Modeling and Control of Propeller-Aided Legged Robot

Università di Genova

Nicolò Torre

DIBRIS - Department of Computer Science, Bioengineering, Robotics and System Engineering

University of Genova

Supervisors:
Prof. Giorgio Cannata
Prof. Marco Baglietto

In partial fulfillment of the requirements for the degree of

Laurea Magistrale in Robotics Engineering

march 23, 2026

Abstract

Legged robots are very common for harsh terrain navigation, but struggle with big obstacles, having to inefficiently move around them. flying vehicles, in particular multirotors, don't have this problem thanks to their 6-DOF motion capabilities, but are less energy efficient having to counteract the force of gravity, leading to shorter mission durations. This Thesis introduces a design approach towards the combination of the two families of robot. The modeling, control strategy and state estimation of a legged robot aided by propellers are introduced. The propellers provide jumping and air control capabilities to the robot. An Extended Kalman Filter is implemented to address the noisy Inertial Measurement Units and Global Positioning System sensors used for the robot. A simple estimation and control loop of the robot is described. The robot's jumping capabilities are demonstrated inside of a simulation.

Contents

List of Figures

Chapter 1

Introduction

The exploration of harsh environments is currently a topic of great interest for many different purposes, such as search and rescue ([12]), environment mapping ([14]), and payload transportation ([13]). These environments often have large obstacles and uneven terrain that severely limit the mobility of conventional robots. The robots most commonly employed for such exploration are legged robots and flying robots, each with their own advantages and disadvantages.

Legged robots are currently a major focus of research due to their ability to move through uneven terrain and other harsh environments ([15], [9]), but are limited to ground movement, lacking the ability to move over tall objects and requiring inefficient circumnavigation of such obstacles, particularly disadvantageous when very large.

In the literature there are some legged robots with the ability to jump, but they either do not have very high jumping capabilities ([17]) or rotate significantly in mid-air and may fall upon landing, requiring time to self-stabilize to a standing position ([18]).

Flying robots can easily move around large obstacles thanks to their ability to move and rotate in all directions of a three-dimensional space. However, they are less energy efficient than ground robots because they have to constantly counteract gravity, resulting in a lower payload capacity and reduced available exploration time. Many flying robots in the literature are based on multi-rotors ([16], [1]) due to their simple structure and model, despite being more challenging to control than other flying robots because of their underactuated nature.

In this Thesis, we want to explore the viability of a hybrid robot that combines the locomotion efficiency of legged robots with the use of propellers to support and stabilize the robot's jumps over large obstacles.

Some hybrid aerial-legged robots already exist. In [7], an airplane has been equipped with a pair of bird-inspired robotic legs in order to make takeoffs and landings more energy efficient and to provide the robot with the ability to hop

over small gaps and jump over relatively tall obstacles.

In [2] and [3], a humanoid robot is equipped with thrusters on the end effectors of its limbs to provide flight capabilities, specifically to control of the Center of Mass (COM) of the robot.

A biped robot was equipped with ducted fans near its feet and at its waist by Y. Li et al. in [6]. The purpose of the fans was to control the attitude of the robot in mid-air using a vectored thrust control paradigm.

The paper [8] described a hopper robot with upward and downward-facing propellers and a spring-equipped leg (rigidly connected to the rest of the body). The propellers have the purpose of providing energy lost due to friction and drag, in order to keep the hopping locomotion, and to change the attitude of the robot to achieve the desired hopping trajectory.

F. Shi et al. in [10] equipped various humanoid robots with independently rotating propellers to stabilize their ground locomotion or to provide the ability to fly, and were controlled by combining machine learning (to calculate the required joint torque and propeller wrench control) and optimization-based control (for propeller thrust force and orientation control).

Despite these progresses in the research, most hybrid robots use independent rotating thrusters. The integration of a conventional multi-rotor structure in a legged robot for jumping, landing, and mid-air stabilization assistance is currently unexplored.

Considering this, we designed a basic legged robot with a multi-rotor attached on top, providing support when jumping over obstacles and enabling mid-air control. We also developed a state estimation algorithm and a control strategy to control the velocity and position of the COM of the robot, especially in mid-air.

The control of the COM during the flight phase presents some challenges, especially in the stabilization of the attitude when the COM of the system and the geometric center of the multi-rotor are not aligned.

In [5] this stabilization challenge is addressed by adding a moving weight and by implementing a PID controller for the position of this weight to counterbalance the COM shift caused by the payload.

In [4] the authors showed that it is possible to separate the COM position dynamics and the rotational dynamics into two distinct subsystems for a hybrid robot composed of a multi-rotor and a multi-link manipulator, with the possibility of applying different control objectives and tasks independently to the two subsystems. Although the platform of application is slightly different, this method could also be used in hybrid propeller-aided legged robots to further assist in the attitude stabilization of the robots.

The robot control was structured as a double-loop control. The outer loop controls the velocity error of the COM of the robot towards 0 and stabilizes the oscillations of unactuated joints (if any), while the inner loop controls the attitude

of the multi-rotor and the configuration of the leg.

An accurate control of the COM and the attitude of the robot requires a good estimation of the state of the system.

The most common methods researched for the state estimation of mobile robots use Inertial Measurement Units (IMU) and Global Positioning Systems (GPS) as their main sensors. The GPS provides absolute position (and sometimes also velocity).

Many robots use in their estimators the output of the IMU sensor to correct the attitude and angular velocity estimate of the system, as described by the robots presented in [11], [8], and [6].

In our work, the state estimation is achieved by implementing an Extended Kalman Filter (EKF) using the output of two Inertial Measurement Units (IMU) and a Global Positioning System (GPS) as measurements of the system.

We decided to implement the state estimator using an EKF in order to address the noisy output of the sensors used.

The robot described in this thesis is simplified as a two-dimensional robot composed of a bi-rotor attached to an unactuated single-link leg, modeled and controlled in planar motion (x–z planar motion and rotation about the y-axis). The robot was built and tested in a simulation inside the MuJoCo physics engine.

1.1 State of the art review

Some hybrid legged and flying robots have already been proposed in the literature. In this section, we will analyze the methodology, focusing on the models, control strategies, and state estimation methods developed for the hybrid robots that were the main source of inspiration for our work, in particular robots with propeller-aided locomotion and robots with cooperation for takeoff and landing between the legs and the flight platform.

Traversaro et al. in [2] and [3] added jet thrusters on the end effectors of the limbs of the humanoid robot iCub (as shown in Fig. 1.1).

They implemented a backstepping control on thruster intensity and joint position, designing a proportional–integral controller for the linear and angular momentum of the COM of the robot and a full-model linearization control for the joint velocities.

This work already provides a big step towards the development of hybrid legged-aerial robots, but the use of jet thrusters implies reduced payload capacity and shorter mission durations.

For this robot, they first considered the model of the robot, obtained using the Euler–Poincaré formalism:

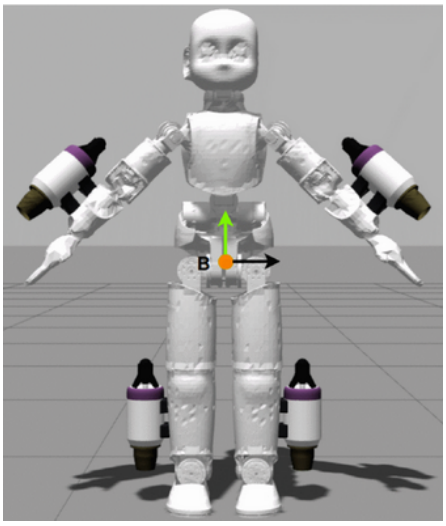


Figure 1.1: image of the humanoid robot and its equipped thrusters controlled in [2], [3].

$$M(q)\,\dot{\nu} + C(q,\nu)\,\nu + G(q) = \begin{bmatrix} 0_{6\times 1} \\ \tau \end{bmatrix} + \sum_{k=1}^{m} J_k^{\intercal}\, F_k \tag{1.1}$$

where q is the robot configuration vector in generalized coordinates, including the position and orientation of the main body of the robot, and ν is the corresponding velocity vector. $M(q)$ represents the inertia matrix, $C(q,\nu)$ is the Coriolis matrix, $G(q)$ is the gravity force vector. Finally, J_k is the Jacobian matrix that maps the velocity vector to the linear velocity of the installation frame of the k-th thruster.

Then, a control strategy for the centroidal momentum of the robot was developed by combining vectored-thrust control and backstepping control (considering from the model of the system that the joint positions and velocities could be feedback-linearized). They also demonstrated the asymptotic stability of their implementation using Lyapunov analysis, first establishing the intensity and orientation required to achieve the desired centroidal momentum of the robot (with a model-predictive proportional–integral control on the momentum error) and then finding the required limb joint velocities and thruster rates of change to reach the desired intensity and orientation while remaining as close as possible to a reference limb posture.

The performance of the controller for flight purposes was demonstrated with different trajectories in simulation.

The design of an estimator was reserved for future work on the robot.

Li Y. et al. in [6] described a biped robot with four ducted fans, two installed on opposite sides of the torso (one at the front and one at the back) and the other

two integrated into the feet of the robot, as shown in Figure 1.2. They derived a dynamic model of the forces acting on the robot COM and implemented a control strategy based on changing the orientation of the fans installed on the legs to stabilize the attitude of the robot. They focused on modeling the contribution of the thrust to the forces and torques acting on the COM of the robot:

$$\begin{cases} {}^{w}F_{COM} = R_y(\theta_{pitch}) \begin{bmatrix} f_L \sin\theta_L + f_R \sin\theta_R \\ 0 \\ f_B + f_F + f_L \cos\theta_L + f_R \cos\theta_R \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ Mg \end{bmatrix} \\ {}^{w}\tau_{COM} = R_y(\theta_{pitch}) \begin{bmatrix} \frac{1}{2} L_f (f_L \cos\theta_L - f_R \cos\theta_R) \\ T_{y_1} + T_{y_2} + T_{y_3} \\ \frac{1}{2} L_f (f_R \sin\theta_R - f_L \sin\theta_L) \end{bmatrix} \\ T_{y_1} = f_B(\frac{L}{2} + x_{COM}) - f_L(\frac{L}{2} - x_{COM}) \\ T_{y_2} = (f_L \cos\theta_L + f_R \cos\theta_R)(x_{COM} - p_{fx}) \\ T_{y_3} = -(f_L \sin\theta_L + f_R \sin\theta_R)(z_{COM} - p_{fz}) \end{cases} \tag{1.2}$$

Where $R_y(\theta_{pitch}$ is the rotation matrix mapping the vector representation from the frame of the torso of the robot to the world frame, f_p symbolizes the intensity of the thrust of the corresponding ducted fan ($_B$ at the back of the torso, $_F$ in the frontal part of the torso, $_L$ on the left foot and $_R$ on the right foot), θ_L and θ_R are the pitch rotation angles between the torso of the robot and the frame of the feet (left and right respectfully), with the thrust direction of the fans aligned with the z-axis of the corresponding foot frame, L_f is the distance along the y-axis between the two leg fans and L is the distance between the two installed on the torso, x_{COM} and z_{COM} are the x-axis and z-axis position of the robot COM relative to the torso frame, p_{fx} and p_{fz} are the x-axis and z-axis position of the two feet relative to the torso frame.

From the force model shown in the previous equation, they controlled the pitch and yaw of the robot by implementing a proportional-derivative control on the pitch of the ducted fans on the leg to generate the desired torque on the robot COM. They used an IMU to estimate the attitude of the robot and a motion capture system to estimate its altitude during the experiments.

Li et al. in [8] described a hybrid hopper robot composed of a quad-rotor attached to a leg composed of two links connected with a prismatic joint and an elastomer acting as spring, designed to carry heavy payloads efficiently. A picture of the robot is shown in Figure 1.3.

This work provides some insight into the available propeller-aided attitude stabilization strategies for hopping and jumping robots.

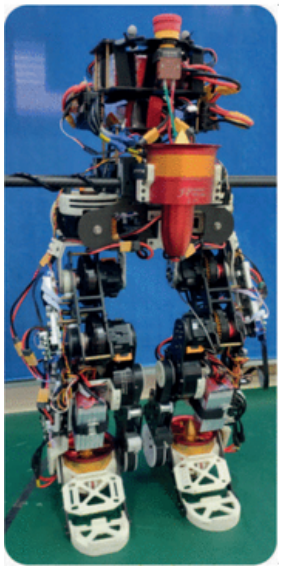


Figure 1.2: image of the biped robot and its equipped propellers controlled in [6].

In this robot, the propellers are used to control the attitude of the robot and to add to the system the energy, lost in the landing and takeoff, required to keep hopping at the same height.

The flight model of the system developed was similar to the model of a quadrotor:

$$\begin{cases} m\,\ddot{\boldsymbol{p}} = R\boldsymbol{e}_3\,(-1)^k 4\sum_{i=1}^{4} f_i + mg\boldsymbol{e}_3 \\ I\,\dot{\omega}_b + \omega_b \times I\,\omega_b = \tau_p \\ \tau_p = \sum_{i=1}^{4} l_i \times \boldsymbol{e}_3\, f_i + \boldsymbol{e}_3 M_i \end{cases} \tag{1.3}$$

Where $\boldsymbol{e}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^\intercal$, m is the total mass of the system including the payload, $\boldsymbol{p}$ is the global position of the robot COM, R represents the rotation matrix between the quadrotor frame and the world frame, f_i is the thrust produced by the i-th propeller, the exponent k is 0 during the ascending phase and 0 during the descending phase and is used to represent whether the thrust direction is upward or downward, I is the inertia matrix of the system, ω_b is the angular rotation vector of the robot, l_i is the position of each propeller relative to the center of the quadrotor, and M_i is the moment caused by the rotation of each propeller.

The grounded robot is modeled assuming non-slip condition and approximating the spring-loaded inverted pendulum model considering the body inertia neglectable compared to the Inertia derived from the mass point modeling of the robot:

$$m\ddot{\boldsymbol{p}}_s = \frac{\boldsymbol{p}_s}{l}\left[k_s(l - l_0 - l_p) - \epsilon\dot{l}\right] - mg\boldsymbol{e}_3 \tag{1.4}$$

Where $\boldsymbol{p}_s$ is the position of the robot COM relative to the contact point of the robot with the ground, $l = \|\boldsymbol{p}_s\|$, l_0 is the length of the leg at rest, l_p is the length the elastomer is pre-compressed during the assembly of the robot, k_s is the linear stiffness of the elastomer and ϵ represents a damping coefficient.

The current nominal hopping height is computed from an energetic analysis based on the speed of the robot at the apex of the hop and the time it takes to move from the apex to the landing position.

A neural network was developed to map the desired takeoff velocity and the expected landing velocity with the required robot attitude at the landing instant.

An IMU was used to estimate the attitude of the robot.

The position and velocity of the COM were estimated using a motion capture system indoor and a LIDAR for the outdoor experiments; these sensor systems were also used to provide attitude estimation to the robot, especially during the unpowered free-fall stages of the hop.

The robot control was structured into multiple control blocks.

The height controller computed the thrust to apply uniformly during the ascending phase of the hop, and during the descending phase as well if required by the payload, in order to stabilize the nominal hopping height with the desired one.

The high-level position controller computes the landing position and velocity of the current hop from the position and velocity at the apex of the hop, determines the necessary takeoff velocity to reach the desired landing position of the next hop, and computes the required landing attitude to achieve such takeoff velocity, with an iterative approach implemented to address the coupling between landing attitude, landing velocity, and takeoff velocity.

The thrust and attitude management module handles the desired thrust and attitude depending on the stage of the robot: grounded, ascending, near the apex of the trajectory, and descending. When grounded and near the apex, the thrust is set to zero; when ascending and descending, the thrust is set to the nominal value provided by the height controller. The desired roll and pitch of the robot are calculated from the desired thrust direction provided by the high-level position controller and from the reference yaw provided as an external input.

The low-level attitude controller converts the desired thrust and rotation from the thrust and attitude management module into the required individual forces and motor intensities by performing an optimization problem that minimizes the thrust error while matching the desired torque to apply to the system.

Shin et al. in [7] presented legs designed to support a plane robot, allowing it to walk, hop over small gaps, jump on top of obstacles, or perform a combined

Figure 1.3: Image of the propeller-based hopper robot described in [8].

jump and propeller-based takeoff, with the aim of decreasing the horizontal space required and increasing energy efficiency. The robot is shown in figure 1.4.

Unlike other works, this article showed the cooperation between the legged and the flying components of the robot for the transition between the grounded locomotion and the flight, although in this work the flight platform of the robot was an airplane instead of a multi-rotor or a set of thruster.

The paper focused on the mechanical structure of the leg and on the control of joint torques to achieve the desired jump velocity during takeoff and to keep the legs in a desired configuration during mid-air motion. To achieve this objective, proportional derivative (PD) control tasks for pitch, horizontal displacement, and vertical displacement were defined. These tasks were then converted into joint accelerations through a prioritized task-based control framework.

First, the authors derived the general dynamic model of the system:

$$\begin{cases} M(\boldsymbol{q})\,\ddot{\boldsymbol{q}} + C(\boldsymbol{q},\dot{\boldsymbol{q}})\,\dot{\boldsymbol{q}} + \boldsymbol{G}(\boldsymbol{q}) + J_c^{\intercal}\,\boldsymbol{f}c = S^{\intercal}\,\boldsymbol{\tau} + \boldsymbol{f}_{ext} \\ \dot{\boldsymbol{p}}_c = J_c\dot{\boldsymbol{q}} = 0 \end{cases} \tag{1.5}$$

where $\boldsymbol{q}$ is the robot configuration vector, M is the inertia matrix, $\boldsymbol{C}$ represents the vector of centripetal and Coriolis forces, $\boldsymbol{G}$ represents the generalized gravitational forces. J_c is the Jacobian matrix that maps the generalized velocities to the velocity of the ground contact point $\boldsymbol{p}_c$, while $\boldsymbol{f}_c$ is the corresponding ground reaction force vector, $\boldsymbol{\tau}$ is the joint torque vector, and S is the selection matrix that maps the joint torque vector to the actuated generalized coordinates. Finally, $\boldsymbol{f}_{ext}$ represents all external forces acting on the robot, including propeller thrust, aerodynamic lift and drag generated by the wings, and the torques produced by the springs installed at the ankle and leg joints.

Based on this model, a recursive prioritized task-based control approach was

used to map the desired robot pitch and position to the required joint accelerations. Subsequently, a full linearization of the dynamic model was performed to compute the joint torques required to generate these accelerations while maintaining the non-slip condition at the contact points.

The performance of the robot was validated in simulation and tested in experiments using a motion-tracking system and four markers to estimate the position and the attitude of the main body, and using the position and velocity trajectories as reference for the experimental tests. The experimental tests showed worse performance compared to the simulations because of difficulty in meeting real-time constraints with the limited computational power of the micro-controllers used, and because of limitations in the activation timing of the leg actuators with the belts used as transmission.

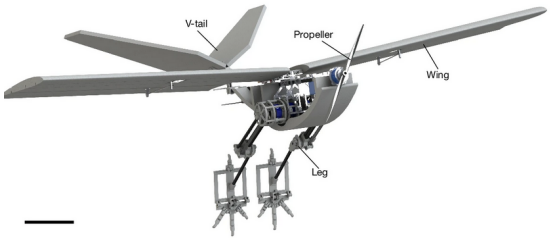


Figure 1.4: image of the bird-inspired robot RAVEN, described in [7].

1.2 Structure of the Thesis

The content of the Thesis is organized as follows:

- In chapter 2, the methodology for modeling (Section 2.1), control (section 2.3), and state estimation (section 2.2) of the robot will be described, including the differences between the grounded robot and the robot in mid-air;
- In chapter 3, we will describe the simulation engine and how the robot's simulation was realized. We will also describe the environment simulated to test the performance of the estimator and the controller;
  - In sections 3.6, 3.7, and 3.8, the results of the tests performed will be analyzed;
- In chapter 4, the results of the methodology and the simulation are summarized and the possible future work on the project is described.

Chapter 2

Methodology

The model realized in simulation in this Thesis is presented in the drawing 2.1. The system is a bi-dimensional robot composed of a bi-rotor connected to a single-link leg using an unactuated joint. It is equipped with two IMUs, a GPS module and a touch sensors. The touch sensor is used to handle the transition between the ground and flight stages of the System

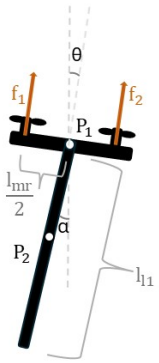


Figure 2.1: Scheme of the simplified model on which modeling, state estimation and control are applied; the system variables are the x and z position of P1, θ and α.

This chapter contains the description of the modeling, state estimation and control procedures implemented in this Thesis for the robot.

2.1 Dynamic modeling

In this section, the procedure applied to obtain the dynamic modeling of the system is described, starting from the drawing 2.1.

2.1.1 Flight Lagrangian mechanics

The model dynamics are obtained using the Lagrangian mechanics. The points of interest considered to compute the Lagrangian are as follows:

- the multi-rotor's COM in world frame $\boldsymbol{P}_1 = \begin{pmatrix} x_b & 0 & z_b \end{pmatrix}^\intercal$;
- the leg's COM in world frame $\boldsymbol{P}_2 = \boldsymbol{P}_1 - \frac{l_{l1}}{2} \begin{pmatrix} \sin\alpha & 0 & \cos\alpha \end{pmatrix}^\intercal$;
- the first propeller's position:$\boldsymbol{P}_{p1} = \boldsymbol{P}_1 + \frac{l_b}{2} \begin{pmatrix} -\cos\theta & 0 & \sin\theta \end{pmatrix}^\intercal$
- the second propeller's position: $\boldsymbol{P}_{p2} = \boldsymbol{P}_1 + \frac{l_b}{2} \begin{pmatrix} \cos\theta & 0 & -\sin\theta \end{pmatrix}^\intercal$

The forces applied by the two propellers to the system are the following:

- $\boldsymbol{f_1} = f_{p1} \begin{pmatrix} \sin\theta & 0 & \cos\theta \end{pmatrix}^\intercal$
- $\boldsymbol{f_2} = f_{p2} \begin{pmatrix} \sin\theta & 0 & \cos\theta \end{pmatrix}^\intercal$

The System is modeled with the generalized coordinates $q = \begin{pmatrix} x_b & z_b & \theta & \alpha \end{pmatrix}^\intercal$, where x_b is the x-coordinate of the multi-rotor in the world frame, z_b is its z-coordinate in the world frame, θ is the rotation of the reference frame with respect to the world frame relative to the axis entering the x-z plane and α is the rotation of the reference frame of the leg with respect to the world frame relative to the axis entering the x-z plane. the kinetic energy of the system is:

$$K = \frac{1}{2}(m_b \dot{\boldsymbol{P}}_1^2 + m_{l1} \dot{\boldsymbol{P}}_2^2 + I_b \dot{\theta}^2 + I_{leg1} \dot{\alpha}^2) \tag{2.1}$$

which simplifies to:

$$K = \frac{1}{2} \left[m_t \left(\dot{x}_b^2 + \dot{z}_b^2 \right) + I_b \dot{\theta}^2 + \dot{\alpha}^2 \left(I_{leg1} + m_{l1} \frac{l_{l1}^2}{4} \right) + 2 m_{l1} l_{l1} \dot{\alpha} \left(\dot{z}_b \sin\alpha - \dot{x}_b \cos\alpha \right) \right] \tag{2.2}$$

Where $m_t = m_b + m_{l1}$ is the total mass of the System. The only potential energy applied in the system is due to the force of gravity:

$$U = U_g = -m_b \, \boldsymbol{P}_1 \, {}^w\boldsymbol{e}_3 g - m_{l1} \boldsymbol{P}_2 \, {}^w\boldsymbol{e}_3 g = -g(z_b \, m_t - m_{l1} \frac{l_{l1}}{2} \cos\alpha) \tag{2.3}$$

From equations 2.2 and 2.3 we obtain the Lagrangian of the system:

$$L = K + U = \frac{1}{2}\Big[m_t(\dot{x}_b^2 + \dot{z}_b^2) + I_b\dot{\theta}^2 + (I_{leg1} + m_{l1}\frac{l_{l1}^2}{4})\dot{\alpha}^2 + \\ + 2m_{l1}l_{l1}\dot{\alpha}(\dot{z}_b\sin\alpha - \dot{x}_b\cos\alpha) - g(z_b\,m_t - \frac{1}{2}m_{l1}l_{l1}\cos\alpha)\Big] \tag{2.4}$$

Including also natural damping in the joint, the generalized forces vector is:

$$\boldsymbol{F} = \frac{\partial P_{p1}}{\partial q_k}\boldsymbol{f_1} + \frac{\partial P_{p2}}{\partial q_k}\boldsymbol{f_2} + \mu\begin{pmatrix}0\\0\\-1\\1\end{pmatrix}(\dot{\theta} - \dot{\alpha}) \tag{2.5}$$

which simplifies to:

$$\boldsymbol{F} = \begin{pmatrix} sin\theta(f_{p1} + f_{p2}) \\ \cos\theta(f_{p1} + f_{p2}) \\ \frac{l_b}{2}(f_{p1} - f_{p2}) - \mu(\dot{\theta} - \dot{\alpha}) \\ -\mu(\dot{\alpha} - \dot{\theta}) \end{pmatrix} \tag{2.6}$$

Using equations 2.4 and 2.6 and the Euler-Lagrange equations:

$$\frac{\partial}{\partial t}(\frac{\partial L}{\partial \dot{q}_i}) - \frac{\partial L}{\partial q_i} = \boldsymbol{F}_i \tag{2.7}$$

we obtain the following system of equations that describe the dynamic model of the robot:

$$\begin{cases} \dfrac{\partial}{\partial t}(\dot{x}_b\,m_t - m_{l1}\frac{l_{l1}}{2}\dot{\alpha}\cos\alpha) = (f_{p1} + f_{p2})\sin\theta \\ \\ \dfrac{\partial}{\partial t}(\dot{z}_b\,m_t + m_{l1}\frac{l_{l1}}{2}\dot{\alpha}\sin\alpha) + g\,m_t = (f_{p1} + f_{p2})\cos\theta \\ \\ \dfrac{\partial}{\partial t}(\dot{\theta}I_b) = \frac{l_b}{2}(f_{p1} - f_{p2}) - \mu(\dot{\theta} - \dot{\alpha}) \\ \\ \dfrac{\partial}{\partial t}\Big[\dot{\alpha}(I_{leg1} + m_{l1}\frac{l_{l1}^2}{4}) + m_{l1}\frac{l_{l1}}{2}(\dot{z}_b\sin\alpha - \dot{x}_b\cos\alpha)\Big] + \\ -\Big[m_{l1}\frac{l_{l1}}{2}\dot{\alpha}(\dot{z}_b\cos\alpha + \dot{x}_b\sin\alpha) - g\,m_{l1}\frac{l_{l1}}{2}\sin\alpha\Big] = -\mu(\dot{\alpha} - \dot{\theta}) \end{cases} \tag{2.8}$$

Solving equation 2.8 provides the following flight dynamics for the system:

$$
\begin{cases}
\dot{x}_b = & v_{bx} \\
\dot{z}_b = & v_{bz} \\
\dot{\theta} = & \omega_\theta \\
\dot{\alpha} = & \omega_\alpha \\
\dot{v}_{bx} = & -\frac{m_{l1}}{m_t}\frac{l_{l1}}{2}\omega_\alpha^2 \sin\alpha + \frac{f_{p1}+f_{p2}}{m_t}\left[\sin\theta + \beta\cos\alpha\sin(\theta-\alpha)\right] + \\
& -\frac{m_{l1}}{m_t}\frac{l_{l1}}{2}\frac{\mu}{I_{l1}}(\omega_\alpha - \omega_\theta)\cos\alpha \\
\dot{v}_{bz} = & -\frac{m_{l1}}{m_t}\frac{l_{l1}}{2}\omega_\alpha^2 \cos\alpha + \frac{f_{p1}+f_{p2}}{m_t}\left[\cos\theta - \beta\sin\alpha\sin(\theta-\alpha)\right] + \\
& +\frac{m_{l1}}{m_t}\frac{l_{l1}}{2}\frac{\mu}{I_{l1}}(\omega_\alpha - \omega_\theta)\sin\alpha - g \\
\dot{\omega}_\theta = & \frac{l_b}{2I_b}(f_{p1} - f_{p2}) - \frac{\mu}{I_b}(\omega_\theta - \omega_\alpha) \\
\dot{\omega}_\alpha = & \frac{1}{I_{l1}}\left[(f_{p1}+f_{p2})\frac{m_{l1}}{m_t}\frac{l_{l1}}{2}\sin(\theta-\alpha) - \mu(\omega_\alpha - \omega_\theta)\right]
\end{cases}
\tag{2.9}
$$

Where $I_{l1} = I_{leg} + m_{l1}\frac{l_{l1}^2}{4}\frac{m_b}{m_t}$ and $\beta = \frac{m_{l1}^2}{I_{l1}m_t}\frac{l_{l1}^2}{4}$. In equation 2.9 v_{bx} and v_{bz} represent the velocity of the multi-rotor's Center of Mass in the world frames, while ω_θ and ω_α represent the angular velocity (along the y-axis of the world reference frame) of the frames of the multi-rotor and of the leg

2.1.2 Ground Lagrangian mechanics

The grounded system dynamics are modeled using Lagrangian mechanics and assuming no slip condition on the foot. Under this assumption, the points of interest of the system become as follow:

- the multi-rotor's COM in the world frame $\boldsymbol{P}_1 = \boldsymbol{P}_{foot} + l_{l1}\begin{pmatrix}\sin\alpha & 0 & \cos\alpha\end{pmatrix}^\intercal$;
- the leg's COM in world frame $\boldsymbol{P}_2 = \boldsymbol{P}_{foot} + \frac{l_{l1}}{2}\begin{pmatrix}\sin\alpha & 0 & \cos\alpha\end{pmatrix}^\intercal$;
- the first propeller's position:$\boldsymbol{P}_{p1} = \boldsymbol{P}_1 + \frac{l_b}{2}\begin{pmatrix}-\cos\theta & 0 & \sin\theta\end{pmatrix}^\intercal$
- the second propeller's position: $\boldsymbol{P}_{p2} = \boldsymbol{P}_1 + \frac{l_b}{2}\begin{pmatrix}\cos\theta & 0 & -\sin\theta\end{pmatrix}^\intercal$

with $\dfrac{\partial}{\partial t}\boldsymbol{P}_{foot} = \boldsymbol{0}$.
The forces applied by the two propellers to the system are:

- $\boldsymbol{f_1} = f_{p1}\begin{pmatrix}\sin\theta & 0 & \cos\theta\end{pmatrix}^\intercal$
- $\boldsymbol{f_2} = f_{p2}\begin{pmatrix}\sin\theta & 0 & \cos\theta\end{pmatrix}^\intercal$

The generalized coordinates considered are: $q = \begin{pmatrix}\theta & \alpha\end{pmatrix}^\intercal$, where θ is the rotation of the reference frame with respect to the world frame relative to the axis entering the x-z plane and α is the rotation of the leg's reference frame with respect to the world frame relative to the axis entering the x-z plane.

The kinetic energy of the system is:

$$K = \frac{1}{2}(m_b \dot{\boldsymbol{P}}_1^2 + m_{l1} \dot{\boldsymbol{P}}_2^2 + I_b \dot{\theta}^2 + I_{leg1} \dot{\alpha}^2) \tag{2.10}$$

which simplifies to:

$$K = \frac{1}{2}\left\{ I_b \dot{\theta}^2 + \dot{\alpha}^2 [I_{leg1} + l_{l1}^2 (m_b + \frac{m_{l1}}{4})] \right\} \tag{2.11}$$

The only potential energy applied in the system is due to the force of gravity:

$$U = U_g = -m_b \, \boldsymbol{P}_1 \, {}^w\boldsymbol{e}_3 g - m_{l1} \boldsymbol{P}_2 \, {}^w\boldsymbol{e}_3 g = -g l_{l1} \cos\alpha (m_b + \frac{m_{l1}}{2}) \tag{2.12}$$

Combining equations 2.11 and 2.12, the Lagrangian of the system is:

$$L = K + U = \frac{1}{2}\left\{ I_b \dot{\theta}^2 + \dot{\alpha}^2 [I_{leg1} + l_{l1}^2 (m_b + \frac{m_{l1}}{4})] \right\} - g \, l_{l1} \cos\alpha (m_b + \frac{m_{l1}}{2}) \tag{2.13}$$

The generalized forces of the system, including the leg joint damping, are as follow:

$$\boldsymbol{F} = \begin{pmatrix} \frac{l_b}{2}(f_{p1} - f_{p2}) - \mu(\dot{\theta} - \dot{\alpha}) \\ l_{l1}(f_{p1} + f_{p2}) \sin(\theta - \alpha) - \mu(\dot{\alpha} - \dot{\theta}) \end{pmatrix} \tag{2.14}$$

Using equations 2.4 and 2.6 and the Euler-Lagrange equations:

$$\frac{\partial}{\partial t}(\frac{\partial L}{\partial \dot{q}_i}) - \frac{\partial L}{\partial q_i} = \boldsymbol{F}_i \tag{2.15}$$

we obtain the following equations for the grounded dynamics of the robot:

$$\begin{cases} \dot{x}_b = v_{bx} \\ \dot{z}_b = v_{bz} \\ \dot{\theta} = \omega_\theta \\ \dot{\alpha} = \omega_\alpha \\ v_{bx} = l_{l1} \, \omega_\alpha \cos\alpha \\ v_{bz} = -l_{l1} \, \omega_\alpha \sin\alpha \\ \dot{\omega}_\theta = \frac{l_b}{2I_b}(f_{p1} - f_{p2}) - \frac{\mu}{I_b}(\omega_\theta - \omega_\alpha) \\ \dot{\omega}_\alpha = \frac{(f_{p1} + f_{p2})}{I_{l1}} \sin(\theta - \alpha) - \frac{\mu}{I_{l1}}(\dot{\alpha} - \dot{\theta}) \end{cases} \tag{2.16}$$

where $I_{l1} = I_{leg} + l_{l1}^2 (m_b + \frac{m_{l1}}{4})$, v_{bx} and v_{bz} represent the velocity of the multi-rotor's COM in the world frames, while ω_θ and ω_α represent the angular velocity (along the y-axis of the world reference frame) of the frames of the multi-rotor and of the leg

2.2 State Estimation

For the estimation of the state an Extended Kalman filter has been implemented. This choice was made in order to address the parameter, modeling and input uncertainty and the noisiness of the sensors included, two IMUs and a GPS module.

2.2.1 System Model

A Kalman filter works with a discrete-time linear model expressed in state-space form:

$$\begin{cases} \boldsymbol{x}_{k+1} = F_k \boldsymbol{x}_k + B_k \boldsymbol{u}_k + \boldsymbol{\xi}_k \\ \boldsymbol{z}_k = H_k \boldsymbol{x}_k + D_k \boldsymbol{u}_k + \boldsymbol{w}_k \end{cases}, \tag{2.17}$$

Where v_k is the value of the vector/matrix v at the k_{th} discrete time instant (i.e. $v(k\,dt)$), ξ_k and w_k are gaussian noises with covariance matrices of value Q_k and R_k related with disturbances, unmodeled dynamics, noises and parameter uncertainties. The Model and covariance matrices are used in the prediction and update stages to compute the state estimate ($\hat{x}_k$) and its covariance matrix (P_k). An extended Kalman filter, on the other hand, operates with a nonlinear system:

$$\begin{cases} \boldsymbol{x}_{k+1} = f_d(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{\xi}_k \\ \boldsymbol{z}_k = h(\boldsymbol{x}_k, \boldsymbol{u}_k) + \boldsymbol{w}_k \end{cases} \tag{2.18}$$

The algorithm used by the extended Kalman filter is the same as the algorithm for the Kalman filter, but the matrices used are the Jacobian matrices of the functions of the model:

$$F_k = \left[\frac{\partial \mathbf{f_d}(\mathbf{x}, \mathbf{u})}{\partial x_1} \quad \cdots \quad \frac{\partial \mathbf{f_d}(\mathbf{x}, \mathbf{u})}{\partial x_n} \right] \Bigg|_{x=\hat{x}_{k|k}, u=u_k} \tag{2.19}$$

$$B_k = \left[\frac{\partial \mathbf{f_d}(\mathbf{x}, \mathbf{u})}{\partial u_1} \quad \cdots \quad \frac{\partial \mathbf{f_d}(\mathbf{x}, \mathbf{u})}{\partial u_n} \right] \Bigg|_{x=\hat{x}_{k|k}, u=u_k} \tag{2.20}$$

$$H_k = \left[\frac{\partial \mathbf{h}(\mathbf{x}, \mathbf{u})}{\partial x_1} \quad \cdots \quad \frac{\partial \mathbf{h}(\mathbf{x}, \mathbf{u})}{\partial x_n} \right] \Bigg|_{x=\hat{x}_{k|k}, u=u_k} \tag{2.21}$$

The complete analytical formulas for these Jacobian matrices are described in appendix A

2.2.2 Dynamic Model Discretization

The state-space equations 2.9 and 2.16 , described in section 2.1, are continuous-time; therefore for the implementation of the extended Kalman filter we needed to discretize the equations:

$$\boldsymbol{f}_d(\boldsymbol{x}, \boldsymbol{u}) = \boldsymbol{x} + dt\, \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}) \tag{2.22}$$

Equation 2.22 applied to the robot becomes the following equation:

$$\boldsymbol{x}_k = \begin{pmatrix} x_b(k-1) + dt\, v_{bx}(k-1) \\ z_b(k-1) + dt\, v_{bz}(k-1) \\ \theta(k-1) + dt\, \omega_\theta(k-1) \\ \alpha(k-1) + dt\, \omega_\alpha(k-1) \\ v_{bx}(k-1) + dt\, \dot{v}_{bx}(k-1) \\ v_{bz}(k-1) + dt\, \dot{v}_{bz}(k-1) \\ \omega_\theta(k-1) + dt\, \dot{\omega}_\theta(k-1) \\ \omega_\alpha(k-1) + dt\, \dot{\omega}_\alpha(k-1) \end{pmatrix} \tag{2.23}$$

the values of $\dot{v}_{bx}$, $\dot{v}_{bz}$, $\dot{\omega}_\theta$ and $\dot{\omega}_\alpha$ are calculated exactly as the values of $\ddot{x}_b$, $\ddot{z}_b$, $\ddot{\theta}$ and $\ddot{\alpha}$ from equation 2.9 if the robot is in the air or from equation 2.16 when the robot is touching the ground.

2.2.3 Measurements

The measurements implemented in the Extended Kalman Filter to correct its estimates are provided by two IMUs and one GPS module installed on the robot. The IMUs are placed on the COM of the multi-rotor and on the leg's COM, while the GPS module is placed l_{GPS} along the x-axis of the reference frame of multi-rotor relative to the COM of the multi-rotor. The frame of each sensor is aligned with the body frame of the part of the robot it is installed on.

2.2.3.1 IMUs

The two IMUs measure the angular velocity and the acceleration of the COM of the multi-rotor and the leg, in the local reference frame of the two components of the system (rotated of θ and α along the y-axis with respect to the world frame). The output function describing the relationship between the measured values and the state is expressed differently when the robot is flying or when the robot is on the ground. This is because the robot is affected by the contact forces when on the ground. The state of the system is mapped to the output of the two IMUs

while flying by equation 2.24, while the relationship between the state and the IMU output when the robot is grounded is described in equation 2.25.

$$\boldsymbol{h}_{IMU}(\boldsymbol{x},\boldsymbol{u}) = \begin{pmatrix} \omega_\theta \\ \frac{m_{l1}}{m_t}\frac{l_{l1}}{2}\left(\dot{\omega}_\alpha \cos(\theta-\alpha) - \omega_\alpha^2 \sin(\theta-\alpha)\right) \\ \frac{f_{p1}+f_{p2}}{m_t} + \frac{m_{l1}}{m_t}\frac{l_{l1}}{2}\left(\dot{\omega}_\alpha \sin(\theta-\alpha) + \omega_\alpha^2 \cos(\theta-\alpha)\right) \\ \omega_\alpha \\ \frac{f_{p1}+f_{p2}}{m_t}\sin(\theta-\alpha) - \frac{m_b}{m_t}\frac{l_{l1}}{2}\dot{\omega}_\alpha \\ \frac{f_{p1}+f_{p2}}{m_t}\cos(\theta-\alpha) - \frac{m_b}{m_t}\frac{l_{l1}}{2}\omega_\alpha^2 \end{pmatrix} \tag{2.24}$$

$$\boldsymbol{h}_{IMU}(\boldsymbol{x},\boldsymbol{u}) = \begin{pmatrix} \omega_\theta \\ l_{l1}\left(\dot{\omega}_\alpha \cos(\theta-\alpha) - \omega_\alpha^2 \sin(\theta-\alpha)\right) - g\sin\theta \\ l_{l1}\left(\dot{\omega}_\alpha \sin(\theta-\alpha) - \omega_\alpha^2 \cos(\theta-\alpha)\right) + g\cos\theta \\ \omega_\alpha \\ \frac{l_{l1}}{2}\dot{\omega}_\alpha - g\sin\alpha \\ \frac{l_{l1}}{2}\omega_\alpha^2 + g\cos\alpha \end{pmatrix} \tag{2.25}$$

2.2.3.2 GPS

The GPS module provides its position in the world reference frame. The output function describing its output's relationship with the state (excluding measurement noise) is described in the following equation:

$$\boldsymbol{h}_{GPS}(\boldsymbol{x},\boldsymbol{u}) = \begin{pmatrix} x_b + l_{GPS}\cos\theta \\ z_b - l_{GPS}\sin\theta \end{pmatrix} \tag{2.26}$$

2.2.4 Model Disturbances

The process disturbance is modeled as a white noise disturbance on each state variable. As such, its covariance matrix is modeled in the following way:

$$Q_k = \begin{bmatrix} \sigma^2_{proc1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma^2_{proc8} \end{bmatrix} \tag{2.27}$$

w_k is instead modeled as white noise added directly to the sensor's output on each axis and has the following covariance matrices:

$$R_{IMU} = \begin{bmatrix} \sigma^2_{gyro_1} & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma^2_{acc_1} & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma^2_{acc_1} & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma^2_{gyro_2} & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma^2_{acc_2} & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma^2_{acc_2} \end{bmatrix} \tag{2.28}$$

$$R_{GPS} = \begin{bmatrix} \sigma_{GPS}^2 & 0 \\ 0 & \sigma_{GPS}^2 \end{bmatrix} \tag{2.29}$$

2.2.5 Operation stages

The Extended Kalman Filter operates on two computation stages: the prediction stage, where the next state is calculated from the current estimate and the input to the system, and the update stage, where the output of the sensors is compared with the output function estimate and the error between these two vectors is used to correct the state estimate.

2.2.5.1 Prediction stage

In the prediction stage the computation based only on the *a priori* knowledge of the system, meaning that the state and covariance matrix are updated based on the input of the system:

$$\hat{\boldsymbol{x}}_{k|k-1} = \boldsymbol{f}_d(\hat{\boldsymbol{x}}_{k-1|k-1}, \boldsymbol{u}_{k-1}) \tag{2.30}$$

$$P_{k|k-1} = F_{k-1}\, P_{k-1|k-1}\, F_{k-1}^{\intercal} + Q \tag{2.31}$$

2.2.5.2 Update stage

In the update stage the output (or part of the output) z_k, if available at the current time instant, is used to update the state estimate and its covariance:

- At first the measurement estimate is computed:

$$\hat{\boldsymbol{z}}_{k|k-1} = \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{u}_k)$$

- Then the measure uncertainty matrix is computed:

$$S_k = H_k\, P_{k|k-1}\, H_k^{\intercal} + R \tag{2.32}$$

- Then the optimal Kalman gain is evaluated:

$$K_k = P_{k|k-1}\, H_k^{\intercal}\, S_k^{-1} \tag{2.33}$$

- Finally the measure uncertainty matrix and the Kalman gain are used to update the state estimate and covariance matrix:

$$\hat{\boldsymbol{x}}_{k|k} = \hat{\boldsymbol{x}}_{k|k-1} + K_k\, (\boldsymbol{z}_k - \hat{\boldsymbol{z}}_{k|k-1}) \tag{2.34}$$

$$P_{k|k} = P_{k|k-1} - K_k\, S_k\, K_k^{\intercal} \tag{2.35}$$

2.2.6 State initialization

The state is initialized by adding gaussian noise to the initial state of the simulation, and the state covariance matrix is initialized as a diagonal matrix identical to the covariance matrix of the initial noise:

$$\begin{cases} \hat{\boldsymbol{x}}_0 = \boldsymbol{x}_{sim}^{start} + \boldsymbol{\xi}_0 \\ \Sigma = \begin{bmatrix} \sigma_{start0}^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sigma_{start8}^2 \end{bmatrix} \\ \boldsymbol{\xi}_0 \approx \mathcal{N}(0, \Sigma) \\ P_0 = \Sigma \end{cases} \tag{2.36}$$

2.2.7 Model switching

The switch between ground and flight model is handled through a touch sensor on the foot: if a force intensity above a minimum threshold is measured the ground model is applied, otherwise the flight model is applied.

Algorithm 1 algorithm describing the sequence of instruction implemented to apply the switch between the grounded and the mid-air models of the robot.

1: $contact\_force \leftarrow read\_measure(foot\_touch\_sensor)$
2: $grounded \leftarrow contact\_force > contact\_force\_threshold$

2.3 Control

The control strategy is characterized by a cascading structure: at first the translational control forces $\boldsymbol{f}_{lin}$ are computed and used to set the total control thrust force λ^* and target orientation of the multi-rotor (θ^{ref}), which is then used to regulate the torque applied on the multi-rotor M_θ. The control thrust force and the control torque are then split in the desired propeller force intensity f_{p1}, f_{p2}. All these virtual forces and moments are considered as if applied on the multi-rotor's COM, since that is the cumulative effect of the forces applied by the propellers on the multi-rotor. Figure 2.2 shows these forces and its control components on the drawing of the robot.

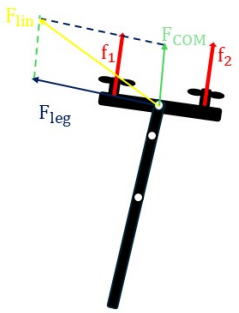


Figure 2.2: simple drawing of the robot with the propeller and virtual control forces drawn.

2.3.1 Linear forces

The translational forces $\boldsymbol{f}_{lin}$ is the resultant of three components:

- the gravity compensation force $\boldsymbol{f}_g$;
- the COM control force $\boldsymbol{f}_{COM}$;
- and the leg control force $\boldsymbol{f}_\alpha$.

.

$$\boldsymbol{f}_{lin} = \boldsymbol{f}_{COM} + \boldsymbol{f}_\alpha + \boldsymbol{f}_g \tag{2.37}$$

2.3.1.1 Gravity compensation forces

The gravity compensation force $\boldsymbol{f}_g$ has a different value depending on whether the robot is flying ($\boldsymbol{f}_g^{flight}$) or on the ground ($\boldsymbol{f}_g^{ground}$):

$$\boldsymbol{f}_g^{flight} = (m_b + m_{l1})\, \boldsymbol{e}_3\, g \tag{2.38}$$

$$\boldsymbol{f}_g^{ground} = (m_b + \frac{m_{l1}}{2})\, \boldsymbol{e}_3\, g \tag{2.39}$$

with $\boldsymbol{e}_3 = \begin{pmatrix} 0 & 0 & 1 \end{pmatrix}^\intercal$.

2.3.1.2 COM forces

The control force for the COM $\boldsymbol{f}_{COM}$ is designed in a back-stepping approach, with a dynamic layer providing velocity control and a kinematic layer computing the reference velocity $\boldsymbol{v}_{\boldsymbol{COM}}^{\boldsymbol{*}}$ with different possible strategies depending on the current phase of the robot:

- trajectory tracking:

$$\boldsymbol{P}_{COM} = \frac{1}{m_t}(m_b\, \boldsymbol{P}_1 + m_{l1}\, \boldsymbol{P}_2) = \begin{pmatrix} x_b \\ 0 \\ z_b \end{pmatrix} - \frac{m_{l1}}{m_t}\frac{l_{l1}}{2} \begin{pmatrix} \sin\alpha \\ 0 \\ \cos\alpha \end{pmatrix}$$

$$\boldsymbol{v}_{COM} = \begin{pmatrix} \dot{x}_b \\ 0 \\ \dot{z}_b \end{pmatrix} + \frac{m_{l1}}{m_t}\, \frac{l_{l1}}{2}\, \dot{\alpha}_1 \begin{pmatrix} -cos(\alpha) \\ 0 \\ sin(\alpha) \end{pmatrix}$$

$$\boldsymbol{v}_{COM}^{*} = \gamma_v\, (\boldsymbol{P}_{COM}^{ref} - \boldsymbol{P}_{COM}) + \boldsymbol{v}_{COM}^{ref} \tag{2.40}$$

- constant reference velocity:

$$\boldsymbol{v}_{COM}^{*} = \boldsymbol{v}_{COM}^{ref}$$

The dynamic layer controls the velocity by setting the goal acceleration $\boldsymbol{a}_{COM}^{*}$ proportional to the velocity error $(\boldsymbol{v}_{COM}^{*} - \boldsymbol{v}_{COM})$ and then adding the goal acceleration $\boldsymbol{a}^{ref}$:

$$\boldsymbol{a}_{COM}^{*} = \gamma_a\, (\boldsymbol{v}_{COM}^{*} - \boldsymbol{v}_{COM}) + \boldsymbol{a}^{ref} \tag{2.41}$$

$$\boldsymbol{f}_{COM} = m_t\, \boldsymbol{a}_{COM}^{*} \tag{2.42}$$

2.3.1.3 Leg control

The leg control force $\boldsymbol{f}_\alpha$ is computed as the force perpendicular to the leg needed to apply a virtual dampening moment M_α, with added compensation for the leg's dynamics:

$$\dot{\omega}_\alpha^{ref} = -\gamma_{\omega_\alpha}\,\omega_\alpha \tag{2.43}$$

$$M_\alpha = \dot{\omega_\alpha}^{ref}\, I_{l1} + \mu\,(\omega_\alpha - \omega_\theta) \tag{2.44}$$

$$\boldsymbol{f}_\alpha = \frac{M_\alpha}{l_{l1}} \begin{pmatrix} cos(\alpha) \\ 0 \\ -sin(\alpha) \end{pmatrix} \tag{2.45}$$

This force is computed and added to the translational forces only because the joint between the multi-rotor and the rest of the robot is not actuated with a motor.

2.3.2 Multi-rotor angle control

For the multi-rotor angle control a back-stepping approach is applied:

- At first the kinematic layer sets the target angular velocity ω_θ^{ref} proportional to the error between the current multi-rotor orientation and the direction (θ^{ref}) of the desired translational control forces $\boldsymbol{f}_{lin}$; if too high the target angular velocity is saturated;
- Then the dynamic layer sets the target multi-rotor moment M_θ proportional to the angular velocity error ($\omega_{theta}^{ref} - \omega_\theta$) with compensation for the model dynamics.

$$\begin{cases} \lambda = \|\boldsymbol{f}_{lin}\| \\ \hat{\boldsymbol{f}}_{lin} = \frac{\boldsymbol{f}_{lin}}{\|\boldsymbol{f}_{lin}\|} \end{cases} \tag{2.46}$$

$$\theta^{ref} = atan(\frac{\hat{f}_{lin_x}}{\hat{f}_{lin_z}}) \tag{2.47}$$

$$\omega_\theta^{ref} = \gamma_\theta\,(\theta^{ref} - \theta) \tag{2.48}$$

$$\dot{\omega}_\theta^{ref} = \gamma_{\omega_\theta}\,(\omega_{theta}^{ref} - \omega_\theta) \tag{2.49}$$

$$M_\theta = I_b\,\dot{\omega}_\theta^{ref} + \mu\,(\omega_\theta - \omega_\alpha) \tag{2.50}$$

2.3.3 Propeller intensity

The control thrust force λ and the control multi-rotor torque M_θ are distributed between the thrust of the two propellers (f_{p1}, f_{p2}) by solving the following system of equations:

$$\begin{cases} f_{p1} + f_{p2} = \lambda \\ \frac{l_b}{2}(f_{p1} - f_{p2}) = M_\theta \end{cases} \tag{2.51}$$

Equation 2.51 results in the following values for the propellers intensities:

$$f_{p1} = \frac{\|\boldsymbol{f}_{lin}\|}{2} + \frac{M_\theta}{l_b} \tag{2.52}$$

$$f_{p2} = \frac{\|\boldsymbol{f}_{lin}\|}{2} - \frac{M_\theta}{l_b} \tag{2.53}$$

Chapter 3

System Simulation

The control and state estimation of the simplified model has been tested inside of a MuJoCo (3.3.5) simulation.

3.1 MuJoCo

MuJoCo (**Mu**lti-**Jo**ints dynamics with **Co**ntact) is a general purpose physics engine that operates directly on generalized coordinates (instead of numerically imposing the joint constraints and computing the dynamics on cartesian space) while also providing modern and efficient contact dynamics methods. MuJoCo is built in C/C++ and provides a C API and python bindings. It allows for environment/systems definitions using URDf files, MJCF files (xml files with MuJoCo-specific extensions), or inside of a program using the APIs provided by the engine. In MuJoCo the fluid dynamics simulation capabilities are limited to very simple models, which is the main reason the modeling, state estimation and control of the robot are based directly on the thrust of the propellers and not on their angular velocity.

3.2 Robot definition

The robot is realized as a MJCF model composed by two bodies. The first body part is a horizontal bar with two sliding joints, one along the x-axis and one along the z-axis, and one rotational joint defining the degrees of freedom of the robot to reproduce the multi-rotor. The second component of the robot is a vertical bar connected to the horizontal one with a damped joint that replicates the leg. The propellers are simulated with force actuators oriented along the z-axis of the horizontal bar. MuJoCo does not provide actual IMUs among its default sensors but provides sensor definitions for both accelerometers and gyroscopes. The two

IMUs are realized by placing an accelerometer and a gyroscope on the same position. MuJoCo also does not provide a true GPS sensor, but was replicated using a *framepos* sensor, which provides the position of a frame with respect to another frame (world frame by default). Zero-mean Gaussian noise was added to the output of the sensors after reading the measures from the simulation data to simulate the noise from real-life sensors. Contact with the ground is detected using a *touch* sensor installed on the foot that measures the norm of the contact forces acting on the foot. The output from the touch sensor is used to detect ground contact and takeoff by the model switch algorithm, as described in algorithm 1.

3.3 Simulation Environment

The simulation shows the robot in four consecutive phases, handled with a state machine:

- During phase 0 the robot starts on the ground and follows the task of standing upright, i.e. the robot COM is above the contact point with the ground);
- When given the command (through a key press or 10 seconds after the start of the simulation), phase 1 starts and the robot tracks a parabolic trajectory with vertical acceleration smaller than or equal to the gravity acceleration in order to jump over a box placed in front of it;
- When in the "descending" arc of the parabolic trajectory and below a certain height above the ground, the robot starts phase 2: the robot is tasked to move at a constant reference velocity: ${}^{w}\boldsymbol{v}_{\boldsymbol{COM}}^{\boldsymbol{ref}} = \boldsymbol{v}_{\boldsymbol{land}} = \begin{bmatrix} 0 & 0 & -0.05 \end{bmatrix}^{\intercal} m/s$ until ground contact is detected;
- After ground contact is detected, the robot returns to the task of phase 0, stabilizing itself from the landing impact and reaching the standing position. A press of the jump key button would start a new phase 1 with a new trajectory.

3.4 Iteration sequence for the robot's simulation

Algorithm 2 shows the pseudo code that describes the implementation of the control and estimation loop of the simulated robot. As specified in the Algorithm, at every iteration loop the estimation is performed before the computations related

to the control of the robot. A diagram of the control and estimation loop is also shown in Figure 3.1.

Algorithm 2 Pseudo code describing the implementation of the estimation and control loop for the system.

$k \leftarrow 1$
$\xi^{start} \leftarrow \mathcal{N}(0, \Sigma_{start})$
$x_0 \leftarrow x^{start} + \xi^{start}$
$P_0 \leftarrow \Sigma_{start}$
$\Sigma_p \leftarrow Diag(\sigma_{p1}^2, \sigma_{p2}^2)$
while $robot\_running$ **do**
  $grounded_k \leftarrow detect\_ground\_contact()$
  $F_{k-1}, B_{k-1} \leftarrow Model\_jacobian\_matrix(x_{k-1}, u_{k-1}, grounded)$
  $Q_{k-1} \leftarrow Disturbance\_matrix(B_{k-1}, \Sigma_p)$
  $x_k, P_k \leftarrow EKF\_predict(x_{k-1}, u_{k-1}, P_{k-1}, F_{k-1}, Q_{k-1}, grounded)$
  if $IMU\_measure\_available$ **then**
    $z_k \leftarrow read\_IMU\_measurement()$
    $H_k^{IMU} \leftarrow IMU\_measurement\_jacobian\_matrix(x_k, u_{k-1}, grounded)$
    $x_k, P_k \leftarrow EKF\_update(x_k, h^{IMU}(x_k, u_{k-1}, grounded), P_k, H_k^{IMU}, R_{IMU})$
  end if
  if $GPS\_measure\_available$ **then**
    $z_k \leftarrow read\_GPS\_measurement()$
    $H_k^{GPS} \leftarrow GPS\_meas\_jacob\_mat(x_k)$
    $x_k, P_k \leftarrow EKF\_update(x_k, h^{GPS}(x_k), P_k, H_k^{GPS}, R_{GPS})$
  end if
  $u_k \leftarrow control\_law(x_k, grounded)$
  $send\_to\_actuators(u_k)$
  $k \leftarrow k + 1$
end while

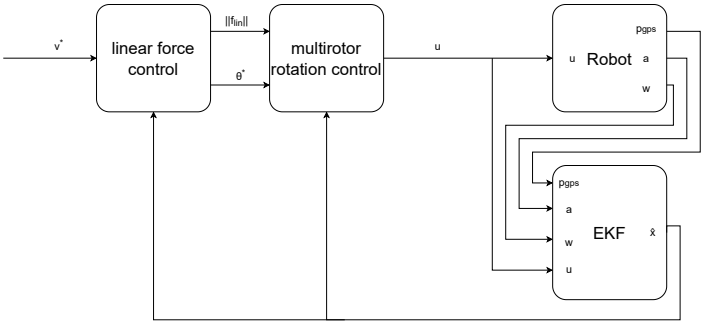


Figure 3.1: diagram showing the estimation and control loop implemented to control the robot.

3.5 Simulation layout and parameters

An overlay of the robot during the four main stages of the simulation (start of the simulation, start of parabolic trajectory tracking, start of landing sequence, standing position after landing) is shown in picture 3.2

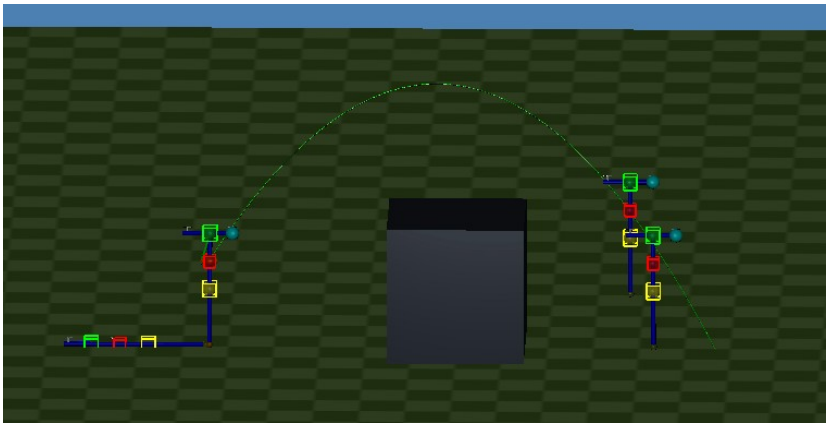


Figure 3.2: environment and Initial robot configuration for the jumping experiment.

At this link is also available a recording of the robot performance during the whole simulation.

In the picture and the video the programmed jumping trajectory is displayed during the jumping phase, and the performance of the estimation is visually shown by displaying the simulated (line edges of boxes) and estimated (transparent spheres) position of:

- multi-rotor COM (green ■);
- leg COM (yellow ■);
- forward-facing propeller (cyan ■);
- robot COM (red ■).

The desired COM position is also shown when used by the controller, represented as an orange ■ ellipsoid.

The linear control forces are also displayed (as arrows originating from the multi-rotor's COM):

- total control force (without gravity compensation)(white)
- COM control force (green ■)
- leg control force (blue ■)

In the following pictures are included some plots showing the performance of the state estimator and of the controller, over 3 experiments:

- experiment 1: the simulated gyroscope and accelerometers are set with the default covariance values considered; ($\sigma_{gyro} = 10^{-3}\frac{rad^2}{s^2}$, $\sigma_{acc} = 10^{-3}\frac{m^2}{s^4}$);
- experiment 2: a white gaussian noise of higher variance than in simulation 1 is added to the simulated IMU sensors (($\sigma_{gyro} = 10^{-2}\frac{rad^2}{s^2}$, $\sigma_{acc} = 10^{-2}\frac{m^2}{s^4}$);
- experiment 3: the simulated gyroscopes and accelerometers have reduced variance with respect to the first simulation (($\sigma_{gyro} = 10^{-4}\frac{rad^2}{s^2}$, $\sigma_{acc} = 10^{-4}\frac{m^2}{s^4}$).

All the parameters used for the implementation of the estimator and of the controller are listed in tables 3.1 and 3.2.

control gain	value	saturation velocity	value
γ_{COM}	1.5	$v_{COM}^{max}(grounded)$	0.2
$\gamma_{v_{COM}}$	10	$v_{COM}^{max}(flight)$	0.5
γ_{θ}	10	ω_{θ}^{max}	0.5
$\gamma_{\omega_{\theta}}$	100	ω_{α}^{max}	0.5
$\gamma_{\omega_{\alpha}}$	15	-	-

Table 3.1: Table listing all the control parameters used in the simulations.

parameter	value
R_{IMU}	$\begin{bmatrix} 0.001 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.001 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.001 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.001 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.001 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.001 \end{bmatrix}$
R_{GPS}	$\begin{bmatrix} 0.5 & 0 \\ 0 & 0.5 \end{bmatrix}$
Q	$\begin{bmatrix} 10^{-5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 10^{-6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 10^{-4} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 10^{-6} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 5 \cdot 10^{-3} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 \cdot 10^{-3} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 \cdot 10^{-4} & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 10^{-6} \end{bmatrix}$
Σ_{start}	$\begin{bmatrix} 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.07 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.07 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.02 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0.02 \end{bmatrix}$

Table 3.2: Table listing all the estimator parameters used in the simulations.

3.6 Simulation 1

This simulation was performed adding to the simulated IMUs the nominal noise value (see section 3.5). The plots can be analyzed in 4 different phases: phase 0, in the time intervals $[0, 10] \cup (20.1, 30]s$; phase 1, in the time interval $(10, 17.3]s$; phase 2, in the time interval $(17.3, 20.1]s$. The robot's estimation performance for this simulation is shown in figures 3.3 to 3.9.

Figure 3.3 shows the comparison over time between the simulation truth value and the estimation of the multi-rotor orientation and of its angular velocity relative to the world frame (θ and ω_θ in the state of the model). The estimation follows almost perfectly the actual simulated value after the first few updates of the filter.

In the plots for the orientation there are 3 noticeable spikes: the first one, in the time interval $[1, 3]s$ of the simulation, corresponds to the tilt of the multi-rotor necessary to raise the COM of the robot to the desired standing position; the one just after $10s$ is due to the start of the jumping phase and the related rapid change in COM reference velocity; the peak at around $17s$ is due to the end of phase 1 and of the parabolic trajectory tracking and the start of the vertical landing phase, with a consequent rapid change in COM reference velocity.

The estimate is slightly noisy, with a root mean square error of $6.8 \cdot 10^{-3} rad$ for the angle and $1.3 \cdot 10^{-2} \frac{rad}{s}$ for the angular velocity. The signal has visible oscillations over time. These are related to the noise in the position and velocity estimation of the COM of the robot resulting oscillations in the desired multi-rotor orientation.

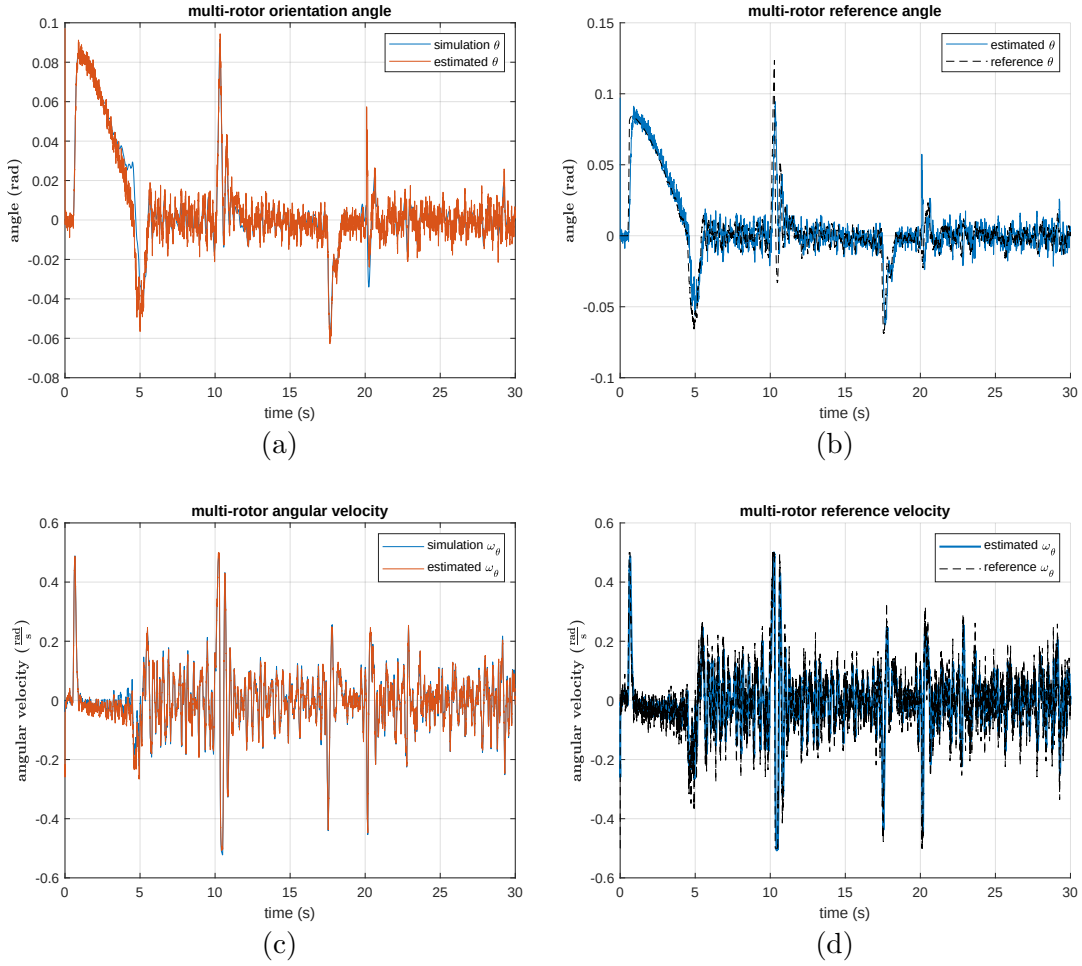
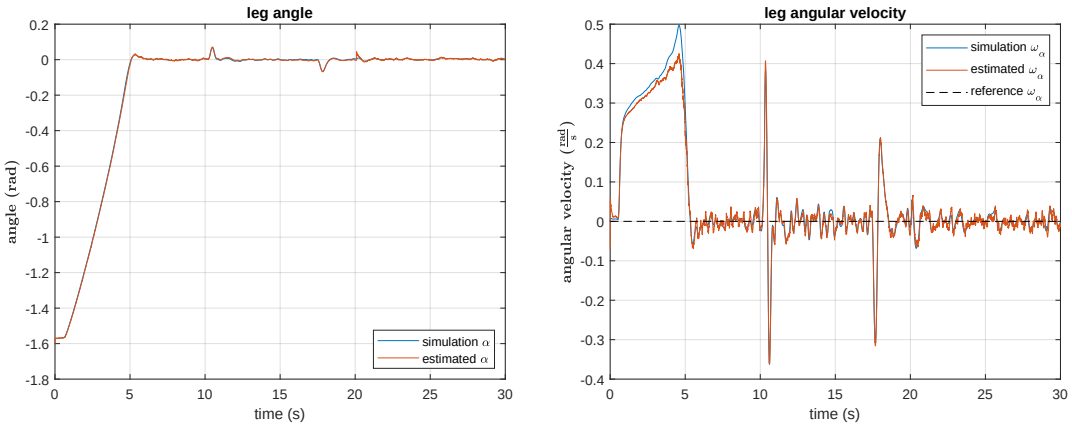


Figure 3.3: comparison over time of the values of the simulated and estimated multi-rotor angle (a), estimated and desired multi-rotor angle (b), simulated and estimated angular velocity (c), estimated and desired angular velocity (d) over time for simulation 1.

Figure 3.4 contains the plots of the estimated values of the leg orientation and angular velocity with respect to the world frame (α and ω_α in the state of the model) compared with the corresponding values from the simulation. The two graphs show good convergence of the estimated values towards the simulation truth, excluding a slight velocity error during the standing up phase of the simulation. The 3 spikes corresponding to the robot standing up ($[1, 3]s$), the start of phase 1 and the start of phase 2 are visible. The last two are caused by the rapid change in desired velocity of the COM of the robot, leading to high acceleration for the robot COM which affect also the angular acceleration of the leg. This sudden acceleration cannot be countered directly due to the lack of motor

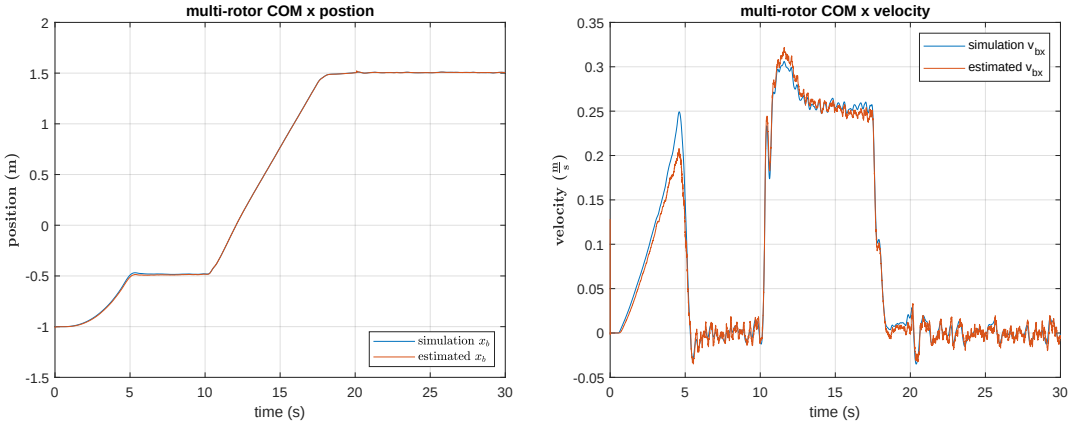
installed on the leg joint, but is controlled using the control strategy described in section 2.3. The noise is noticeable but tiny, with a root mean square error between the simulated and estimated signal of $4.6 \cdot 10^{-3} rad$ for the angle and $1.62 \cdot 10^{-2} \frac{rad}{s}$ for the angular velocity.



(a) (b)

Figure 3.4: value of the leg angle (a) and angular velocity (b) over time for simulation 1.

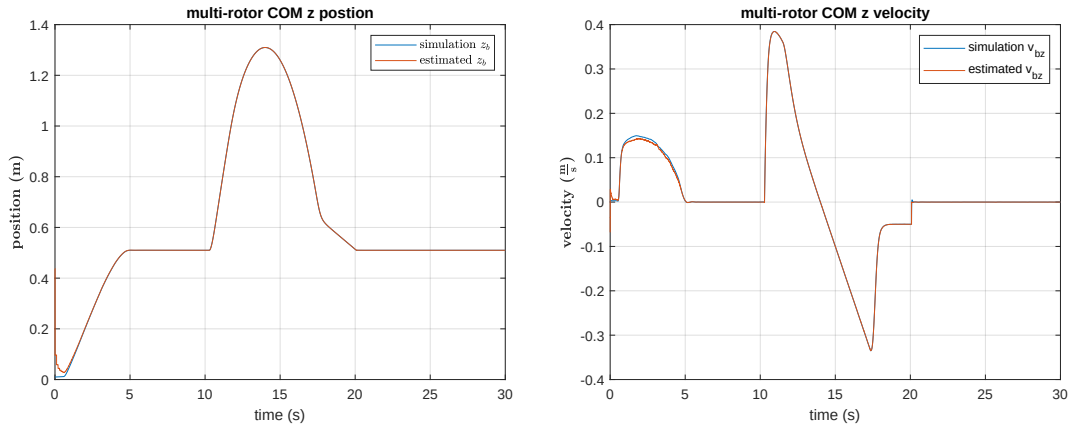
Figure 3.5 shows the evolution of the estimated position and velocity of the multi-rotor along the x-axis compared with the values from the simulation. The position estimation has little noise, with a root mean square error of $5.4 \cdot 10^{-3} m$, while the velocity estimation is more disturbed, with a root mean square error of $9.5 \cdot 10^{-3} \frac{m}{s}$. During the time intervals corresponding to phase 0 there are noticeable similarities between the x-axis velocity of the COM of the multi-rotor (3.5b) and the angular velocity of the leg (3.4b): in these time intervals the robot is grounded and, as described in paragraph 2.1.2, the velocity of the COM of the multi-rotor is computed in function of the leg angular velocity (based on a no-slip assumption on the foot of the robot). During most of the simulation the value over time of both the position and the velocity of the COM of the multi-rotor are very similar to the position and velocity of the COM of the robot (3.7). In the plots are noticeable the planned movement trajectories of the COM of the robot during the various phases of the simulation (after the control inputs converge to zero): static standing position during phase 0 and after $5s$ from the start of the simulation, non-zero constant velocity for parabolic trajectory in phase 1 and zero-velocity for the vertical landing during phase 2.



(a) (b)

Figure 3.5: value of the multi-rotor COM x-axis position (a) and velocity (b) over time for simulation 1.

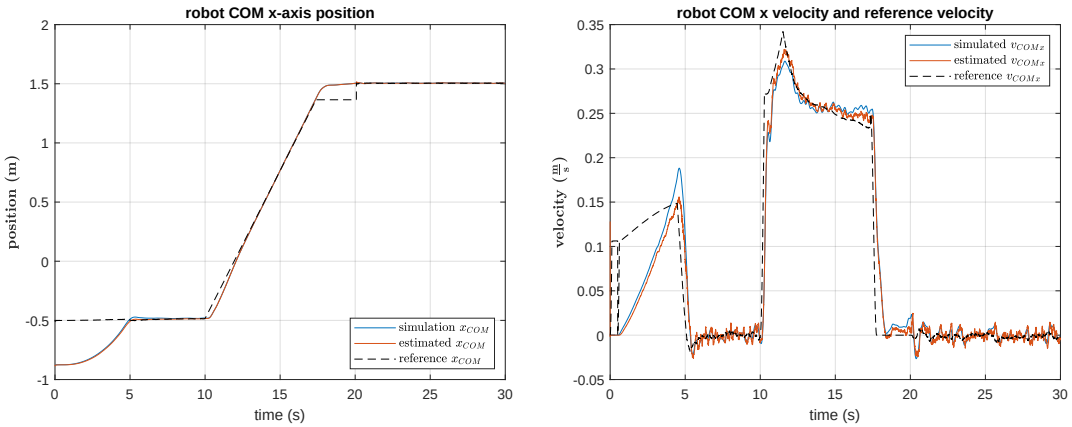
Figure 3.6 shows the plots comparing the estimates and the simulated values for the position and velocity along the z-axis of the COM of the multi-rotor. The estimation is very close to the simulated value after the first update cycles at the beginning of the simulation, with a root mean square error of $9.0 \cdot 10^{-3} m$ for the position and $2.6 \cdot 10^{-3} \frac{m}{s}$ for the velocity. During most of the simulation the value over time of both the position and the velocity of the COM of the multi-rotor are very similar to the position and velocity of the COM of the robot (figure 3.8). From these plots are noticeable the desired trajectories for the various phases (after the control inputs converge to zero): static standing position when grounded during phase 0, parabolic trajectory for phase 1 and constant downward velocity during the vertical landing of phase 2.



(a) (b)

Figure 3.6: value of the multi-rotor COM z-axis position (a) and velocity (b) over time for simulation 1.

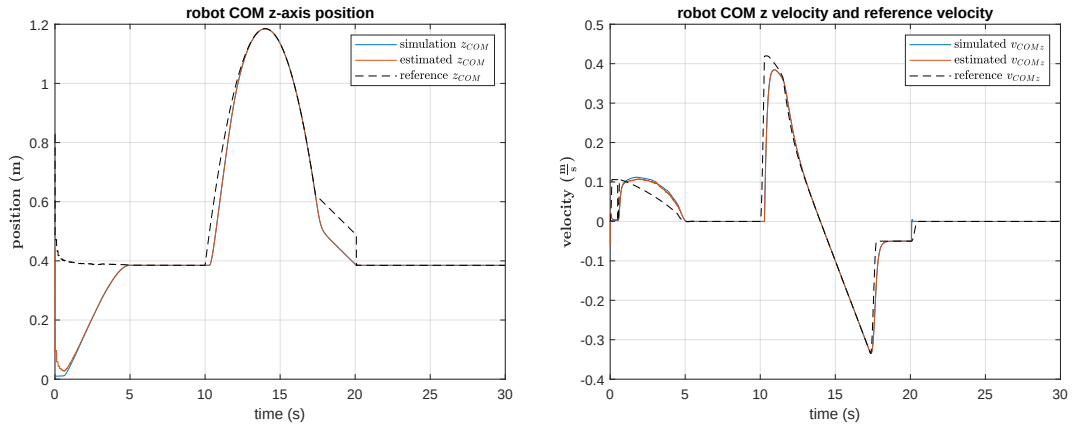
Figure 3.7 contains the plots comparing the estimated and simulated values of the position and velocity of the COM of the robot along the x-axis. The estimation for the position has little noise, while the estimated velocity has visible oscillating differences from the simulated value but little and not diverging. From the plots are noticeable the desired movement for the COM of the robot along the x-axis, in particular the standing position while grounded in phase 0 and the constant velocity parabolic trajectory (after the position error control converges to zero) for phase 1.



(a) (b)

Figure 3.7: value of the Robot COM x-axis position (a) and velocity (b) over time for simulation 1.

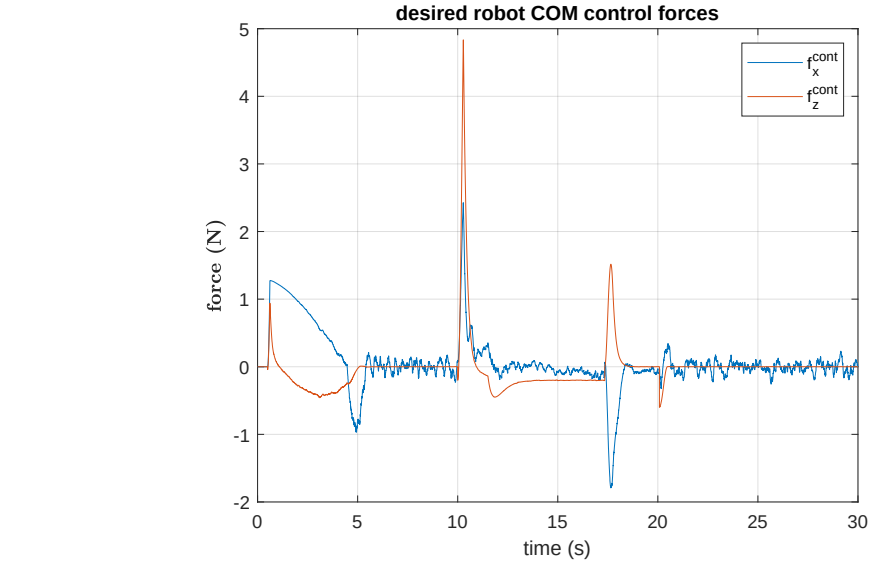
Figure 3.8 shows the evolution over time of the estimated and simulated position and velocity along the z-axis of the COM of the robot. The estimation errors are small for both the variables after the first update cycles. In the plots is also visible the tracking of the desired trajectories for the various phases, in particular the parabolic trajectory during phase 1 and the constant-velocity vertical landing of phase 2.



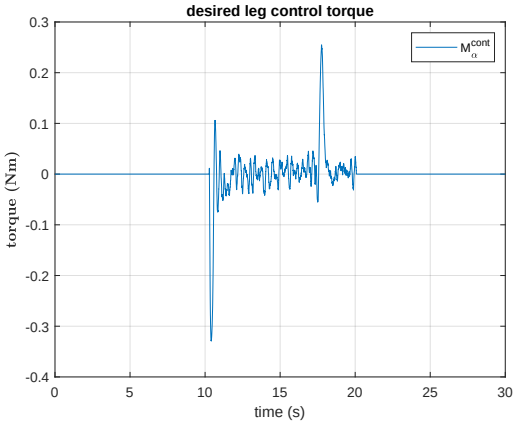
(a) (b)

Figure 3.8: value of the Robot COM z-axis position (a) and velocity (b) over time for simulation 1.

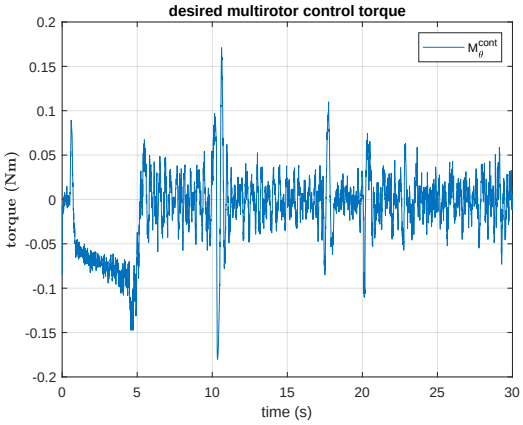
Figure 3.9 shows the plots for the control forces and torques relevant to control the state of the system before being converted into propellers intensity: COM control forces (excluding gravity compensation), leg control torque and multi-rotor control torque. All plots show peaks followed by convergence of the control forces towards 0 (excluding the z COM control force, which during the jump phase holds a negative constant value corresponding to the constant negative acceleration of the parabolic trajectory) at each change of phase, followed by lower intensity oscillation due to noisy estimates of the controlled state variables or (in the case of the multi-rotor orientation control) noisy reference value due to the noise of other state variables. The control torque for the leg (figure 3.16b) has value zero during the time intervals corresponding to phase 0: during these intervals the robot is grounded and as such, as stated in section 2.3, the leg control torque is not applied.



(a)



(b)



(c)

Figure 3.9: value of the Robot COM control force (a) and the control torques for the leg (b) and the multi-rotor (c) over time for simulation 1.

3.7 Simulation 2

This simulation was run using higher noise for the IMUs than the nominal value (see section 3.5). The plots can be analyzed in 4 different phases: phase 0, in the time intervals $[0, 10] \cup (20.0, 30]s$; phase 1, in the time interval $(10, 17.3]s$; phase 2, in the time interval $(17.3, 20.0]s$. The robot's estimation performance is shown in figures 3.10 to 3.16.

Figure 3.10 shows the comparison over time between the simulation truth value and the estimation of the multi-rotor orientation and of its angular velocity relative to the world frame (θ and ω_θ in the state of the model). The estimate is more noisy than for the previous simulation, with a root mean square error of $1.7 \cdot 10^{-2} rad$ for the angle and $2.9 \cdot 10^{-2} \frac{rad}{s}$ for the angular velocity. The signal has bigger oscillations over time and saturates at the maximum planned angular velocity of $0.5 \frac{rad}{s}$ more frequently compared to the previous simulation. This is due to higher oscillations in the position and velocity estimates of the COM of the robot causing more oscillations in the desired multi-rotor orientation. There is an estimation error of a few hundredths of radians during the flight phase, mirrored in an error in the estimation of the multi-rotor x-axis velocity. Despite the elevated oscillations of the signal, three relevant points of interest are noticeable in the plot of the orientation angle (3.10a: the first one is the non-zero tilt in the time interval $[0.6, 3.5]s$ of the simulation and corresponds to the tilt of the multi-rotor necessary to raise the COM of the robot to the desired standing position; the peak after $10s$ is related to the start of the trajectory tracking of phase 1 and the rapid change in COM reference velocity caused by this change of phase; the peak at around $17s$ is related to the end of the trajectory tracking of phase 1 and the start of the vertical landing of phase 2, with a consequent rapid change in COM reference velocity.

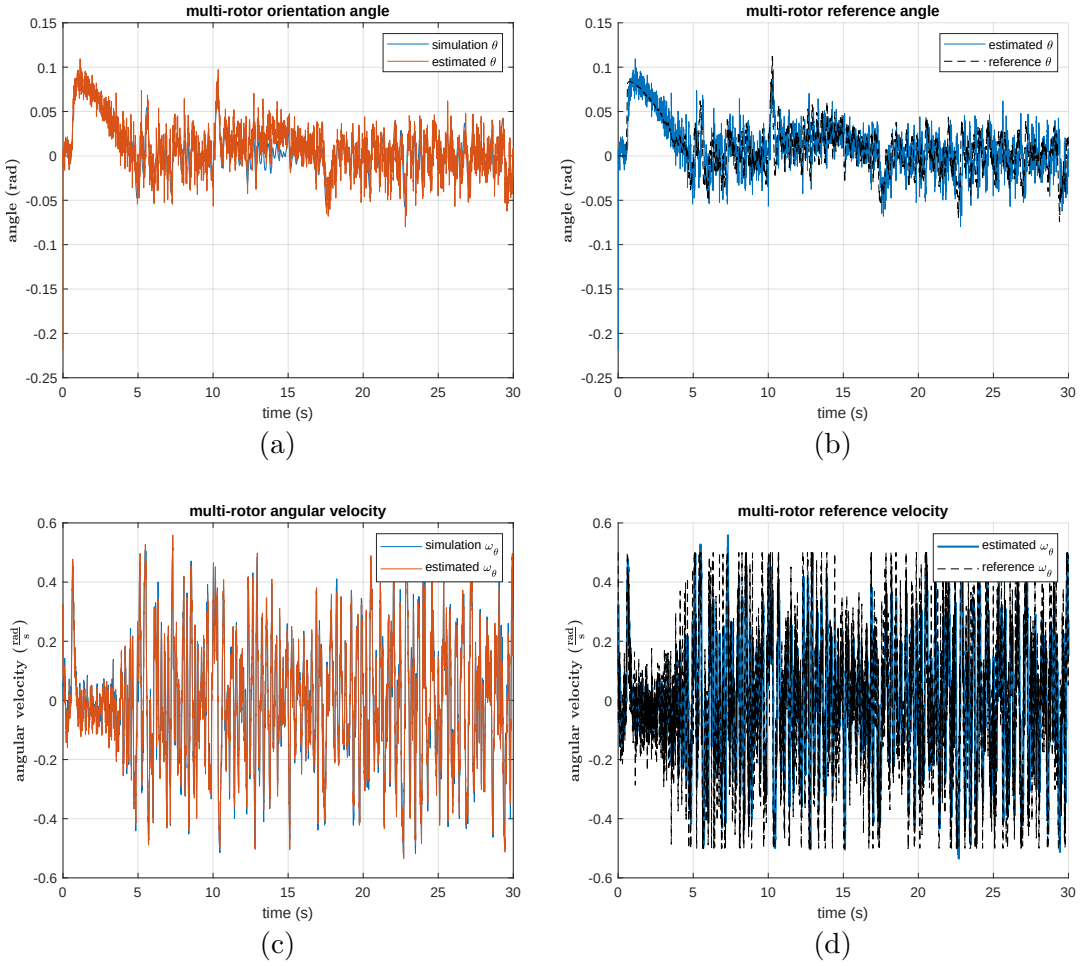
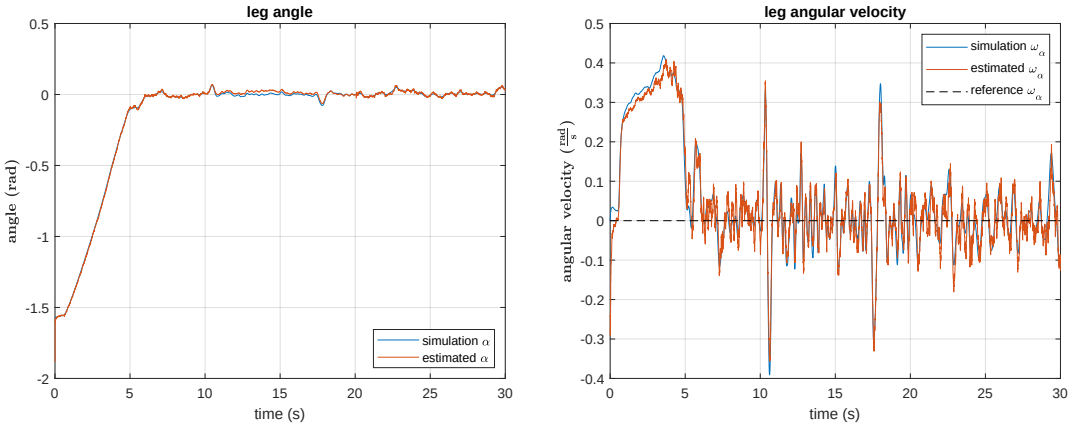


Figure 3.10: comparison over time of the values of the simulated and estimated multi-rotor angle (a), estimated and desired multi-rotor angle (b), simulated and estimated angular velocity (c), estimated and desired angular velocity (d) over time for simulation 2.

Figure 3.11 contains the plots of the estimated values of the leg orientation and angular velocity with respect to the world frame (α and ω_α in the state of the model) compared with the corresponding values from the simulation. The two graphs show good convergence of the estimated values towards the simulation truth, excluding a slight velocity error during the standing up phase of the simulation. The 2 peaks corresponding to the start of phase 1 and the start of phase 2 are visible; this is because the un-actuated joint between the leg and the multi-rotor causes the leg angle to be affected by the control accelerations applied on the COM of the robot.

The noise is bigger than in simulation 1, with a root mean square error between the simulated and estimated signal of $1.1 \cdot 10^{-2} rad$ for the angle and $3.1 \cdot 10^{-2} \frac{rad}{s}$ for the angular velocity.

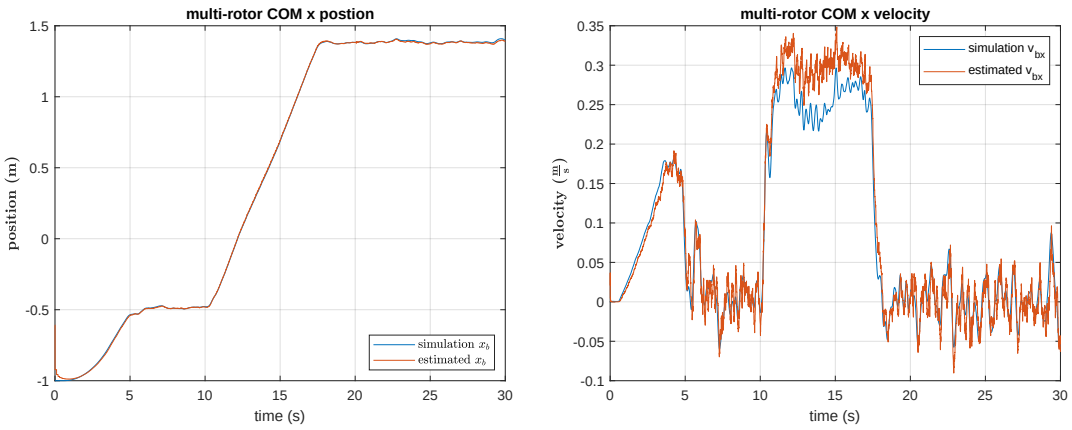


(a) (b)

Figure 3.11: value of the leg angle (a) and angular velocity (b) over time for simulation 2.

Figure 3.12 shows the evolution of the estimated position and velocity of the multi-rotor along the x-axis compared with the values from the simulation. The position estimation has little noise, with a root mean square error of $9.4 \cdot 10^{-3} m$, while the velocity estimation is more disturbed, with a root mean square error of $2.5 \cdot 10^{-2} \frac{m}{s}$, and its estimation error is marginally stable, staying visibly far from zero for extended intervals of time but not diverging, during phase 1. In the time intervals corresponding to phase 0 there are noticeable similarities between the x-axis velocity of the COM of the multi-rotor (3.12b) and the angular velocity of

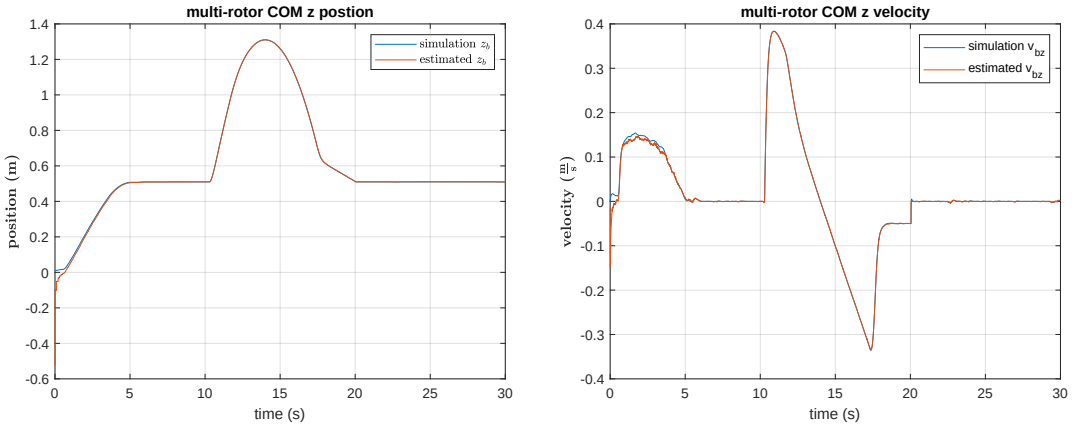
the leg (3.11b): in these time intervals the robot is grounded and, as described in paragraph 2.1.2 the velocity of the COM of the multi-rotor is computed in function of the leg angular velocity (based on a no-slip assumption on the foot of the robot). During most of the simulation the values over time of both the position and the velocity of the COM of the multi-rotor are very similar to the position and velocity of the COM of the robot (3.14). Despite the noise in the estimation in the plots are noticeable the desired trajectories of the COM of the robot during the various phases of the simulation (after the control inputs converge to zero): static standing position during phase 0 after $5s$ from the start of the simulation, non-zero constant velocity for parabolic trajectory during phase 1 and zero-velocity for the vertical landing in phase 2.



(a) (b)

Figure 3.12: value of the multi-rotor COM x-axis position (a) and velocity (b) over time for simulation 2.

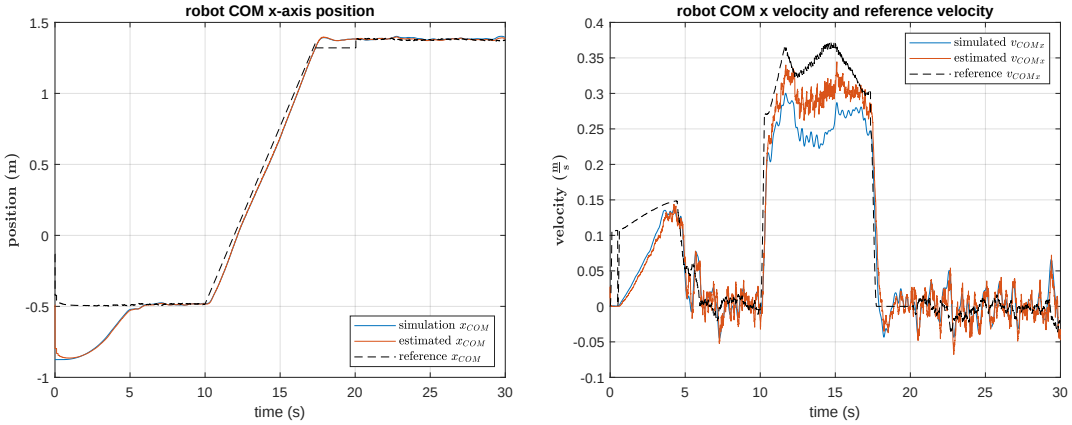
Figure 3.13 shows the plots comparing the estimates and the simulated values for the position and velocity along the z-axis of the COM of the multi-rotor. The estimation is very close to the simulated value after the first update cycles at the beginning of the simulation, with a root mean square error of $1.2 \cdot 0^{-2}m$ for the position and $6.9 \cdot 10^{-3} \frac{m}{s}$ for the velocity. During most of the simulation the value over time of both the position and the velocity of the COM of the multi-rotor are very similar to the position and velocity of the COM of the robot (figure 3.15). From these plots are noticeable the desired trajectories for the various phases (after the control inputs converge to zero): static standing position when grounded during phase 0, parabolic trajectory in phase 1 and constant downward velocity during the vertical landing of phase 2.



(a) (b)

Figure 3.13: value of the multi-rotor COM z-axis position (a) and velocity (b) over time for simulation 2.

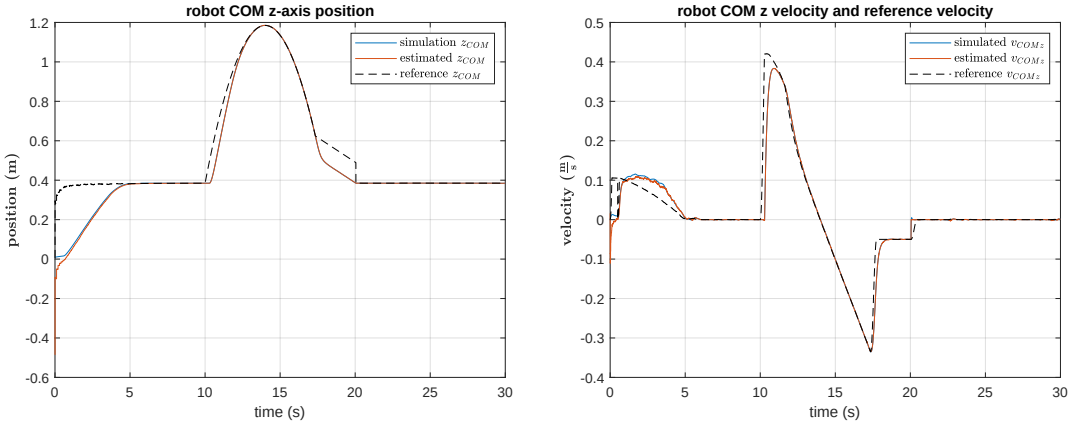
Figure 3.14 contains the plots comparing the estimated and simulated values of the position and velocity of the COM of the robot along the x-axis. The estimation for the position has little noise, while the estimated velocity has visible oscillating differences from the simulated value; little and not diverging while grounded and with a little offset during the flight phases. From the plots are noticeable the desired movement for the COM of the robot along the x-axis, in particular the standing position while grounded in phase 0 and the constant velocity parabolic trajectory (after the position error control converges to zero) for phase 1.



(a) (b)

Figure 3.14: value of the Robot COM x-axis position (a) and velocity (b) over time for simulation 2.

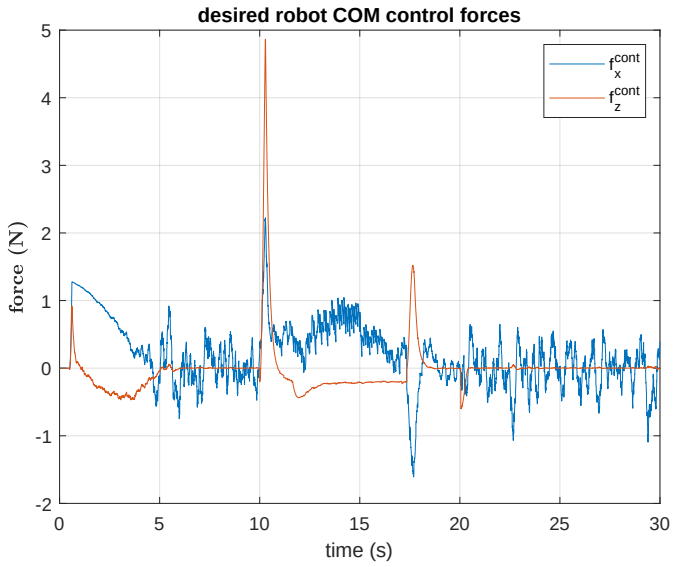
Figure 3.15 shows the evolution over time of the estimated and simulated position and velocity along the z-axis of the COM of the robot. The estimation errors are small for both the variables after the first update cycles. In the plots is also visible the tracking of the desired trajectories for the various phases, in particular the parabolic trajectory during phase 1 and the constant-velocity vertical landing of phase 2.



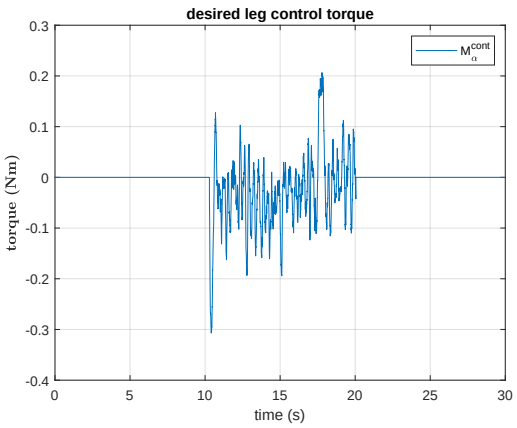
(a) (b)

Figure 3.15: value of the z-axis Robot COM position (a) and velocity (b) over time for simulation 2.

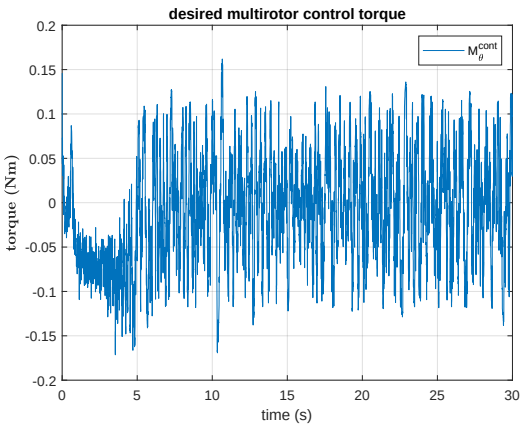
Figure 3.16 shows the plots for the control forces and torques relevant to control the state of the system before being converted into propellers intensity: COM control forces (excluding gravity compensation), leg control torque and multi-rotor control torque. All plots show peaks followed by convergence of the control forces towards 0 (excluding the z COM control force, which during the jump phase holds a negative constant value corresponding to the constant negative acceleration of the parabolic trajectory) at each change of phase, followed by lower intensity oscillation due to noisy estimates of the controlled state variables or (in the case of the multi-rotor orientation control) noisy reference value caused by the noise of other state variables. The multi-rotor control torque has higher amplitude oscillations compared to the previous simulation, and the phase change peaks are less visible: this is due to both higher variance in all the estimates of this simulation and the reference angular velocity for the multi-rotor computed being saturated to $0.5\frac{rad}{s}$ The control torque for the leg (figure 3.16b) has value zero during the time intervals corresponding to phase 0: during these intervals the robot is grounded and as such, as stated in section 2.3, the leg control torque is not applied.



(a)



(b)



(c)

Figure 3.16: value of the Robot COM control force (a) and the control torques for the leg (b) and the multi-rotor (c) over time for simulation 2.

3.8 Simulation 3

This simulation was run using lower noise for the IMUs than the nominal value of the experiment 1 (see section 3.5), closer to the variance of common commercially available sensors. The plots can be analyzed in 4 different phases: phase 0, in the time intervals $[0, 10] \cup (20.1, 30]s$; phase 1, in the time interval $(10, 17.3]s$; phase 2, in the time interval $(17.3, 20.1]s$. The robot's estimation performance, as shown in figures 3.17 to 3.23, is better for all the states excluding the multi-rotor cardinal velocity (and, as a consequence, the velocity of the COM of the robot) compared to the first simulation.

Figure 3.17 shows the comparison over time between the simulation truth value and the estimation of the multi-rotor orientation and of its angular velocity relative to the world frame (θ and ω_θ in the state of the model). The estimate is less noisy than for the previous simulations, with a root mean square error of $5.5 \cdot 10^{-3} rad$ for the angle and $9.4 \cdot 10^{-3} \frac{rad}{s}$ for the angular velocity. The signal has also less oscillations over time compared to the previous simulations, thanks to a minor oscillation in the position and velocity estimation of the COM of the robot resulting in less oscillations in the desired multi-rotor orientation. Despite the elevated oscillations of the signal, three relevant points of interest are noticeable in the plot of the orientation angle (3.17a: the first one is the non-zero tilt in the time interval $[0.6, 3.5]s$ of the simulation and corresponds to the tilt of the multi-rotor necessary to raise the COM of the robot to the desired standing position; the peak after $10s$ is related to the start of the jumping phase and the rapid change in COM reference velocity (from close to zero to the starting speed of the parabolic trajectory); the peak at around $17s$ is related to the end of thee trajectory tracking of phase 1 and the start of the vertical landing of phase 2, with a consequent rapid change in COM reference velocity (from non-zero constant horizontal velocity in the parabolic trajectory to the vertical reference velocity of the vertical landing phase).

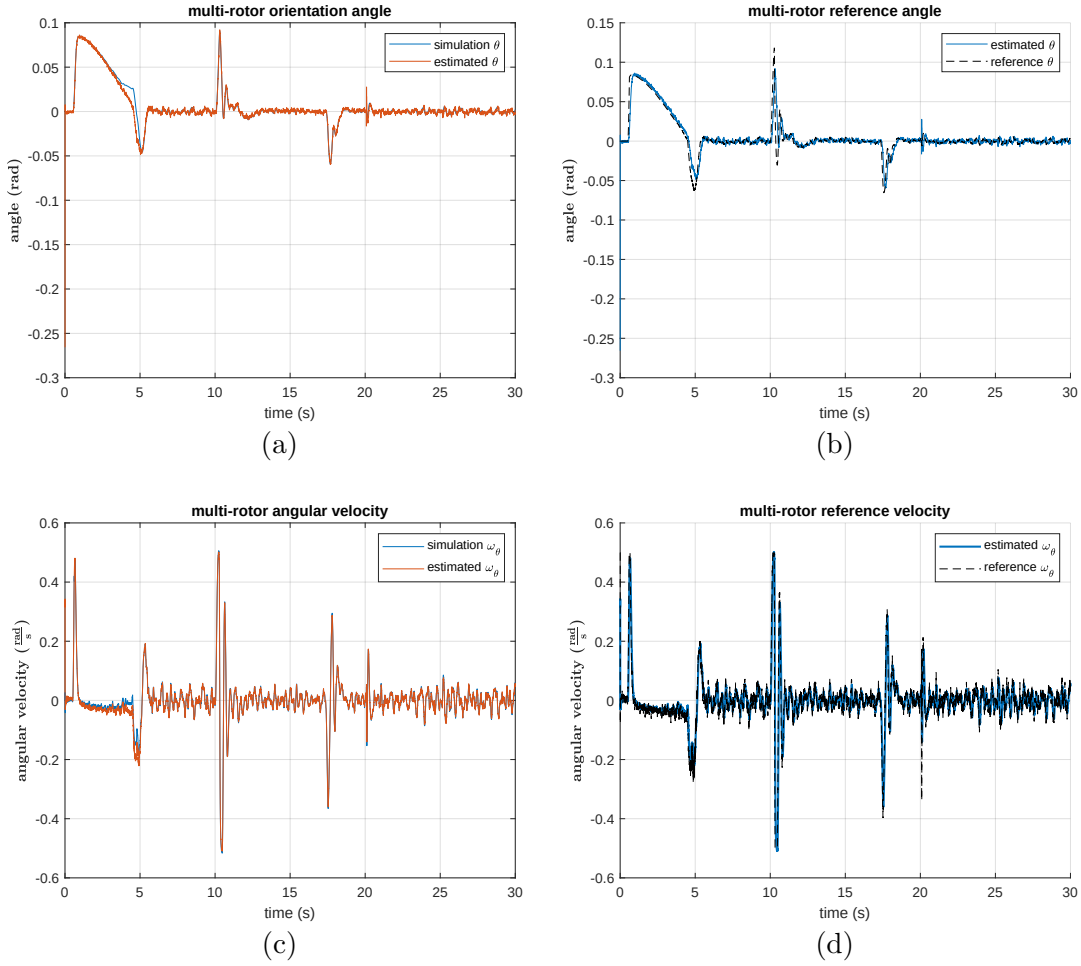
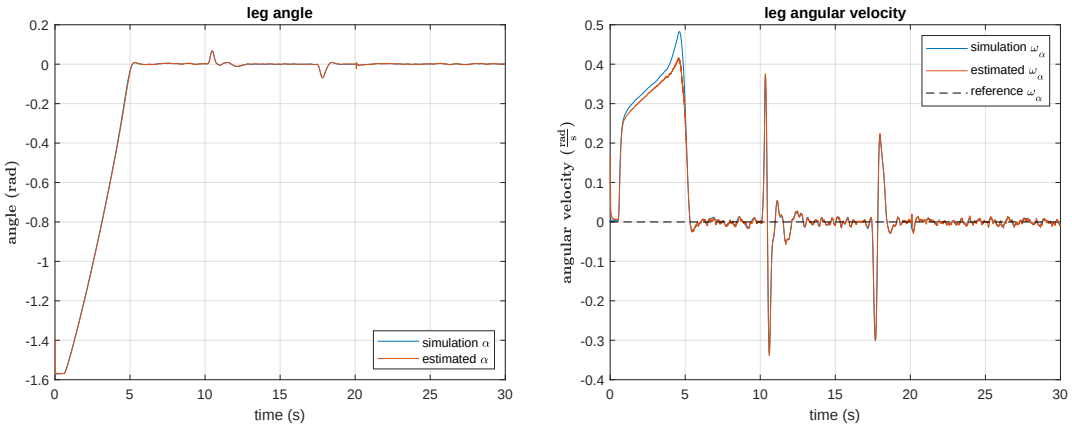


Figure 3.17: comparison over time of the values of the simulated and estimated multi-rotor angle (a), estimated and desired multi-rotor angle (b), simulated and estimated angular velocity (c), estimated and desired angular velocity (d) over time for simulation 3.

Figure 3.18 contains the plots of the estimated values of the leg orientation and angular velocity with respect to the world frame (α and ω_α in the state of the model) compared with the corresponding values from the simulation. The two graphs show little noise along the entire simulation and a bit of velocity error during the standing up stage of the simulation. The 2 peaks corresponding to the start of phase 1 and the start of phase 2 are visible; this is because the unactuated joint between the leg and the multi-rotor causes the acceleration of the COM of the robot to affect also the angular acceleration of the leg.

The estimation noise is tiny, with a root mean square error between the simulated and estimated signal of $4.4 \cdot 10^{-3} rad$ for the angle and $1.3 \cdot 10^{-2} \frac{rad}{s}$ for the

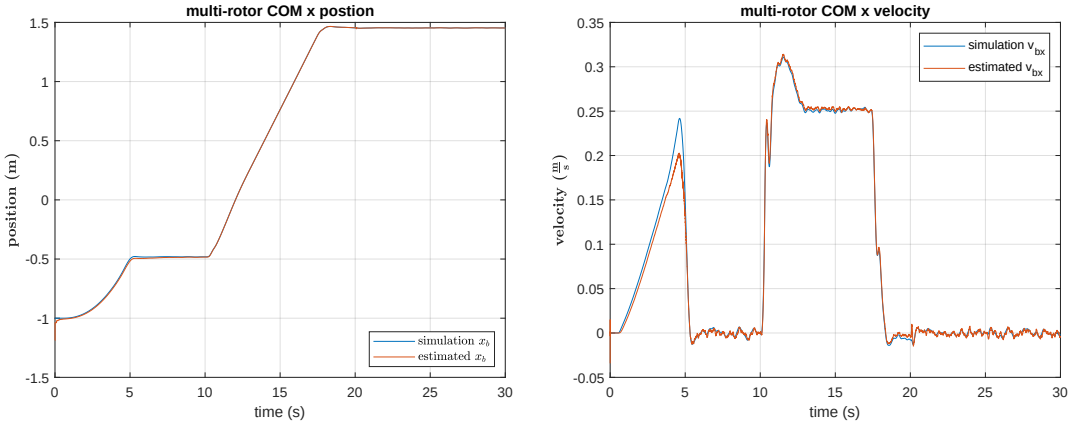
angular velocity.



(a) (b)

Figure 3.18: value of the leg angle (a) and angular velocity (b) over time for simulation 3.

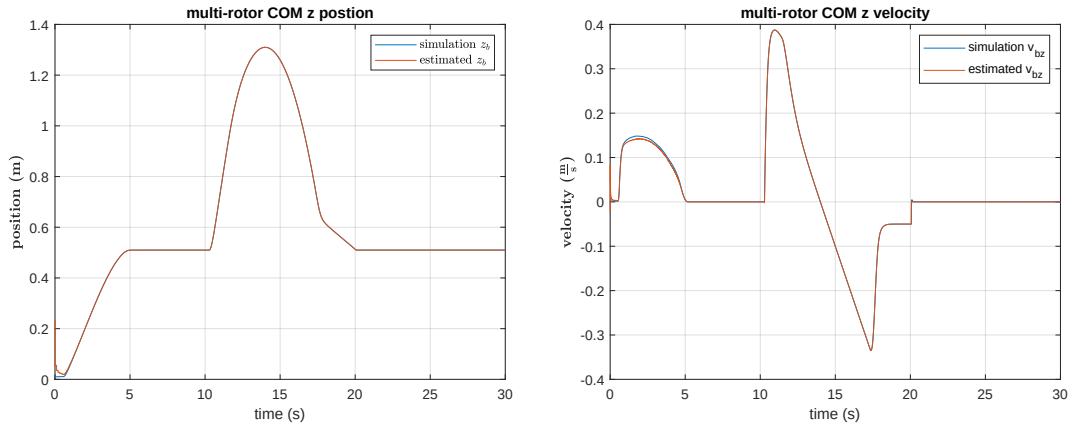
Figure 3.19 shows the evolution of the estimated position and velocity of the multi-rotor along the x-axis compared with the values from the simulation. The position estimation has little noise, with a root mean square error of $6.4 \cdot 10^{-3} m$, while the velocity estimation is more disturbed, with a root mean square error of $7.2 \cdot 10^{-3} \frac{m}{s}$. During the time interval corresponding to phase 0 there are noticeable similarities between the x-axis velocity of the COM of the multi-rotor (3.19b) and the angular velocity of the leg (3.18b): in these time intervals the robot is grounded and, as described in paragraph 2.1.2, the velocity of the COM of the multi-rotor is computed in function of the leg angular velocity (based on a no-slip assumption on the foot of the robot). During most of the simulation the value over time of both the position and the velocity of the COM of the multi-rotor are very similar to the position and velocity of the COM of the robot (3.21). In the plots are noticeable the planned movement trajectories of the COM of the robot during the various phases of the simulation (after the control inputs converge to zero): static standing position during phase 0 (after $5s$ from the start of the simulation), non-zero constant velocity for parabolic trajectory tracking during phase 1 and zero-velocity for the vertical landing for phase 2.



(a) (b)

Figure 3.19: value of the multi-rotor COM x-axis position (a) and velocity (b) over time for simulation 3.

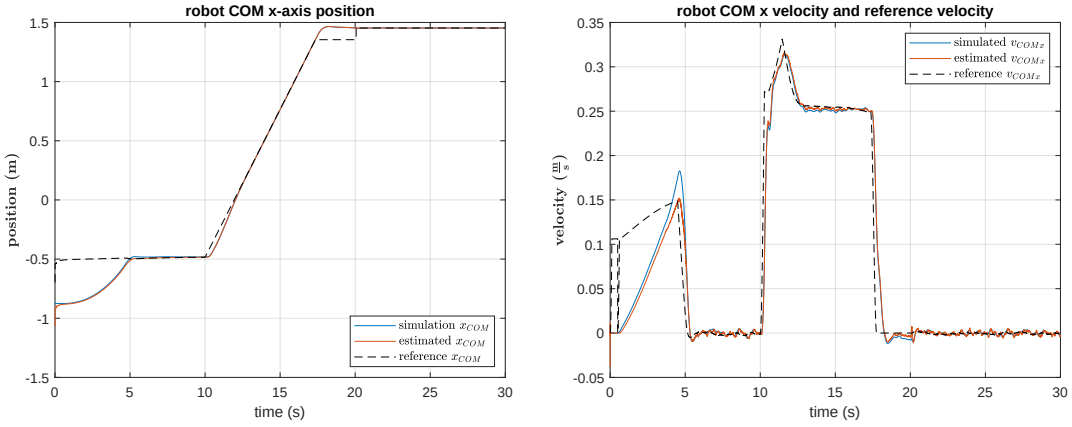
Figure 3.20 shows the plots comparing the estimates and the simulated values for the position and velocity along the z-axis of the COM of the multi-rotor. The estimation is very close to the simulated value after the first update cycles at the beginning of the simulation, with a root mean square error of $4.7 \cdot 10^{-3} m$ for the position and $2.6 \cdot 10^{-3} \frac{m}{s}$ for the velocity. Only exception is for the velocity during the standing up phase ($[0.6, 4]s$), where a discrepancy between the estimated and the simulated velocity is noticeable, related to the discrepancy in the same interval on the angular velocity of the leg (3.18b) because of the no-slip assumption for the grounded model (see paragraph 2.1.2). During most of the simulation the value over time of both the position and the velocity of the COM of the multi-rotor are very similar to the position and velocity of the COM of the robot (figure 3.22). From these plots are noticeable the planned trajectories for the various phases (after the control inputs converge to zero): static standing position when grounded during phase 0, parabolic trajectory for phase 1 and constant downward velocity during the vertical landing of phase 2.



(a) (b)

Figure 3.20: value of the multi-rotor COM z-axis position (a) and velocity (b) over time for simulation 3.

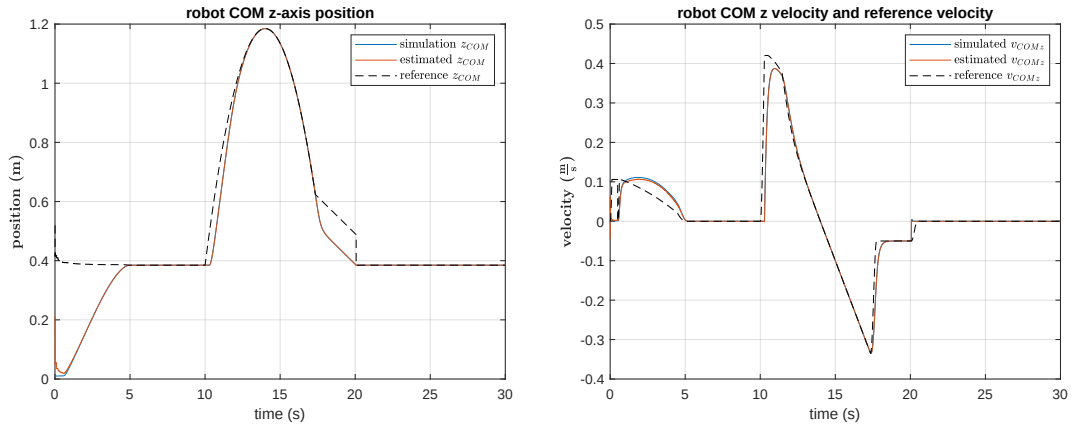
Figure 3.21 contains the plots comparing the estimated and simulated values of the position and velocity of the COM of the robot along the x-axis. The estimation for the position is very accurate, while the estimated velocity has visible oscillating differences from the simulated value but little and not diverging. From the plots are noticeable the desired movement for the COM of the robot along the x-axis, in particular the standing position while grounded in phase 0 and the constant velocity parabolic trajectory (after the position error control converges to zero) for phase 1.



(a) (b)

Figure 3.21: value of the x-axis Robot COM position (a) and velocity (b) over time for simulation 3.

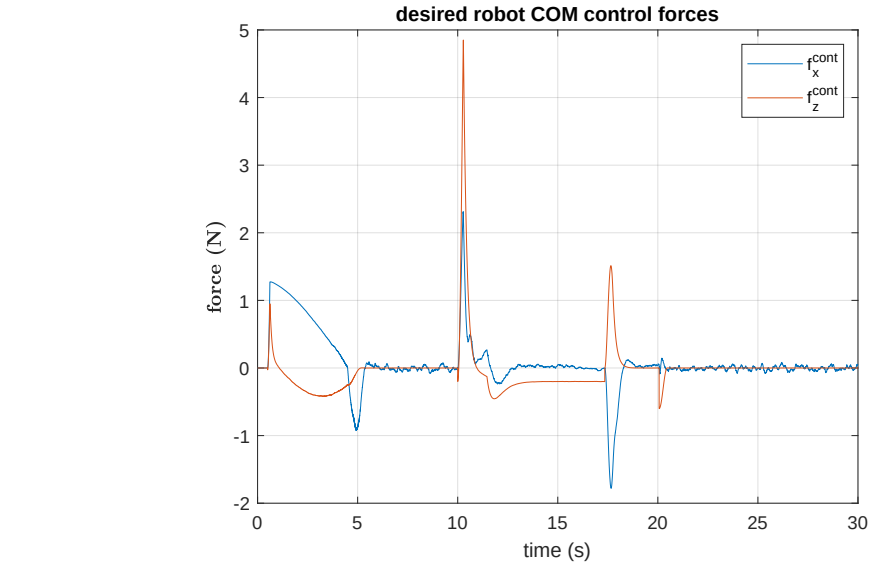
Figure 3.22 shows the evolution over time of the estimated and simulated position and velocity along the z-axis of the COM of the robot. The estimation errors are small for both the variables after the first update cycles. In the plots is also visible the tracking of the desired trajectories for the various phases, in particular the parabolic trajectory during phase 1 and the constant-velocity vertical landing of phase 2.



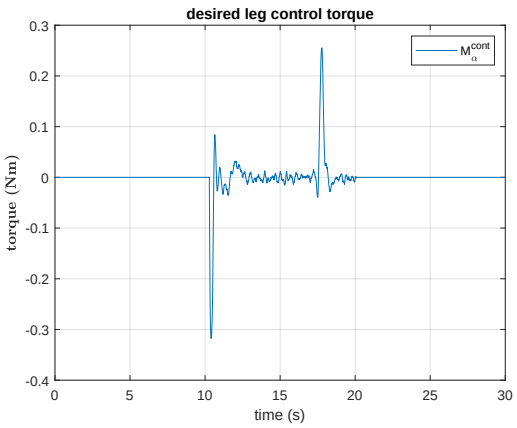
(a) (b)

Figure 3.22: value of the z-axis Robot COM position (a) and velocity (b) over time for simulation 3.

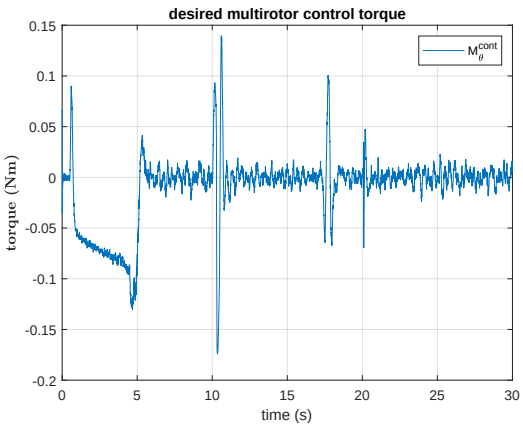
Figure 3.23 shows the plots for the control forces and torques relevant to control the state of the system before being converted into propellers intensity: COM control forces (excluding gravity compensation), leg control torque and multi-rotor control torque. All plots show peaks followed by convergence of the control forces towards 0 (excluding the z COM control force, which during the jump phase holds a negative constant value corresponding to the constant negative acceleration of the parabolic trajectory) at each change of phase, followed by very low intensity oscillation caused by noisy estimates of the controlled state variables or (in the case of the multi-rotor orientation control) noisy reference value due to the noise of other state variables. The control torque for the leg (figure 3.16b) has value zero during the time intervals corresponding to phase 0: during these intervals the robot is grounded and as such, as stated in section 2.3, the leg control torque is not applied.



(a)



(b)



(c)

Figure 3.23: value of the Robot COM control force (a) and the control torques for the leg (b) and the multi-rotor (c) over time for simulation 3.

Chapter 4

Conclusions

In this thesis we considered the challenges a legged robot faces in the exploration of harsh terrains with large obstacles present and designed the first iteration of a hybrid robot equipped with legs and a multi-rotor with walking, jumping and flight capabilities. We developed a model for the robot, implemented a state estimator based on an Extended Kalman Filter using IMU and GPS sensors and designed a proportional-derivative cascading control strategy for the velocity of the COM of the robot and for the orientation of the multi-rotor.

The performance of the estimator and of the controller were tested in simulations with different levels of noise added to the measurements from the simulated sensors. These different noise levels show a good estimation for all the robot's state variables, except for the cardinal velocity of the multi-rotor (and, as a consequence, of the robot COM) during the flight stage with high sensor variance. the lack of direct observability on the multi-rotor velocity while in mid-air using only IMU and GPS as sensors contributes to this estimation inaccuracy.
Future works on this project may focus on:

- improving the performance of the estimator, with particular focus on the cardinal velocity of the multi-rotor;
- extending the model implementation of the Kalman filter and the control strategy to three dimensions;
- implementing an improved transition method between the flight and ground stages of the robot;
- adding more links and actuated joint to the leg and implement joint configuration estimation and control;
- implementing a cooperation strategy while on the ground between propellers and joint motors for robot stabilization, landing and to start the

jump;

- verify the estimator's robustness with external disturbances (such as wind);
- achieving a physical realization of the robot and its estimation and control.

Appendix A

Jacobian matrices computation

The analytical Jacobian matrices for the model are shown here. section A.1 shows the state Jacobian matrices $F_k = \left[\frac{\partial \mathbf{f_d}(\mathbf{x},\mathbf{u})}{\partial x_n}\right]$ for both flight and ground model, sections A.2 describe the analytical values of the IMU measurement Jacobian matrices $H_{IMU} = \left[\frac{\partial \mathbf{h}_{IMU}(\mathbf{x},\mathbf{u})}{\partial x_n}\right]$ for ground and flight measurements; finally section A.3 of the GPS measurements $H_{GPS} = \left[\frac{\partial \mathbf{h}_{GPS}(\mathbf{x},\mathbf{u})}{\partial x_n}\right]$.

A.1 state Jacobian

A.1.1 flight state Jacobian matrix

$$^{flight}F_{1:4,1:8} = \begin{bmatrix} 1 & 0 & 0 & 0 & dt & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & dt & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & dt & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & dt \end{bmatrix}$$

$$^{flight}F_{5,3} = dt \cdot \frac{f_{p1} + f_{p2}}{m_t} \cdot (\cos\theta + \frac{m_{l1}^2}{I_{l1} \cdot m_t} \cdot \frac{l_{l1}^2}{4} \cos\alpha \cos(\theta - \alpha))$$

$$^{flight}F_{5,4} = \frac{dt}{m_t}(-m_{l1}\frac{l_{l1}}{2}\omega_\alpha^2 \cos\alpha - (f_{p1}+f_{p2})\frac{m_{l1}^2}{I_{l1}m_t}\frac{l_{l1}^2}{4}\cos(\theta - 2\alpha) + \frac{m_{l1}}{I_{l1}}\mu\frac{l_{l1}}{2}\sin(\alpha)(\omega_\alpha - \omega_\theta))$$

$$^{flight}F_{5,5} = 1$$

$$^{flight}F_{5,7} = dt\frac{m_{l1}}{m_t}\frac{l_{l1}}{2}\frac{\mu}{I_{l1}}\cos\alpha$$

$$^{flight}F_{5,8} = -dt\frac{m_{l1}}{m_t}l_{l1}(\omega_\alpha\sin\alpha + \frac{\mu}{2I_{l1}}\cos\alpha)$$

$$^{flight}F_{6,3} = dt\frac{f_{p1}+f_{p2}}{m_t}(-\sin\theta - \frac{m_{l1}^2}{I_{l1}m_t}\frac{l_{l1}^2}{4}\sin\alpha\cos(\theta-\alpha))$$

$$^{flight}F_{6,4} = \frac{dt}{m_t}(m_{l1}\frac{l_{l1}}{2}\omega_\alpha^2\sin\alpha - (f_{p1}+f_{p2})\frac{m_{l1}^2}{I_{l1}m_t}\frac{l_{l1}^2}{4}\sin(\theta-2\alpha) + \frac{m_{l1}}{I_{l1}}\mu\frac{l_{l1}}{2}\cos(\alpha)(\omega_\alpha - \omega_\theta))$$

$$^{flight}F_{6,6} = 1$$

$$^{flight}F_{6,7} = -dt\frac{m_{l1}}{m_t}\frac{l_{l1}}{2}\frac{\mu}{I_{l1}}\sin\alpha$$

$$^{flight}F_{6,8} = dt\frac{m_{l1}}{m_t}l_{l1}(\omega_\alpha\cos\alpha + \frac{\mu}{2I_{l1}}\sin\alpha)$$

$$^{flight}F_{7,7} = 1 - dt\frac{\mu}{I_{l1}}$$

$$^{flight}F_{7,8} = dt\frac{\mu}{I_{l1}}$$

$$^{flight}F_{8,3} = dt\frac{f_{p1}+f_{p2}}{I_{l1}}\frac{m_{l1}}{m_t}\frac{l_{l1}}{2}\cos(\theta-\alpha)$$

$$^{flight}F_{8,4} = -dt\frac{f_{p1}+f_{p2}}{I_{l1}}\frac{m_{l1}}{m_t}\frac{l_{l1}}{2}\cos(\theta-\alpha)$$

$$^{flight}F_{8,7} = dt\frac{\mu}{I_{l1}}$$

$$^{flight}F_{8,8} = 1 - dt\frac{\mu}{I_{l1}}$$

A.1.2 ground state Jacobian matrix

$$^{ground}F_{1:4,1:8} = \begin{bmatrix} 1 & 0 & 0 & 0 & dt & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & dt & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & dt & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & dt \end{bmatrix}$$

$$^{ground}F_{5,3} = dt\frac{f_{p1}+f_{p2}}{I_{l1}}l_{l1}^2\cos(\theta-\alpha)\cos\alpha$$

$$
\begin{aligned}
{}^{ground}F_{5,4} =& dt(\frac{l_{l1}^2}{I_{l1}}(g\cos(2\alpha)(m_b+\frac{m_{l1}}{2})-(f_{p1}+f_{p2})\cos(\theta-2\alpha))+\\
&+\frac{l_{l1}}{I_{l1}}\mu(\omega_\alpha-\omega_\theta)\sin\alpha-l_{l1}\omega_\alpha^2\cos\alpha)
\end{aligned}
$$

$$
{}^{ground}F_{5,5}=1
$$

$$
{}^{ground}F_{5,7}=dt\frac{l_{l1}}{I_{l1}}\mu\cos\alpha
$$

$$
{}^{ground}F_{5,8}=-dt(2l_{l1}\omega_\alpha\sin\alpha+\frac{l_{l1}}{I_{l1}}\mu\cos\alpha)
$$

$$
{}^{ground}F_{6,3}=-dt\frac{f_{p1}+f_{p2}}{I_{l1}}l_{l1}^2\cos(\theta-\alpha)\sin\alpha
$$

$$
\begin{aligned}
{}^{ground}F_{6,4} =& dt(-\frac{l_{l1}^2}{I_{l1}}(g\sin(2\alpha)(m_b+\frac{m_{l1}}{2})+(f_{p1}+f_{p2})\sin(\theta-2\alpha))+\\
&+\frac{l_{l1}}{I_{l1}}\mu(\omega_\alpha-\omega_\theta)\cos\alpha+l_{l1}\omega_\alpha^2\sin\alpha)
\end{aligned}
$$

$$
{}^{ground}F_{6,6}=1
$$

$$
{}^{ground}F_{6,7}=-dt\frac{l_{l1}}{I_{l1}}\mu\sin\alpha
$$

$$
{}^{ground}F_{6,8}=-dt(2l_{l1}\omega_\alpha\cos\alpha-\frac{l_{l1}}{I_{l1}}\mu\sin\alpha)
$$

$$
{}^{ground}F_{8,3}=dt\frac{f_{p1}+f_{p2}}{I_{l1}}l_{l1}\cos(\theta-\alpha)
$$

$$
{}^{ground}F_{8,4}=dt\frac{l_{l1}}{I_{l1}}(g\cos\alpha(m_b+\frac{m_{l1}}{2})-(f_{p1}+f_{p2})\cos(\theta-\alpha)
$$

$$
{}^{ground}F_{8,7}=dt\frac{\mu}{I_{l1}}
$$

$$
{}^{ground}F_{8,8}=1-dt\frac{\mu}{I_{l1}}
$$

$$
{}^{ground}F_{7,7}=1-dt\frac{\mu}{I_{l1}}
$$

$$
{}^{ground}F_{7,8}=dt\frac{\mu}{I_{l1}}
$$

A.2 IMU measure Jacobian matrix

A.2.1 Flight IMU Jacobian matrix

$$^{flight}H_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$^{flight}H_{2,3} = (\frac{m_{l1}}{m_t})^2 \frac{l_{l1}^2}{4} \frac{f_{p1}+f_{p2}}{I_{l1}} \cos(2(\theta-\alpha)) + \frac{m_{l1}}{m_t}\frac{l_{l1}}{2}(\omega_\alpha^2 \cos(\theta-\alpha) + \frac{\mu}{I_{l1}}(\omega_\alpha - \omega_\theta)\sin(\theta-\alpha)$$

$$^{flight}H_{2,4} = -(\frac{m_{l1}}{m_t})^2 \frac{l_{l1}^2}{4} \frac{f_{p1}+f_{p2}}{I_{l1}} \cos(2(\theta-\alpha)) - \frac{m_{l1}}{m_t}\frac{l_{l1}}{2}(\omega_\alpha^2 \cos(\theta-\alpha) + \frac{\mu}{I_{l1}}(\omega_\alpha - \omega_\theta)\sin(\theta-\alpha)$$

$$^{flight}H_{2,7} = \frac{m_{l1}}{m_t}\frac{l_{l1}}{2}\frac{\mu}{I_{l1}}\cos(\theta-\alpha)$$

$$^{flight}H_{2,8} = -\frac{m_{l1}}{m_t} l_{l1}(\omega_\alpha \sin(\theta-\alpha) + \frac{\mu}{2I_{l1}}\cos(\theta-\alpha)$$

$$^{flight}H_{3,3} = (\frac{m_{l1}}{m_t})^2 \frac{l_{l1}^2}{4} \frac{f_{p1}+f_{p2}}{I_{l1}} \sin(2(\theta-\alpha)) - \frac{m_{l1}}{m_t}\frac{l_{l1}}{2}(\omega_\alpha^2 \sin(\theta-\alpha) + \frac{\mu}{I_{l1}}(\omega_\alpha - \omega_\theta)\cos(\theta-\alpha)$$

$$^{flight}H_{3,4} = -(\frac{m_{l1}}{m_t})^2 \frac{l_{l1}^2}{4} \frac{f_{p1}+f_{p2}}{I_{l1}} \sin(2(\theta-\alpha)) + \frac{m_{l1}}{m_t}\frac{l_{l1}}{2}(\omega_\alpha^2 \sin(\theta-\alpha) + \frac{\mu}{I_{l1}}(\omega_\alpha - \omega_\theta)\cos(\theta-\alpha)$$

$$^{flight}H_{3,7} = \frac{m_{l1}}{m_t}\frac{l_{l1}}{2}\frac{\mu}{I_{l1}}\sin(\theta-\alpha)$$

$$^{flight}H_{3,8} = \frac{m_{l1}}{m_t} l_{l1}(\omega_\alpha \cos(\theta-\alpha) - \frac{\mu}{2I_{l1}}\sin(\theta-\alpha)$$

$$^{flight}H_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{flight}H_{5,3} = \frac{f_{p1}+f_{p2}}{m_t}\cos(\theta-\alpha)(1 - \frac{m_b}{m_t}\frac{m_{l1}l_{l1}^2}{4I_{l1}})$$

$$^{flight}H_{5,4} = -\frac{f_{p1}+f_{p2}}{m_t}\cos(\theta-\alpha)(1 - \frac{m_b}{m_t}\frac{m_{l1}l_{l1}^2}{4I_{l1}})$$

$$^{flight}H_{5,7} = -\frac{m_b}{m_t}\frac{l_{l1}}{2}\frac{\mu}{I_{l1}}$$

$$^{flight}H_{5,8} = \frac{m_b}{m_t}\frac{l_{l1}}{2}\frac{\mu}{I_{l1}}$$

$$^{flight}H_{6,3} = -\frac{f_{p1}+f_{p2}}{m_t}\sin(\theta-\alpha)$$

$$^{flight}H_{6,4} = \frac{f_{p1}+f_{p2}}{m_t}\sin(\theta-\alpha)$$

$$^{flight}H_{6,8} = -\frac{m_b}{m_t} l_{l1}\omega_\alpha$$

A.2.2 Ground IMU Jacobian matrix

$$^{ground}H_1 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

$$^{ground}H_{2,3} = \frac{l_{l1}^2}{I_{l1}}(-g(m_b + \frac{m_{l1}}{2})\sin\alpha\sin(\theta-\alpha) + (f_{p1}+f_{p2})\cos(2(\theta-\alpha))) + \\ + l_{l1}(\frac{\mu}{I_{l1}}(\omega_\alpha - \omega_\theta)\sin(\theta-\alpha) - \omega_\alpha^2\cos(\theta-\alpha)) - g\cos\theta$$

$$^{ground}H_{2,4} = \frac{l_{l1}^2}{I_{l1}}(g(m_b+\frac{m_{l1}}{2})\cos(\theta-2\alpha) - (f_{p1}+f_{p2})\cos(2(\theta-\alpha))) + l_{l1}(\omega_\alpha^2\cos(\theta-\alpha) - \frac{\mu}{I_{l1}}(\omega_\alpha-\omega_\theta)\sin(\theta-\alpha))$$

$$^{ground}H_{2,7} = l_{l1}\frac{\mu}{I_{l1}}\cos(\theta-\alpha)$$

$$^{ground}H_{2,8} = -2l_{l1}(\omega_\alpha\sin(\theta-\alpha) - l_{l1}\frac{\mu}{I_{l1}}\cos(\theta-\alpha)$$

$$^{ground}H_{3,3} = \frac{l_{l1}^2}{I_{l1}}(g(m_b + \frac{m_{l1}}{2})\sin\alpha\cos(\theta-\alpha) + (f_{p1}+f_{p2})\sin(2(\theta-\alpha))) + \\ - l_{l1}(\frac{\mu}{I_{l1}}(\omega_\alpha - \omega_\theta)\cos(\theta-\alpha) + \omega_\alpha^2\sin(\theta-\alpha)) - g\sin\theta$$

$$^{ground}H_{3,4} = \frac{l_{l1}^2}{I_{l1}}(g(m_b+\frac{m_{l1}}{2})\sin(\theta-2\alpha) - (f_{p1}+f_{p2})\sin(2(\theta-\alpha))) + l_{l1}(\omega_\alpha^2\sin(\theta-\alpha) + \frac{\mu}{I_{l1}}(\omega_\alpha-\omega_\theta)\cos(\theta-\alpha))$$

$$^{ground}H_{3,7} = l_{l1}\frac{\mu}{I_{l1}}\sin(\theta-\alpha)$$

$$^{ground}H_{3,8} = 2l_{l1}\omega_\alpha\cos(\theta-\alpha) - l_{l1}\frac{\mu}{I_{l1}}\sin(\theta-\alpha)$$

$$^{ground}H_4 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$^{ground}H_{5,3} = \frac{l_{l1}^2}{2I_{l1}}(f_{p1}+f_{p2})\cos(\theta-\alpha)$$

$$^{ground}H_{5,4} = \frac{l_{l1}^2}{2I_{l1}}(g(m_b+\frac{m_{l1}}{2})\cos\alpha - (f_{p1}+f_{p2})\cos(\theta-\alpha)) - g\cos\alpha$$

$$^{ground}H_{5,7} = \frac{\mu}{I_{l1}}\frac{l_{l1}}{2}$$

$$^{ground}H_{5,8} = -\frac{\mu}{I_{l1}}\frac{l_{l1}}{2}$$

$$^{ground}H_{6,4} = -g\sin\alpha$$

$$^{ground}H_{6,8} = l_{l1}\omega_\alpha$$

A.3 GPS measure Jacobian matrix

$$H_{GPS} = \begin{bmatrix} 1 & 0 & -l_{GPS}\sin\theta & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & -l_{GPS}\cos\theta & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Bibliography

[1] Yang, H., Lee, Y., Jeon, SY. et al. Multi-rotor drone tutorial: systems, mechanics, control and state estimation. Intel Serv Robotics 10, 79–93 (2017). https://doi.org/10.1007/s11370-017-0224-y 8

[2] G. Nava, L. Fiorio, S. Traversaro and D. Pucci, "Position and Attitude Control of an Underactuated Flying Humanoid Robot," 2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids), Beijing, China, 2018, pp. 1-9, doi: 10.1109/HUMANOIDS.2018.8624985. keywords: Humanoid robots;Symmetric matrices;Task analysis;Attitude control;Dynamics;Legged locomotion 6, 9, 10, 11

[3] D. Pucci, S. Traversaro and F. Nori, "Momentum Control of an Underactuated Flying Humanoid Robot," in IEEE Robotics and Automation Letters, vol. 3, no. 1, pp. 195-202, Jan. 2018, doi: 10.1109/LRA.2017.2734245 keywords: Humanoid robots;Robot kinematics;Robot sensing systems;Force;Mobile robots;Acceleration;Aerial systems;mechanics and control;humanoid robots;motion control 6, 9, 10, 11

[4] H. Yang and D. Lee, "Dynamics and control of quadrotor with robotic manipulator," 2014 IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, 2014, pp. 5544-5549, doi: 10.1109/ICRA.2014.6907674. keywords: Manipulator dynamics;Vehicle dynamics;Gravity;Aerodynamics 9

[5] K. Chaisena, K. Chamniprasart and S. Tantrairatn, "An Automatic Stabilizing System for Balancing a Multi-Rotor Subject to Variations in Center of Gravity and Mass," 2018 Third International Conference on Engineering Science and Innovative Technology (ESIT), North Bangkok, Thailand, 2018, pp. 1-5, doi: 10.1109/ESIT.2018.8665339. keywords: Payloads;Gravity;Rotors;Batteries;Helicopters;Pins;Stability;Balancing;PID Controller;Counter balance;Quad - Rotor 9

[6] Y. Li et al., "Jet-HR2: A Flying Bipedal Robot Based on Thrust Vector Control," in IEEE Robotics and Automation Letters, vol. 7, no. 2, pp. 4590-4597, April 2022, doi: 10.1109/LRA.2022.3152231. keywords: Robots;Legged locomotion;Fans;Torque;Actuators;Robot kinematics;Hip;Humanoid robot systems;aerial systems: mechanics and control 6, 9, 10, 11, 13

[7] Shin, W.D., Phan, HV., Daley, M.A. et al. Fast ground-to-air transition with avian-inspired multifunctional legs. Nature 636, 86–91 (2024). https://doi.org/10.1038/s41586-024-08228-9 6, 8, 14, 16

[8] S. Li et al., "A High-Payload Robotic Hopper Powered by Bidirectional Thrusters," in IEEE Transactions on Robotics, vol. 41, pp. 5307-5326, 2025, doi: 10.1109/TRO.2025.3600127. keywords: Robots;Legged locomotion;Payloads;Attitude control;Quadrotors;Propulsion;Propellers;Elastomers;Computational modeling;Trajectory;Autonomous navigation;high payload;hopping;legged robots;neural network (NN);spring-loaded inverted pendulum (SLIP) 6, 9, 10, 12, 15

[9] Carpentier, J., Wieber, PB. Recent Progress in Legged Robots Locomotion Control. Curr Robot Rep 2, 231–238 (2021). https://doi.org/10.1007/s43154-021-00059-0 8

[10] F. Shi, T. Anzai, Y. Kojio, K. Okada and M. Inaba, "Learning Agile Hybrid Whole-body Motor Skills for Thruster-Aided Humanoid Robots," 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Kyoto, Japan, 2022, pp. 12986-12993, doi: 10.1109/IROS47612.2022.9981974. keywords: Legged locomotion;Attitude control;Humanoid robots;Reinforcement learning;Birds;Computational efficiency;Complexity theory. 9

[11] M. Menner and K. Berntorp, "Simultaneous State Estimation and Contact Detection for Legged Robots by Multiple-Model Kalman Filtering," 2024 European Control Conference (ECC), Stockholm, Sweden, 2024, pp. 2768-2773, doi: 10.23919/ECC64448.2024.10590751. keywords: Legged locomotion;Switched systems;Simulation;Filtering algorithms;Hardware;Real-time systems;Kalman filters 10

[12] K. Tong, Y. Hu, B. Dikic, S. Solmaz, F. Fraundorfer and D. Watzenig, "Robots Saving Lives: A Literature Review About Search and Rescue (SAR) in Harsh Environments," 2024 IEEE Intelligent Vehicles

Symposium (IV), Jeju Island, Korea, Republic of, 2024, pp. 953-960, doi: 10.1109/IV55156.2024.10588685. keywords: Intelligent vehicles;Terrorism;Disasters;Bibliographies;Focusing;Software;Hardware, 8

[13] A. Delbene and M. Baglietto, "On Formation Control Strategies in a Failure-Prevention Scenario for Multi-UAV Payload Transportation," 2025 IEEE 21st International Conference on Automation Science and Engineering (CASE), Los Angeles, CA, USA, 2025, pp. 185-191, doi: 10.1109/CASE58245.2025.11163951. keywords: Heuristic algorithms;Transportation;Autonomous aerial vehicles;Mathematical models;Formation control;Vehicle dynamics;State estimation;Payloads;Cables;Videos, 8

[14] Lluvia, I.; Lazkano, E.; Ansuategi, A. Active Mapping and Robot Exploration: A Survey. Sensors 2021, 21, 2445. https://doi.org/10.3390/s21072445 8

[15] Torres-Pardo, Adriana, et al. "Legged locomotion over irregular terrains: State of the art of human and robot performance." Bioinspiration & Biomimetics 17.6 (2022): 061002, 8

[16] Gupte, Shweta, Paul Infant Teenu Mohandas, and James M. Conrad. "A survey of quadrotor unmanned aerial vehicles." 2012 Proceedings of IEEE Southeastcon (2012): 1-6. 8

[17] Klemm, Victor, et al. "Ascento: A two-wheeled jumping robot." 2019 International conference on robotics and automation (ICRA). IEEE, 2019 8

[18] Zhang, Jun, et al. "A bio-inspired jumping robot: Modeling, simulation, design, and experimental results." Mechatronics 23.8 (2013): 1123-1140 8