.3}$$

2. the covariance matrix Σ

$$\Sigma = E[(x - \mu)(x - \mu)^T] \tag{2.4}$$

Running the experiment N times and using a random variable x with an associated PDF p(x), it is useful to consider the sample mean and the sample covariance, representing the realization of the random variable:

$$\begin{aligned} \mu_{meas} &= \frac{1}{N}\sum_{k=1}^{N} x_{k,meas} \\ \Sigma_{meas} &= \frac{1}{N-1}\sum_{k=1}^{N}(x_{k,meas} - \mu_{meas})(x_{k,meas} - \mu_{meas})^T \end{aligned} \tag{2.5}$$

Two random variables, x and y, are *statistically independent* if their joint density factor is:

$$p(x, y) = p(x)p(y) \tag{2.6}$$

while the variables are *uncorrelated* if:

$$E[xy^T] = E[x]E[y]^T \tag{2.7}$$

At the basis of the working principle of the Kalman Filter there is the theory about Gaussian Filter and *Gaussian probability density functions*. A gaussian PDF is in the following fashion:

$$p(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp\Big(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\Big) \tag{2.8}$$

where μ is the mean,σ^2 is the variance and σ is the standard deviation. A multivariate Gaussian PDF can be expressed ,using the mean μ and the covariance Σ,as follows:

$$p(x|\mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^N det\Sigma}} exp\Big(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\Big) \tag{2.9}$$

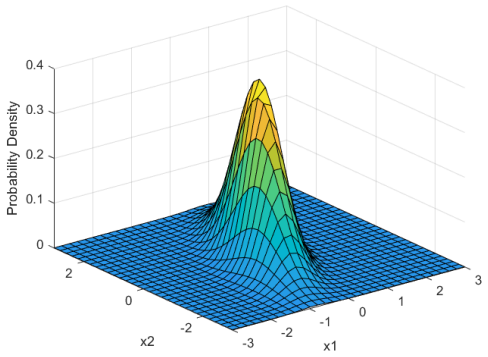


Figure 2.2: Gaussian Probability Density Function in 3D

In case of Gaussian PDF,uncorrelated variables are also statistically independent and statistically independent variables are also uncorrelated. As a consequence,the following equation is valid:

$$E[xy^T] = E[x]E[y]^T \tag{2.10}$$

Gaussian distribution is at the basis of Bayes Filter. In the Chapter 2.2, starting from Gaussian Filters, the procedure of Kalman Filter will be analyzed in details, considering that the *belief* are represented by Gaussian distribution.

The concept of the *belief* distribution is an important aspect in Probabilistic Robotics. Since the state of the robot is not directly measurable, the *belief* indicates the *state of knowledge* of the robot about the real state. To better understand this concept it is important to distinguish the environment interaction in two categories:

1. **sensor measurements**. The robot obtains informations about the surrounding environment by using its sensors.

2. **control actions**. These interactions are necessary to modify the state of the world.

From these types of interaction the robot collects two types of data:

1. **measurement data** represented as z_t. They are data about the state of the environment in a certain time.

2. **control data** represented as $u_{t1:t2}$ They are data about the *change of the state* in the environment.

After this premise, it is possible to define the *belief* as:

$$bel(x_t) = p(x_t|z_{1:t}, u_{1:t}). \tag{2.11}$$

this equation indicates the posterior obtained considering the state x_t as a probability distribution, conditioned by all the past data $z_{1:t}$ and $u_{1:t}$. To be more precise, the *belief* can be evaluated after incorporating the measurements data z_t, obtaining as a consequence:

$$\bar{bel}(x_t) = p(x_t|z_{1:t-1}, u_{1:t}). \tag{2.12}$$

that indicates the *prediction* of the state x_t, considering the previous state posterior.

Another important aspect for the working principle of filtering is the presence of noise in all the measurements. In order to obtain an estimated output from a corrupted measured signal, it is necessary to understand the characteristics of the noise and how it works, introducing its main properties. The noise can be considered as a stochastic process, or random process, in which the distribution, the density function, the mean and the covariance are functions of time.

$$\mu(t) = \int_{-\infty}^{+\infty} xp(x,t)dx; \tag{2.13}$$

$$\begin{aligned} \Sigma(t) &= E[(x(t) - \mu(t))(x(t) - \mu(t))^T \\ &= \int_{-\infty}^{+\infty} [x - \mu(t)][x - \mu(t)]^T p(x,t)dx; \end{aligned} \tag{2.14}$$

A stochastic process x(t) can be one of the followings different types:
• continuous random process, in which both RV and time are continuous.
• discrete random process, where RV is discrete and time is continuous.
• continuous random sequence, RV is continuous while time is discrete.
• discrete random sequence,both RV and time are discrete.

In optimal filtering and state estimation, it is often used, for simulating the model, the correlated white noise, in which a RV $x(t_1)$ is independent from a RV $x(t_2)$ for all time instant $t_1 \neq t_2$ [52]. In order to have a correlated white noise, it is necessary to create a random vector in which the elements are correlated using defined covariance matrices. The idea is to generate a random vector w with zero mean and covariance Q:

$$Q = \begin{bmatrix} \sigma_1^2 & \dots & \sigma_{1n} \\ \vdots & \vdots & \vdots \\ \sigma_{1n} & \dots & \sigma_n^2 \end{bmatrix} \tag{2.15}$$

In which all of its eigenvalues are non-negative and real:

$$\lambda(Q) = \mu_k^2 \qquad (k = 1, ..., n) \tag{2.16}$$

In order to do this, 4 steps have to be followed:

1. Analyze Q, find its eigenvalues and write them as $\mu_1^2, .., \mu_n^2$.
2. Evaluate the eigenvectors of Q and then write them as $d_1, ..., d_n$, to obtain the vector $D = [d_1, ..., d_n]$.
3. Compute the RV $v_i = \mu_i r_i$ where r_i is a random number with a unity variance.
4. Obtain in the end $w = Dv$.

This part related to probability, random variables and stochastic process is at the basis of the working principle of filtering technique, introduced in the Chapter 2.2.

2.1.2 Modelling Theory

To perform State Estimation, it is important to evaluate and compute the so called *state* of the robot, finding for example its position, orientation, velocity and acceleration, during time, in a certain reference frame. The pose of a rigid robot is composed by its position and orientation in the space respect to a reference frame.

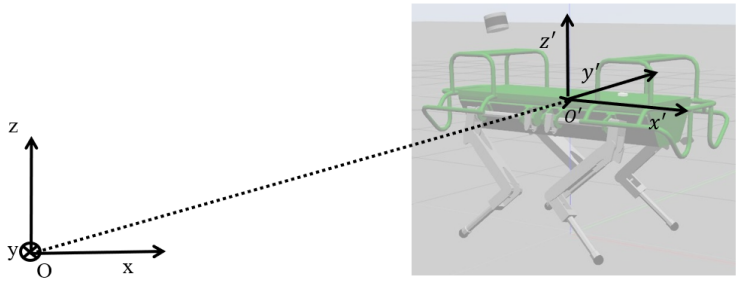


Figure 2.3: Position and orientation of a rigid body in the space - The world frame is represented by O_{xyz}, while the body frame of HyQ is $O'_{x'y'z'}$

Two important reference frames are considered for the computation of the pose:
• the inertial world coordinate system with origin x_w.
• the body fixed coordinate system with origin x_b.
To fully describe the pose of the robot, two elements are used: i) the *translation* in order to express the position of the rigid body and ii) the *rotation*, to express its orientation. To describe the *orientation* of the robot, in world frame or body frame, it is necessary to compute a 3x3 matrix called coordinate rotations matrix.

$$R = \begin{bmatrix} x' & y' & z' \end{bmatrix} = \begin{bmatrix} x'_x & y'_x & z'_x \\ x'_y & y'_y & z'_y \\ x'_z & y'_z & z'_z \end{bmatrix} \tag{2.17}$$

The representation of a point P in the space with respect to the frame $O - xyz$ is:

$$p = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{2.18}$$

while respect to the frame $O' - x'y'z'$ is:

$$p' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \tag{2.19}$$

so the vector p can be expressed as:

$$p = Rp' \tag{2.20}$$

where R is the transformation matrix of the vector p' in the coordinate frame $O' - x'y'z'$ into the vector p of coordinate frame $O - xyz$ [51].

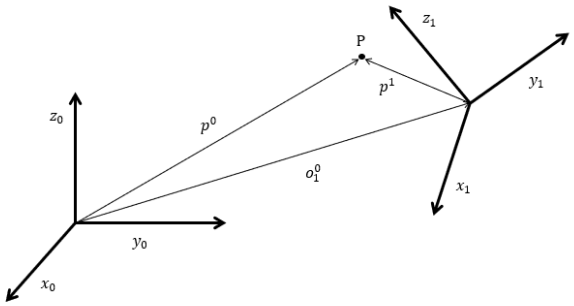


Figure 2.4: Representation of a point P in two differents coordinate frames

Considering a point P in the space, as shown in the figure 2.4, it is possible to express its position using *translation* and *rotation matrix* in the following equation:

$$p^0 = o_1^0 + R_1^0 p^1 \tag{2.21}$$

The combination of translation and rotation in a compact form gives rise to the homogeneous transformations matrix:

$$A_i^j = \begin{bmatrix} R_i^j & t \\ 0^T & 1 \end{bmatrix} \tag{2.22}$$

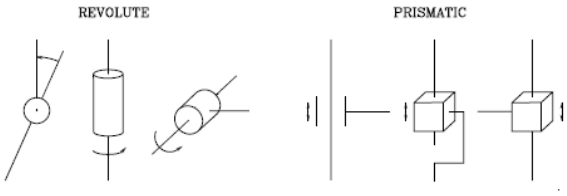


Figure 2.5: Prismatic and revolute joints [51].

where R_i^j is the rotation from reference frame i to reference frame j and t is the translation.
The relationship between joint variables and end-effector pose is called *robot's kinematics*. Joints are divided in two categories: *prismatic* and *revolute*, as shown in [Fig. 2.5]. The combination of all the joints of a structure gives rise to a *kinematics chain*, that could be i)*open* when there is a sequence of links until the end-effector, or ii) *closed*, when sequential links form a sort of loop. To easily find the *direct kinematics* of a robot structure, it is often used the DH Convention,explained in Appendix A.3.
An important aspect of *kinematics*, discussed also in *motion trajectory*, is the difference between *Operational space* and *Joint Space*. The first one indicates the space in which the end-effector task is specified, while the second one is the space related to joint variables. In relation to the operational, it is important to define also the *workspace*, the region in which all the joints execute all possible motions.

The relationship between joint velocities and end-effector is represented by the *differential kinematics*:

$$\dot{p_e} = J_P(q)\dot{q} \tag{2.23}$$

$$\dot{w_e} = J_O(q)\dot{q} \tag{2.24}$$

where $\dot{q}$ is the joint velocity. Considering the two types of joints, the angular and linear velocity are calculated as:

1. **prismatic joint**

$$\omega_i = \omega_{i-1} \text{ angular velocity} \tag{2.25}$$

$$\dot{p}_i = \dot{p}_{i-1} + \dot{d}_i z_{i-1} + \omega_i x r_{i-1,i} \text{ linear velocity} \tag{2.26}$$

2. **revolute joint**

$$\omega_i = \omega i - 1 + \dot{\theta}_i z_{i-1} \text{ angular velocity} \tag{2.27}$$

$$\dot{p}_i = \dot{p}_{i-1} + \omega_i x r_{i-1,i} \text{ linear velocity} \tag{2.28}$$

From the above equations it is possible to find linear velocity, linear acceleration, angular velocity and angular acceleration expressed in the world frame. If the point is rigidly attached to the body, the velocity and the acceleration in world frame are the following [23]:

$$\dot{x}_p = \dot{x}_b + R^T S(w') x'_{p/b}. \tag{2.29}$$

$$\ddot{x}_p = \ddot{x}_b + R^T[S(\dot{w}') + S(w)^2]x'_{p/b}. \tag{2.30}$$

while if the particle is in a moving frame the equations are:

$$\dot{x}_p = \dot{x}_b + R^T(\dot{x}'_{p/b} + S(w')x'_{p/b}). \tag{2.31}$$

$$\ddot{x}_p = \ddot{x}_b + R^T[[S(\dot{w}') + S(w)^2]x'_{p/b} + \ddot{x}'_{p/b} + 2S(w')\dot{x}'_{p/b}]. \tag{2.32}$$

where Body-fixed quantities are noted with a prime symbol, S is the skew-symmetric matrix and $x_{p/b}$, $\dot{x}_{p/b}$, $\ddot{x}_{p/b}$ are the position,velocity and acceleration of the particle p relative to the body b. Once the model is built using kinematics and differential kinematics, it is possible to use it inside the state estimation algorithms, filtering it and providing as a consequence results less corrupted by noise and by uncertainties.

2.2 Filtering

Filtering is an operation performed to extract informations about quantities of interest, generally corrupted by noise, using measurements coming from data sensors. The process of filtering is characterized by two main steps: i) *Prediction*,that is an *a priori* form of estimation, used to infer informations about the quantity of interest, starting from the measured data; ii) *Smoothing* that is the *a posteriori* form of estimation in which data measured are used for estimating the quantity on the basis of informations acquired before [18].

2.2.1 Linear Systems Theory

State-space systems provide mathematical descriptions of processes that happen in real world and they can be of two types: i) linear models, ii) non-linear models. Real situations present usually non-linearities in their behaviour, this is why a real environment can be described by non-linear models and simplified by a linear-model representation. As a consequence a continuous-time, deterministic linear system is described as:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{2.33}$$

where x is the state vector,u is the control vector and y is the output vector. The matrices A,B and C are important for describing the system: A is the system matrix, B is the input matrix while C is the output matrix. A non-linear system is represented as follows:

$$\begin{aligned} \dot{x} &= f(x, u, w) \\ y &= h(x, v) \end{aligned} \tag{2.34}$$

where w is the process noise, while v is the measurements noise. Moreover if f(.) and h(.) are functions of time, the system is *time-variant* otherwise it is called as *time-invariant* system.

Another important "splitting" is between *continuous-time* systems and *discrete-time* systems.A continuous time, linear, system is expressed as:

$$\begin{aligned} \dot{x} &= Ax + Bu \\ y &= Cx \end{aligned} \tag{2.35}$$

and a discrete time,linear system is defined as:

$$\begin{aligned} x_{k+1} &= Fx_k + Gu_k \\ y_k &= Hx_k \end{aligned} \tag{2.36}$$

In State Estimation and Control, most of the algorithms use systems in discrete-time, since the models are implemented in digital electronic devices. The transformation from a continuous-time system to a discrete-time system is called *discretization*. This process can be executed using three different methods:

1. **rectangular integration or Euler integration**. This is an approximation of $x(t)$ considered as a constant for the small time interval $(t_{n+1} - t_n)$.

$$x(t_n) = x(0) + \sum_{k=0}^{n} f[x(t_k), u(t_k), t_k]T \tag{2.37}$$

2. **Trapezoidal integration**. Each area, obtained under the curve f(x) from time interval t_n to time interval t_{n+1} is considered as a trapezoid and not as a rectangular.

$$x(t_{n+1}) = x(t_n) + \frac{1}{2}(\triangle x_1 + \triangle x_2) \tag{2.38}$$

3. **Runge-Katta integration**. Using this approach, at each time step it is possible to perform n functions calculations. The first order Runge-Katta integration is equivalent to rectangular integration, the second order Runge-Katta integration is equivalent to trapezoidal integration.

2.2.2 Bayes Filter

The simplest and most general algorithm, based on recursive estimation, for calculating the *belief* is represented by the Bayes Filter. This type of filtering is used extensively in robotics because it continuously updates the state of the robot from the recently acquired sensor data that are normally noisy.
Bayesan Theory was first implemented in 1763 by the researcher Thomas Bayes in the publication "Essay towards solving a problem in the doctrine of chances" [13]. Initially this approach was not considered, until its rediscovery thanks to the contribution of Laplace, who published its theory in "Théorie analytique des probailités". Bayesian Inference [15] [40] [43] got a lot of success and applications in several fields, such as statistics, estimation, machine learning and pattern recognition.

The working principle of the Bayes Filter can be described in two steps: i)the prediction or control update and ii) the measurement update. The first one is based on the evaluation of the control u_t, starting from the calculation of the *belief* over the state x_t on the basis of the prior *belief* over the state x_{t-1} and the control u_t. In the second step the *belief* of the state x_t is multiplied by the probability that the measurement z_t can be observed before. The general algorithm is shown as follows:

Bayes Filters were realized in order to estimate the *bel(x_t)*, where the state x_t is a random variable, using all the informations coming from sensors data. As a consequence the probability distribution *bel(x_t)* represents the uncertainty over x_t and it is defined as a *posterior density*. In general, computing such posterior density is really demanding and its complexity grows exponentially over time in relation with the increasing number of sensor measurements data. To make the system computationally solvable [28], it is assumed that the model is *Markovian*. Under this assumption the state x_t is a *complete state*:

Table 2.1: Algorithm for Bayes Filter [53].

1:	$(bel(x_{t-1}), u_t, z_t)$:
2:	**for all** x_t **do**
3:	$\bar{bel}(x_t) = \int p(x_t\|u_t, x_{t-1})bel(x_{t-1})dx$
4:	$bel(x_t) = \eta p(z_t\|x_t)\bar{bel}(x_t)$
5:	**endfor**
6:	**return** $bel(x_t)$

- Markov Property related to the states: x_t depends only on the previous state x_{t-1} obtaining the following equation

$$p(x_t|x_{t-1}, x_{t-2}, ..., x_0) = p(x_t|x_{t-1}). \tag{2.39}$$

  where the states $x_t : t = 0, 1, 2...$ are a Markovian sequence.

- Independence of measured data. This assumptions implies that the sensor measurements z_t depend only on the current state x_t :

$$p(z_t|x_{0:t}, z_{1:t}, u_{1:t}) = p(z_t|x_t). \tag{2.40}$$

Under Markov assumptions it is easier to compute the *bel* without losing informations. It is possible now to analyze the mathematical derivation of the Bayes Filter and its recursive estimation. The first step is to apply the *Bayes' rule*:

$$\frac{p(x|y) = p(y|x)p(x)}{p(y)} \tag{2.41}$$

using the posterior distribution $p(x_t|z_{1:t}, u_{1:t})$ it is obtained that:

$$\begin{aligned} p(x_t|z_{1:t}, u_{1:t}) &= \frac{p(z_t|x_t, z_{1:t-1}, u_{1:t})p(x_t|z_{1:t-1}, u_{1:t})}{p(z_t|z_{1:t-1}, u_{1:t})} \\ &= \eta p(z_t|x_t, z_{1:t-1}, u_{1:t})p(x_t|z_{1:t-1}, u_{1:t}) \end{aligned} \tag{2.42}$$

Using the information that x_t is a complete state and applying the *conditional independence* of Markov assumption, the equation (2.42) is simplified as:

$$p(x_t|z_{1:t}, u_{1:t}) = \eta p(z_t|x_t)p(x_t|z_{1:t-1}, u_{1:t}) \tag{2.43}$$

and hence

$$bel(x_t) = \eta p(z_t|x_t)\bar{bel}(x_t) \tag{2.44}$$

Expanding the term $\bar{bel}(x_t)$ and using the first Markov assumption (2.39), the recursive update equation is:

$$\bar{bel}(x_t) = \int p(x_t|x_{t-1}, u_t)p(x_{t-1}|z_{1:t-1}, u_{1:t-1})dx_{t-1} \tag{2.45}$$

In conclusion, Bayes Filter calculates the posterior distribution *bel* over the state x_t, on the basis of measured z_t and control u_t data up to time t,assuming that the world is *Markovian* and the state is complete. Bayes Filter, in general, is an abstract algorithm that provides only a probabilistic framework for recursive estimation. To implement it in practice, it is important to

consider three elements: i)the initial belief $p(x_0)$, the perceptual model $p(z_t|x_t)$ and the transition probability $p(x_t|u_t, x_{t-1})$. This type of estimation may have a variety of implementations that can differ on the basis of different assumptions regarding the initial belief, the measurements and the state transition probabilities. The challenging work is to find the suitable belief approximation for the considered robotic problem, considering three important aspects:

1. **Computational efficiency**. Linear Gaussian Estimation: time polynomial for calculating belief. Particle Filter: *any-time* characteristic for computing belief
2. **Accuracy of the Approximation**. Linear Gaussian: unimodal distributions for approximations. Particles Filter: can approximate wide array of distributions.
3. **Ease of Implementation**. The difficulty of algorithm implementation depends on two main factors: i) form of the measurment probability $p(z_t|x_t)$ and ii) the dynamic model $p(x_t|u_t, x_{t-1})$

After explaining the main characteristics of the Bayes Filter and its recursive estimation, it is possible now to introduce, in the following chapters, concrete algorithms based on its working principle.

2.2.3 Gaussian Estimation

Gaussian Filters were the first implementation of Bayes filtering in the continuous space. The main characteristic of Gaussian estimation is that the *belief* is represented as a multivariate distribution, as expressed in (2.9). In this equation two parameters are foundamentals for the Gaussian estimation: the mean μ and the covariance Σ, their dimensions depend on the states x. Following the rules presented in the previous Chapter, about the Bayesian Inference, it is now possible to build a Gaussian System.
The first step is to express the state probability as a linear function, including a Gaussian noise:

$$x_t = A_t x_{t-1} + B_t u_t + \epsilon_t. \tag{2.46}$$

where, as explained before, x_t and x_{t-1} are state vectors, while u_t is the control vector at time t. A_t and B_t are matrices related respectively to the state and to the control input. In particular A_t is a square matrix n x n where n is the state's dimension and B_t is a m x n matrix where m is the dimension of u_t. For what concern ϵ_t it is represented as a Gaussian random vector, its dimension is related to the state's dimension and its mean and covariance is expressed as R_t. It is possible to express now the equation (2.9), using A_t, B_t and R_t:

$$p(x_t|u_t, x_{t-1}) = det(2\pi R_t)^{-\frac{1}{2}} exp\Big\{ -\frac{1}{2}(x_t - A_t X_{t-1} - B_t u_t)^T R_t^{-1}(x_t - A_t x_{t-1} - B_t u_t)\Big\} \tag{2.47}$$

The second step is to express the measurements probability $p(z_t|x_t)$ adding a Gaussian noise:

$$z_t = C_t x_t + \delta_t. \tag{2.48}$$

where C_t is a matrix whose dimension k x n depends on the measurement vector z_t. As ϵ_t also δ_t is a Gaussian density vector with zero mean and covariance Q_t. Respecting these conditions, the following equation is obtained:

$$p(z_t|x_t) = det(2\pi Q_t)^{-\frac{1}{2}} exp\Big\{ -\frac{1}{2}(z_t - C_t x_t)^T Q_t^{-1}(z_t - C_t x_t)\Big\} \tag{2.49}$$

The third step is to use the $bel(x_0)$ as a normal distribution, obtaining:

$$bel(x_0) = p(x_0) = det(2\pi\Sigma_0)^{-\frac{1}{2}} exp\Big[-\frac{1}{2}(x_0 - \mu_0)^T \Sigma_0^{-1}(x_0 - \mu_0)\Big] \tag{2.50}$$

This three steps are sufficient to obtain that the posterior *bel* is always Gaussian over the time. After this premise is now possible to build the *Kalman Filter*.

The Kalman Filter was implemented for the first time by Rudolph Emil Kalman in 1950. Considering linear systems with Gaussian noise, the Kalman Filter is the BLUE filtering, best linear unbiased estimate filtering. For its working principle it is necessary to define as initial state x_0 and P_0 and to initialize them. Moreover the measurement datas z_t are the actual reading received by the sensors.In the following section, its algorithm is analyzed.

2.2.4 Discrete-Time Kalman Filter

This type of Filter wil be used in Chapter 4 for estimating the contact point of shin collision during robot locomotion. The dynamics of the Kalman Filter and its working principle is presented in the following lines. Considering a linear discrete-time system, the dynamic is, according to (2.36):

$$\begin{aligned} x_k &= F_{k-1}x_{k-1} + G_{k-1}u_{k-1} + w_{k-1} \\ y_k &= H_k x_k + v_k \end{aligned} \tag{2.51}$$

where w_k and v_k are represented by a Gaussian distribution, so they are white,uncorrelated, zero-mean and they have covariance Q_t and R_t respectively:

$$\begin{aligned} w_k &\sim (0, Q_k) \\ v_k &\sim (0, R_k) \\ E[w_k w_j^T] &= Q_k \delta_{k-j} \\ E[v_k v_j^T] &= R_k \delta_{k-j} \\ E[v_k w_j^T] &= 0 \end{aligned} \tag{2.52}$$

As in Bayesian Inference the two main steps are:

1. the *a posteriori estimate* in which the expected state $\hat{x}_k^+$ is computed on the basis of all the measurements included at that time k:

$$\hat{x}_k^+ = E[x_k[y_1, y_2, ..., y_k]] \tag{2.53}$$

2. the a priori estimate in which the expected value is obtained on the basis of the previous measurements not included at time k:

$$\hat{x}_k^- = E[x_k[y_1, y_2..., y_{k-1}]] \tag{2.54}$$

and it is possible to summarise that:

- $\hat{x}_k^-$ is the estimate of state x_k before the measurement at time k is processed.
- $\hat{x}_k^+$ is the estimate of state x_k after the measurement at time k is processed.

As said before, in order to have the Kalman Filter working , it is necessary to initialize x_0 and P_0:

$$\begin{aligned} \hat{x}_0^+ &= E(x_0) \\ P_0^+ &= E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T] \end{aligned} \tag{2.55}$$

Then its algorithm is expressed as:

$$\begin{aligned} P_k^- &= F_{k-1} P_{k-1}^+ F_{k-1}^T + Q_{k-1} \\ K_k &= P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \\ &= P_k^+ H_k^T R_k^{-1} \\ \hat{x}_k^- &= F_{k-1} \hat{x}_{k-1}^+ + G_{k-1} u_{k-1} \text{ a priori estimate} \\ \hat{x}_k^+ &= \hat{x}_k^- + K_k (y_k - H_k \hat{x}_k^-) \text{ a posteriori estimate} \\ P_k^+ &= (I - K_k H_k) P_k^- \end{aligned} \tag{2.56}$$

An important role is performed by the Kalman gain K that weight the innovation's contribution $y_k - H_k \hat{x}_k^-$ to the estimate; it indicates how the estimate is reliable compared to the actual measurements. After developing the model of the Kalman Filter, it is performed a step called *tuning* of the parameters. This phase is executed in practice to obtained a desired output with certain performances. The main parameters for tuning are the covariance matrices Q and R. Q is related to the model, if it is increased also the process noise increases, giving more importance to the measurements. R is related to the measurements, if R is really small the measurements are more reliable, if R is large the filter doesn't trust on the measurements. P_0 represents the velocity of convergence, if this value is small, the model trust perfectly on the initial state and it doesn't diverge so much from this value during the computation, while if this value is large, the initial condition is considered uncertain.

In conclusion the advantages of using the Kalman Filter are:

- this filter can be the solution if w_k and v_k are gaussian, zero-mean, uncorrelated and white noise.
- this filter is the best linear solution if w_k and v_k are zero-mean, uncorrelated and white.
- this filter can also be implemented if w_k and v_k are colored or correlated.
- this filter can be used also for nonlinear systems,using appropriate equations.

2.2.5 Continuous-Time Kalman Filter

One of the application of the Continuous-time Kalman Filter is for analog circuits. Its equations are obtained from the discrete-time filter equations expressed above, defining the sample time T that decreases to zero.In order to build the system in continuous is also necessary to define the process noise and measurement noise as continuous-time white noises. The covariance of the process noise is expressed as:

$$E[w(t)w^T(\tau)] = \frac{Q}{T}\delta(t-\tau) \tag{2.57}$$

where T and Q are the same as in discrete-time model, while $\delta(t-\tau)$ is the continuous-time impulse response (called also Dirac δ).Considering the continuous-time system:

$$\dot{x}(t) = w(t) \tag{2.58}$$

and doing the integration,as explained in [33], it is obtained that:

$$E[x(t)x^T(t)] = \frac{Qt}{T} = kQ \tag{2.59}$$

Comparing this result with the covariance of a discrete-time system, it can be noticed that it increases with time as the same in discrete-system.As a consequence the covariance Q in discrete-time is equivalent to the covariance $Q_c\delta(t)$ in continuous-time system where $Q_c = Q/T$:

$$w(t) \sim (0, Q_c) \tag{2.60}$$

$$E[w(t)w^T(\tau)] = Q_c\delta(t-\tau) \tag{2.61}$$

For what concern the measurement noise it is defined,in discrete-time, as:

$$\begin{aligned} y_k &= x_k + v_k \\ v_k &\sim (0, R) \end{aligned} \tag{2.62}$$

considering the sample time T that tends to 0:

$$\lim_{T\to 0} R = R_c\delta(t) \tag{2.63}$$

The relationship between R in discrete-time and R_c in continuous-time is represented by:

$$E[v(t)v(\tau)] = R_c\delta(t-\tau) \tag{2.64}$$

After derving the process noise and measurement noise in continuous, it is now possible to derive the whole system in continuous-time:

$$\begin{aligned} \dot{x} &= Ax + Bu + w \\ y &= Cx + v \\ w &\sim (0, Q_c) \\ v &\sim (0, R_c) \end{aligned} \tag{2.65}$$

Considering $T \to 0$ the following equations are obtained:

$$\begin{aligned} \hat{x}(0) &= E[x(0)] \\ P(0) &= E[(x(0) - \hat{x}(0))(x(0) - \hat{x}(0))^T] \\ K &= PC^T R_c^{-1} \\ \dot{\hat{x}} &= A\hat{x} + Bu + K(y - C\hat{x}) \\ \dot{P} &= -PC^T R_c^{-1} CP + AP + PA^T + Q_c \end{aligned} \tag{2.66}$$

2.2.6 Extendend Kalman Filter

In a real framework also the Kalman Filter has to be considered in a non-linear environment. The most commonly used technique for non linear estimation is the extended Kalman Filter (EKF).This type of filter is used in this thesis for estimating the contact point,as a consequence it is necessary to understand the theory for its working principle. The idea is to linearize the system around the estimation, and the estimation of the Filter is then based on the linearized system.

In continuous-time it is expressed as:

$$\begin{aligned} \dot{x} &= f(x, u, w, t) \\ y &= h(x, v, t) \\ w &\sim (0, Q) \\ v &\sim (0, R) \end{aligned} \tag{2.67}$$

Then in order to consider it in a *linearized* way, it is necessary to do the partial derivative around the current state estimate:

$$\begin{aligned} A &= \frac{\partial f}{\partial x}\Big|_{\hat{x}} \\ L &= \frac{\partial f}{\partial w}\Big|_{\hat{x}} \\ C &= \frac{\partial h}{\partial x}\Big|_{\hat{x}} \\ M &= \frac{\partial h}{\partial v}\Big|_{\hat{x}} \end{aligned} \tag{2.68}$$

After computing the matrices $\bar{Q} = LQL^T$ and $\bar{R} = MRM^T$,the filter equations can be evaluated:

$$\begin{aligned} \hat{x}(0) &= E[x(0)] \\ P(0) &= E[(x(0) - \hat{x}(0))(x(0) - \hat{x}(0))^T] \\ \dot{\hat{x}} &= f(\hat{x}, u, w_0, t) + K[y - h(\hat{x}, v_0, t)] \\ K &= PC^T\bar{R}^{-1} \\ \dot{P} &= AP + PA^T + \bar{Q} - PC^T\bar{R}^{-1}CP \end{aligned} \tag{2.69}$$

The discrete-time EKF is expressed as follow:

$$\begin{aligned} x_k &= f_{k-1}(x_{k-1}, u_{k-1}, w_{k-1}) \\ y_k &= h_k(x_k, v_k) \\ w_k &\sim (0, Q_k) \\ v_k &\sim (0, R_k) \end{aligned} \tag{2.70}$$

The initialization of $\hat{x}_0^+$ and P_0^+ is:

$$\begin{aligned} \hat{x}_0^+ &= E(x_0) \\ P_0^+ &= E[(x_0 - \hat{x}_0^+)] \end{aligned} \tag{2.71}$$

To write the dynamics of the Filter,it is necessary to compute the partial derivative as before:

$$\begin{aligned} F_{k-1} &= \frac{\partial f_{k-1}}{\partial x}\Big|_{\hat{x}_{k-1}^+} \\ L_{k-1} &= \frac{\partial f_{k-1}}{\partial w}\Big|_{\hat{x}_{k-1}^+} \\ H_k &= \frac{\partial h_k}{\partial x}\Big|_{\hat{x}_k^-} \\ M_k &= \frac{\partial h_k}{\partial v}\Big|_{\hat{x}_k^-} \end{aligned} \tag{2.72}$$

The *predict* equations for the estimation are the following:

$$\begin{aligned} P_k^- &= F_{k-1} P_{k-1}^+ F_{k-1}^T + L_{k-1} Q_{k-1} L_{k-1}^T \\ \hat{x}_k^- &= f_{k-1}(\hat{x}_{k-1}^+, u_{k-1}, 0) \end{aligned} \tag{2.73}$$

while the *update* equations for the estimation are:

$$\begin{aligned} K_k &= P_k^- H_k^T (H_k P_k^- H_k^T + M_k R_k M_k^T)^{-1} \\ \hat{x}_k^+ &= \hat{x}_k^- + K_k [y_k - h_k(\hat{x}_k^-, 0)] \\ P_k^+ &= (I - K_k H_k) P_k^- \end{aligned} \tag{2.74}$$

In this dissertation it is used the hybrid EKF, a combination of continuous-time and discrete-time equations, for the estimation of the contact point. Its implementation is fully described in the Chapter 4, in order to use the same equations needed for building the model of this thesis.

Chapter 3

Experimental Setup

The target platform used for evaluating the contact point during the locomotion is called HyQ, Hydraulically actuated quadruped robot. This robot is the final result of a series of implementations and studies, starting from the work of Dr. Claudio Semini in 2010 [46]. From that moment Hyq became the centre of research of the Dynamic Legged System Laboratory inside Istituto Italiano di Tecnologia. IIT is a national research Institute whose purpose is based on technological development and scientific progress [1], competing with realities in all over the world, as ETH or MIT. This is why inside this Institute,it is possible to enter in contact with modern and innovative systems and projects. DLS is one of the laboratories that investigates robotics, in particular legged robotics which aim is to realize a robust architecture able to walk on rough terrains [2], despite also external disturbances [27] which can cause robot task's failure.

3.1 System Overview

The starting point of HyQ robot realization was the first prototypes of the legs in 2008, built to carry heavy loads and at the same time to guarantee elasticity of movements.

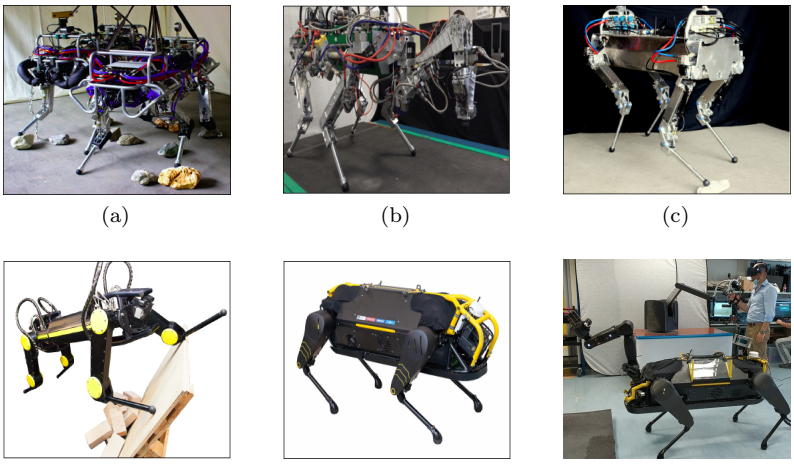
Since 2008 several versions of HyQ are realized to implement its structure and also its performances:

1. HyQ Green (2010) [49] Fig.3.1a. It is the robot used for doing experiments in this thesis. Usually this robot is considered as a "testing hardware", because before try new changes on HyQ Real, they are performed on HyQ Green.

2. HyQ Centaur (2015) [55] Fig.3.1b. It has the same architecture as before, but in addition it was mounted on it an hydraulic arm. This arm is compact (0.743m at its maximum extension), light-weight (12.5 Kg), fast (maximum speed of 4 m/s with no loads at end-effector) and it is characterized by six-degree of freedom.

3. HyQ2Max (2015) [48] Fig. 3.1d. The goal at the basis of its realization was to realize a more robust and versatile robot, able to do the so called *self-righting*, laying on its back and then turning back on its belly to stand-up again. Another difference is that all the cables or sentitive hardware parts, as electronic devices, sensors and actuators, are covered with a protection shield, in order to avoid their breaking during robot fall.

4. HyQMini (2015) [31] Fig.3.1c. It is the smallest version of HyQ robot, realized with small hydraulic actuators and also with small link legs (15% less in flex configurations).



(a) (b) (c)

(d) (e) (f)

Figure 3.1: HyQGreen(a), HyQCentaur(b),HyQMini (c), HyQ2Max (d), HyQ Real (e), HyQ Real + TeleOp Arm (f).

5. HyQReal (2019) [45]Fig.3.1e. This is the newest version of HyQ, presented for the first time at ICRA 2019. The new structure is composed by two HPUs (Front and Hind), a battery of 48V and four computers mounted on it. It is a result of a collaboration with the MOOG Inc. expecially for what concern the design of integrated smart actuators (ISA) and smart manifolds [11], obtaining as a consequence a more compact structure. The weight is approximately of 130 Kg and it is able to carry heavy weights as a small passenger airplane (Piaggio P180 Avanti).

6. HyQ real and Teleoperation Arm mounted on its torso (2020) Fig.3.1f. This project was born in collaboration with INAIL and its aim is to reduce the risk for workers safety and health in particular scenarios such as nuclear, chemical, disaster response, construction/demolition. In this type of environments hazards can be present in the form of radiation,toxicity,exposure to chemicals. The teleoperation arm can be a solution to perform tasks guaranteeing the well-being of workers or humans involved in the rescue operation [3] [44].

3.1.1 Mechanical Design

HyQ was designed to perform different types of motions, varying from agile and fast locomotion to slow and careful motions on uneven terrains. Its weight is about 90 Kg and its dimensions are 1.0m x 0.5 m x 0.98m (Length x Width x Height). The robot is composed by four legs: Left Front (LF), Right Front (RF), Left Hind (LH) and Right Hind (RH). Each leg is characterized by three DoFs, for an amount of twelve torque-controlled joints,hydraulically actuated:

- 4 Hip Abduction-Adduction (HAA)
- 4 Hip Flexion-Extension (HFE)
- 4 Knee Flexion-Extension (KFE)

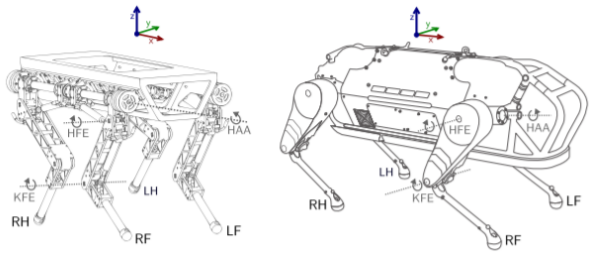


Figure 3.2: Representation of joints,legs and frame on Hyq(left) and HyQ Real(right). The legs are left-front (LF), right-front(RF), left-hind(LH) and right-hind (RH). The joints are HAA,HFE and KFE [56]

The pressure of about 16MPa allows the joints to reach torque from 120N for the HAA to 145N for HFE and KFE. In the following table are resumed the main characteristics of HyQGreen and HyQReal

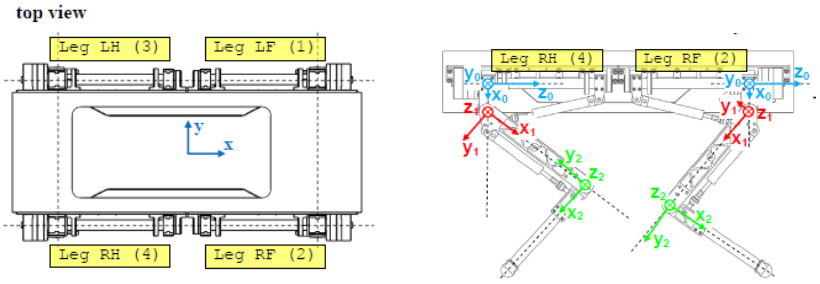
Table 3.1: Table of HyqGreen and HyQReal Parameters

	Green	Real
Robot's mass	90Kg	130Kg
Operating pressure	160bar	160bar
Dimension (fully extended leg)	1.0m x 0.5m x 0.98m	1.3m x 0.67m x 0.9m
Leg length/link length	from 0.339m to 0.789m	hip (HAA-HFE):0.017 upper leg (HFE-KFE):0.36m upper leg (KFE-foot):0.38m
Foot radius	2cm	2cm
Number of joints	12	12
HAA range motion	$-90°$ $+30°$	$60°$
HFE range motion	$-70°$ $+50°$	$110°$
KFE range motion	$+20°$ $+140°$	$130°$
Maximum torque HAA	120Nm	165Nm
Maximum torque HFE	150Nm	270Nm
Maximum torque KFE	150Nm	240Nm
onboard computer		Intel core i7 with real-time Linux

3.1.2 Frames of Reference

In order to evaluate the contact point using direct kinematics an important aspect is the definition of the References Frames adopted in the equations. As it can be seen in the Fig. 3.3 the base frame is at the geometric centre of the torso and from it ,it is possible to obtain the HAA, HFE, KFE reference frames following the DH Conventions. The world frame is an inertial frame whose origin coincides with a fixed point on Earth. When the robot is in its starting position, the base frame coincides with the world frame, considering an offset along z-axis of exactly the height of the robot. The IMU frame is located at the origin of sensor IMU and its transformation from the base frame is obtained using a CAD model of the robot and considering datasheet. The

horizontal frame, with its origin aligned with the base frame and its z-axis aligned along the gravity vector, is useful for trajectory planner applications.



(a) top view of HyQ.

(b) Reference Frames of HyQ.

Figure 3.3: HyQ Design [47]

3.1.3 Sensors

The starting point of State Estimation is getting informations from data sensors, as explained in the Chapter 2.1. HyQ is equipped with different types of sensors to obtain informations as accurate as possible; this is why this section is dedicated to their description. The *proprioceptive* senors mounted on HyQ are: Encoders, IMU and F/T sensors. In each joint of HyQ are implemented two encoders: absolute (AMS model AS50445) and relative/optical (Avago model AEDA3300 BE1 Fig.3.4a). They are useful to measure the joint position q and the joint velocity $\dot{q}$ obtained computing the derivative from the position. These sensor measurements data are important to evaluate position, velocity and acceleration of the knee and of the foot.
Since HFE and KFE joints are moved by pistons, they are equipped with loadcells (Burster 8417 Fig. 3.4b) in order to measure forces. The joint HAA is equipped with a custom made torque sensor. These sensors are important to detect the torque τ used for dynamic approaches.
The IMU is then used to measure the base angular velocity and the base acceleration, specifically it measures the force $g + acc$. The IMU mounted on HyQ is the KVH 1775 Fig. 3.4c,this is also a device used by the quadruped legged robot Cheetah of MIT.

(a) Avago AEDA3300 BE1.

(b) Burster 8417.

(c) KVH1775.

Figure 3.4: Proprioceptive Sensors mounted on HyQ. 3.4a encoder, 3.4bloadcell, 3.4c IMU

The exteroceptive sensors mounted on HyQ are depth sensors, stereo cameras and LIDARS. It is important to underline that, for the experiments done in this thesis, the robot is completely blind and the informations are obtained only from proprioceptive sensors data. However, in a real scenario, both exteroceptive and proprioceptive sensors are used, in order to provide informations regarding position, orientation and description of the environment, as accurate as possible.
The RGB-D sensors or depth sensor cameras mounted on HyQ are Realsense. The output obtained by these sensors is a color point cloud.
For what concern the LIDARs, they are deeply used in robotics because they provide high precision measurements despite not optimal weather conditions such as fog and dust. On HyQ LIDARS are mainly used for SLAM purposes. The LIDAR mounted on HyQ robot is Velodyne puck.

3.1.4 Software/Hardware Architecture

The robot is equipped with two on board PCs that are used to communicate with the framework and allows the robot to perform all the required tasks. The *Control* PC runs the *Supervisor*, a real-time Linux kernel,and its main function is to calculate the control in Real-Time. The *Vision* PC runs a regular kernel and it is responsible for collecting the exteroceptive inputs, building a map of the environment and sending it to the *Control* PC. In case of failure of the *Vision* PC, the robot can move blindly, using only informations coming from the state-estimator. The user commands are coming from the *Operator* PC through a Wi-Fi or Ethernet connection.

In HyQ the low-level control signals are computed inside the *Control* PC and they are sent to the actuators. The first step is performed by the state-estimator that receives the signal coming from the sensors and it sends it to the *Control* PC. At the same time the cameras create the map of the environment to be sent to the *Vision* PC. In the second step, the obtained map and the states informations are sent to the *Supervisor* controller which runs in a real-time and it sends the desired torque to the low-level control module. The latter manages the actuators sending the valve commands. The hardware/software architecture is shown in the figure Fig. 3.5.

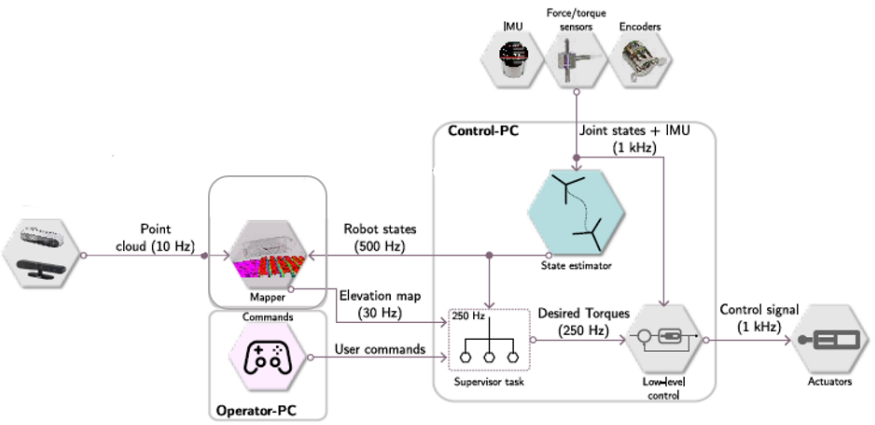


Figure 3.5: Scheme block of the software/hardware architecture [56]

3.2 HyQ Controller

The locomotion of HyQ robot can be of different type: crawl [27], bound [38] and trot. In this thesis it is analyzed the trotting of the robot using a specific locomotion framework: the Reactive Controller Framework (RCF).

3.2.1 Reactive Controller Framework

The RCF is mainly used for uneven and rough terrains thanks to its capability to perform robust dynamic locomotion. This modular framework was first designed to receive informations only provided by proprioceptive sensors, then its usage was extended cosnidering exteroceptive sensors as well. It is characterized by two parts: i) a motion control block and ii) a motion generation block Fig. 3.6. The latter is responsible for assigning the trajectories of the feet with respect to the horizontal frame. The main characteristic of choosing this frame is that the generation of feet trajectories is indipendent from the trunk attitude, in this way it is possible to deal with non-flat terrain. The motion control block has the role to provide corrective actions in order to obtain the desired trunk motion. These actions can be defined at the kinematic level using the push recovery, or at torque level using the whole body control.
Moreover RCF contains a number of modules, used for increasing the realiability of robot locomotion is difficult terrains. Examples of these modules are: push recovery, foot collision detection, terrain adjustment and shin collision detection. Their contribution is to send reactive components to the motion control block or to the motion generation block in order to adjust the trajectory. The contribution of this thesis is to extend the part related to the shin collision detection, in order to adjust the trajectory of the robot considering a 3D plane.

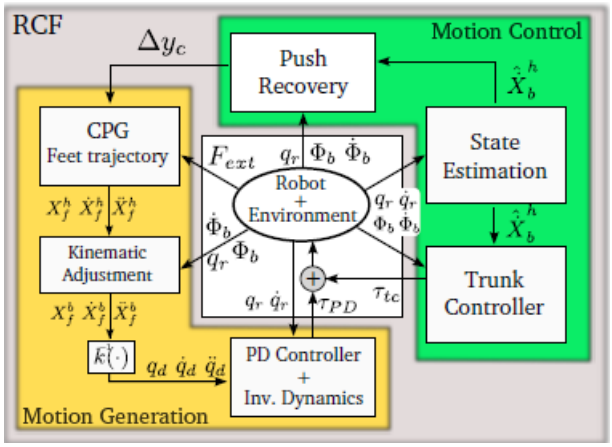


Figure 3.6: Scheme Block of the Reactive Controller Framework RCF [9]

3.3 Gazebo and Robot Operating Systems

The validation of the results in simulation is performed using ROS, Gazebo as a simulator and RVIZ as a visualizer [42]. The communication between ROS and Gazebo is allowed thanks to

a set of packages *gazebo ros pkgs* that are added to the framework. They used topics and services to send the informations from ROS to Gazebo and vice-versa.

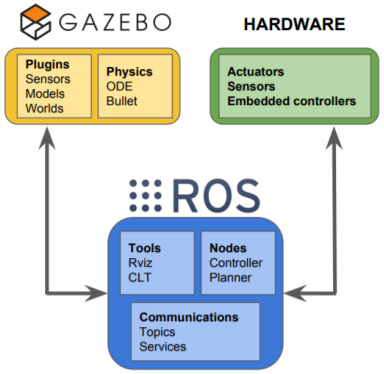


Figure 3.7: Scheme representing the communications between the simulation in Gazebo, the controller in ROS and the real hardware [4]

3.3.1 ROS working principle

ROS,*Robot Operating System*, is a open source meta-operating system that is able to provide a software platform to perform robotic applications [4]. The characteristic of ROS is that every code can be implemented infinite times on different kind of applications and robots. Moreover, it provides tools such as rqt and RVIZ, used for the phases of debugging and visualization. It is organized in basics units known as packages, which contain nodes that are the smallest unit of ROS environment, used for one single purpose. Every node is characterized by Name, message,type and URI address. ROS is a communication based program that uses specific elements to send the informations:

- **Topic** [5]: it represents the communication between a publisher node and a subscriber node in an unidirectional way. The *publisher* has the role to write a message on a topic, while the *subscriber* receives the information published in the connected topic. The action of receiving a message performed by the subscriber is called *Callback Function*.

- **Service** [6]: it is a bidirectional communications used for exchanging response/request informations between service client node and service server node. The latter provides a service that can be called by the client node using a request message. When the service is concluded,a response message is sent to the client.

- **Action**: it is a bidirectional communication between an action server node and an action client node, used to send goal/result/feedback messages.

The working principle of the message communication is shown in Fig.3.8.
As said before, ROS provides an ever-growing quantity of packages used for programming robots. One of the most important is the *ros control* [19], a set of packages related to the lower level

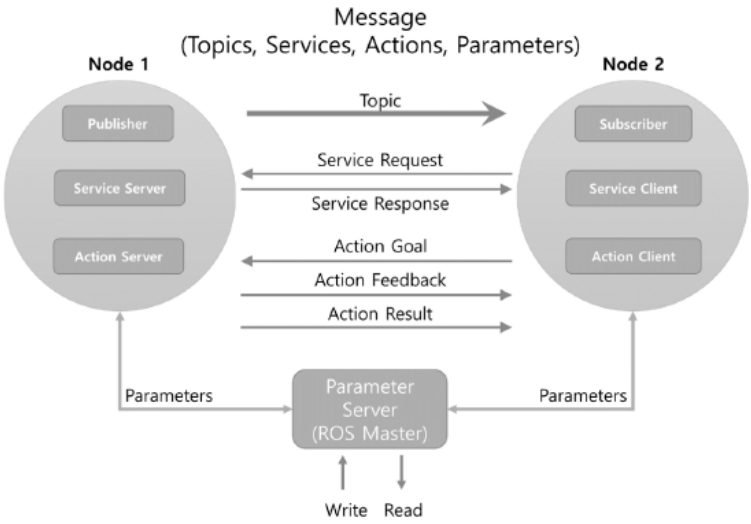


Figure 3.8: Message communication scheme [41]

controllers. In this work the communication between the hardware and ROS, passes through the *Supervisor* controller,which takes the informations from the sensors mounted on the robot and it sends them directly to ROS. The *Supervisor* controller used for HyQ robot, manages several type of different controllers with different perfomances and goals: i)Crawl Controller, ii)Leap Controller, iii)Prep Controller, iv)RCF Controller and v)VM Force Optimization Controller. The one used in this thesis is the RCF controller and the working principle of the communication is described briefly as follows:

1. informations related to the position, velocity and angular velocity of the robot's base are subscribed from the Supervisor controller and sent to the RCF Controller.

2. the topic related to the contact point,evaluated through the state estimation, is published in ROS for: i) saving the datas using *ros bag* and then post-processing the result in Matlab, ii) stabilizing the controller using the obtained result as a feedback.

As it can be seen in the Fig.3.9 the RCF controller receives the informations from the State Estimator, which computes the actual state pose and twist of the base at a frequency of 250 Hz. The required inputs for evaluating the contact point are: q, $\dot{q}$, p^w_{base}, v^w_{base}, ω^w_{base}.
Inside the RCF controller the first step is to initialize it, in order to activate the trunk controller, to get informations about the robot pose in the environment, to evaluate the ground reaction forces and to stabilize the robot base. The second step is to decide which Trajectory Generator Methods applied to the robot: i) TROT or ii) PushUp. The former allows the robot to perform the trotting, a type of walking in which two opposite legs are in contact with the ground,while the other two are in swing phase. The latter represents the PushUp motion, used mainly to verify velocity or acceleration in each joint. The velocity of the robot or the legs motion are examples of parameters that can be directly chosen by the user, allowing the robot to perform different types of locomotion in different types of scenarios.

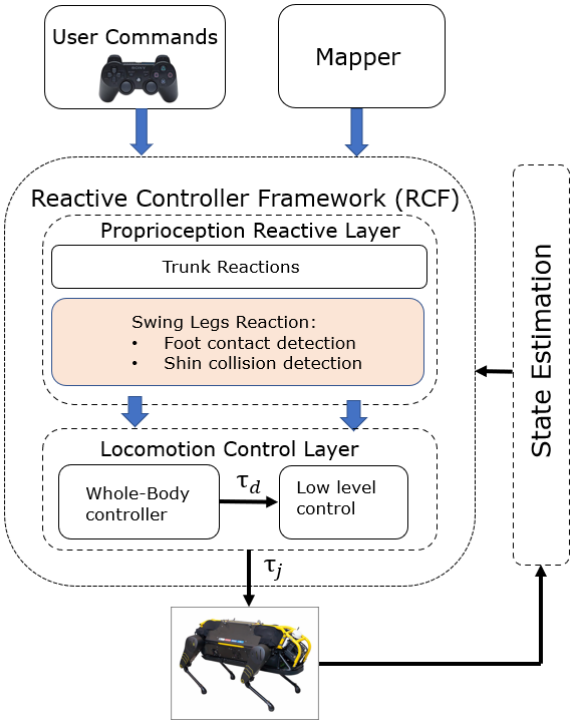


Figure 3.9: Overview of the locomotion framework - in orange the target analyzed in this thesis

3.3.2 GAZEBO platform

Gazebo is an open source 3D dynamics simulator [32], that is able to provide tools, models and worlds, for testing and simulating robots in a realistic way through the physics engine [7]. The physics engine availables is very wide, depending on the user requirements: ODE, DART, Symbody and Bullet. Gazebo has its own format, the SDF - Simulation Description Format [8], used for describing robot models or environments that can be freely available.
To perform a simulation in Gazebo using ROS control, it is necessary to add two interfaces: i)an *hardware interface* used to represent a simulated or real robot, ii) a *controller*, that, as explained before, is responsible of the communication with the hardware sending commands. The connection between ROS and the real robot Fig. 3.7, is fulfilled through the usage of the Hardware Abstraction Layer, represented by the `RobotHW` interface. The latter sends commands to the real controller and at the same time, it reads the informations providing by encoders mounted on the robot. The most known default hardware interfaces are:

- PositionJointInterface: used to command position-based joints.
- VelocityJointInterface: used to command velocity-based joints.
- EffortJointInterface: used to command effort-based joints.

In simulation, the `RobotHW` interface is substituted by the `RobotHWSim`. It has the same functionalities and as a consequence it is able to read states and to command joints using Gazebo simulator.

Chapter 4

Contact Detection and Localization

4.1 Model Development

The equations for the system are found using only the kinematics. In both 2D and 3D approach, are used only proprioceptive sensors providing informations on position,velocity and acceleration of the robot's base in the environment. This method can be considered as a sensor-less approach because the localization of the collision is detected using only kinematic equations starting from the trunk of the robot. In this way the presence of noise,provided by the usage of sensors, is reduced. To deal with noisy data sensor measurements,a filtering stage is added in the end to obtain a *clear* output and use it to stabilize the trunk of the robot.

4.1.1 Model in 2D

The starting point of this thesis is the model in 2D, performed to detect the collision point in order to then allows the robot to overcome a step, avoiding the slippage, during blind locomotion. Find correctly the contact point is useful to have a better locomotion of the robot, as proven in [10]. In fact an high delay on the estimation of the contact point, gives rise to a shin slippage of the robot, due to the presence of errors in the estimation. Another consideration is based on the analysis of the velocity: the higher is the velocity, the less the collision is detrimental for the motion. Morevover positive and negative errors on the estimation, have opposite effect on the whole motion of the robot: i) positive errors can cause the stuck of the robot, due to the slippage of the shin while ii) negative errors help the robot to overcome the step, this is why the trunk controller provides lower joint torques and as a consequence lower ground reaction forces.

A solution to detect correctly the contact point is to build the 2D model, based only on the evaluation of position and velocity of the contact point and of the knee. The model is simplified in a 2D plane in order to reduce the complexity of the system.The shin contact point C_s and the shin velocity $\dot{C}_s$ are found using trigonometric equations:

$$\begin{aligned} P_{k_x} &= r_s cos(\pi/2 + \theta_s) + C_s cos(\theta_s) \\ P_{k_z} &= r_s sin(\pi/2 + \theta_s) + C_s sin(\theta_s) \end{aligned} \tag{4.1}$$

where $P_k = (P_{k_x}, P_{k_z})$ is the position in x and z axis of the knee. From the position is then

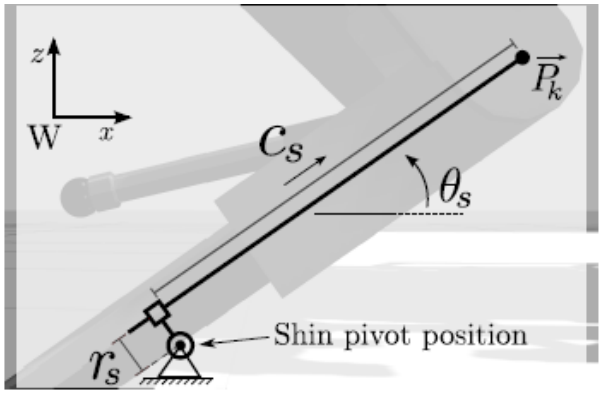


Figure 4.1: Representation of the shin collision in a 2D plane [10]

necessary to find the velocity, simply doing the derivative by time, in order to evaluate $\dot{C}_s$.

$$\begin{aligned} \dot{P}_{k_x} &= -r_s cos(\theta_s)\dot{\theta}_s + \dot{C}_s cos(\theta_s) - C_s sin(\theta_s)\dot{\theta}_s \\ \dot{P}_{k_z} &= -r_s sin(\theta_s)\dot{\theta}_s + \dot{C}_s sin(\theta_s) + C_s cos(\theta_s)\dot{\theta}_s \end{aligned} \tag{4.2}$$

The goal is to find C_s and $\dot{C}_s$, so everything is then written in function of C_s and $\dot{C}_s$:

$$\begin{bmatrix} \dot{P}'_{k_x} \\ \dot{P}'_{k_z} \end{bmatrix} = \begin{bmatrix} cos(\theta_s) & -sin(\theta_s)\dot{\theta}_s \\ sin(\theta_s) & cos(\theta_s)\dot{\theta}_s \end{bmatrix} \begin{bmatrix} \dot{C}_s \\ C_s \end{bmatrix} \tag{4.3}$$

and using the inverse of the Jacobian is obtained that:

$$\begin{bmatrix} \dot{C}_s \\ C_s \end{bmatrix} = \begin{bmatrix} cos(\theta_s) & sin(\theta_s) \\ -sin(\theta_s)\dot{\theta}_s^{-1} & cos(\theta_s)\dot{\theta}_s^{-1} \end{bmatrix} \begin{bmatrix} \dot{P}'_{k_x} \\ \dot{P}'_{k_z} \end{bmatrix} \tag{4.4}$$

Doing experiments, the robot is able to overcome a step stopping the shin slippage during the collisions and improving as a condequence the locomotion. However this 2D model has limitations, this is why then the model is implemented in 3D:

- the 2D model doesn't consider the velocity and also the movement along y axis. If the robot has slippage along y axis, the controller doesn't stabilize the robot.
- the 2D model is evaluated considering only position and velocity,so it doesn't consider the case in which the velocity is equal to zero. In this specific scenario there is a singularity and a solution to deal with it, is to add in the model the acceleration equations.

4.1.2 Model in 3D

The model in 3D is built in order to provide a solution to the limitations described in the 2D system. To obtain the model, the starting point is to compute the position, velocity and acceleration of the knee and of the foot using the kinematics. The first step is the evaluation of the homogeneous transformation matrices, following the reference frames shown in Fig. 3.3b. Considering LF and RH leg, the following homogeneous transformation matrices are obtained:

$$T_{LF} = \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} & t_1 \\ R_{2,1} & R_{2,2} & R_{2,3} & t_2 \\ R_{3,1} & R_{3,2} & R_{3,3} & t_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.5}$$

where:
$R_{1,1} = -sin(q_2)cos(q_3) - cos(q_2)sin(q_3)$
$R_{1,2} = sin(q_2)sin(q_3) - cos(q_2)cos(q_3)$
$R_{1,3} = 0$
$t_1 = -a_4cos(q_3)sin(q_2) - a_4sin(q_3)cos(q_2) + d_1 - a_3sin(q_2)$
$R_{2,1} = -sin(q_1)cos(q_2)cos(q_3) + sin(q_1)sin(q_2)sin(q_3)$
$R_{2,2} = sin(q_1)cos(q_2)sin(q_3) + sin(q_1)sin(q_2)cos(q_3)$
$R_{2,3} = cos(q_1)$
$t_2 = -sin(q_1)cos(q_2)a_4cos(q_3) + a_4sin(q_3)sin(q_1)sin(q_2) - a_3cos(q_2)sin(q_1) - a_2sin(q_1) + d_0$
$R_{3,1} = -cos(q_1)cos(q_2)cos(q_3) + sin(q_2)cos(q_1)sin(q_3)$
$R_{3,2} = cos(q_1)cos(q_2)sin(q_3) + sin(q_2)cos(q_1)cos(q_3)$
$R_{3,3} = -sin(q_1)$
$t_3 = -a_4cos(q_3)cos(q_1)cos(q_2) + a_4sin(q_3)sin(q_2)cos(q_1) - a_3cos(q_2)cos(q_1) - a_2cos(q_1)$

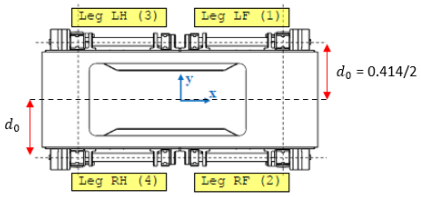
$$T_{RH} = \begin{bmatrix} R'_{1,1} & R'_{1,2} & R'_{1,3} & t'_1 \\ R'_{2,1} & R'_{2,2} & R'_{2,3} & t'_2 \\ R'_{3,1} & R'_{3,2} & R'_{3,3} & t'_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.6}$$

where:
$R'_{1,1} = -sin(q_2)cos(q_3) - cos(q_2)sin(q_3)$
$R'_{1,2} = -sin(q_2)sin(q_3) + cos(q_2)cos(q_3)$
$R'_{1,3} = 0$
$t'_1 = -a_4cos(q_3)sin(q_2) - a_4sin(q_3)cos(q_2) - d_1 - a_3sin(q_2)$
$R'_{2,1} = sin(q_1)cos(q_2)cos(q_3) - sin(q_1)sin(q_2)sin(q_3)$
$R'_{2,2} = sin(q_1)cos(q_2)sin(q_3) + sin(q_1)sin(q_2)cos(q_3)$
$R'_{2,3} = cos(q_1)$
$t_2 = sin(q_1)cos(q_2)a_4cos(q_3) - a_4sin(q_3)sin(q_1)sin(q_2) + a_3cos(q_2)sin(q_1) + a_2sin(q_1) - d_0$
$R'_{3,1} = -cos(q_1)cos(q_2)cos(q_3) + sin(q_2)cos(q_1)sin(q_3)$
$R'_{3,2} = cos(q_1)cos(q_2)sin(q_3) + sin(q_2)cos(q_1)cos(q_3)$
$R'_{3,3} = sin(q_1)$
$t'_3 = -a_4cos(q_3)cos(q_1)cos(q_2) + a_4sin(q_3)sin(q_2)cos(q_1) - a_3cos(q_2)cos(q_1) - a_2cos(q_1)$

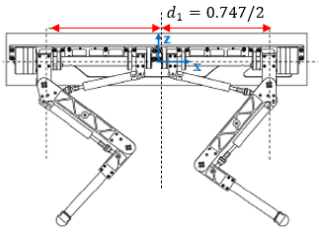
The characteristic of an homogeneous transformation matrix is that the submatrix R represents the rotation matrix from the base of the robot to the desired joint(knee/foot), while the last column represents the translation t along the three axes x, y and z. The inputs q_1, q_2 and q_3 are the joints angle taken directly from the sensors, in particular from the encoders mounted on the robot, while the measurements d_0, d_1, a_2, a_3 and a_4 are related to the stucture and geometry of the robot as shown in the Fig. 4.2. From T_{LF} and T_{RH} is then easy to find the position of the foot for LF leg and for RH leg, simply considering the last column of the matrices.

$$p_{foot_{LF}} = \begin{bmatrix} -a_3s(q_2) + d_1 - a_4c(q_2)s(q_3) - a_4s(q_2)c(q_3) \\ -a_3s(q_1)c(q_2) - a_2s(q_1) + d_0 - s(q_1)c(q_2)a_4c(q_3) + s(q_1)s(q_2)a_4s(q_3) \\ -c(q_1)a_3c(q_2) - a_2c(q_1) - a_4c(q_3)c(q_1)c(q_2) + a_4s(q_3)c(q_1)s(q_2) \end{bmatrix} \tag{4.7}$$

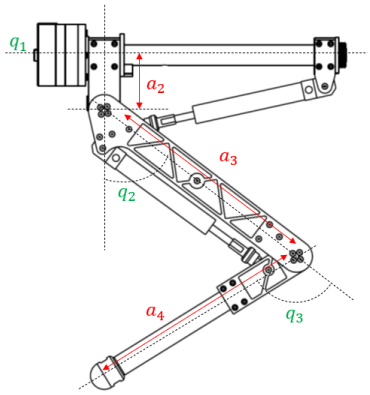
$$p_{foot_{RH}} = \begin{bmatrix} -a_3s(q_2) - d_1 - a_4c(q_2)s(q_3) - a_4s(q_2)c(q_3) \\ +a_3s(q_1)c(q_2) + a_2s(q_1) - d_0 + s(q_1)c(q_2)a_4c(q_3) - s(q_1)s(q_2)a_4s(q_3) \\ -c(q_1)a_3c(q_2) - a_2c(q_1) - a_4c(q_3)c(q_1)c(q_2) + a_4s(q_3)c(q_1)s(q_2) \end{bmatrix} \tag{4.8}$$



(a) top view HyQ with dimension d_0.



(b) side view of HyQ with dimension d_1.



(c) Leg of HyQ.

Figure 4.2: Structure of HyQ robot.

In order to find then the velocity and acceleration, it is necessary to evaluate also the Jacobian for each leg.This particular matrix is obtained doing the partial derivative in δq_1, δq_2 and δq_3 for each element of the vector position:

$$J_{foot_{LF}} = \begin{bmatrix} \frac{\partial p_{f_{LF}}[1]}{\partial q_1} & \frac{\partial p_{f_{LF}}[1]}{\partial q_2} & \frac{\partial p_{f_{LF}}[1]}{\partial q_3} \\ \frac{\partial p_{f_{LF}}[2]}{\partial q_1} & \frac{\partial p_{f_{LF}}[2]}{\partial q_2} & \frac{\partial p_{f_{LF}}[2]}{\partial q_1 3} \\ \frac{\partial p_{f_{LF}}[3]}{\partial q_1} & \frac{\partial p_{f_{LF}}[3]}{\partial q_2} & \frac{\partial p_{f_{LF}}[3]}{\partial q_3} \end{bmatrix} \tag{4.9}$$

where the terms inside the Jacobian matrix are equal to:

$J_{1,1} = 0$
$J_{1,2} = -a_4 cos(q_3)cos(q_2) + a_4 sin(q_3)sin(q_2) - a_3 cos(q_2)$
$J_{1,3} = a_4 sin(q_3)sin(q_2) - a_4 cos(q_3)cos(q_2)$
$J_{2,1} = -cos(q_1)cos(q_2)a_4 cos(q_3) + a_4 sin(q_3)cos(q_1)sin(q_2) - a_3 cos(q_1)cos(q_2) - a_2 cos(q_1)$
$J_{2,2} = sin(q_1)sin(q_2)a_4 cos(q_3) + a_4 sin(q_3)sin(q_1)cos(q_2) + a_3 sin(q_1)sin(q_2)$
$J_{2,3} = sin(q_1)cos(q_2)a_4 sin(q_3) + a_4 cos(q_3)sin(q_1)sin(q_2)$
$J_{3,1} = sin(q_1)cos(q_2)a_4 cos(q_3) - sin(q_1)sin(q_2)a_4 sin(q_3) + sin(q_1)a_3 cos(q_2) + a_2 sin(q_1)$
$J_{3,2} = cos(q_1)sin(q_2)a_4 cos(q_3) + cos(q_1)cos(q_2)a_4 sin(q_3) + cos(q_1)a_3 sin(q_2)$
$J_{3,3} = cos(q_1)cos(q_2)a_4 sin(q_3) + cos(q_1)sin(q_2)a_4 cos(q_3)$

$$J_{foot_{RH}} = \begin{bmatrix} \frac{\partial p_{f_{RH}}[1]}{\partial q_1} & \frac{\partial p_{f_{RH}}[1]}{\partial q_2} & \frac{\partial p_{f_{RH}}[1]}{\partial q_3} \\ \frac{\partial p_{f_{RH}}[2]}{\partial q_1} & \frac{\partial p_{f_{RH}}[2]}{\partial q_2} & \frac{\partial p_{f_{RH}}[2]}{\partial q_1 3} \\ \frac{\partial p_{f_{RH}}[3]}{\partial q_1} & \frac{\partial p_{f_{RH}}[3]}{\partial q_2} & \frac{\partial p_{f_{RH}}[3]}{\partial q_3} \end{bmatrix} \tag{4.10}$$

where:
$J_{1,1} = 0$
$J_{1,2} = -a_4 cos(q_3)cos(q_2) + a_4 sin(q_3)sin(q_2) - a_3 cos(q_2)$
$J_{1,3} = a_4 sin(q_3)sin(q_2) - a_4 cos(q_3)cos(q_2)$
$J_{2,1} = cos(q_1)cos(q_2)a_4 cos(q_3) - a_4 sin(q_3)cos(q_1)sin(q_2) + a_3 cos(q_1)cos(q_2) + a_2 cos(q_1)$
$J_{2,2} = -sin(q_1)sin(q_2)a_4 cos(q_3) - a_4 sin(q_3)sin(q_1)cos(q_2) - a_3 sin(q_1)sin(q_2)$
$J_{2,3} = -sin(q_1)cos(q_2)a_4 sin(q_3) - a_4 cos(q_3)sin(q_1)sin(q_2)$
$J_{3,1} = sin(q_1)cos(q_2)a_4 cos(q_3) - sin(q_1)sin(q_2)a_4 sin(q_3) + sin(q_1)a_3 cos(q_2) + a_2 sin(q_1)$
$J_{3,2} = cos(q_1)sin(q_2)a_4 cos(q_3) + cos(q_1)cos(q_2)a_4 sin(q_3) + cos(q_1)a_3 sin(q_2)$
$J_{3,3} = cos(q_1)cos(q_2)a_4 sin(q_3) + cos(q_1)sin(q_2)a_4 cos(q_3)$

For what concerns RH and LF knee,the matrices are simply found without considering the last joint q_3 and,as a consequence, computing the kinematics from the base to KFE:

$$p_{knee_{LF}} = \begin{bmatrix} -a_3 s(q_2) + d_1 \\ -a_3 s(q_1)c(q_2) - a_2 s(q_1) + d_0 \\ -c(q_1)a_3 c(q_2) - a_2 c(q_1) \end{bmatrix} \tag{4.11}$$

$$p_{knee_{RH}} = \begin{bmatrix} -a_3 s(q_2) - d_1 - \\ +a_3 s(q_1)c(q_2) + a_2 s(q_1) - d_0 \\ -c(q_1)a_3 c(q_2) - a_2 c(q_1) \end{bmatrix} \tag{4.12}$$

From the position vectors is then possible, as before, to obtain the Jacobian matrices:

$$J_{knee_{LF}} = \begin{bmatrix} \frac{\partial p_{k_{LF}}[1]}{\partial q_1} & \frac{\partial p_{k_{LF}}[1]}{\partial q_2} & \frac{\partial p_{k_{LF}}[1]}{\partial q_3} \\ \frac{\partial p_{k_{LF}}[2]}{\partial q_1} & \frac{\partial p_{k_{LF}}[2]}{\partial q_2} & \frac{\partial p_{k_{LF}}[2]}{\partial q_1 3} \\ \frac{\partial p_{k_{LF}}[3]}{\partial q_1} & \frac{\partial p_{k_{LF}}[3]}{\partial q_2} & \frac{\partial p_{k_{LF}}[3]}{\partial q_3} \end{bmatrix} \tag{4.13}$$

$$J_{knee_{RH}} = \begin{bmatrix} \frac{\partial p_{k_{RH}}[1]}{\partial q_1} & \frac{\partial p_{k_{RH}}[1]}{\partial q_2} & \frac{\partial p_{k_{RH}}[1]}{\partial q_3} \\ \frac{\partial p_{k_{RH}}[2]}{\partial q_1} & \frac{\partial p_{k_{RH}}[2]}{\partial q_2} & \frac{\partial p_{k_{RH}}[2]}{\partial q_1 3} \\ \frac{\partial p_{k_{RH}}[3]}{\partial q_1} & \frac{\partial p_{k_{RH}}[3]}{\partial q_2} & \frac{\partial p_{k_{RH}}[3]}{\partial q_3} \end{bmatrix} \tag{4.14}$$

As written before in the section 2.1.2,it is now possible to find the position in the world frame:

$$\begin{aligned} p^w_{f_{RH}} &= R^w_b \cdot p^b_{f_{RH}} + p^w_{base} \\ p^w_{f_{LF}} &= R^w_b \cdot p^b_{f_{LF}} + p^w_{base} \\ p^w_{k_{RH}} &= R^w_b \cdot p^b_{k_{RH}} + p^w_{base} \\ p^w_{k_{LF}} &= R^w_b \cdot p^b_{k_{LF}} + p^w_{base} \end{aligned} \tag{4.15}$$

where p_f^w is the position of the foot in the world frame, p_k^w is the position of the knee in the world frame and p_{base}^w is the position of the robot's base in the world frame. After finding the position the next step is to evaluate the velocity. The general formulas are expressed as:

$$\begin{aligned} v^b &= J \cdot \dot{q} - \omega_{base} \times p^b \\ v^w &= R_b^w \cdot v^b + v_{base}^w \end{aligned} \tag{4.16}$$

where $\dot{q}$ is the joint velocity and it is provided by the proprioceptive sensors mounted on the robot, v_{base}^w and ω_{base} are respectively the linear and angular velocity of the robot's base computed by the IMU (Chapter 3.1.3).From the equation 4.16,it is possible to evaluate the linear velocity of the knee and of the foot for RH/LF legs:

$$\begin{aligned} v_{f_{RH}}^w &= R_b^w \cdot (J_{f_{RH}} \cdot \dot{q} - \omega_{base} \times p_{f_{RH}}^b) + v_{base}^w \\ v_{f_{LF}}^w &= R_b^w \cdot (J_{f_{LF}} \cdot \dot{q} - \omega_{base} \times p_{f_{LF}}^b) + v_{base}^w \\ v_{k_{RH}}^w &= R_b^w \cdot (J_{k_{RH}} \cdot \dot{q} - \omega_{base} \times p_{k_{RH}}^b) + v_{base}^w \\ v_{k_{LF}}^w &= R_b^w \cdot (J_{k_{LF}} \cdot \dot{q} - \omega_{base} \times p_{k_{LF}}^b) + v_{base}^w \end{aligned} \tag{4.17}$$

To evaluate the angular velocity it is necessary to compute the angular Jacobian Jo. It is a $3xn$ matrix, where n depends on the number of joints involved in the evaluation of the differential kinematics,describing the relation between $\dot{q}$ and the angular velocity ω. For RH/LF leg it is computed as follows:

$$\begin{aligned} Jo_{f_{RH}} &= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos(q_1) & cos(q_1) \\ 0 & sin(q_1) & sin(q_1) \end{bmatrix} \\ Jo_{f_{LF}} &= \begin{bmatrix} -1 & 0 & 0 \\ 0 & cos(q_1) & cos(q_1) \\ 0 & -sin(q_1) & -sin(q_1) \end{bmatrix} \\ Jo_{k_{RH}} &= \begin{bmatrix} 1 & 0 \\ 0 & cos(q_1) \\ 0 & sin(q_1) \end{bmatrix} \\ Jo_{k_{LF}} &= \begin{bmatrix} -1 & 0 \\ 0 & cos(q_1) \\ 0 & -sin(q_1) \end{bmatrix} \end{aligned} \tag{4.18}$$

and, as a consequence, the angular velocity in the world frame is:

$$\begin{aligned} \omega_{f_{RH}}^w &= R_b^w \cdot (Jo_{f_{RH}} \cdot \dot{q}) + \omega_{base} \\ \omega_{f_{LF}}^w &= R_b^w \cdot (Jo_{f_{LF}} \cdot \dot{q}) + \omega_{base} \\ \omega_{k_{RH}}^w &= R_b^w \cdot (Jo_{k_{RH}} \cdot \dot{q}) + \omega_{base} \\ \omega_{k_{LF}}^w &= R_b^w \cdot (Jo_{k_{LF}} \cdot \dot{q}) + \omega_{base} \end{aligned} \tag{4.19}$$

The novel approach of this thesis is to consider the collision in a 3D plane. To do that it is necessary to compute also the acceleration and having as a consequence a whole description of the robot shin motion in the environment. The general formulas for acceleration are the following:

$$\begin{aligned} a^b &= (\dot{J} \cdot \dot{q} + J \cdot \ddot{q}) - S(\dot{\omega}) \cdot p^b - S(\omega) \cdot (J \cdot \dot{q}) \\ a^w &= a_{base}^w + S(\omega) \cdot R_b^w \cdot v^b + R_b^w \cdot a^b \end{aligned} \tag{4.20}$$

where $S(\omega)$ and $S(\dot{\omega})$ are the skew-symmetric matrices evaluated as follows:

$$S(\omega) = \begin{bmatrix} 0 & -\omega_3 & \omega_2 \\ \omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} S(\dot{\omega}) = \begin{bmatrix} 0 & -\dot{\omega}_3 & \dot{\omega}_2 \\ \dot{\omega}_3 & 0 & -\dot{\omega}_1 \\ -\dot{\omega}_2 & \dot{\omega}_1 & 0 \end{bmatrix} \tag{4.21}$$

The derivative by time of the Jacobian, $\dot{J}$, for LF/RH knee/foot is the following matrix:

$$\dot{J} = \begin{bmatrix} \frac{\partial J_{1,1}}{\partial t} & \frac{\partial J_{1,2}}{\partial t} & \frac{\partial J_{1,3}}{\partial t} \\ \frac{\partial J_{2,1}}{\partial t} & \frac{\partial J_{2,2}}{\partial t} & \frac{\partial J_{2,3}}{\partial t} \\ \frac{\partial J_{3,1}}{\partial t} & \frac{\partial J_{3,2}}{\partial t} & \frac{\partial J_{3,3}}{\partial t} \end{bmatrix} \tag{4.22}$$

The last elements, needed for building the model to find the contact point, are the *shin angle* and the *shin angle rate*. They are related to the shin orientation and as a consequence they can be obtained directly from the rotation matrices (eq. 4.5 and 4.6), using the inverse formulas of the Euler angles (eq. A.6).The first step is to find the Rotation matrices,from the robot base to the foot, in the world frame

$$\begin{aligned} R_e &= R_b^w \cdot R \\ R'_e &= R_b^w \cdot R' \end{aligned} \tag{4.23}$$

The second step is to evaluate the shin angle:

$$\begin{aligned} [\phi_{LF}, \theta_{LF}, \psi_{LF}] &= \begin{bmatrix} \phi(R_e) \\ \theta(R_e) \\ \psi(R_e) \end{bmatrix} = \begin{bmatrix} atan2(R_{e2,3}, R_{e_{3,3}}) \\ -asin(R_{e_{1,3}}) \\ atan2(R_{e_{1,2}}, R_{e_{1,1}}) \end{bmatrix} \\ [\phi_{RH}, \theta_{RH}, \psi_{RH}] &= \begin{bmatrix} \phi(R'_e) \\ \theta(R'_e) \\ \psi(R'_e) \end{bmatrix} = \begin{bmatrix} atan2(R'_{e2,3}, R'_{e_{3,3}}) \\ -asin(R'_{e_{1,3}}) \\ atan2(R'_{e_{1,2}}, R'_{e_{1,1}}) \end{bmatrix} \end{aligned} \tag{4.24}$$

From these equations, the shin angle rate is computed as follows:

$$[\dot{\phi}, \dot{\theta}, \dot{\psi}] = E^{-1} \cdot \omega_f^w \tag{4.25}$$

where:

$$E^{-1} = \frac{1}{cos(\theta)} \begin{bmatrix} cos(\psi) & \sin(\psi) & 0 \\ -cos(\theta)sin(\psi) & cos(\theta)cos(\psi) & 0 \\ cos(\psi)sin(\theta) & sin(\psi)sin(\theta) & cos(\theta) \end{bmatrix} \tag{4.26}$$

Then substituting the equations (4.19, 4.24) in (4.25), it is obtained that:

$$\begin{aligned} [\dot{\phi}_{LF}, \dot{\theta}_{LF}, \dot{\psi}_{LF}] &= E^{-1} \cdot \omega_{f_{LF}}^w \\ [\dot{\phi}_{RH}, \dot{\theta}_{RH}, \dot{\psi}_{RH}] &= E^{-1} \cdot \omega_{f_{RH}}^w \end{aligned} \tag{4.27}$$

The derivatives of 4.27 are the following equations:

$$\begin{aligned} \begin{bmatrix} \ddot{\phi}_{LF} \\ \ddot{\theta}_{LF} \\ \ddot{\psi}_{LF} \end{bmatrix} &= \dot{E}^{-1} \cdot \omega_{f_{LF}}^w + E^{-1} \cdot [S(\omega) \cdot R_b^w \cdot \omega_{f_{LF}} + R_b^w \cdot (\dot{Jo} \cdot \dot{q} + Jo \cdot \ddot{q}) + \dot{\omega}_{base}^w] \\ \begin{bmatrix} \ddot{\phi}_{RH} \\ \ddot{\theta}_{RH} \\ \ddot{\psi}_{RH} \end{bmatrix} &= \dot{E}^{-1} \cdot \omega_{f_{RH}}^w + E^{-1} \cdot [S(\omega) \cdot R_b^w \cdot \omega_{f_{RH}} + R_b^w \cdot (\dot{Jo} \cdot \dot{q} + Jo \cdot \ddot{q}) + \dot{\omega}_{base}^w] \end{aligned} \tag{4.28}$$

where:

$$\dot{E}^{-1} = \frac{sin(\theta)}{cos(\theta)^2}E_1 + \frac{1}{cos(\theta)}E_2$$

$$E_1 = \begin{bmatrix} cos(\psi) & sin(\psi) & 0 \\ -cos(\theta)sin(\psi) & cos(\theta)cos(\psi) & 0 \\ cos(\psi)sin(\theta) & sin(\psi)sin(\theta) & cos(\theta) \end{bmatrix}$$

$$E_2 = \begin{bmatrix} -sin(\psi)\dot{\psi} & cos(\psi)\dot{\psi} & 0 \\ sin(\theta)\dot{\theta}sin(\psi) - cos(\theta)cos(\psi)\dot{\psi} & -sin(\theta)\dot{\theta}cos(\psi) - cos(\theta)sin(\psi)\dot{\psi} & 0 \\ -sin(\psi)\dot{\psi}sin(\theta) + cos(\psi)cos(\theta)\dot{\theta} & cos(\psi)\dot{\psi}sin(\theta) + sin(\psi)cos(\theta)\dot{\theta} & -sin(\theta)\dot{\theta} \end{bmatrix} \tag{4.29}$$

In order to verify if all the computations evaluated till now are consistent with the reality, the robot is simulated in Real-Time on Gazebo. The outputs are analyzed in Real-Time using the *Scope* tool and datas are then processed on Matlab to study in details the behaviour during the time. Their shape is shown and described in the Appendix B.

After computing all the parameters needed for building the system, it is now possible to evaluate the contact point in a 3D plane using Trigonometric Equations.

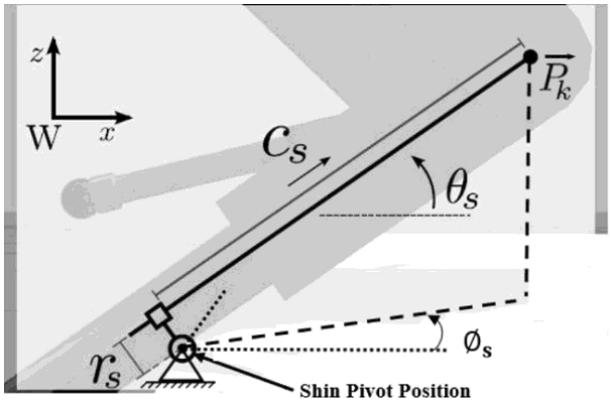


Figure 4.3: Representation of the shin collision in a 3D plane

Considering the collisions as a distance along the shin, from the knee joint, and taking into account that θ is the *pitch* while ϕ is the *yaw* of the shin (Fig. 4.3), the following equations are obtained for the position:

$$\begin{aligned} P_{kx_{LF}} &= -[C_s cos(\theta_{LF}) - r_s sin(\theta_{LF})]cos(\phi_{LF}) \\ P_{ky_{LF}} &= -[C_s cos(\theta_{LF}) - r_s sin(\theta_{LF})]sin(\phi_{LF}) \\ P_{kz_{LF}} &= C_s sin(\theta_{LF}) + r_s cos(\theta_{LF}) \end{aligned} \tag{4.30}$$

$$\begin{aligned} P_{kx_{RH}} &= [C_s cos(\theta_{RH}) - r_s sin(\theta_{RH})]cos(\phi_{RH}) \\ P_{ky_{RH}} &= [C_s cos(\theta_{RH}) - r_s sin(\theta_{RH})]sin(\phi_{RH}) \\ P_{kz_{RH}} &= -C_s sin(\theta_{RH}) + r_s cos(\theta_{RH}) \end{aligned} \tag{4.31}$$

In reference to the figure Fig. 4.3, the unknown parameter is C_s that is the distance between the knee and the point in which the collision happens. Moreover, since it is a model in a 3D plane, it is necessary to consider also the thickness of the leg $r_s = 0.02$ [cm]. The idea at the basis of contact detection, is that the collision happens when the velocity and acceleration are equal to zero. This means that the movement of the robot leg is stuck due to the presence of an obstacle. Therefore the evaluation of contact point velocity and acceleration is a fundamental step for the analysis of the model in a 3D plane.
Doing the derivative by time of equations (4.30) and (4.31), the velocities are evaluated as follows:

$$\begin{aligned} \dot{P}'_{kx_{LF}} &= -\dot{C}_s[cos(\theta_{LF})cos(\phi_{LF})] + C_s[sin(\theta_{LF})\dot{\theta}_{LF}cos(\phi_{LF}) + cos(\theta_{LF})sin(\phi_{LF})\dot{\phi}_{LF}] \\ \dot{P}'_{ky_{LF}} &= -\dot{C}_s[cos(\theta_{LF})sin(\phi_{LF})] + C_s[sin(\theta_{LF})\dot{\theta}_{LF}sin(\phi_{LF}) - cos(\theta_{LF})cos(\phi_{LF})\dot{\phi}_{LF}] \\ \dot{P}_{kz_{LF}} &= \dot{C}_s sin(\theta_{LF}) + C_s cos(\theta_{LF})\dot{\theta}_{LF} \end{aligned} \tag{4.32}$$

$$\begin{aligned} P'_{kx_{RH}} &= \dot{C}_s[cos(\theta_{RH})cos(\phi_{RH})] + C_s[-sin(\theta_{RH})\dot{\theta}_{RH}cos(\phi_{RH}) - cos(\theta_{RH})sin(\phi_{RH})\dot{\phi}_{RH}] \\ \dot{P}'_{ky_{RH}} &= \dot{C}_s[cos(\theta_{RH})sin(\phi_{RH})] + C_s[-sin(\theta_{RH})\dot{\theta}_{RH}sin(\phi_{RH}) + cos(\theta_{RH})cos(\phi_{RH})\dot{\phi}_{RH}] \\ \dot{P}'_{kz_{RH}} &= \dot{C}_s sin(\theta_{RH}) + C_s cos(\theta_{RH})\dot{\theta}_{RH} \end{aligned} \tag{4.33}$$

where for simplicity in calculations $\dot{P}'_{kx_{LF}}$,$\dot{P}'_{ky_{LF}}$,$\dot{P}'_{kz_{LF}}$,$\dot{P}'_{kx_{RH}}$,$\dot{P}'_{ky_{RH}}$,$\dot{P}'_{kz_{RH}}$ are equal to:

$$\begin{aligned} \dot{P}'_{kx_{LF}} &= \dot{P}_{kx_{LF}} - r_s cos(\theta_{LF})\dot{\theta}_{LF}cos(\phi_{LF}) + r_s sin(\theta_{LF})sin(\phi_{LF})\dot{\phi}_{LF} \\ \dot{P}'_{ky_{LF}} &= \dot{P}_{ky_{LF}} - r_s cos(\theta_{LF})\dot{\theta}_{LF}sin(\phi_{LF}) - r_s sin(\theta_{LF})cos(\phi_{LF})\dot{\phi}_{LF} \\ \dot{P}'_{kz_{LF}} &= \dot{P}_{kz_{LF}} + r_s sin(\theta_{LF})\dot{\theta}_{LF} \end{aligned} \tag{4.34}$$

$$\begin{aligned} \dot{P}'_{kx_{RH}} &= \dot{P}_{kx_{RH}} + r_s cos(\theta_{RH})\dot{\theta}_{RH}cos(\phi_{RH}) - r_s sin(\theta_{RH})sin(\phi_{RH})\dot{\phi}_{RH} \\ \dot{P}'_{ky_{RH}} &= \dot{P}_{ky_{RH}} + r_s cos(\theta_{RH})\dot{\theta}_{RH}sin(\phi_{RH}) + r_s sin(\theta_{RH})cos(\phi_{RH})\dot{\phi}_{RH} \\ \dot{P}'_{kz_{RH}} &= \dot{P}_{kz_{RH}} + r_s sin(\theta_{RH})\dot{\theta}_{RH} \end{aligned} \tag{4.35}$$

Deriving by time the equations (4.32) and (4.33),it is possible to compute also the accelerations:

$$\begin{aligned} \ddot{P}'_{kx_{LF}} &= -a_{LF} \cdot \ddot{C}_s + 2b_{LF} \cdot \dot{C}_s + \dot{b}_{LF} \cdot C_s \\ \ddot{P}'_{ky_{LF}} &= -c_{LF} \cdot \ddot{C}_s + 2d_{LF} \cdot \dot{C}_s + \dot{d}_{LF} \cdot C_s \\ \ddot{P}'_{kz_{LF}} &= e_{LF} \cdot \ddot{C}_s + 2f_{LF} \cdot \dot{C}_s + \dot{f}_{LF} \cdot C_s \end{aligned} \tag{4.36}$$

$$\begin{aligned} \ddot{P}'_{kx_{RH}} &= a_{RH} \cdot \ddot{C}_s - 2b_{RH} \cdot \dot{C}_s - \dot{b}_{RH} \cdot C_s \\ \ddot{P}'_{ky_{RH}} &= c_{RH} \cdot \ddot{C}_s - 2d_{RH} \cdot \dot{C}_s - \dot{d}_{RH} \cdot C_s \\ \ddot{P}'_{kz_{RH}} &= e_{RH} \cdot \ddot{C}_s + 2f_{RH} \cdot \dot{C}_s + \dot{f}_{RH} \cdot C_s \end{aligned} \tag{4.37}$$

For the sake of readable, in the above formulas are used coefficients which are equal to:

$$
\begin{aligned}
a &= cos(\theta)cos(\phi) \\
b &= sin(\theta)\dot{\theta}cos(\phi) + cos(\theta)sin(\phi)\dot{\phi} \\
c &= cos(\theta)sin(\phi) \\
d &= sin(\theta)\dot{\theta}sin(\phi) - cos(\theta)cos(\phi)\dot{\phi} \\
e &= sin(\theta) \\
f &= cos(\theta)\dot{\theta} \\
\dot{b} &= cos(\theta)\dot{\theta}^2cos(\phi) + sin(\theta)\ddot{\theta}cos(\phi) - sin(\theta)\dot{\theta}sin(\phi)\dot{\phi} - sin(\theta)\dot{\theta}sin(\phi)\dot{\phi} + \\
&\quad + cos(\theta)cos(\phi)\dot{\phi}^2 + cos(\theta)sin(\phi)\ddot{\phi} \\
\dot{d} &= cos(\theta)\dot{\theta}^2sin(\phi) + sin(\theta)\ddot{\theta}sin(\phi) + sin(\theta)\dot{\theta}cos(\phi)\dot{\phi} + sin(\theta)\dot{\theta}cos(\phi)\dot{\phi} + \\
&\quad + cos(\theta)sin(\phi)\dot{\phi}^2 - cos(\theta)cos(\phi)\ddot{\phi}
\end{aligned}
$$

The parameters $\ddot{P}'_{kx_{LF}}$,$\ddot{P}'_{ky_{LF}}$,$\ddot{P}'_{kz_{LF}}$,$\ddot{P}'_{kx_{RH}}$,$\ddot{P}'_{ky_{RH}}$,$\ddot{P}'_{kz_{RH}}$ are obtained just deriving by time the equations (4.34 and 4.35).

The purpose of this thesis is to evaluate and estimate the contact point C_s and the contact velocity $\dot{C}_s$. From the equations written above it is necessary to compute these two parameters, in order to use them as measured inputs, $\tilde{C}_s$ and $\dot{\tilde{C}}_s$, and then estimate $\hat{C}_s$ and $\dot{\hat{C}}_s$ during the phase of filtering using the Kalman Filter method. The extension in a 3D model is useful to build two models for the detection of the contact point:

- *velocity model*. It is more precise and accurate for the contact detection because it has a lower complexity in the implementation. When the velocity is about zero the collision is happening. As a consequence C_s is a constant value corresponding to the distance from the knee.
- *acceleration model*.In practice, it is less accurate than the velocity model, due to its complexity in the equations. Moreover for this model it is necessary to compute the discrete derivative of the angular velocity, that is very noisy and more sensitive to phase shifts. As a consequence this model is used to avoid singularities when in the *velocity model*, the velocity is equal to zero. It is, in fact, guaranteed that when *velocity model* is in singularity, the *acceleration model* provides a better output.

From the equations (4.32 and 4.33), the *velocity model* is obtained as follows for LF leg and for RH leg respectively:

$$
\begin{gathered}
\dot{P}_{LF} = \begin{bmatrix} \dot{P}_{kx_{LF}} \\ \dot{P}_{ky_{LF}} \\ \dot{P}_{kz_{LF}} \end{bmatrix} = \begin{bmatrix} b & -a \\ d & -c \\ f & e \end{bmatrix} \begin{bmatrix} \tilde{C}_s \\ \dot{\tilde{C}}_s \end{bmatrix} \\
\text{inverting the formula it is obtained that} \\
\begin{bmatrix} \tilde{C}_s \\ \dot{\tilde{C}}_s \end{bmatrix} = J^+_{LF} \cdot \dot{P}_{LF}
\end{gathered}
\tag{4.38}
$$

$$\dot{P}_{RH} = \begin{bmatrix} \dot{P}_{kx_{RH}} \\ \dot{P}_{ky_{RH}} \\ \dot{P}_{kz_{RH}} \end{bmatrix} = \begin{bmatrix} -b & a \\ -d & c \\ f & e \end{bmatrix} \begin{bmatrix} \tilde{C}_s \\ \dot{\tilde{C}}_s \end{bmatrix}$$

inverting the formula it is obtained that (4.39)

$$\begin{bmatrix} \tilde{C}_s \\ \dot{\tilde{C}}_s \end{bmatrix} = J^{+}_{RH} \cdot \dot{P}_{RH}$$

where J^{+} is the Moore-Penrose inverse also known as *pseudo-inverse* matrix [14] [30]. For what concerns the *acceleration model* it is computed as follows:

$$\ddot{P}_{LF} = \begin{bmatrix} \ddot{P}_{kx_{LF}} \\ \ddot{P}_{ky_{LF}} \\ \ddot{P}_{kz_{LF}} \end{bmatrix} = \begin{bmatrix} \dot{b} & 2b & -a \\ \dot{d} & 2d & -c \\ \dot{f} & 2f & e \end{bmatrix} \begin{bmatrix} \tilde{C}_s \\ \dot{\tilde{C}}_s \\ \ddot{\tilde{C}}_s \end{bmatrix}$$

inverting the formula it is obtained that (4.40)

$$\begin{bmatrix} \tilde{C}_s \\ \dot{\tilde{C}}_s \\ \ddot{\tilde{C}}_s \end{bmatrix} = J^{-1}_{LF} \cdot \ddot{P}_{LF}$$

and for the RH leg:

$$\ddot{P}_{RH} = \begin{bmatrix} \ddot{P}_{kx_{RH}} \\ \ddot{P}_{ky_{RH}} \\ \ddot{P}_{kz_{RH}} \end{bmatrix} = \begin{bmatrix} -\dot{b} & 2b & a \\ -\dot{d} & 2d & c \\ \dot{f} & 2f & e \end{bmatrix} \begin{bmatrix} \tilde{C}_s \\ \dot{\tilde{C}}_s \\ \ddot{\tilde{C}}_s \end{bmatrix}$$

inverting the formula it is obtained that (4.41)

$$\begin{bmatrix} \tilde{C}_s \\ \dot{\tilde{C}}_s \\ \ddot{\tilde{C}}_s \end{bmatrix} = J^{-1}_{RH} \cdot \ddot{P}_{RH}$$

All the elements for building the dynamics of the Kalman Filter are now computed. Its implementation is described properly in the following section.

4.2 Filtering

As explained in Chapter 2.1, State Estimation is characterized by two main steps: i) the model development,analyzed in Chapter 4.1.2 and ii) the phase of filtering. The former is used to describe a robot's behaviour using mathematical equations, the latter has the role to improve the results obtained analytically, providing an *estimation* of the desired variables with certain features. In this thesis the technique adopted to filtered out the signal is the Kalman Filter, widely described in Chapter 2.2.4. Since the model is non linear time-variant, the EKF framework is adopted for the phase of Filtering. The main goal is to estimate the mean and the covariance of the Gaussian distribution over the state $x_k = [C_s, \dot{C}_s]^T$ at time k.
As it can be seen in Fig. 4.4, the signal is coming from sensors:

- the IMU is adopted to then estimate the signal related to the linear/angular velocity/acceleration of the robot's base in the world frame: ω_{base}, $\dot{\omega}_{base}$, v^w_{base} and a^w_{base}.

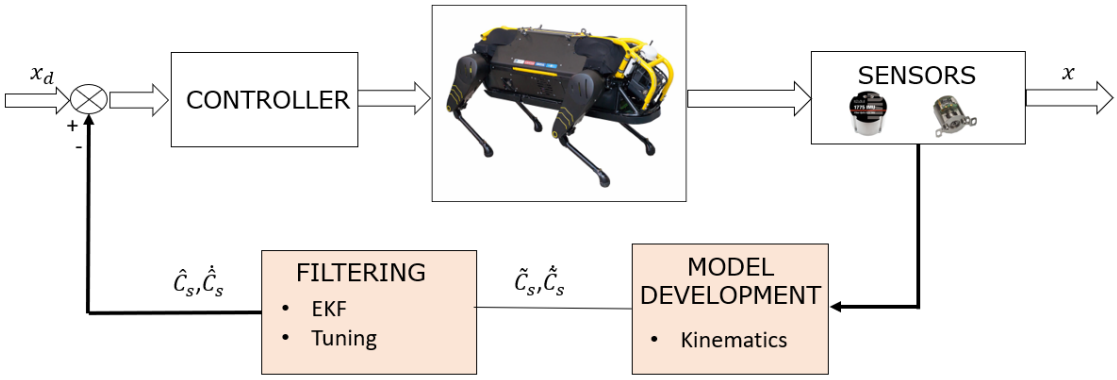


Figure 4.4: Scheme block of the shin collision state estimation model

- the encoders are used to acquire informations about the joints position and velocity: q, $\dot{q}$

All these signals are corrupted by noise, this is way the usage of the Kalman Filter is fundamental to obtain a good estimation of the state x. To build this filter the following elements are necessary:

- inputs: they correspond to knee velocity/acceleration $\dot{P}_k$, $\ddot{P}_k$ and shin orientation θ, ϕ, $\dot{\theta}$, $\dot{\phi}$, $\ddot{\theta}$, $\ddot{\phi}$.
- the measured state indicated as z: it corresponds to

$$\tilde{x} = \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} = \begin{bmatrix} \tilde{C}_s \\ \dot{\tilde{C}}_s \end{bmatrix} \tag{4.42}$$

  So it is equal to the measured state computed using the *velocity model* obtained in 4.39.

- the dynamics of the filter

$$\dot{\hat{x}} = \begin{bmatrix} \dot{\hat{x}_1} \\ \dot{\hat{x}_2} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} \tilde{x}_1 \\ \tilde{x}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \ddot{\tilde{C}}_s \end{bmatrix} \tag{4.43}$$

  In this formula the parameter $\ddot{\tilde{C}}_s$ is computed using the *acceleration model*. This is why the EKF is a combined model in which both velocity and acceleration are involved to estimate the contact point.

- the covariance matrices Q and R
- the equations of the *hybrid model* EKF

In the section 2.2.6, are explained the continuous-time EKF and the discrete-time EKF. Considering the two models, the *hybrid* version can be easily obtained. Its application is used when the dynamics is in continuous-time, while the measurements are taken at discrete time instant from the sensors. Considering this situation, that it's very common in real application, the following equations are obtained:

$$\begin{aligned} \dot{x} &= f(x, u, w, t) \\ y_k &= h_k(x_k, v_k) \\ w(t) &\sim (0, Q) \\ v_k &\sim (0, R_k) \end{aligned} \tag{4.44}$$

The process noise $w(t)$ has the covariance Q and it is in continuous-time, while the measurement noise v_k with covariance R_k is in discrete-time.
Then the filter is initialized as follows:

$$\begin{aligned} \hat{x}_0^+ &= E[x_0] \\ P_0^+ &= E[(x_0 - \hat{x}_0^+)(x_0 - \hat{x}_0^+)^T] \end{aligned} \tag{4.45}$$

The first step is to compute $\dot{\hat{x}}$ and $\dot{P}$, using the equations in the continuous-time Kalman filter (2.68) and (2.69):

$$\begin{aligned} \dot{\hat{x}} &= f(\hat{x}, u, w_0, t) \\ \dot{P} &= AP + PA^T + LQL^T \end{aligned} \tag{4.46}$$

at the end of the integration it is obtained that $\hat{x} = \hat{x}_k^-$ and $P = P_k^-$. The updating of the state-estimate and the covariance, following the equations of the discrete-time Kalman Filter (2.74),is computed as:

$$\begin{aligned} K_k &= P_k^- H_k^T (H_k P_k^- H_k^T + M_k R_k M_k^T)^{-1} \\ \hat{x}_k^+ &= \hat{x}_k^- + K_k[y_k - h_k(\hat{x}_k^-, v_0, t_k)] \\ P_k^+ &= (I - K_k H_k) P_k^- (I - K_k H_k)^T + K_k M_k R_k M_k^T K_k^T \end{aligned} \tag{4.47}$$

The above equations can be now applied to the contact detection model,obtaining in the C++ code:

Table 4.1: Hybrid model EKF algorithm .

1: **predict**$(t, u)\{$
2: $\quad dt = (t - t_{prev})/0.004$
3: $\quad xdot = calc\_f(t, xhat, u)$
4: $\quad F = calc\_F(t, xhat, u)$
5: $\quad Pdot = A \cdot P + P \cdot A^T + LQL^T$
6: $\quad xhat = xhat + xdot \cdot dt$
7: $\quad P = P + Pdot \cdot dt$
8: $\quad t_{prev} = t\}$
9: **update**$(t, z)\{$
10: $\quad H = calc\_H(t, xhat)$
11: $\quad h = calc\_h(t, xhat)$
12: $\quad K_k = P_k^- H_k^T (H_k P_k^- H_k^T + M_k R_k M_k^T)^{-1}$
13: $\quad xhat = xhat + K \cdot (z - h)$
14: $\quad P = (I - K \cdot H) \cdot P\}$

---

where *calc_f* is equal to the dynamics of the filter (4.43):

$$\begin{aligned} \textbf{calc\_f}(t, x, u)\{ \\ x1dot &= x(2) \\ x2dot &= \ddot{C}\} \end{aligned} \tag{4.48}$$

then *calc_F* is the partial derivative of the EKF around the state estimate $\hat{x}$:

$$\begin{aligned} \textbf{calc\_F}(t, x, u)\{ \\ F &= [0, 1, 0, 0]\} \end{aligned} \tag{4.49}$$

In the **update** function, *calc_h* is used to compute *xhat* tha correponds to the output y_k (2.70), while *calc_H* is the partial derivative of h around the state estimate $\hat{x}$:

$$\begin{aligned} &\textbf{calc\_H}(t, x)\{ \\ &\qquad H = [1, 0, 0, 1]\} \end{aligned} \tag{4.50}$$

The idea of the Extended Kalman Filter is to remove the noise in the signal and obtain as a consequence a flat output during the collision, corresponding to the distance between the knee and the contact point. To avoid peaks and change also the shape of the estimation, the tuning of the parameters Q and R is performed. This is an important step during the filtering phase, because it allows to obtain the desired output with certain features.
The tuning of the parameters is explained as follows:

- Q is the covariance matrix of the *process noise*. It represents the reliability of the dynamics. A small value of Q means that the dynamics is reliable, while increasing too much Q means that the model are uncertain. This happens because K is a function of Q, so when the process noise increases (i.e. Q increases), P^- converges to a larger value and also K converges to a larger steady-state value, making as a consequence the filter less confident to the model.In summary, if Q is for assumption equal to zero, the measurements are completely ignored and the system trusts in the model. In this thesis Q is as follows:

$$Q = \begin{bmatrix} Q_{11} & Q_{12} \\ Q_{21} & Q_{22} \end{bmatrix} \tag{4.51}$$

  where Q_{11} is equal to a small costant value as 0.01 while Q_{22} is a function that depends on the acceleration. This is because on a covariance matrix, the values along the diagonal corresponds to the variance of each state, since w(a,b)$\sim$ Q where a=0 is the mean and b is the variance. In this case Q_{22} is the variance of $\dot{C}$,it is in fact referred to the state $\dot{x}_2 = \ddot{C}$, computed using the acceleration model 4.40.When the acceleration is closed to zero, Q reaches its maximum value, while for large values of the acceleration, Q is equal to its minimum value. This means that for acceleration closed to zero,the *acceleration model* is zero, providing a singularity. This is the reason why Q is larger for acceleration closed to zero, in this way the filter doesn't trust in $\hat{\dot{x}}_2 = \ddot{C}$. To obtain this shape, Q_{22} is equal to a saturation function as described in the following formula:

$$\begin{aligned} f_{sat} &= -\frac{L}{1 + exp(-k \cdot (f_{acc} - x_0))} + L + min \\ &\text{where:} \\ &L = max - min \\ &f_{acc} \text{ is a function that depends on the acceleration} \\ &x_0 \text{ is the inversion point of the saturation function} \\ &k \text{ is the slope of the saturation function} \end{aligned} \tag{4.52}$$

  For this phase it is important to change the parameters *min*, *max* x_0 and k according to the desired performance and the presence of noise in the measured signal.

- R is the covariance matrix of the *sensor noise* and as a consequence it is related to the measured signal. Decreasing the value of R means that the measurements are reliable, while increasing R means that the measured signal is corrupted by noise. In summary a large

measurements noise (i.e large R), implies that K decreases, so as a consequence the filter has less confidence in the measurements. The covariance matrix is the following:

$$R = \begin{bmatrix} R_{11} & R_{12} \\ R_{21} & R_{22} \end{bmatrix} \tag{4.53}$$

where R_{11} is equal to the saturation function, while $R_{22} = f_{sat}10$.For R matrix, the first element along the diagonal (i.e. R_{11}) is referred to the state $\dot{\hat{x_1}} = \bar{x_2}$, where $\bar{x_2}$ is the measured signal $\dot{\bar{C}}$. The idea is to have a big covariance R for small values of the velocity. It implies that for a velocity closed to zero, the system doesn't trust in the measurements because there is a singularity. In this case, the acceleration is different from zero and the filter is more confident in the model. The equation of the saturation function is the following:

$$\begin{aligned} f_{sat} &= -\frac{L}{1 + exp(-k \cdot (f_{vel} - x_0))} + L + min \\ &\text{where:} \\ &L = max - min \\ &f_{vel} \text{ is a function that depends on the velocity of the knee} \\ &x_0 \text{ is the inversion point of the saturation function} \\ &k \text{ is the slope of the saturation function} \end{aligned} \tag{4.54}$$

- P increases when Q and R increase. So a large noise in the system model (i.e. Q) or in the measurements (i.e. R), implies that the system is less confident in the state estimate $\hat{x}$.

The phase of tuning is performed in the best way building the model on Matlab and changing the parameters directly on the model,avoiding to do tests on the hardware that could damage the robot. As a consequence all the datas obtained during the simulation are then collected and post-processed, analyzing the variation of the output at every change of the parameters.
After this step, in which the desired output is analyzed accurately, the new and correct parameters for the Kalman Filter are ready to be uploaded directly on the robot.

Chapter 5

Simulation and Experimental results

To analyze if the behaviour of the robot is consistent with the model built in the Chapter 2.1.2, the simulation is a fundamental step for checking and testing the system and then implement it on the hardware. The two phases of simulation and experimental validation are performed thanks to the strict relation between the controller and the system built for contact point estimation. The controller,in fact, has the role to compute the reference for the low-level variables of the robot, as joint torques and joint positions, and track them.
The target platform for doing experiments is the HyQGreen Robot of Istituto Italiano di Tecnologia in Genova, a 90 Kg quadruped legged robot, characterized by 12 joints (3 for each leg),powered by hydraulic actuators (Fig.5.1). The controller implemented on HyQ and used for the purpose of this thesis,is the RCF controller, which is fully described in [9]. This type of controller is specifically designed for coping with uneven and irregular terrains, errors in trajectory tracking and poor state estimation. The final goal of this thesis is to improve the robot locomotion, using the estimation of the contact point for stabilizing the trunk controller.
The communication between the controller and the computed model is performed by the usage of ROS environment. A brief introduction on its working principle and structure is described in the following section. Then the last sections are devoted to the presentations and descriptions of the obtained results in experiments.

5.1 HyQ locomotion

As said in the previous chapters, the capability of HyQ robot is to help human and perform tasks in difficult scenarios, in which the terrain is non-flat and characterized by obstacles that could destabilize the robot. The aim of the researchers at the DLS lab is to implement controllers, based on trajectory planning or on state estimation, that can improve the stability and also the motion of HyQ. In particular, the goal of this thesis, is to detect the contact point during collision allowing the robot to overcome the obstacle,keeping its balance. The locomotion of HyQ can be of two types: the trot and the crawl.The former is a particular motion, used in this thesis, in which two opposite legs are in swing phase, while the others two are touching the ground. The latter is a slower motion in which only one leg is in swing phase,while the remaining legs are touching the ground. This technique provide more stability to the robot, this is why it is mainly used to overcome stairs.

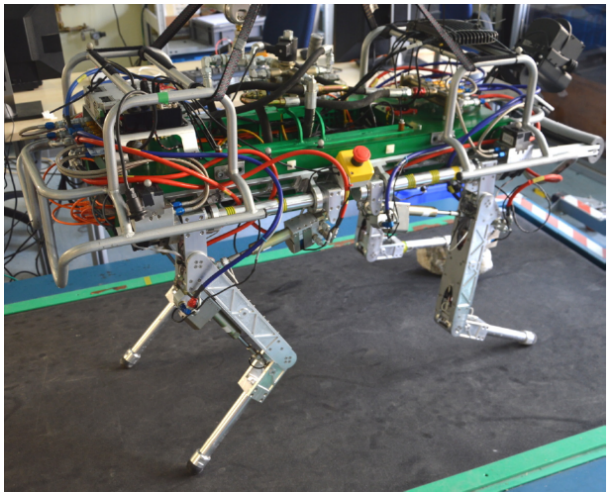


Figure 5.1: HyQ Green robot - Target platform of experiments

The first step in order to analyze the correct locomotion of the robot is to study the contact point between the foot of HyQ and the ground. In this way, the quadruped robot, is able to walk in difficult terrains, checking if the collision happens properly and then use this information to stabilize the trunk controller. Since the contact point is the distance from the knee,during the walking this value corresponds to the length of the lower leg: 0.356 [m].

5.1.1 Simulation

1. **PushUp motion** The first scenario analyzed inside Gazebo is the PushUp motion performed by the quadruped robot. This step is then fundamental for the experiment phase in order to check if the filter is working properly on the robot, acquiring the correct informations from the sensors mounted on it.
   The idea of using the Kalman Filter is to have an output that follows the measured input C, but when the measured signal goes to infinity, the estimated one is cutted in order to obtain a convergence on the value 0.356. The pushUp motion shown in figure Fig. 5.2 is performed with an Amplitude of 0.05 and a frequency of 0.5 Hz. As it can be possible to see in the figure, *velocity model* depends on the variation of the velocity, while the yellow line (*acceleration model*), follows the acceleration. To cut off the peaks to infinity, the covariance matrix R is large when the velocity is closed to zero, so as a consequence, the estimated output doesn't trust in the measurements. The tuning of the parameters for the PushUp is shown in the table 5.1.

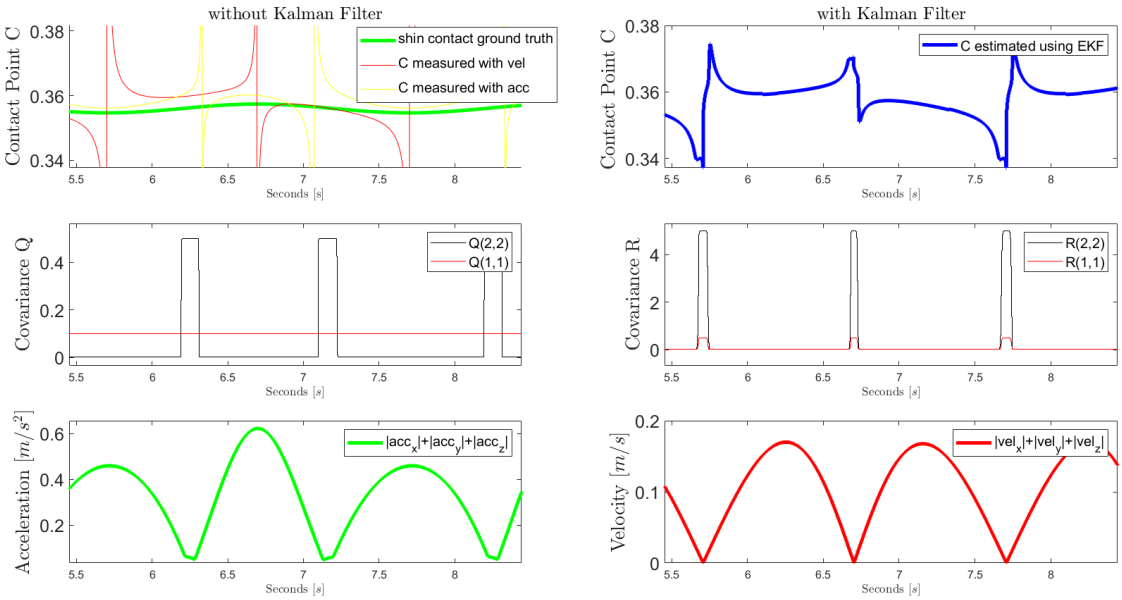


Figure 5.2: Estimation of C during the pushUp motion in Simulation with Amplitude of 0.05 and frequency of 0.5 Hz

Table 5.1: Table of tuning Parameters for Kalman Filter during PushUp motion

parameters	R	Q
min	0.0002 m^2	0.001 m^2/s^2
max	0.5 m^2	0.5 m^2/s^2
k	100	100
x0	0.015 m^2	0.1m^2/s^2

2. **TROT motion** To see better if the collision happens, the simulation are performed considering the robot moving forward with a velocity of 0.3 m/s. In this case the estimation is called *foot collision detection.*As it can be seen in the figure Fig. 5.3, the peaks generated at the collision moment (see 5.00[s]), are completely removed, obtaining a flat output during all the contact period. The velocity and the acceleration are closed to zero and their variation during the time is necessary for the tuning of Q and R parameters. Their values are shown in the table 5.2.

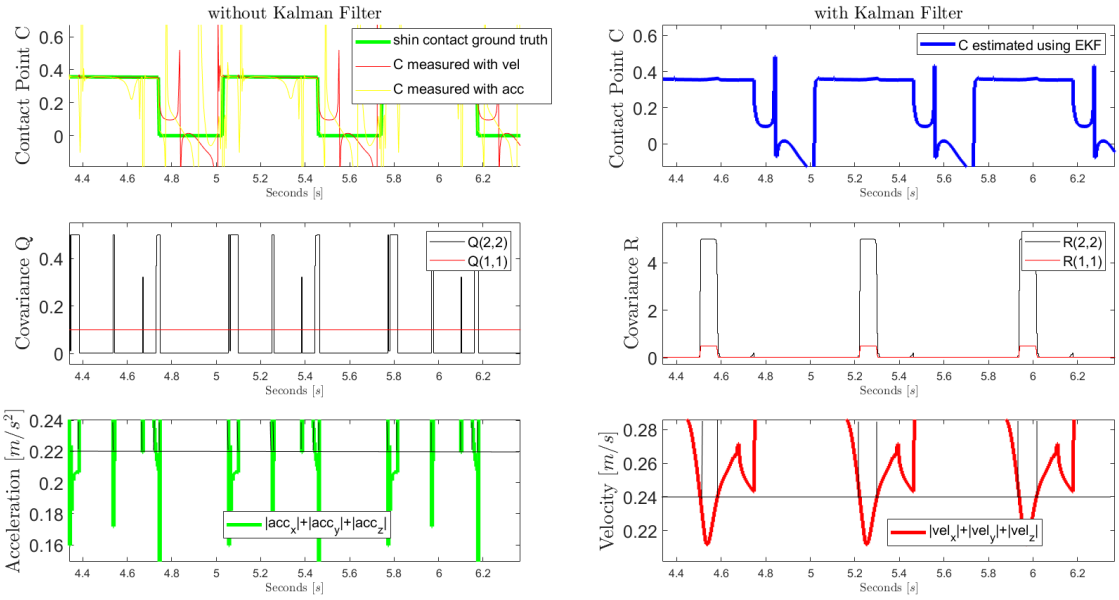


Figure 5.3: Estimation of C during trotting with a velocity of 0.3 m/s

Table 5.2: Table of tuning Parameters for Kalman Filter during trot motion in Simulation

parameters	R	Q
min	$0.0002m^2$	$0.001\ m^2/s^2$
max	$0.5m^2$	$0.5m^2/s^2$
k	100	100
x0	$0.24m^2$	$0.22m^2/s^2$

5.1.2 Experiments

The experiments are performed after the sanity check, guaranteeing the safety to all the operators.The crane is used to avoid that the robot drops down when it enters in contact with obstacles. The first step during experiments is to turn on the hydraulic pump, in order to have a pressure of 160 bar inside the valves of the robot. After the initialization phase in which all the forces are compensated with the command **goTau0**,removing all the offsets, the robot is moved down using the crane to have all the feet touching the ground. After activating the State Estimator the robot is ready for going forward and backward. For the experiments, the joystick is used to have more control on the robot motion.

The first experiment done is the PushUp motion, used to check the correct behaviour of velocity, acceleration and the working principle of the Kalman Filter 5.4.

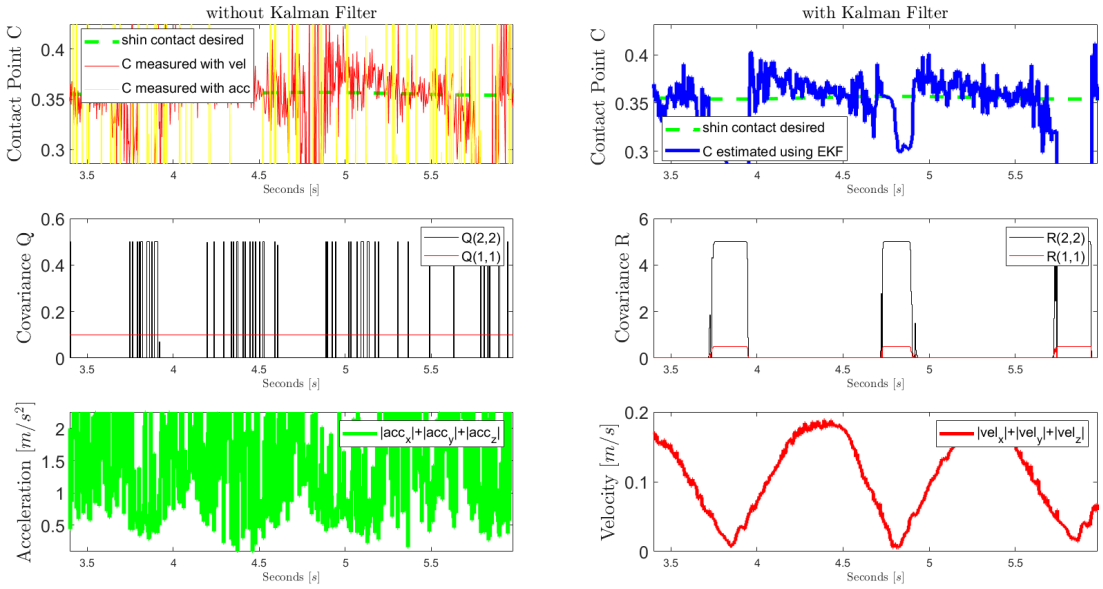


Figure 5.4: PushUp motion to check mainly the shape of knee velocity and acceleration

For this experiment the tuning is the following:

Table 5.3: Table of tuning Parameters for Kalman Filter during PushUp motion in experiments

parameters	R	Q
min	$0.0002m^2$	$0.001m^2/s^2$
max	$0.5m^2$	$0.5m^2/s^2$
k	100	100
x0	$0.05m^2$	$0.5m^2/s^2$

As it can be possible to see in the figure Fig. 5.4 the knee acceleration is a lot corrupted by noise and as a consequence the *acceleration model* is noisy as well. The main contribution is provided by the *velocity model*. During the PushUp motion, the values are around the foot contact point of 0.356 cm.

Considering the trot motion, the results are shown in the figure. Fig. 5.5

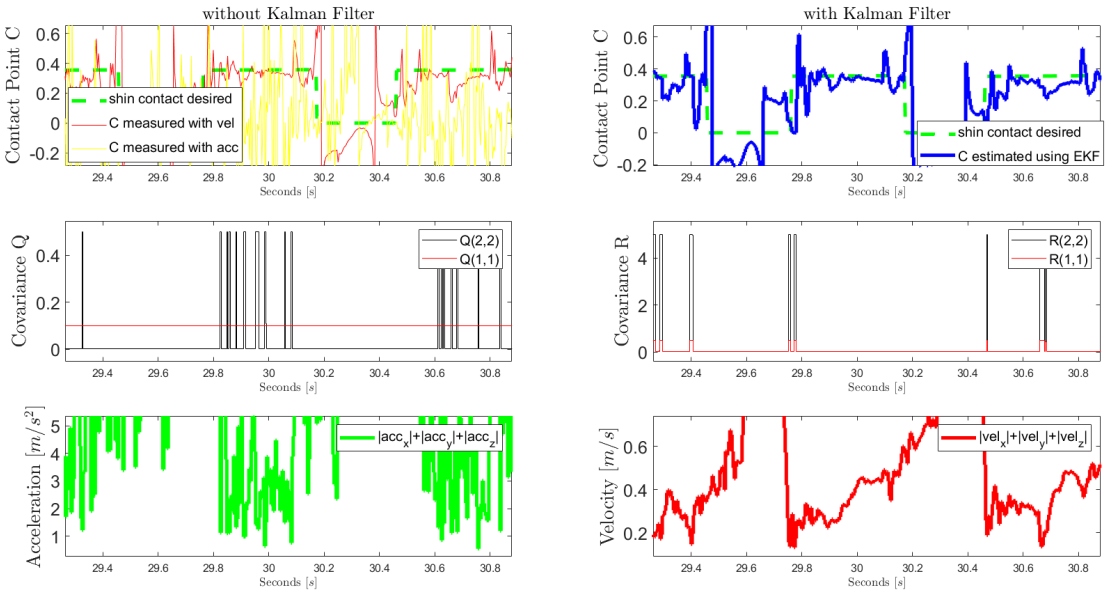


Figure 5.5: Trot motion during experiments - *foot contact detection*

The tuning parameters are set as follows:

Table 5.4: Table of tuning Parameters for Kalman Filter during trot motion in experiments

parameters	R	Q
min	$0.0002m^2$	$0.001\ m^2/s^2$
max	$0.5m^2$	$0.5m^2/s^2$
k	100	100
x0	$0.2m^2$	$1.5m^2/s^2$

5.2 HyQ locomotion with obstacle collisions

The main goal of this thesis is to detect the shin collisions due to the presence of obstacles during the walking of the robot.To do this both in Simulation and in Experiment it is used a pallet of dimension 300x1000x1600 cm, located under the trunk of the robot, between two opposite legs. To represents the collision, the robot moves forward and backward, having a certain velocity and acceleration that become zero at the touching moment.

5.2.1 Simulation

1. **One collision point** The implementation of the Kalman Filter is adopted to detect only one collision point along the shin.In simulation it can happen that two simultaneous contact points are detected and this particular situation is described in the following paragraph. To obtain a proper estimation of the contact point during shin collisions, several attempts

of tuning are performed using Matlab models and working directly on the shape of the estimated contact point. The final result is shown in the figure Fig. 5.7

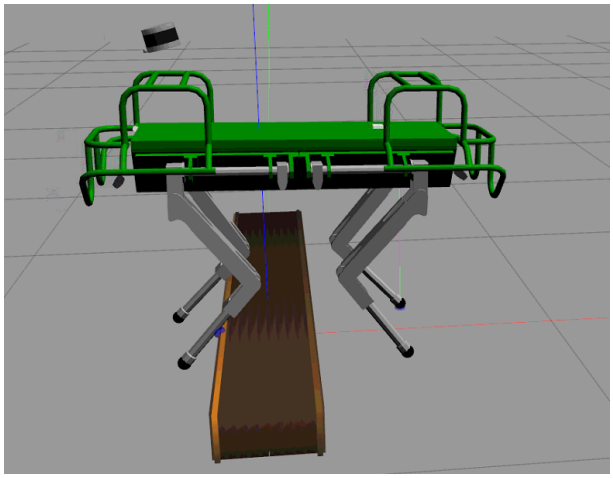


Figure 5.6: Contact collision on the shin during the Simulation on Gazebo

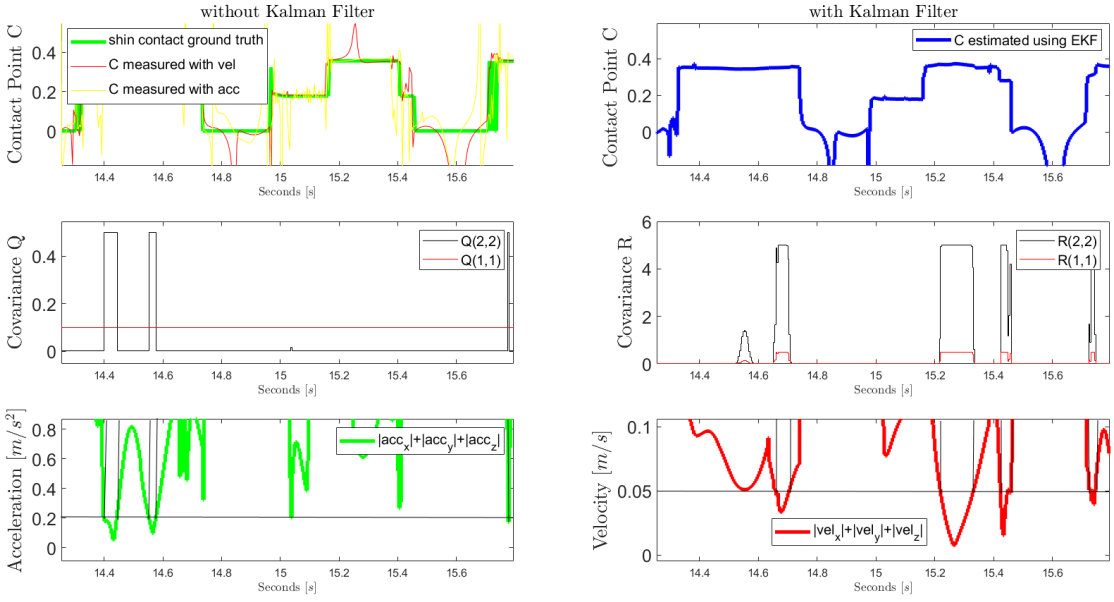


Figure 5.7: Analysis of shin contact points performed on Matlab. Collision at time 15.00 [s] in which the distance from the knee is 0.2 [m].

The shin collision happens at point 15.00[s],in fact it is possible to see that the distance from the knee is about 0.2[m] insted of 0.356 [m]. The procedure followed for having the

desired output is the *trial and error* method and the final tuning of the two parameters is set as follows:

Table 5.5: Table of tuning Parameters for Kalman Filter during collision

parameters	R	Q
min	$0.0002m^2$	$0.001m^2/s^2$
max	$0.5m^2$	$0.5m^2/s^2$
k	100	100
x0	$0.05m^2$	$0.2m^2/s^2$

To explain better the phase of tuning it is possible to see in the figure Fig. 5.7 that the covariance matrix Q is strictly related to the variation of the acceleration during time, while the covariance matrix R depends on the velocity. In the top left picture it is possible to compare the different outputs using the *velocity model* and the *acceleration model.*At point 15.2[s],thanks to the usage of Kalman Filter it is possible to "switch" from the *velocity model* to the *acceleration model* obtaining a flat output.

2. **Two collision points** As it can be possible to see in the figure Fig. 5.8 from 11.8-to 12.2 [s],the green line and the blue line are different for few seconds. In that moment the collision happens in two different points simultaneously: the foot and the shin. This is why the ground truth (the green line) oscillates between the two values, while the blue line remains to the costant value of 0.356 [m] because it is the first contact point detected.

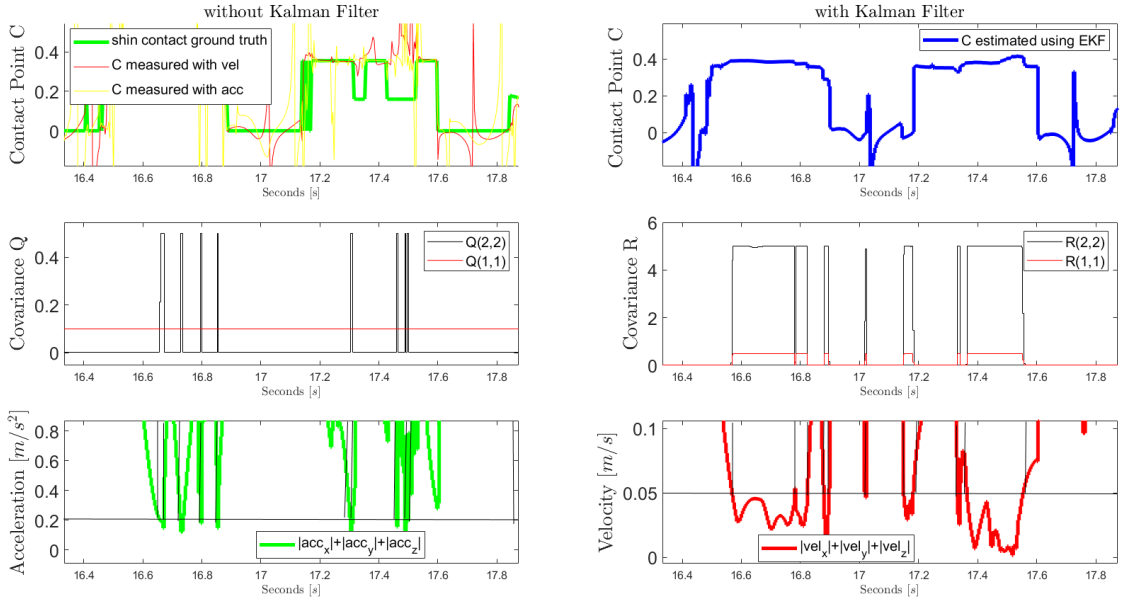


Figure 5.8: Tuning of the parameters performed on Matlab - two simultaneous collision points

5.2.2 Experiments

To analyze the contact point from experimental data, it is necessary to compare the obtained results with videos recorded during the phase of experiment. In this way it is possible to find the exact moment of collision between the shin and the pallet, since the ground truth is not provided. To be more precise in the evaluation of the contact point,the legs are marked with lines, drawn at a certain distance Fig. 5.9.

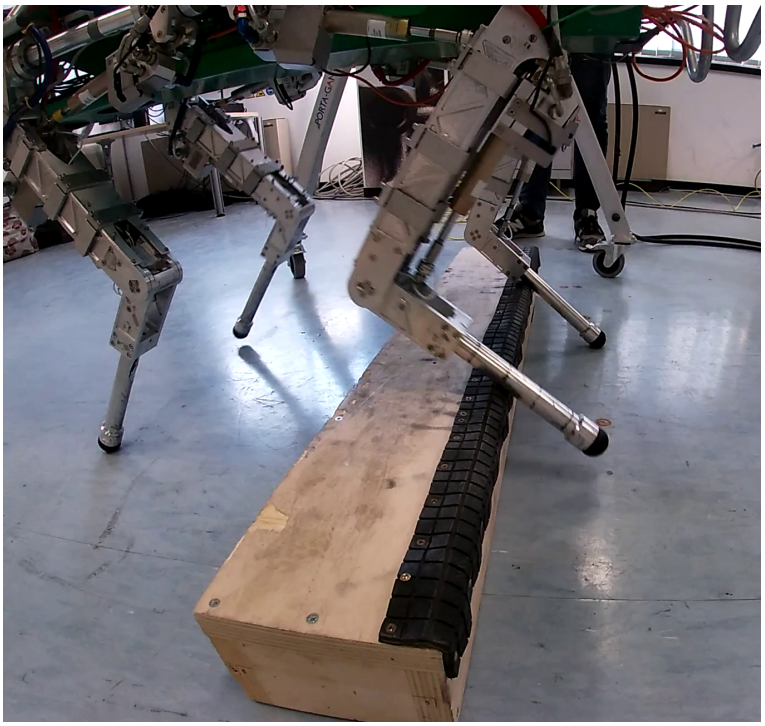


Figure 5.9: Shin collision moment during experiments

Performing trot of the robot and moving it against a pallet,the results are shown in the figure Fig. 5.10. As it can be possible to see in the picture, the collision happens at point 59.2 [s], providing for 1 sec the distance from the knee equal to 0.2 [cm]. In the same figure it is shown the *foot collision detection*, from 58.4 [s] to 58.7 [s] since the value reaches 0.356 [cm].

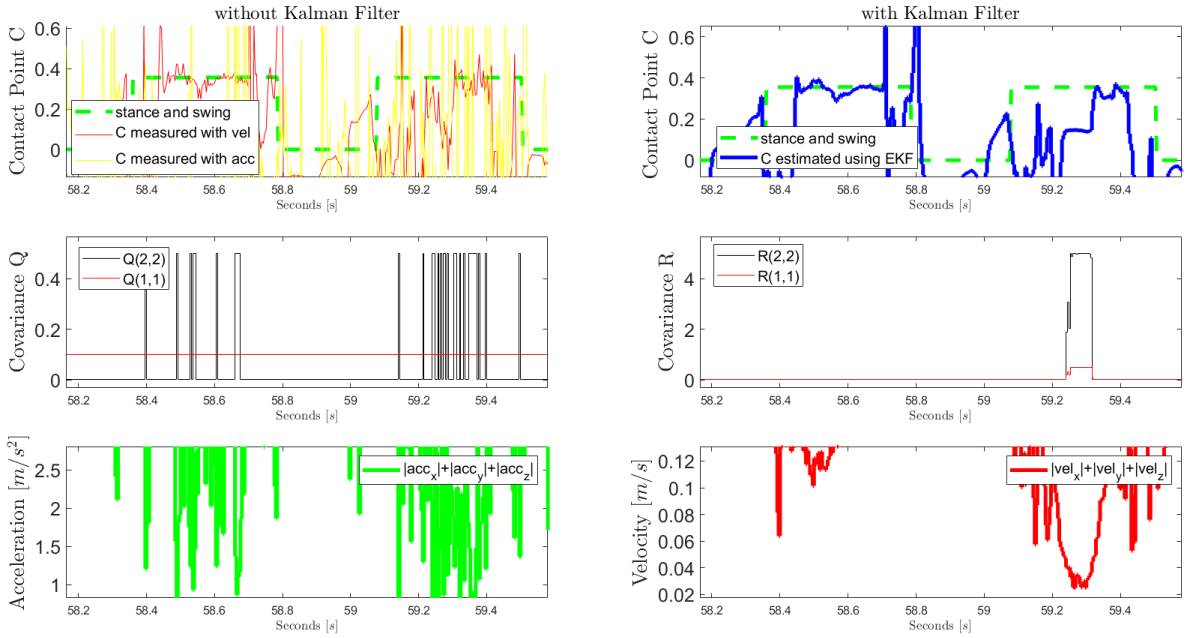


Figure 5.10: Kalman Filter for collision during experiment

The tuning parameters R and Q chosen for this experiments are shown in the table 5.6.

Table 5.6: Table of tuning Parameters for Kalman Filter during experiment collision with pallet

parameters	R	Q
min	$0.002m^2$	$0.001m^2/s^2$
max	$0.5m^2$	$0.5m^2/s^2$
k	100	100
x0	$0.04m^2$	$1.5m^2/s^2$

5.3 Closed-Loop Controller

At the end of the work, the results obtained from the Kalman Filter estimation are used as a feedback in order to stabilize the trunk controller. The idea is to update the current jacobians of contact position, providing the information about the contact point along the shin estimated by the EKF. In order to do this, a range of values are set for considering the presence of collisions and for allowing the robot to overcome the step. In fact the legs rotate along the pivot position, avoiding the slippage and letting the robot to go over the obstacle. The results achieved are shown in the figure Fig. 5.11.

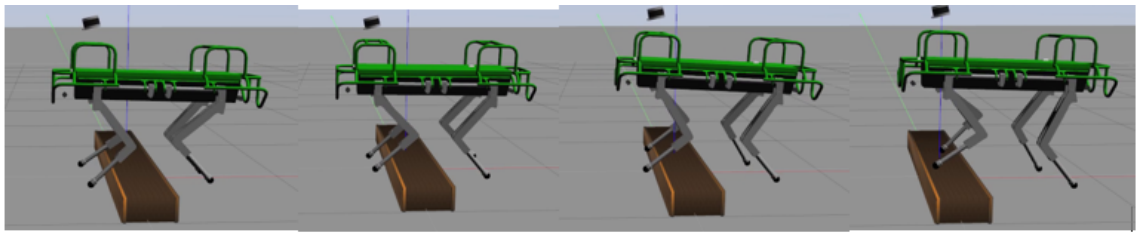


Figure 5.11: Closed-loop controller in Simulation

In order to design the controller to allow the robot to overcome the step, it is important to take into account the following considerations:

- the collision along the shin must happens on a certain range. As a consequence, the estimated contact point C is considered only when it is detected in an area that has a distance from the knee of about 0.1 [m] to 0.3 [m]. It implies that: $\hat{C} > param_{min}$ and $\hat{C} < param_{max}$.
- in the moment of the collision the leg has to be in stance status.
- the standard deviation of the covariance matrix P is smaller than a fixed value: $\sigma < param_{standard}$.

The logic is expressed as follows:

Table 5.7: Algorithm for feedback control

```
1:  for(leg=LF;leg<=RH;leg++){
2:      if shin_ekf[leg]>=param_min &&
3:          shin_ekf[leg]<= param_max &&
4:          stance[leg] &&
5:          sqrt(P)<param_standard {
6:              contactEstimate[leg]=shin_ekf[leg]}
7:      else {
8:          if(!stance[leg])
9:          contactEstimate[leg]=defaultShinLength
10:     }
11: }
```

Chapter 6

Conclusion

In dynamic legged robots, used for help human in challenging scenarios, one of the main issues is to always maintain stable the base of the robot and guarantee a robust motion in order to accomplish the desired tasks. The aim of this dissertation is to provide a contribution in the development of algorithms and techniques for stabilizing the trunk controller of HyQ robot when it enters in contact with obstacles or undesired objects.
The design of a state estimator for foot collision and shin collision has been realized using the knowledge of kinematics,for what concerns the model development, and of filtering for the estimation phase. This work has been developed in a 3D plane, taking into account the velocity, the acceleration and the orientation of the robot, providing as a consequence a wide analysis on the shin state during collision moment. Since this approach is a kinematics-based model, it can be easily implemented without the usage of extra sensors, detecting and localizing the contact points during a blind locomotion. Moreover the velocity and acceleration fusion inside the EKF allows to switch from one model to another, tuning properly the covariance matrices and increasing as a consequence the estimation performances.
The obtained results demonstrate that it is possible to estimate in a very accurate way the contact point along the shin, using then this information as a feedback to stabilize the trunk controller and help the robot to overcome the step.

6.1 Future Works

This thesis can be the starting point for future researcher works, in order to improve its algorithm for experiments and to extend its applications considering different scenarios. Some suggestions for future works can be:

- analysis of this model during the slippage of the shin, entering in contact with an obstacle. In fact it can happen that during collision, the leg of the robot slides along the surface of the obstacle. This model, implemented in 3D, can be useful to analyze in details this behaviour.
- implementation of the described model in different types of environments, considering for example soft terrains. The foot collision can be studied considering the ground-touching with different value of roughness. Tuning the values of the Kalman Filter it is possible to adapt this model to various terrains.

- improve the leg odometry of the robot using the knowledge of the collision. The leg odometry estimates the motion of the robot base starting from the kinematics of the legs. As a consequence the informations about the collision can be used to improve the whole motion of the robot in a challenging scenario.

Appendix A

Attitude of the robot

A.1 Elementary Rotation

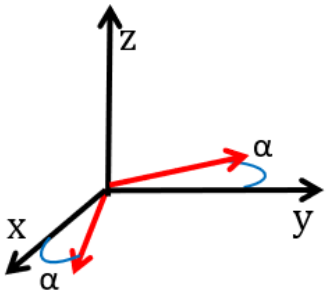


Figure A.1: Rotation of Frame O_{xyz} about z-axis by an angle α

The rotation around z-axis by an angle α is the following:

$$R_z(\alpha) = \begin{bmatrix} cos\alpha & -sin\alpha & 0 \\ sin\alpha & cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{A.1}$$

The rotation around x-axis by an angle γ is:

$$R_x(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\gamma & -sin\gamma \\ 0 & sin\gamma & cos\gamma \end{bmatrix} \tag{A.2}$$

while the rotation around y-axis by an agle β is:

$$R_y(\beta) = \begin{bmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix} \tag{A.3}$$

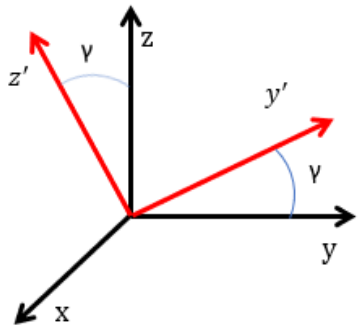


Figure A.2: Rotation of Frame O_{xyz} about x-axis by an angle γ

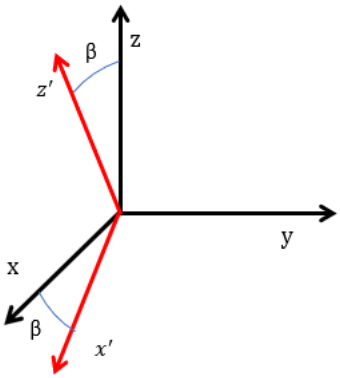


Figure A.3: Rotation of Frame O_{xyz} about y-axis by an angle β

A.2 Euler Angles

Euler angles are the minimal representation of orientation using three independent parameters:

- **ZXZ Angles**

The euler angles are ϕ,θ and ψ also known as spin, nutation and precession.

$$R_{313}(\phi,\theta,\psi) = R_3(\phi)R_1(\theta)R_3(\psi) = \begin{bmatrix} c_\phi c_\psi - s_\phi c_\theta s_\psi & c_\phi s_\psi + s_\phi c_\theta c_\psi & s_\phi s_\theta \\ -s_\phi c_\psi - c_\phi c_\theta s_\psi & -s_\phi s_\psi + c_\phi c_\theta c_\psi & c_\phi s_\theta \\ s_\theta s_\psi & -s_\theta c_\psi & c_\theta \end{bmatrix} \tag{A.4}$$

- **XYZ Angles**

These angles are also known as RPY angles: roll,pitch,yaw.

$$R_{123}(\phi,\theta,\psi) = R_1(\phi)R_2(\theta)R_3(\psi) = \begin{bmatrix} c_\theta c_\psi & c_\theta s_\psi & -s_\theta \\ s_\phi s_\theta c_\psi - c_\phi s_\psi & s_\phi s_\theta s_\psi + c_\phi c_\psi & s_\phi c_\theta \\ s_\theta c_\psi c_\phi + s_\phi s_\psi & s_\theta s_\psi c_\phi - s_\phi c_\psi & c_\theta c_\phi \end{bmatrix} \tag{A.5}$$

• **Rotation Matrix ⇒ Euler Angles**
Given a general Rotation matrix 3x3, it is possible to find the Euler Angles ϕ, θ and ψ as follows,using the sequence X,Y,Z:

$$\begin{aligned} R &= \begin{bmatrix} R_{1,1} & R_{1,2} & R_{1,3} \\ R_{2,1} & R_{2,2} & R_{2,3} \\ R_{3,1} & R_{3,2} & R_{3,3} \end{bmatrix} \\ \phi &= atan2(R_{2,3}, R_{3,3}) \\ \theta &= -asin(R_{1,3}) \\ \psi &= atan2(R_{1,2}, R_{1,1}) \end{aligned} \tag{A.6}$$

A.3 Denavit-Hartenberg Convention

The DH convention is used to find the direct kinematics in a simpler way than computing homogeneous transformations matrices and multipling them. For any open chain the following procedure can be applied:

1. Number in a crescent sequence the joint axes and name them from z_0 to z_{n-1}.
2. The frame 0 is the one with axis z_0 - set x_0 and y_0 according to the right-handed rule.
3. The origin O_i corresponds to the intersection of z_i with the common normal of z_{i-1} and z_i.
4. Set axis x_i in correspondance of the common normal between z_{i-1} and z_i following the direction from Joint i to $i+1$.
5. Draw axis y_i according to right-handed rule.

After finding all the axis for all the joints, it is possible to build the table with the parameters a_i (distance between the origins O_i and O_i'),d_i (position of O_i' along z_{i-1}), α_i (angle between z_{i-1} and z_i around x_i) and θ_i (angle between x_{i-1} and x_i around z_{i-1}).

Appendix B

Model and Code Verification

The phase of checking is important to understand and to verify if the measured inputs of the Kalman filter are correct or they present errors related to several situations as bugs in their implementations or wrong measurements coming from sensors. This is the first step necessary for then performing the simulation in Real-Time of the whole model. As explained in the Chapter 4.1.2 the inputs needed for the working principle of the Kalman Filter are:

- knee velocity
- knee acceleration
- shin orientation: θ and ϕ
- the measured contact point C_s and the measured contact velocity $\dot{C}_s$

Starting from the knee velocity in the world frame, it is obtained using the following equations:

$$\begin{aligned} v_k^w &= R_b^w \cdot (J_k \cdot \dot{q} - \omega_{base} \times p_k^b) + v_{base}^w \text{ for linear velocity} \\ \omega_k^w &= R_b^w \cdot (Jo_k \cdot \dot{q}) + \omega_{base} \text{ for angular velocity} \end{aligned} \tag{B.1}$$

The checking is performed in fast way using tools as the Scope or PlotJuggler in order to analyze the shape during the time, then stop the simulation, zoom in along the x or y axes and verify if there is a match between the obtained output and the situation shown in simulation using Gazebo. Then all the datas are collected and post-processed on Matlab in order to see the behaviour in a more detailed way.

B.1 Linear and Angular Velocity

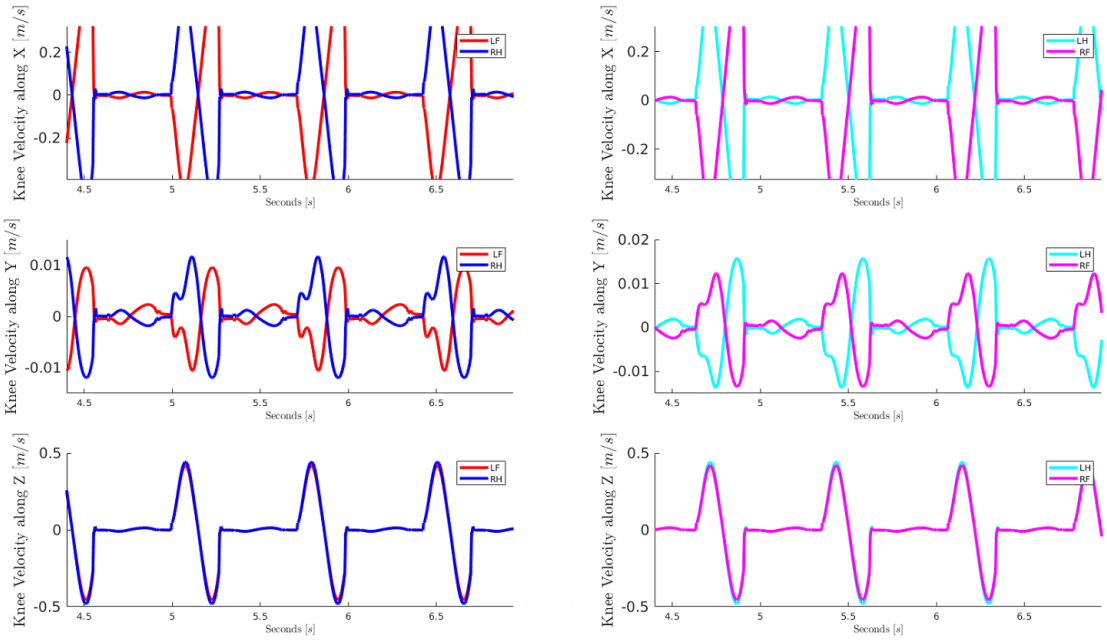


Figure B.1: Check knee linear velocity of RH/LH/LF/RF legs

In the Fig. B.1 are shown the knee linear velocities of the four legs. From the picture it is possible to see the trot motion of the robot, in which two legs are in the swing phase while the others two are touching the ground. Since the robot in this case is just trotting in the same place, without going forward or backward, the velocity along the axes is around zero.
For what concern the angular velocity, it is necessary for then evaluating the shin orientation.

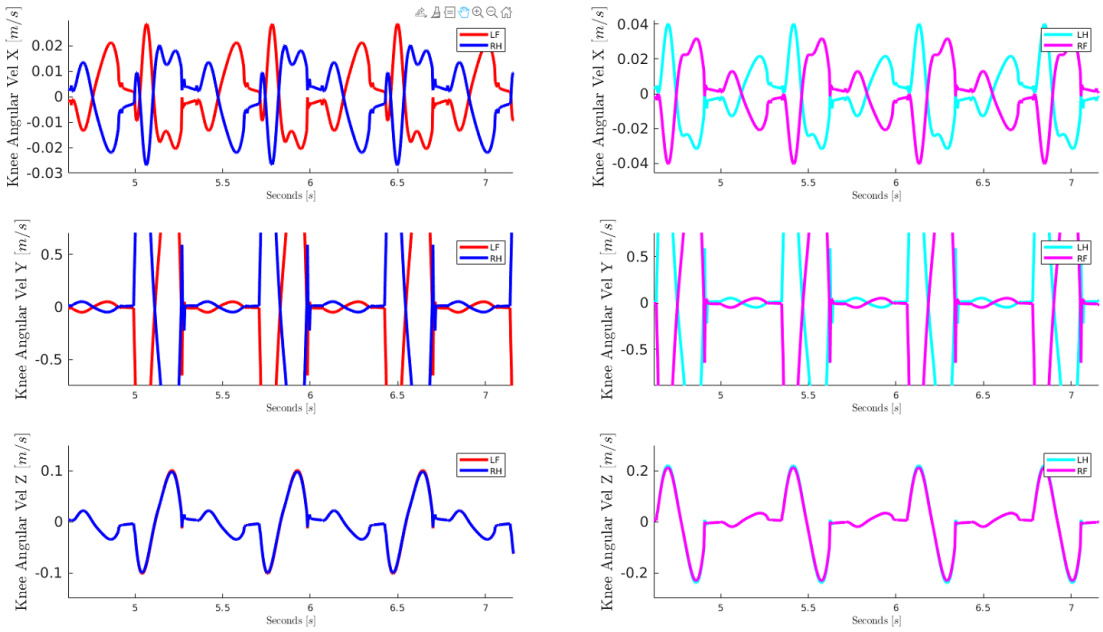


Figure B.2: Check knee angular velocity of RH/LH/LF/RF legs

B.2 Acceleration

The next step is to compute and check the acceleration of the knee, that is foundamental for then building the *acceleration model* used for the estimation of C_s and $\dot{C}_s$. The shape of the acceleration is shown in the figure Fig. B.3. It is computed as:

$$\begin{aligned} a^b &= (\dot{J} \cdot \dot{q} + J \cdot \ddot{q}) - S(\dot{\omega}) \cdot p^b - S(\omega) \cdot (J \cdot \dot{q}) \\ a^w &= a^w_{base} + S(\omega) \cdot R^w_b \cdot v^b + R^w_b \cdot a^b \end{aligned} \tag{B.2}$$

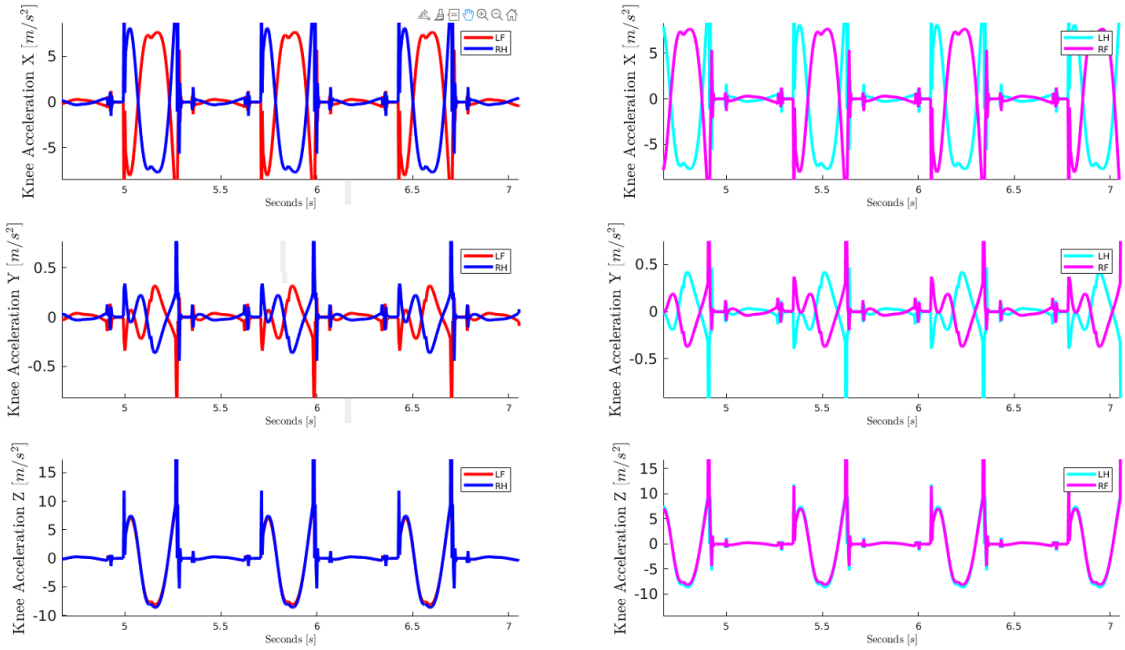


Figure B.3: Check knee acceleration of RH/LH/LF/RF legs

B.3 Shin Orientation

The shin orientation, represented by θ and ϕ, is computed directly from the rotation matrices between the robot base and each knee. Their derivatives are obtained deriving by time the following formulas,obtaining as a consequence the shin angle rate and the shin angle acceleration:

$$[\phi, \theta, \psi] = \begin{bmatrix} \phi(R_e) \\ \theta(R_e) \\ \psi(R_e) \end{bmatrix} = \begin{bmatrix} atan2(R_{e2,3}, R_{e_{3,3}}) \\ -asin(R_{e_{1,3}}) \\ atan2(R_{e_{1,2}}, R_{e_{1,1}}) \end{bmatrix} \tag{B.3}$$

$$[\dot{\phi}, \dot{\theta}, \dot{\psi}] = E^{-1} \cdot \omega_f^w \tag{B.4}$$

$$\begin{bmatrix} \ddot{\phi} \\ \ddot{\theta} \\ \ddot{\psi} \end{bmatrix} = \dot{E}^{-1} \cdot \omega_f^w + E^{-1} \cdot [S(\omega) \cdot R_b^w \cdot \omega_f + R_b^w \cdot (\dot{Jo} \cdot \dot{q} + Jo \cdot \ddot{q}) + \dot{\omega}_{base}^w] \tag{B.5}$$

Their shape is described in the figure Fig. B.4 and B.5.

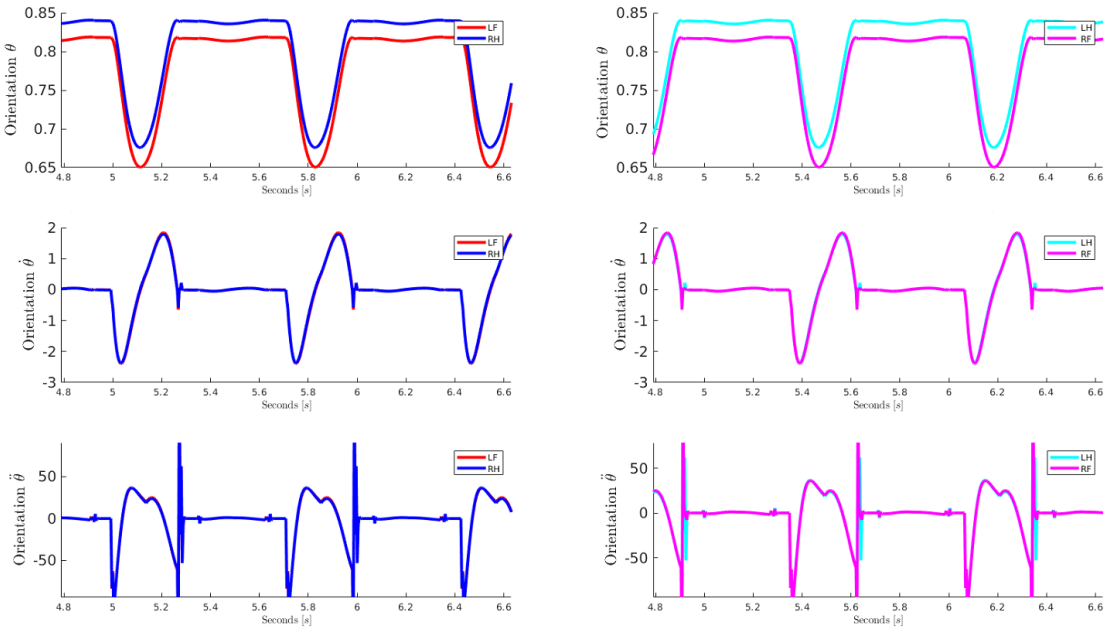


Figure B.4: Check orientation θ of RH/LH/LF/RF legs

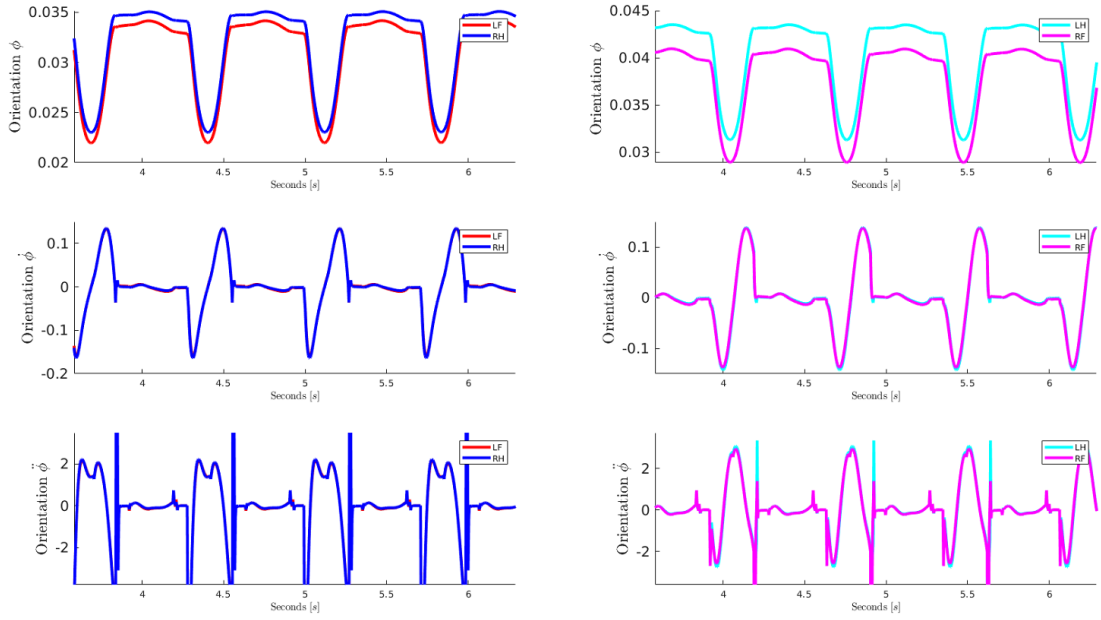


Figure B.5: Check orientation ϕ of RH/LH/LF/RF legs

B.4 Measured Contact Point and Contact Velocity

The last elements needed for the working principle of the Kalman Filter are C_s and $\dot{C}_s$ measured using the *velocity model*. They are computed using measurements coming from data sensors, this is the reason why they are corrupted by noise. The goal using the Filter is to estimate C_s and $\dot{C}_s$, obtaining a clear output. To see properly the shape of the contact point C_s and the contact velocity $\dot{C}_s$, the robot is moving forward with a certain velocity of $0.3m/s$. The contacts happen when the feet are touching the ground, as shown in figure Fig. B.6.

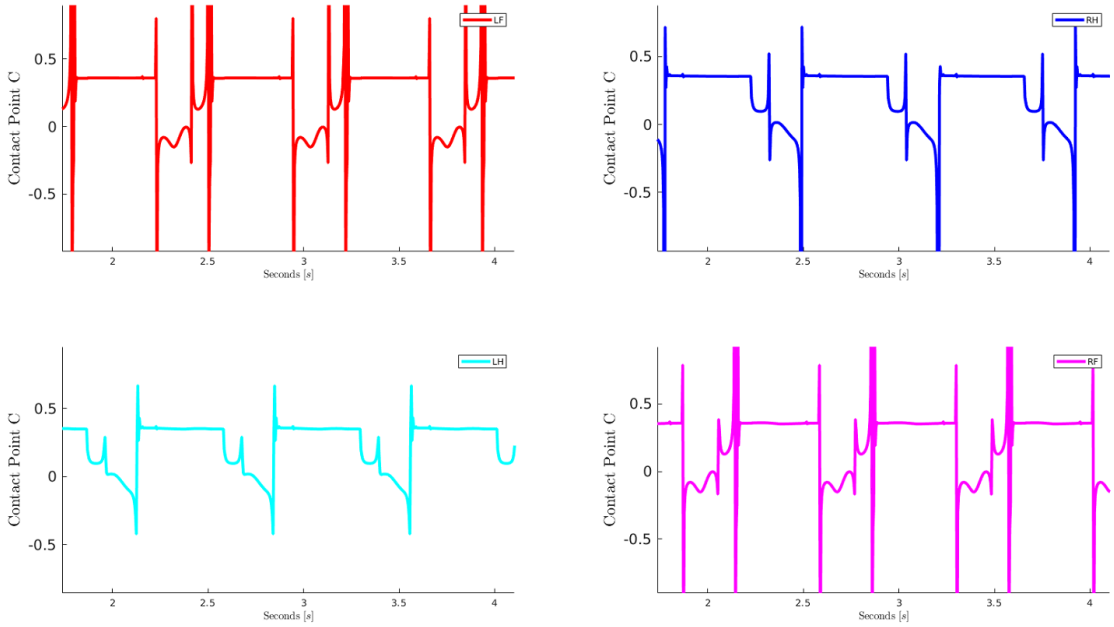


Figure B.6: Check contact point C_s of RH/LH/LF/RF legs

As it can be seen in the figure B.6, the foot collision with the ground happens in an alternate way for the legs: LH and RF are touching the ground while LF and RH are in the swing phase and vice-versa, guaranteeing always the stability to the robot. The contact velocity is then shown in the figure Fig. B.7,it is obviously zero when the collision happens.

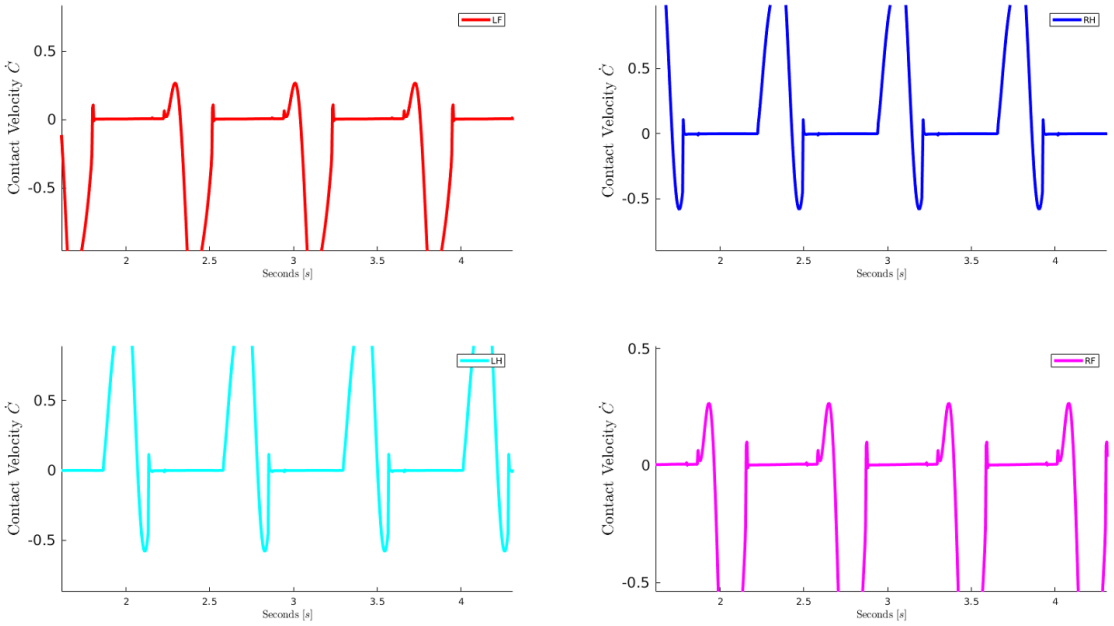


Figure B.7: Check contact velocity $\dot{C}_s$ of RH/LH/LF/RF legs

Bibliography

[1] https://www.iit.it/about-us/institute.

[2] https://www.iit.it/research/lines/dynamic-legged-systems.

[3] https://advr.iit.it/projects/inail-scc/teleoperazione.

[4] http://wiki.ros.org/.

[5] http://wiki.ros.org/Topics.

[6] http://wiki.ros.org/Services.

[7] http://gazebosim.org/.

[8] http://sdformat.org/.

[9] Victor Barasuol, Jonas Buchli, Claudio Semini, Marco Frigerio, Edson R De Pieri, and Darwin G Caldwell. A reactive controller framework for quadrupedal locomotion on challenging terrain. In *2013 IEEE International Conference on Robotics and Automation*, pages 2554–2561. IEEE, 2013.

[10] VICTOR Barasuol, GEOFF Fink, MICHELE Focchi, DARWIN G Caldwell, and CLAUDIO Semini. On the detection and localization of shin collisions and reactive actions in quadruped robots. In *22nd International Conference on Climbing and Walking Robots*, 2019.

[11] Victor Barasuol, Octavio A Villarreal-Magana, Dhinesh Sangiah, Marco Frigerio, Mike Baker, Robert Morgan, Gustavo A Medrano-Cerda, Darwin Gordon Caldwell, and Claudio Semini. Highly-integrated hydraulic smart actuators and smart manifolds for high-bandwidth force control. *Frontiers in Robotics and AI*, 5:51, 2018.

[12] Timothy D Barfoot. *State estimation for robotics*. Cambridge University Press, 2017.

[13] Thomas Bayes. Lii. an essay towards solving a problem in the doctrine of chances. by the late rev. mr. bayes, frs communicated by mr. price, in a letter to john canton, amfr s. *Philosophical transactions of the Royal Society of London*, (53):370–418, 1763.

[14] Adi Ben-Israel. The moore of the moore-penrose inverse. *The Electronic Journal of Linear Algebra*, 9, 2002.

[15] José M Bernardo and Adrian FM Smith. *Bayesian theory*, volume 405. John Wiley & Sons, 2009.

[16] Angelo Bratta, Romeo Orsolino, Michele Focchi, Victor Barasuol, Giovanni Gerardo Muscolo, and Claudio Semini. On the hardware feasibility of nonlinear trajectory optimization for legged locomotion based on a simplified dynamics. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1417–1423. IEEE, 2020.

[17] M Camurri. Multisensory state estimation and mapping on dynamic legged robots. *Istituto Italiano di Tecnologia and Univ. Genoa*, 2017.

[18] Zhe Chen et al. Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics*, 182(1):1–69, 2003.

[19] Sachin Chitta, Eitan Marder-Eppstein, Wim Meeussen, Vijay Pradeep, Adolfo Rodríguez Tsouroukdissian, Jonathan Bohren, David Coleman, Bence Magyar, Gennaro Raiola, Mathias Lüdtke, et al. ros_control: A generic and simple control framework for ros. *The Journal of Open Source Software*, 2(20):456–456, 2017.

[20] Alessandro De Luca and Raffaella Mattone. Actuator failure detection and isolation using generalized momenta. In *2003 IEEE international conference on robotics and automation (cat. No. 03CH37422)*, volume 1, pages 634–639. IEEE, 2003.

[21] Alessandro De Luca and Raffaella Mattone. Sensorless robot collision detection and hybrid force/motion control. In *Proceedings of the 2005 IEEE international conference on robotics and automation*, pages 999–1004. IEEE, 2005.

[22] Andrea Del Prete, Francesco Nori, Giorgio Metta, and Lorenzo Natale. Control of Contact Forces: the Role of Tactile Feedback for Contact Localization. *IEEE/RSJ International Conference on Intelligent Robots and Systems · October 2012*, 2012.

[23] James Diebel. Representing attitude: Euler angles, unit quaternions, and rotation vectors. *Matrix*, 58(15-16):1–35, 2006.

[24] Shamel Fahmi, Carlos Mastalli, Michele Focchi, DG Caldwell, and C Semini. Passivity based whole-body control for quadrupedal locomotion on challenging terrain. *IEEE Robot. Automat. Lett.(RA-L)*, 2019.

[25] Michele Focchi, Victor Barasuol, Ioannis Havoutis, Jonas Buchli, Claudio Semini, and Darwin G Caldwell. Local reflex generation for obstacle negotiation in quadrupedal locomotion. In *Nature-Inspired Mobile Robotics*, pages 443–450. World Scientific, 2013.

[26] Michele Focchi, Andrea Del Prete, Ioannis Havoutis, Roy Featherstone, Darwin G Caldwell, and Claudio Semini. High-slope terrain locomotion for torque-controlled quadruped robots. *Autonomous Robots*, 41(1):259–272, 2017.

[27] Michele Focchi, Romeo Orsolino, Marco Camurri, Victor Barasuol, Carlos Mastalli, Darwin G Caldwell, and Claudio Semini. Heuristic planning for rough terrain locomotion in presence of external disturbances and variable perception quality. In *Advances in Robotics Research: From Lab to Market*, pages 165–209. Springer, 2020.

[28] V Fox, Jeffrey Hightower, Lin Liao, Dirk Schulz, and Gaetano Borriello. Bayesian filtering for location estimation. *IEEE pervasive computing*, 2(3):24–33, 2003.

[29] Sami Haddadin, Alessandro De Luca, and Alin Albu-Schäffer. Robot collisions: Detection, isolation, and identification. *Submitted to IEEE Transactions on Robotics*, 2015.

[30] Vasilios Katsikis and Dimitrios Pappas. Fast computing of the moore-penrose inverse matrix. *The Electronic Journal of Linear Algebra*, 17:637–650, 2008.

[31] Hamza Khan, Satoshi Kitano, Marco Frigerio, Marco Camurri, Victor Barasuol, Roy Featherstone, Darwin G Caldwell, and Claudio Semini. Development of the lightweight hydraulic quadruped robot—minihyq. In *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, pages 1–6. IEEE, 2015.

[32] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[33] Marco Luise, M Vitetta G, Giacomo Bacci, and F Zuccardi Merli. *Teoria dei segnali. Terza edizione*. The McGraw-Hill Companies, 2009.

[34] Octavio Antonio Villarreal Magana, Victor Barasuol, Marco Camurri, Luca Franceschi, Michele Focchi, Massimiliano Pontil, Darwin G Caldwell, and Claudio Semini. Fast and continuous foothold adaptation for dynamic locomotion through cnns. *IEEE Robotics and Automation Letters*, 4(2):2140–2147, 2019.

[35] Lucas Manuelli and Russ Tedrake. Localizing external contact using proprioceptive sensors: The contact particle filter. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5062–5069. IEEE, 2016.

[36] Yoshiyuki Ohmura, Yasuo Kuniyoshi, and Akihiko Nagakubo. Conformable and scalable tactile sensor skin for curved surfaces. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 1348–1353. IEEE, 2006.

[37] Romeo Orsolino, Michele Focchi, Darwin G Caldwell, and Claudio Semini. An asymmetric model for quadrupedal bounding in place, 2016.

[38] ROMEO Orsolino, MICHELE Focchi, DARWIN G Caldwell, and CLAUDIO Semini. A combined limit cycle-zero moment point based approach for omni-directional quadrupedal bounding. In *International Conference on Climbing and Walking Robots (CLAWAR)*. World Scientific, 2017.

[39] Romeo Orsolino, Michele Focchi, Carlos Mastalli, Hongkai Dai, Darwin G Caldwell, and Claudio Semini. Application of wrench-based feasibility analysis to the online trajectory optimization of legged robots. *IEEE Robotics and Automation Letters*, 3(4):3363–3370, 2018.

[40] S James Press. *Subjective and objective Bayesian statistics: Principles, models, and applications*, volume 590. John Wiley & Sons, 2009.

[41] YoonSeok Pyo, HanCheol Cho, R Jung, and T Lim. Ros robot programming. *ROBOTIS, December*, 2017.

[42] Morgan Quigley, Brian Gerkey, and William D Smart. *Programming Robots with ROS: a practical introduction to the Robot Operating System.* " O'Reilly Media, Inc.", 2015.

[43] Christian Robert. *The Bayesian choice: from decision-theoretic foundations to computational implementation*. Springer Science & Business Media, 2007.

[44] Ioannis Sarakoglou, Nadia Garcia-Hernandez, Nikos G Tsagarakis, and Darwin G Caldwell. A high performance tactile feedback display and its integration in teleoperation. *IEEE Transactions on Haptics*, 5(3):252–263, 2012.

[45] C Semini, V Barasuol, M Focchi, C Boelens, M Emara, S Casella, O Villarreal, R Orsolino, G Fink, S Fahmi, et al. Brief introduction to the quadruped robot hyqreal. *Istituto di Robotica e Macchine Intelligenti (I-RIM)*, 2019.

[46] Claudio Semini. Hyq-design and development of a hydraulically actuated quadruped robot. *Doctor of Philosophy (Ph. D.), University of Genoa, Italy*, 2010.

[47] Claudio Semini. Hyq-robot: Standard definition for joint angles and kinematic parameters of the legs and torso. 2017.

[48] Claudio Semini, Victor Barasuol, Jake Goldsmith, Marco Frigerio, Michele Focchi, Yifu Gao, and Darwin G Caldwell. Design of the hydraulically actuated, torque-controlled quadruped robot hyq2max. *IEEE/ASME Transactions on Mechatronics*, 22(2):635–646, 2016.

[49] Claudio Semini, Nikos G Tsagarakis, Emanuele Guglielmino, Michele Focchi, Ferdinando Cannella, and Darwin G Caldwell. Design of hyq–a hydraulically and electrically actuated quadruped robot. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 225(6):831–849, 2011.

[50] Bruno Siciliano and Oussama Khatib. *Springer handbook of robotics*. Springer, 2016.

[51] Bruno Siciliano, Lorenzo Sciavicco, Luigi Villani, and Giuseppe Oriolo. *Robotics: modelling, planning and control*. Springer Science & Business Media, 2010.

[52] Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

[53] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probalistic robotics. *Kybernetes*, 2006.

[54] John Ulmen and Mark Cutkosky. A robust, low-cost and low-noise artificial skin for human-friendly robots. In *2010 IEEE International conference on robotics and automation*, pages 4836–4841. IEEE, 2010.

[55] BILAL UR REHMAN, Michele Focchi, Marco Frigerio, Jake Goldsmith, Darwin G Caldwell, and Claudio Semini. Design of a hydraulically actuated arm for a quadruped robot. In *ASSISTIVE ROBOTICS: Proceedings of the 18th International Conference on CLAWAR 2015*, pages 283–290. World Scientific, 2016.

[56] Octavio Villarreal. *Bridging Vision and Dynamic Legged Locomotion*. PhD thesis, Tese (Doutorado)—Istituto Italiano di Tecnologia (IIT) and University of Genova, 2020.

[57] Jonathan Vorndamme, Moritz Schappler, and Sami Haddadin. Collision detection, isolation and identification for humanoids. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4754–4761. IEEE, 2017.

[58] Sean Wang, Ankit Bhatia, Matthew T Mason, and Aaron M Johnson. Contact localization using velocity constraints. In *International Conference on Intelligent Robots and Systems*, 2020.

[59] Alexander W Winkler, Carlos Mastalli, Ioannis Havoutis, Michele Focchi, Darwin G Caldwell, and Claudio Semini. Planning and execution of dynamic whole-body locomotion for a hydraulic quadruped on challenging terrain. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5148–5154. IEEE, 2015.