SAPIENZA
UNIVERSITÀ DI ROMA

Reactive obstacle avoidance for quadrotor UAVs based on dynamic feedback linearization

Facoltà di Ingegneria dell'Informazione, Informatica e Statistica
Dipartimento di Ingegneria Informatica, Automatica e Gestionale
Corso di laurea in Intelligenza Artificiale e Robotica

Candidato
Andriy Sarabakha
1192958

Relatore
Prof. Giuseppe Oriolo

Correlatore
Dott. Lorenzo Rosa

A/A 2014/2015

"A helicopter is a mechanical engineer's dream and an aerodynamicist's nightmare."

John Watkinson, British teacher

"If you are in trouble anywhere, an airplane can fly over and drop flowers, but a helicopter can land and save your life."

Igor Sikorsky, Ukrainian American aviation pioneer

Abstract

This work addresses the problem of ensuring a safe navigation in an unknown cluttered environment for a quadrotor-like Vertical Take-Off and Landing (VTOL) Unmanned Aerial Vehicle (UAV). Consequently, the main issue is to perform effective obstacle avoidance, and we want to solve this problem by designing a fast reactive behaviour able to ensure absence of collisions both with static and moving obstacles. Moreover, we want to consider the motion capabilities of the vehicle, so that the obstacle avoidance controller produces feasible inputs for the robot.

To this aim, we rely on classical obstacle avoidance technique which uses Artificial Potential Fields (APF). In a first moment, we consider the hypotheses of quasi-stationary flight and we feed the input given by the APF in a controller based on Dynamic Feedback Linearization (DFL). We show that the closed-loop behaviour of the resulting linearized system is not acceptable, especially when the vehicle approaches the obstacle with high speed. Thus, to solve the issue, we consider the use of a first-order feed-forward, which is used to generate a short-term trajectory. By differentiating the generated trajectory, we compute suitable inputs for the DFL controller. At the end, by considering the motion capabilities of the VTOL UAV in the design of the controller, we obtain a control framework which ensures the feasibility of the commanded trajectory for the original system.

Contents

List of Figures

Abbreviations

BLDC	Brush-Less Direct Current
DFL	Dynamic Feedback Linearization
DOF	Degrees Of Freedom
GPS	Global Positioning System
HLP	High Level Processor
IARC	International Aerial Robotics Competition
IMU	Inertial Measurement Unit
LLP	Low Level Processor
MAV	Micro Aerial Vehicle
MFR	Miniature Flying Robots
MIMO	Multiple-Input Multiple-Output
PD	Proportional-Derivative
PID	Proportional-Integral-Derivative
PPM	Pulse Position Modulation
RGB	Red-Green-Blue
ROS	Robot Operating System
UAV	Unmanned Aerial Vehicle
VM	Virtual Model
VTOL	Vertical Take-Off and Landing

Symbols

$\mathbf{A}$	quadrotor linear acceleration in world frame	$[\mathrm{m/s^2}]$
$\mathbf{A}_{\mathcal{B}}$	quadrotor linear acceleration in body frame	$[\mathrm{m/s^2}]$
A_x	aerodynamic forces about x-axis	[N]
A_y	aerodynamic forces about y-axis	[N]
A_z	aerodynamic forces about z-axis	[N]
A_p	aerodynamic moments about x-axis	$[\mathrm{N \cdot m}]$
A_q	aerodynamic moments about y-axis	$[\mathrm{N \cdot m}]$
A_r	aerodynamic moments about z-axis	$[\mathrm{N \cdot m}]$
$\mathcal{B}$	quadrotor body	
b	propellers thrust coefficient	$[\mathrm{N \cdot s^2}]$
$\mathcal{C}$	configuration space	
$\mathcal{CO}$	image of obstacle in configuration space	
D_x	unmodelled forces about x-axis	[N]
D_y	unmodelled forces about y-axis	[N]
D_z	unmodelled forces about z-axis	[N]
D_p	external disturbance moments about x-axis	$[\mathrm{N \cdot m}]$
D_q	external disturbance moments about y-axis	$[\mathrm{N \cdot m}]$
D_r	external disturbance moments about z-axis	$[\mathrm{N \cdot m}]$
d	propellers drag coefficient	$[\mathrm{N \cdot m \cdot s^2}]$
$\mathbf{F}_e$	external forces vector	[N]
f_i	force generated by i^{th} propeller	[N]
g	gravity acceleration	$[\mathrm{m/s^2}]$
$\mathbf{I}$	inertia matrix	$[\mathrm{kg \cdot m^2}]$
I_x	moment of inertia about x-axis	$[\mathrm{kg \cdot m^2}]$
I_y	moment of inertia about y-axis	$[\mathrm{kg \cdot m^2}]$

I_z	moment of inertia about z-axis	$[\mathrm{kg} \cdot \mathrm{m}^2]$
l	quadrotor arm length	[m]
l_p	quadrotor propeller length	[m]
m	quadrotor mass	[kg]
$\mathcal{O}_i$	i_{th} obstacle	
p	quadrotor angular velocity in body frame x-axis	[rad/s]
$\mathbf{q}$	quadrotor configuration	
q	quadrotor angular velocity in body frame y-axis	[rad/s]
$\mathbf{R}$	rotation matrix	
R	radius of quadrotor bounding sphere	[m]
r	quadrotor angular velocity in body frame z-axis	[rad/s]
$\mathbf{T}$	transformation matrix	
T	quadrotor thrust force	[N]
U_r	repulsive potential field	[N]
$\mathbf{u}$	control inputs to quadrotor system	
$\hat{\mathbf{u}}$	control inputs to extended quadrotor system	
u	quadrotor linear velocity in world frame x-axis	[m/s]
$\mathbf{V}$	quadrotor linear velocity in world frame	[m/s]
V	quadrotor volume	$[\mathrm{m}^3]$
$\mathbf{V}_\mathcal{B}$	quadrotor linear velocity in body frame	[m/s]
v	quadrotor linear velocity in world frame y-axis	[m/s]
$\mathcal{W}$	workspace	
w	quadrotor linear velocity in world frame z-axis	[m/s]
$\mathbf{x}$	quadrotor state	
$\hat{\mathbf{x}}$	quadrotor extended state	
x	quadrotor position along world frame x-axis	[m]
$\mathbf{y}$	output of quadrotor system	
y	quadrotor position along world frame y-axis	[m]
$\mathbf{z}$	state of linearized system	
z	quadrotor position along world frame z-axis	[m]
θ	pitch angle in world frame	[rad]
$\boldsymbol{\tau}_e$	external torques vector	$[\mathrm{N} \cdot \mathrm{m}]$

τ_i	torque generated by i^{th} propeller	[N · m]
τ_θ	pitch torque in body frame	[N · m]
τ_ϕ	roll torque in body frame	[N · m]
τ_ψ	yaw torque in body frame	[N · m]
ϕ	roll angle in world frame	[rad]
ψ	yaw angle in world frame	[rad]
$\boldsymbol{\omega}$	quadrotor angular velocity in world frame	[rad/s]
$\boldsymbol{\omega}_\mathcal{B}$	quadrotor angular velocity in body frame	[rad/s]
ω_i	rotational speed of i^{th} propeller	[rad/s]

Chapter 1

Introduction

1.1 Motivations and objectives

Since the beginning of time flying objects have exerted a great fascination on man. The last decades has seen many exciting developments in the area of *Unmanned Aerial Vehicles* (UAVs). Therefore, the scientific challenge in UAV design and control is very

FIGURE 1.1: Quadrotor aircraft.

motivating. On the other hand, UAVs are gaining increasing interest because of a wide area of applications from military to civilian fields. At the same time, in the past years many works were concerned on developing methods for navigation and obstacle avoidance for the autonomous ground vehicles. The existing ground robots have limitations on reaching the desired location in several applications. Thus, the recent progresses in technology push towards developing new mobility concepts, which include flying systems. However, despite the dynamics of flying vehicles is more complex than the one of ground robots, one could still apply the techniques already developed for ground robots. An attractive group of flying robots is made up of quadrotor aircraft (shown in Figure 1.1).

In particular, a *quadrotor* or *quadcopter* is an UAV which has *Vertical Take-Off and Landing* (VTOL) characteristics, like helicopters, but lifted and propelled by four rotors in a cross configuration. Due to its simple symmetric mechanical configuration, it is capable of flying without all those complex linkages appearing in typical helicopters. However, like a classical helicopter, a quadrotor has non-linear dynamics. Moreover, it is really hard to model all secondary order effects. Thus, a control system capable of dealing with non-linearity, unmodeled dynamics and disturbances is needed. Therefore, the interest comes not only from its dynamics, which represent an attractive control problem, but also from the design issue. Integrating the sensors, actuators and intelligence into a lightweight flying system is not trivial.

Currently, technology provides a new generation of integrated micro *Inertial Measurement Unit* (IMU), and the latest progresses in high density power storage offers very promising results especially for *Micro Aerial Vehicles* (MAV); thus, the cost and size reduction of flying systems makes them very interesting both for civilian and military applications. In particular, quadrotors are very promising platforms: because of their increased mobility in the environment, they are able to be effectively employed both indoor and outdoor. Moreover, quadrotor propulsion system is very robust and guarantees a stable flight. Besides, quadrotors are low-noise, emission-free and environmentally friendly devices. On the other hand, small quadrotors have limited payload, that implies a selection of light sensors, and the limited computational power on-board makes the development of fully autonomous quadrotors very challenging. In the last ten years, the UAVs' autonomy was dramatically improved with the development of new navigation, communication, control and image processing technologies. Thus, nowadays we can consider some UAVs as intelligent robotic systems integrating perception, reasoning and decision making capabilities, which allow to operate in complex environments.

Recently, small quadrotors had a quick growth and had attracted much interest in the research community. Indeed, the quadrotor has become a standard platform in the

experimental applications. Great manoeuvrability and small size of those vehicles make them suitable for indoor use. However, unsupervised flight of aerial vehicles is a hard challenge, especially in indoor applications, where it is not possible to use GPS[1].

Most of quadrotor's applications can in principle be achieved by using straightforward linear control systems, such as *Proportional-Derivative* (PD) controller. Nevertheless, to fly in cluttered environments it could be useful to obtain a very accurate stabilization, a highly precise navigation and collision avoidance with tracking of aggressive manoeuvres. To these purposes, non-linear control techniques based on *Dynamic Feedback Linearization* (DFL) should be designed to provide better convergence and robustness performances. Through a change of variables, DFL transforms a non-linear system into an equivalent linear system with suitable control input and output functions. This equivalent linear system consists of a number of decoupled controllable and observable canonical forms. However, this technique needs exact cancellations and full accessibility to the state of the quadrotor. For these reasons it presents several critical issues when dealing with real implementation in the physical world.

1.2 Overview

The outline of this work is as follows. After this introductory chapter (Chapter 1), in Chapter 2 we present the quadrotor dynamical model and we show the basic principles of DFL. In Chapter 3 a control analysis of the quadrotor is reported in order to obtain the feed-forward+feedback controller with obstacle avoidance. In Chapter 4 the simulation results are given to check the designed controller behaviour. In Chapter 5 the experimental system is described and the experimental results are presented. Finally, we will draw some concluding remarks in Chapter 6. The derivation of 3D rotation matrix and its derivative is reported in Appendix A. The bases of Lie derivative are described in Appendix B. The computation of inertia matrices for solid cuboids and cylinders is done in Appendix C. The reprojection of an image point to a correspondent point in space is shown in Appendix D.

In Figure 1.2 is depicted the overall system architecture. High level navigation commands are provided by an *operator* through a *joystick* or keyboard. This command is interpreted by the *controller* as a desired velocity. In order to compute the snap[2] needed to achieve this desired velocity, the controller needs both the desired quantities and the actual ones. Some quantities, like altitude, attitude, vertical velocity, angular velocities

[1] *Global Positioning System* (GPS) signal has very poor performance in indoor environments.
[2] *Snap* is the fourth derivative of the position vector with respect to time. The choice of commanding with snap is argued out later in Subsection 2.2.2.

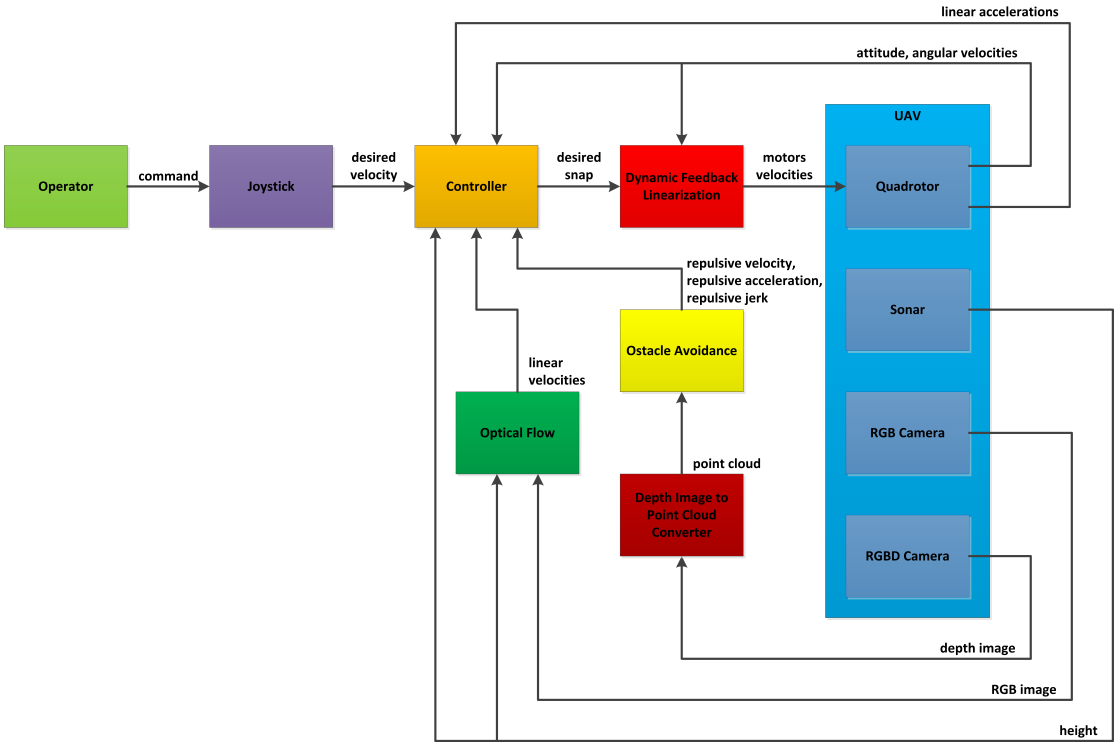


FIGURE 1.2: System architecture.

and linear accelerations, are directly measured by using *UAV*'s sensors. Others need to be estimated by additional processes. The *obstacle avoidance* block needs to compute the desired velocities, accelerations and jerks in order to prevent hitting an object during the execution of the desired velocities given by the user; the *optical flow* block estimates velocities on the horizontal plane. Furthermore, *depth image to point cloud converter* processes the depth map in order to return a set of 3D points. Once the desired snap is computed by the controller, it is sent to the *dynamic feedback linearization*, which, inverting the quadrotor's dynamics, computes the commanded motors velocities, and in turn send them to the *quadrotor*.

Chapter 2

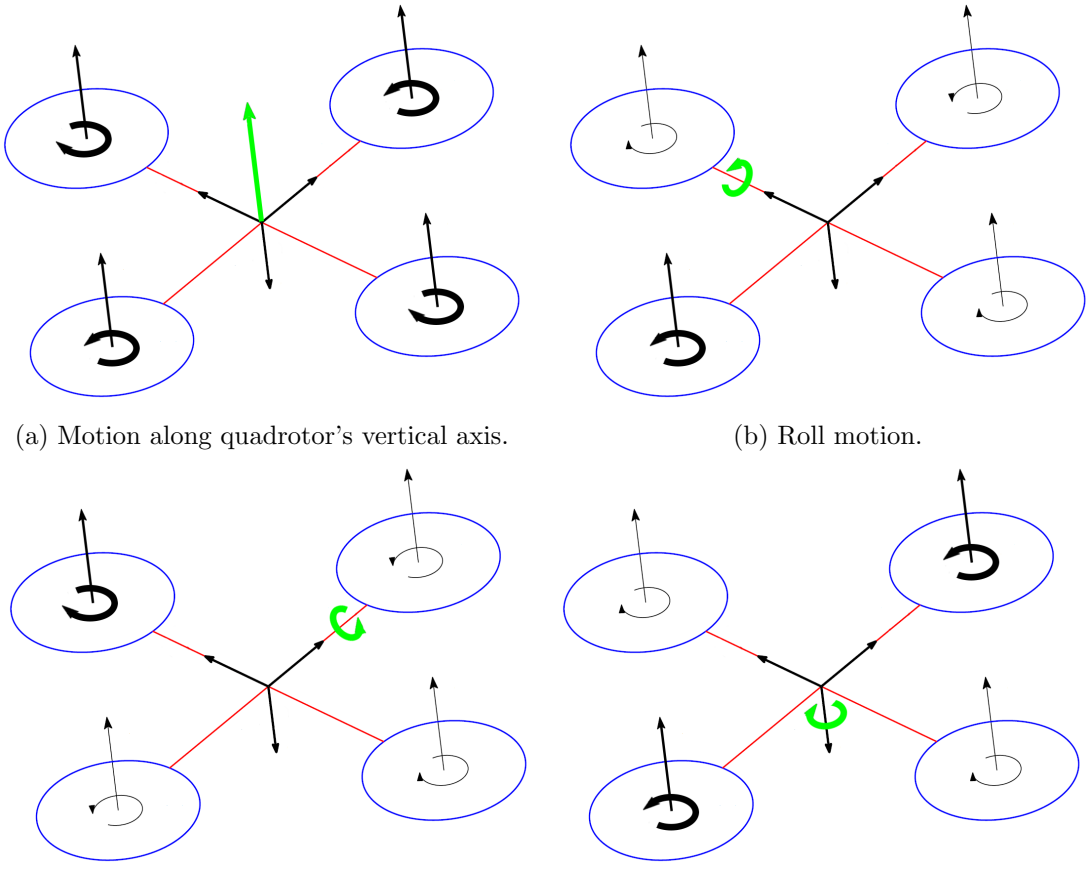
Modelling and control

In this chapter we derive and describe a dynamic model representing a quadrotor. A complete quadrotor's model is provided, afterwards further assumptions and simplifications are made. The second part of this chapter introduces the dynamic feedback linearization technique in order to design a feedback control law for a quadrotor.

2.1 Quadrotor dynamics

To develop control laws and estimation schemes, it is needed to model the dynamics of the quadrotor and understand how forces and torques act on the vehicle. First, dynamic and kinematic differential equations based on the quadrotor model are derived, after which complete quadrotor model will be developed. Then, a state variable notation is introduced.

A quadrotor is an aerial vehicle actuated by modulating the speed command of each of the four motors. It consists of four identical rotors and propellers located at the extremities of a cross-shaped frame. In a quadrotor all the movements are the consequence of the propellers' speed (as shown in Figure 2.1): two propellers rotate in a clockwise direction (front and rear propellers) while the other two rotate in a counter-clockwise direction (left and right propellers). Changing simultaneously the throttle of all motors, while the vehicle is horizontal, produces vertical motion (Figure 2.1a). A difference of speed between the blades on the same axis carries a rotation of the aircraft along the other axis. Roll moments are produced by adjusting the thrust of the left motor with respect to the right one (Figure 2.1b). Pitching moments are produced in a similar way by increasing the thrust of the front motor while decreasing that of the rear motor or vice versa (Figure 2.1c). Yawing moments are slightly more subtle:



(a) Motion along quadrotor's vertical axis.

(b) Roll motion.

(c) Pitch motion.

(d) Yaw motion.

FIGURE 2.1: The quadrotor concept. The width of the arrows is proportional to the propellers' angular speed.

if the front and rear motors (which spin clockwise) spin faster than the left and right motors (which spin counter-clockwise), yawing results due to the difference in rotor drag moments on the respective motors (Figure 2.1d). Therefore, the quadrotor is a highly non-linear dynamic system with four control inputs (angular speed of four rotors) and six degrees of freedom (position and orientation in space), resulting in a multiple-input multiple-output (MIMO) under-actuated system.

2.1.1 Configuration definition

Let the world fixed inertial reference frame be $\{\vec{x}_{\mathcal{W}}, \vec{y}_{\mathcal{W}}, \vec{z}_{\mathcal{W}}\}$ and the body frame be $\{\vec{x}_{\mathcal{B}}, \vec{y}_{\mathcal{B}}, \vec{z}_{\mathcal{B}}\}$. The origin of the body frame is located at the center of mass of the quadrotor. Axes $\vec{x}_{\mathcal{B}}$ and $\vec{y}_{\mathcal{B}}$ lie in the plane defined by the centres of the four rotors and respectively point toward motor 1 and motor 2, as illustrated in Figure 2.2. Axis $\vec{z}_{\mathcal{W}}$ points downward, opposite to the direction of gravity, as well as axis $\vec{z}_{\mathcal{B}}$ which is opposite to the direction of the total thrust.

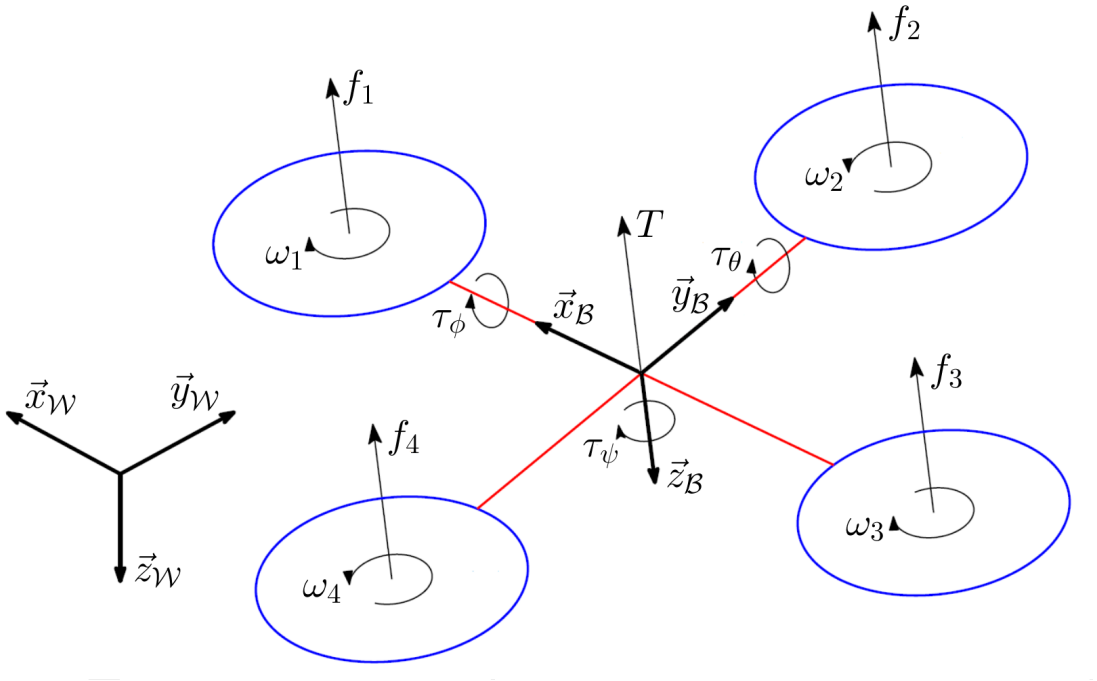


FIGURE 2.2: Quadrotor model with reference frames.

The four propellers rotations generate four forces (f_1, f_2, f_3 and f_4), directed along the axis of rotation $\vec{z}_\mathcal{B}$ and with module proportional to the speed of rotation, and four torques (τ_1, τ_2, τ_3 and τ_4), around the axis of rotation $\vec{z}_\mathcal{B}$ and with module proportional to the speed of rotation [1]:

$$\begin{aligned} f_i &= b\omega_i^2 \\ \tau_i &= dl\omega_i^2 \end{aligned}, \qquad i = 1, \ldots, 4, \tag{2.1}$$

where b is the propeller thrust coefficient, d is the propeller drag coefficient and ω_i is the rotational speed of the i^{th} propeller.

From the point of view of control, it is easier to consider these four forces and four torques as the union of the forces, directed towards the vertical axis of the quadrotor $\vec{z}_\mathcal{W}$ and applied to its center of mass, and three rotation torques, along roll ($\vec{x}_\mathcal{W}$), pitch ($\vec{y}_\mathcal{W}$) and yaw ($\vec{z}_\mathcal{W}$). The quadrotor electric motors are velocity controlled, so the vector $\mathbf{u}$ of control inputs may by considered directly as

$$\mathbf{u} = \begin{bmatrix} T & \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix}^T,$$

where T is the total thrust and acts along $\vec{b}_{\mathcal{W}}$ axis, whereas τ_ϕ, τ_θ and τ_ψ are the moments acting around $\vec{x}_{\mathcal{W}}$, $\vec{y}_{\mathcal{W}}$ and $\vec{z}_{\mathcal{W}}$ axes, respectively. Under these considerations, the relation between u and ω_i, $i = 1, \ldots, 4$, becomes algebraic [2]:

$$\begin{aligned} T &= f_1 + f_2 + f_3 + f_4 \\ \tau_\phi &= l(-f_2 + f_4) \\ \tau_\theta &= l(f_1 - f_3) \\ \tau_\psi &= -\tau_1 + \tau_2 - \tau_3 + \tau_4, \end{aligned} \tag{2.2}$$

where l is the arm length. Applying (2.1) to (2.2) and writing it in matrix form:

$$\begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix} = \begin{bmatrix} b & b & b & b \\ 0 & -bl & 0 & bl \\ bl & 0 & -bl & 0 \\ -dl & dl & -dl & dl \end{bmatrix} \begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix}.$$

This relation is always invertible, when $l \neq 0$, $b \neq 0$ and $d \neq 0$. Therefore, inputs can then be brought back to the speed of the individual propellers using the following inverse transformation:

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \frac{1}{4bdl} \begin{bmatrix} dl & 0 & 2d & -b \\ dl & -2d & 0 & b \\ dl & 0 & -2d & -b \\ dl & 2d & 0 & b \end{bmatrix} \begin{bmatrix} T \\ \tau_\phi \\ \tau_\theta \\ \tau_\psi \end{bmatrix}.$$

The absolute position of a quadrotor (three DOF) is described by the three Cartesian coordinates (x, y and z) of its center of mass in the world frame and its attitude (three DOF) by the three Euler's angles (ϕ, θ and ψ). These three angles are respectively called roll ($-\frac{\pi}{2} < \phi < \frac{\pi}{2}$), pitch ($-\frac{\pi}{2} < \theta < \frac{\pi}{2}$) and yaw ($0 \leq \psi < 2\pi$).

The derivative with respect to time of the position (x, y, z) is given by

$$\mathbf{V} = \begin{bmatrix} \dot{x} & \dot{y} & \dot{z} \end{bmatrix}^T = \begin{bmatrix} u & v & w \end{bmatrix}^T,$$

where $\mathbf{V}$ is the absolute velocity of the quadrotor's center of mass expressed with respect to the world fixed inertial reference frame. Let $\mathbf{V}_{\mathcal{B}} \in \mathbb{R}^3$ be the absolute velocity of the quadrotor expressed in the body fixed reference frame. So, $\mathbf{V}$ and $\mathbf{V}_{\mathcal{B}}$ are related by

$$\mathbf{V} = \mathbf{R}\mathbf{V}_{\mathcal{B}}, \tag{2.3}$$

where $\mathbf{R} \in \mathsf{SO}(3)$[1] is the rotation matrix from the body frame to the world frame and is computed in Appendix A:

$$\mathbf{R} = \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\phi\sin\theta - \cos\phi\sin\psi & \sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\psi\sin\theta & \cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\phi\cos\theta \end{bmatrix}.$$

Similarly, the derivatives with respect to time of the angles (ϕ, θ, ψ) are given by

$$\boldsymbol{\omega} = \begin{bmatrix} \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T,$$

and the angular velocities expressed in the body frame are

$$\boldsymbol{\omega}_{\mathcal{B}} = \begin{bmatrix} p & q & r \end{bmatrix}^T.$$

The relation between $\boldsymbol{\omega}$ and $\boldsymbol{\omega}_{\mathcal{B}}$ is given by

$$\boldsymbol{\omega} = \mathbf{T}\boldsymbol{\omega}_{\mathcal{B}}, \tag{2.4}$$

in which $\mathbf{T}$ is the transformation matrix given by

$$\mathbf{T} = \begin{bmatrix} 1 & \sin\phi\tan\theta & \cos\phi\tan\theta \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi\sec\theta & \cos\phi\sec\theta \end{bmatrix}.$$

Correspondingly, let $\mathbf{A} \in \mathbb{R}^3$ be the absolute acceleration of the quadrotor expressed in the world fixed reference frame and $\mathbf{A}_{\mathcal{B}} \in \mathbb{R}^3$ be the absolute acceleration of the quadrotor expressed in the body reference frame. So, the relation between $\mathbf{A}$ and $\mathbf{A}_{\mathcal{B}}$ is computed by the analytical derivative of (2.3):

$$\mathbf{A} = \dot{\mathbf{R}}\mathbf{V}_{\mathcal{B}} + \mathbf{R}\mathbf{A}_{\mathcal{B}}.$$

[1] $\mathsf{SO}(3)$ denotes the 3D rotation group.

In Appendix A.1 is shown how the time derivative of $\mathbf{R}$ can be expressed as a function of $\mathbf{R}$ itself. Using this expression, the final relation is obtained[2]:

$$\mathbf{A} = [\boldsymbol{\omega}]_\times \mathbf{R}\mathbf{V}_\mathcal{B} + \mathbf{R}\mathbf{A}_\mathcal{B} = [\boldsymbol{\omega}]_\times \mathbf{V} + \mathbf{R}\mathbf{A}_\mathcal{B}.$$

2.1.2 Complete quadrotor model

Using the Newton-Euler equations about the center of mass, the dynamic equations for the quadrotor are the following [3]:

$$\begin{aligned} m\dot{\mathbf{V}} &= \mathbf{F}_e \\ \mathbf{I}\dot{\boldsymbol{\omega}}_\mathcal{B} &= -\boldsymbol{\omega}_\mathcal{B} \times \mathbf{I}\boldsymbol{\omega}_\mathcal{B} + \boldsymbol{\tau}_e, \end{aligned} \tag{2.5}$$

where m is the mass and $\mathbf{I}$ is the diagonal inertia matrix given by

$$\mathbf{I} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix} \tag{2.6}$$

and $\mathbf{F}_e$ is the vector of external forces and $\boldsymbol{\tau}_e$ is the vector of external torques. They contain the quadrotor's mass, the aerodynamic forces, the thrust and the torques developed by the four rotors. Some calculations yield the following form for theses two vectors

$$\begin{aligned} \mathbf{F}_e &= \begin{bmatrix} -\left(\cos\phi \sin\theta \cos\psi + \sin\phi \sin\psi\right) T + A_x + D_x \\ -\left(\cos\phi \sin\theta \sin\psi - \sin\phi \cos\psi\right) T + A_y + D_y \\ -\cos\phi \cos\theta T + mg + A_z + D_z \end{bmatrix} \\ \boldsymbol{\tau}_e &= \begin{bmatrix} \tau_\phi + A_p + D_p + G_p \\ \tau_\theta + A_q + D_q + G_q \\ \tau_\psi + A_r + D_r + G_r \end{bmatrix}, \end{aligned} \tag{2.7}$$

in which

[2] $[\boldsymbol{\omega}]_\times$ denotes the skew-symmetric matrix, given by (A.5).

- A_x, A_y and A_z are the aerodynamic forces acting on the UAV[3];
- D_x, D_y and D_z are the external disturbances and unmodelled forces, given by wind, contacts or collisions;
- A_p, A_q and A_r are the aerodynamic moments acting on the UAV[4];
- D_p, D_q and D_r are the external disturbance moments, like wind;
- G_p, G_q and G_r are the gyroscopic effects generated by propellers;
- and g is the gravity acceleration ($g = 9.81\text{m/s}^2$).

Using dynamic and kinematic differential equations (2.3), (2.4), (2.5) and (2.7), the following system of non-linear differential equations is obtained

$$\begin{cases} \dot{x} = u \\ \dot{y} = v \\ \dot{z} = w \\ \dot{\phi} = p + \sin\phi\tan\theta q + \cos\phi\tan\theta r \\ \dot{\theta} = \cos\phi q - \sin\phi r \\ \dot{\psi} = \frac{\sin\phi}{\cos\theta}q + \frac{\cos\phi}{\cos\theta}r \\ \dot{u} = -\frac{1}{m}\left(\cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi\right)T + \frac{A_x+D_x}{m} \\ \dot{v} = -\frac{1}{m}\left(\cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi\right)T + \frac{A_y+D_y}{m} \\ \dot{w} = -\frac{1}{m}\cos\phi\cos\theta T + g + \frac{A_z+D_z}{m} \\ \dot{p} = \frac{I_y-I_z}{I_x}qr + \frac{1}{I_x}\tau_\phi + \frac{A_p+D_p+G_p}{I_x} \\ \dot{q} = \frac{I_z-I_x}{I_y}pr + \frac{1}{I_y}\tau_\theta + \frac{A_q+D_q+G_q}{I_y} \\ \dot{r} = \frac{I_x-I_y}{I_z}pq + \frac{1}{I_z}\tau_\psi + \frac{A_r+D_r+G_r}{I_z}. \end{cases} \tag{2.8}$$

2.1.3 Assumptions and simplifications

Let's assume that the quadrotor propeller speed is directly proportional to the current supplied to the motor. If the velocity induced by the wake is omitted, then thrust and torque are proportional to the square of the angular speed of the propeller, as supposed in the relation (2.1).

[3] Aerodynamic forces are computed as $A_i = \frac{1}{2}\rho_{air}C_i|\mathbf{V}|^2V^{2/3}$, where ρ_{air} is the air density, C_i are the aerodynamic coefficients and V is the quadrotor volume [4].

[4] Aerodynamic moments are computed as $A_i = \frac{1}{2}\rho_{air}C_i|\boldsymbol{\omega}|^2V$, where ρ_{air} is the air density, C_i are the aerodynamic coefficients and V is the quadrotor volume [4].

The quadrotor model (2.8) is assumed to be sufficiently accurate in representing all quadrotor functional motions. But is not suitable for control design, because it depends upon the aerodynamic forces (A_x, A_y and A_z) and moments (A_p, A_q and A_r), which are difficult to model in presence of parametric uncertainties. On the other hand, the external disturbance forces (D_x, D_y and D_z) and moments (D_p, D_q and D_r) are unknown in the presence of unpredictable winds and turbulences. Thus, these terms are neglected during the control design and are considered as disturbances. Many of those terms are dissipative (e.g. due to the air friction), so that their contribution partially contributes to the stability of the system. The gyroscopic terms (G_p, G_q and G_r) are second-order terms with respect to the other torques acting on the vehicle, so they can be neglected. Taken all these assumptions, the complete quadrotor model (2.8) becomes:

$$\begin{cases} \dot{x} = u \\ \dot{y} = v \\ \dot{z} = w \\ \dot{\phi} = p + \sin\phi \tan\theta q + \cos\phi \tan\theta r \\ \dot{\theta} = \cos\phi q - \sin\phi r \\ \dot{\psi} = \frac{\sin\phi}{\cos\theta} q + \frac{\cos\phi}{\cos\theta} r \\ \dot{u} = -\frac{1}{m} \left(\cos\phi \cos\psi \sin\theta + \sin\phi \sin\psi\right) T \\ \dot{v} = -\frac{1}{m} \left(\cos\phi \sin\psi \sin\theta - \cos\psi \sin\phi\right) T \\ \dot{w} = -\frac{1}{m} \cos\phi \cos\theta T + g \\ \dot{p} = \frac{I_y - I_z}{I_x} qr + \frac{1}{I_x} \tau_\phi \\ \dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{1}{I_y} \tau_\theta \\ \dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{1}{I_z} \tau_\psi, \end{cases} \tag{2.9}$$

which can be described in state space form

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u},$$

where

$$\mathbf{x} = \begin{bmatrix} x & y & z & \phi & \theta & \psi & u & v & w & p & q & r \end{bmatrix}^T,$$

$$f(\mathbf{x}) = \begin{bmatrix} u \\ v \\ w \\ p + \sin\phi\tan\theta q + \cos\phi\tan\theta r \\ \cos\phi q - \sin\phi r \\ \frac{\sin\phi}{\cos\theta} q + \frac{\cos\phi}{\cos\theta} r \\ 0 \\ 0 \\ g \\ \frac{I_y - I_z}{I_x} qr \\ \frac{I_z - I_x}{I_y} pr \\ \frac{I_x - I_y}{I_z} pq \end{bmatrix},$$

$$g(\mathbf{x}) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ -\frac{1}{m}(\cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi) & 0 & 0 & 0 \\ -\frac{1}{m}(\cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi) & 0 & 0 & 0 \\ -\frac{1}{m}\cos\phi\cos\theta & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \end{bmatrix}$$

and

$$\mathbf{u} = \begin{bmatrix} T & \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix}^T.$$

2.2 Dynamic feedback linearization

Due to under-actuation, the quadrotor system (2.9) can not be transformed into an equivalent linear and controllable system by static state feedback. However, it is possible

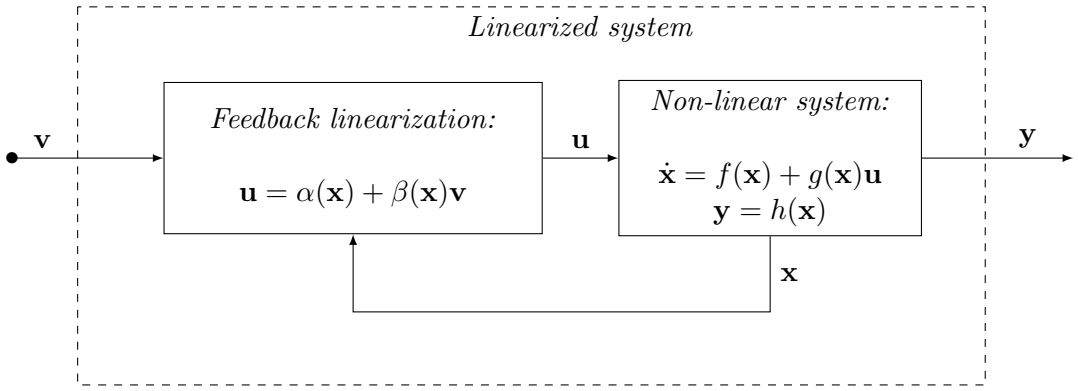


FIGURE 2.3: Block diagram of the feedback linearization.

to resort to dynamic state feedback to obtain full state linearization. *Dynamic feedback linearization* is a commonly used technique to approach non-linear control design that algebraically transforms a non-linear system into an equivalent linear, controllable and observable one, consisting of a number of decoupled canonical forms, so that the linear control theory can be applied. This problem is known as the *exact linearization problem.*

2.2.1 Introduction

Given a non-linear system [5, 6]

$$\begin{cases} \dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u} \\ \mathbf{y} = h(\mathbf{x}), \end{cases} \tag{2.10}$$

where $\mathbf{x}$ is the system state, $\mathbf{u}$ is the system input, $\mathbf{y}$ is the system output, $f(\mathbf{x})$, $g(\mathbf{x})$ and $h(\mathbf{x})$ are vector fields in $\mathbb{R}^n$. The task of feedback linearization is to identify a static state feedback control law of the following form:

$$\mathbf{u} = \alpha(\mathbf{x}) + \beta(\mathbf{x})\mathbf{v}, \tag{2.11}$$

where $\mathbf{v}$ is an new control input, $\alpha(\mathbf{x})$ and $\beta(\mathbf{x})$ are smooth functions defined in a neighbourhood of some point $\mathbf{x}_0 \in \mathbb{R}^n$ and $\beta(\mathbf{x}_0) \neq 0$, such that the closed-loop system in Figure 2.3, composed of (2.10) and (2.11):

$$\begin{cases} \dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\alpha(\mathbf{x}) + g(\mathbf{x})\beta(\mathbf{x})\mathbf{v} \\ \mathbf{y} = h(\mathbf{x}), \end{cases} \tag{2.12}$$

behaves as a linear completely accessible system:

$$\begin{cases} \dot{\mathbf{z}} = \mathbf{A}\mathbf{z} + \mathbf{B}\mathbf{v} \\ \mathbf{y} = \mathbf{C}\mathbf{z}, \end{cases} \tag{2.13}$$

in which $\mathbf{z}$ represents the new state, $\mathbf{A}$, $\mathbf{B}$ and $\mathbf{C}$ are matrices of suitable dimensions.

Let $\begin{bmatrix} r_1 & \cdots & r_o \end{bmatrix}$ be the vector relative degree[5] of the system (2.10). Differentiating each element of the output vector for their relative degree [7]:

$$\begin{bmatrix} y_1^{(r_1)} & \cdots & y_o^{(r_o)} \end{bmatrix}^T = b(\mathbf{x}) + \Delta(\mathbf{x})\mathbf{u},$$

where[6]

$$\Delta(\mathbf{x}) = \begin{bmatrix} L_{g_1} L_f^{r_1 - 1} h_1(\mathbf{x}) & \cdots & L_{g_o} L_f^{r_1 - 1} h_1(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ L_{g_1} L_f^{r_o - 1} h_o(\mathbf{x}) & \cdots & L_{g_o} L_f^{r_o - 1} h_o(\mathbf{x}) \end{bmatrix} \tag{2.14}$$

and

$$b(\mathbf{x}) = \begin{bmatrix} L_f^{r_1} h_1(\mathbf{x}) & \cdots & L_f^{r_o} h_o(\mathbf{x}) \end{bmatrix}^T.$$

The main result about the input-output decoupling problem is that this problem is solvable if and only if the matrix $\Delta(\mathbf{x})$ is non-singular. In that case, the static state feedback (2.11) with

$$\begin{cases} \alpha(\mathbf{x}) = -\Delta^{-1}(\mathbf{x}) b(\mathbf{x}) \\ \beta(\mathbf{x}) = \Delta^{-1}(\mathbf{x}) \end{cases} \tag{2.15}$$

renders the closed-loop system (2.12) linear and decoupled from an input-output point of view. More precisely,

$$y^{(r_i)} = v_i, i = 1, \ldots, o.$$

[5] The relative degree r_i is the number of times one has to differentiate the i^{th} output in order to have at least one component of the input vector $\mathbf{u}$ explicitly appearing.

[6] $L_f h$ denotes the Lie derivative of the function h with respect to the vector field f (see Appendix B).

2.2.2 Application to quadrotor model

First, it is necessary to define the control objective by choosing a suitable output function for the system (2.9). The number of outputs is set to the number of inputs, due to the under-actuation it does not make any sense to choose more outputs than control inputs. The most interesting outputs are the position of the quadrotor (x, y, z) and the yaw orientation ψ [8], because it represents the heading direction, so the output function is

$$\mathbf{y} = h(\mathbf{x}) = \begin{bmatrix} x & y & z & \psi \end{bmatrix}^T. \tag{2.16}$$

In order to feedback linearizing the system, the necessary and sufficient condition for the solvability of the state space exact linearization problem is [9]:

$$\sum_{i=1}^{o} r_i = n. \tag{2.17}$$

For the nonlinear system (2.9), $n = 12$ and

$$\begin{bmatrix} r_1 & r_2 & r_3 & r_4 \end{bmatrix} = \begin{bmatrix} 2 & 2 & 2 & 2 \end{bmatrix}.$$

Since,

$$r_1 + r_2 + r_3 + r_4 = 8 \neq 12,$$

the condition (2.17) is not satisfied and, therefore, the input-output decoupling problem is not solvable for the system (2.9) by means of a static state feedback control law (2.11). In fact, if we compute $\Delta(\mathbf{x})$ using equation (2.14):

$$\Delta(\mathbf{x}) = \begin{bmatrix} \frac{1}{m}\left(\cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi\right) & 0 & 0 & 0 \\ \frac{1}{m}\left(\cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi\right) & 0 & 0 & 0 \\ \frac{1}{m}\cos\phi\cos\theta & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{I_y}\sin\phi\sec\theta & \frac{1}{I_z}\cos\phi\sec\theta \end{bmatrix},$$

which is singular for any $\mathbf{x}$. In order to linearize (2.9), we need to invert $\Delta(\mathbf{x})$ in (2.15). Indeed, due to the under-actuation of the quadrotor, its system (2.9) can not be transformed into an equivalent linear and controllable one by static state feedback.

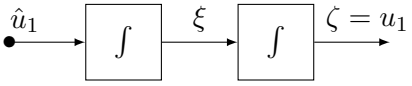


FIGURE 2.4: Block diagram of the double integrator.

The reason is that the second order derivatives of x, y and z are affected only by the control input u_1 and none by u_2, u_3 and u_4. Thus, in order to get $\Delta(\mathbf{x})$ non-singular, the appearance of u_1 should be delayed to higher order derivatives of x, y and z. In order to achieve this result, a dynamic compensator is introduced. A *dynamic compensator* is a feedback structure which incorporates an additional set of state variables and it is modelled by the following equations [3]:

$$\begin{cases} \dot{\boldsymbol{\chi}} = \gamma(\mathbf{x}, \boldsymbol{\chi}) + \delta(\mathbf{x}, \boldsymbol{\chi})\hat{\mathbf{u}} \\ \hat{\mathbf{u}} = \alpha(\mathbf{x}, \boldsymbol{\chi}) + \beta(\mathbf{x}, \boldsymbol{\chi})\mathbf{v}, \end{cases} \tag{2.18}$$

in which $\hat{\mathbf{u}}$ is new reference input, $\boldsymbol{\chi}$ are auxiliary state variables, $\gamma(\mathbf{x}, \boldsymbol{\chi})$ and $\delta(\mathbf{x}, \boldsymbol{\chi})$ are vector fields in $\mathbb{R}^{\hat{n}}$. The purpose of the addition of $\boldsymbol{\chi}$ is to achieve a right relative degrees.

We set u_1 equal to the output of an auxiliary dynamic system driven by a new reference input $\hat{u}_1$ and modelled by the first equation of (2.18). The simplest way in which this result can be achieved is to set this auxiliary dynamic system equal to a double integrator (Figure 2.4). In this case,

$$\boldsymbol{\chi} = \begin{bmatrix} \zeta \\ \xi \end{bmatrix},$$

$$\gamma(\mathbf{x}, \boldsymbol{\chi}) = \begin{bmatrix} \xi \\ 0 \end{bmatrix}$$

and

$$\delta(\mathbf{x}, \boldsymbol{\chi}) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}.$$

So, u_1 is set equal to the output of a double integrator driven by $\hat{u}_1$:

$$\begin{cases} u_1 = \zeta \\ \dot{\zeta} = \xi \\ \dot{\xi} = \hat{u}_1. \end{cases} \tag{2.19}$$

The other input variables are left unchanged:

$$\begin{aligned} u_2 &= \hat{u}_2 \\ u_3 &= \hat{u}_3 \\ u_4 &= \hat{u}_4. \end{aligned} \tag{2.20}$$

Now, u_1 is not any-more an input for the system (2.9), but becomes the internal state ζ for the new dynamical system (2.19). The obtained extended system is

$$\begin{cases} \dot{x} = u \\ \dot{y} = v \\ \dot{z} = w \\ \dot{\phi} = p + \sin\phi \tan\theta q + \cos\phi \tan\theta r \\ \dot{\theta} = \cos\phi q - \sin\phi r \\ \dot{\psi} = \frac{\sin\phi}{\cos\theta} q + \frac{\cos\phi}{\cos\theta} r \\ \dot{u} = -\frac{1}{m} \left(\cos\phi \cos\psi \sin\theta + \sin\phi \sin\psi\right) \zeta \\ \dot{v} = -\frac{1}{m} \left(\cos\phi \sin\psi \sin\theta - \cos\psi \sin\phi\right) \zeta \\ \dot{w} = g - \frac{1}{m} \cos\phi \cos\theta \zeta \\ \dot{p} = \frac{I_y - I_z}{I_x} qr + \frac{1}{I_x} \tau_\phi \\ \dot{q} = \frac{I_z - I_x}{I_y} pr + \frac{1}{I_y} \tau_\theta \\ \dot{r} = \frac{I_x - I_y}{I_z} pq + \frac{1}{I_z} \tau_\psi \\ \dot{\zeta} = \xi \\ \dot{\xi} = \hat{u}_1, \end{cases} \tag{2.21}$$

The system (2.21) can be described in the state space form

$$\dot{\hat{\mathbf{x}}} = \hat{f}(\hat{\mathbf{x}}) + \hat{g}(\hat{\mathbf{x}})\hat{\mathbf{u}},$$

in which

$$\hat{\mathbf{x}} = \begin{bmatrix} \mathbf{x}^T & \boldsymbol{\chi}^T \end{bmatrix}^T = \begin{bmatrix} x & y & z & \phi & \theta & \psi & u & v & w & p & q & r & \zeta & \xi \end{bmatrix}^T, \tag{2.22}$$

$$\hat{f}(\hat{\mathbf{x}}) = \begin{bmatrix} u \\ v \\ w \\ p + \sin\phi\tan\theta q + \cos\phi\tan\theta r \\ \cos\phi q - \sin\phi r \\ \frac{\sin\phi}{\cos\theta} q + \frac{\cos\phi}{\cos\theta} r \\ -\frac{1}{m}\left(\cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi\right)\zeta \\ -\frac{1}{m}\left(\cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi\right)\zeta \\ g - \frac{1}{m}\cos\phi\cos\theta\zeta \\ \frac{I_y - I_z}{I_x} qr \\ \frac{I_z - I_x}{I_y} pr \\ \frac{I_x - I_y}{I_z} pq \\ \xi \\ 0 \end{bmatrix},$$

$$\hat{g}(\hat{\mathbf{x}}) = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_y} & 0 \\ 0 & 0 & 0 & \frac{1}{I_z} \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

and

$$\hat{\mathbf{u}} = \begin{bmatrix} \ddot{T} & \tau_\phi & \tau_\theta & \tau_\psi \end{bmatrix}^T.$$

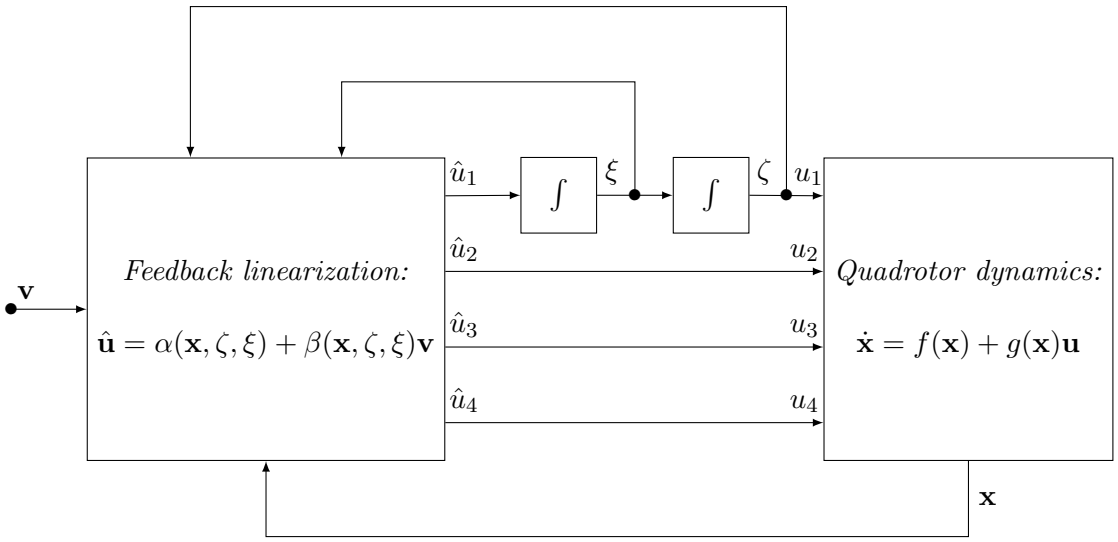


FIGURE 2.5: Block diagram of the control law.

The input-output decoupling problem is solvable for the non-linear system (2.9) by means of a dynamic feedback control law, if it is solvable via static feedback for the extended system (2.21). Since the extended non-linear system (2.21) has dimension $\hat{n} = 14$ and

$$\begin{bmatrix} \hat{r}_1 & \hat{r}_2 & \hat{r}_3 & \hat{r}_4 \end{bmatrix} = \begin{bmatrix} 4 & 4 & 4 & 2 \end{bmatrix},$$

the condition (2.21) is satisfied and, therefore, the input-output decoupling problem is solvable for the system (2.9) by means of a dynamic feedback control law of the form:

$$\hat{\mathbf{u}} = \alpha(\hat{\mathbf{x}}) + \beta(\hat{\mathbf{x}})\mathbf{v}, \tag{2.23}$$

where $\alpha(\hat{\mathbf{x}})$ and $\beta(\hat{\mathbf{x}})$ are computed using equations (2.15). Moreover, we can recompute $\Delta(\hat{\mathbf{x}})$ using equation (2.14):[7]

$$\Delta(\hat{\mathbf{x}}) = \begin{bmatrix} m(s_\phi s_\psi + c_\phi c_\psi s_\theta) & m(c_\psi s_\phi - c_\phi s_\psi s_\theta) & -mc_\phi c_\theta & 0 \\ -\frac{I_x m}{\zeta}(c_\phi s_\psi - c_\psi s_\phi s_\theta) & \frac{I_x m}{\zeta}(c_\phi c_\psi + s_\phi s_\psi s_\theta) & \frac{I_x m}{\zeta} c_\theta s_\phi & 0 \\ -\frac{I_y m}{\zeta} c_\psi c_\theta & -\frac{I_y m}{\zeta} c_\theta s_\psi & \frac{I_y m}{\zeta} s_\theta & 0 \\ \frac{I_z m}{\zeta} c_\psi c_\theta t_\phi & \frac{I_z m}{\zeta} c_\theta t_\phi s_\psi & -\frac{I_z m}{\zeta} t_\phi s_\theta & I_z \frac{c_\theta}{c_\phi} \end{bmatrix}.$$

[7] c_θ, s_θ and t_θ denotes respectively $\cos\theta$, $\sin\theta$ and $\tan\theta$.

The matrix $\Delta(\hat{\mathbf{x}})$ is non-singular at any point characterized by $\zeta \neq 0$, $\phi \neq \pm\frac{\pi}{2}$ and $\theta \neq \pm\frac{\pi}{2}$. Recalling the relations between $\mathbf{u}$ and $\hat{\mathbf{u}}$ in (2.19) and (2.20), the structure for the control law of the original system (2.9) is shown in Figure 2.5.

Finally, the system can be transformed via dynamic feedback into a system which, in suitable coordinates, is fully linear, controllable and observable [5]:

$$\begin{cases} \dddot{x} = v_1 \\ \dddot{y} = v_2 \\ \dddot{z} = v_3 \\ \ddot{\psi} = v_4. \end{cases}$$

The change of coordinates $\mathbf{z} = \Phi(\hat{\mathbf{x}})$ is given by [6]:

$$\begin{cases} z_1 = h_1(\hat{\mathbf{x}}) = x \\ z_2 = L_f h_1(\hat{\mathbf{x}}) = \dot{x} \\ z_3 = L_f^2 h_1(\hat{\mathbf{x}}) = \ddot{x} \\ z_4 = L_f^3 h_1(\hat{\mathbf{x}}) = \dddot{x} \\ z_5 = h_2(\hat{\mathbf{x}}) = y \\ z_6 = L_f h_2(\hat{\mathbf{x}}) = \dot{y} \\ z_7 = L_f^2 h_2(\hat{\mathbf{x}}) = \ddot{y} \\ z_8 = L_f^3 h_2(\hat{\mathbf{x}}) = \dddot{y} \\ z_9 = h_3(\hat{\mathbf{x}}) = z \\ z_{10} = L_f h_3(\hat{\mathbf{x}}) = \dot{z} \\ z_{11} = L_f^2 h_3(\hat{\mathbf{x}}) = \ddot{z} \\ z_{12} = L_f^3 h_3(\hat{\mathbf{x}}) = \dddot{z} \\ z_{13} = h_4(\hat{\mathbf{x}}) = \psi \\ z_{14} = L_f h_4(\hat{\mathbf{x}}) = \dot{\psi}. \end{cases} \tag{2.24}$$

In the new coordinates, the non-linear system (2.21) appears as the linear system (2.13), in which[8]

[8] $\mathbf{0}$ denotes a matrix of zeros of appropriate dimensions.

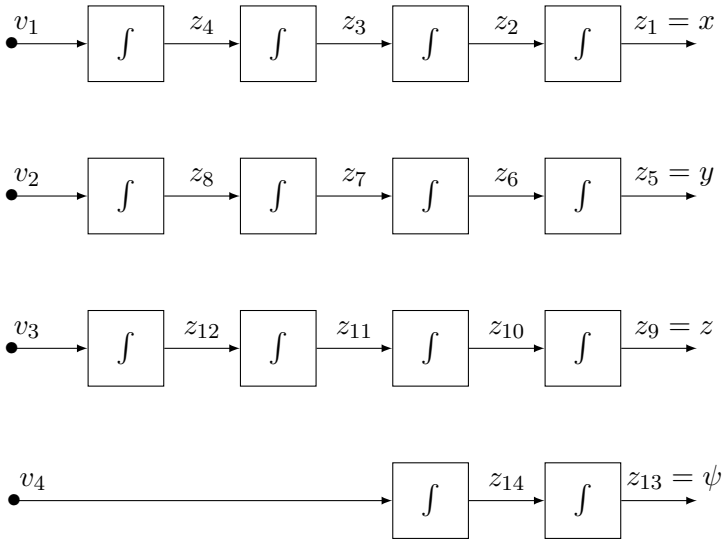


FIGURE 2.6: Block diagram of the closed loop system.

$$\mathbf{z} = \begin{bmatrix} z_1 & \cdots & z_{14} \end{bmatrix}^T,$$
$$\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 & v_4 \end{bmatrix}^T,$$
$$\mathbf{y} = \begin{bmatrix} z_1 & z_5 & z_9 & z_{13} \end{bmatrix}^T = \begin{bmatrix} x & y & z & \phi \end{bmatrix}^T,$$
$$\mathbf{A} = \begin{bmatrix} \mathbf{A_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{A_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{A_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{A_2} \end{bmatrix}, \mathbf{A_1} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \mathbf{A_2} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix},$$
$$\mathbf{B} = \begin{bmatrix} \mathbf{B_1} \\ \mathbf{B_2} \\ \mathbf{B_3} \\ \mathbf{B_4} \end{bmatrix}, \mathbf{B_1} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \mathbf{B_2} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \mathbf{B_3} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$
$$\mathbf{B_4} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$
$$\mathbf{C} = \begin{bmatrix} \mathbf{C_1} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{C_1} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{C_1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{C_2} \end{bmatrix}, \mathbf{C_1} = \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}, \mathbf{C_2} = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

The structure of the system in the new coordinates (2.21) is shown in Figure 2.6.

Chapter 3

Navigation and obstacle avoidance

In this chapter obstacle avoidance problem applied to a quadrotor robot will be considered.

3.1 Driving inputs

3.1.1 First-order feed-forward

On the equivalent linear system (2.24) can be applied all standard control techniques given by linear control system theory. Since we are dealing with four chains of integrators, standard PD control law results the best choice. We want to control the quadrotor position (x, y, z) and the yaw orientation ψ:

$$\mathbf{y}_d = \begin{bmatrix} x_d & y_d & z_d & \psi_d \end{bmatrix}^T$$

and the errors are

$$\mathbf{e} = \mathbf{y_d} - \mathbf{y} = \begin{bmatrix} e_x \\ e_y \\ e_z \\ e_\psi \end{bmatrix} = \begin{bmatrix} x_d - x \\ y_d - y \\ z_d - z \\ \psi_d - \psi \end{bmatrix}.$$

At this point we can define a vector of four desired inputs $\mathbf{v}_d = \begin{bmatrix} v_{d,1} & v_{d,2} & v_{d,3} & v_{d,4} \end{bmatrix}^T$ [10]:

$$\begin{aligned} v_{d,1} &= \dddot{x}_d + \sum_{i=0}^{3} k_{x,i} e_x^{(i)} \\ v_{d,2} &= \dddot{y}_d + \sum_{i=0}^{3} k_{y,i} e_y^{(i)} \\ v_{d,3} &= \dddot{z}_d + \sum_{i=0}^{3} k_{z,i} e_z^{(i)} \\ v_{d,4} &= \ddot{\psi}_d + \sum_{i=0}^{1} k_{\psi,i} e_\psi^{(i)}, \end{aligned} \tag{3.1}$$

where $k_{x,i}$, $k_{y,i}$ and $k_{z,i}$, $i = 0, \ldots, 3$, and $k_{\psi,j}$, $j = 0, 1$, are the gains properly chosen in such a way that the corresponding eigenvalues are placed in the open left half of the complex plane. For this purpose, the coefficients of a *Hurwitz polynomial*[1] are chosen [11].

3.1.2 Towards feasibility of inputs

In order to compute four control inputs in (3.1), we have to know the desired snaps ($\ddddot{x}_d$, $\ddddot{y}_d$, $\ddddot{z}_d$), jerks ($\dddot{x}_d$, $\dddot{y}_d$, $\dddot{z}_d$), accelerations ($\ddot{x}_d$, $\ddot{y}_d$, $\ddot{z}_d$, $\ddot{\psi}_d$) and velocities ($\dot{x}_d$, $\dot{y}_d$, $\dot{z}_d$, $\dot{\psi}_d$) at any point of time. Since we want to control the quadrotor in velocity, all the quantities have to be reconstructed from the desired velocities. If we simply do a numerical derivative of the desired velocity, the result is very noisy. So we need to proceed in an alternative way.

Let's consider a generic septic polynomial[2] function which can represent the velocity profile on x-axis:

$$\dot{x}(\tau) = c_7\tau^7 + c_6\tau^6 + c_7\tau^7 + c_6\tau^6 + c_5\tau^5 + c_4\tau^4 + c_3\tau^3 + c_2\tau^2 + c_1\tau + c_0, \tag{3.2}$$

in which $\tau \in [0, 1]$ is normalized time and c_i, $i = 0, \ldots, 7$, are coefficients to be adequately chosen. If we do the derivative of (3.2), we obtain acceleration, jerk and snap:

[1] Hurwitz polynomial is a polynomial whose coefficients are positive real numbers and whose roots' real part is zero or negative.
[2] Septic polynomial is a polynomial that has 7 as the highest exponent of its terms.

$$\begin{aligned}
\ddot{x}(\tau) &= 7c_7\tau^6 + 6c_6\tau^5 + 5c_5\tau^4 + 4c_4\tau^3 + 3c_3\tau^2 + 2c_2\tau + c_1 \\
\dddot{x}(\tau) &= 42c_7\tau^5 + 30c_6\tau^4 + 20c_5\tau^3 + 12c_4\tau^2 + 6c_3\tau + 2c_2 \\
\ddddot{x}(\tau) &= 210c_7\tau^4 + 120c_6\tau^3 + 60c_5\tau^2 + 24c_4\tau + 6c_3.
\end{aligned} \tag{3.3}$$

Now, we set the initial conditions:

$$\begin{cases}
\dot{x}(0) = c_0 = \dot{x}_0 \\
\ddot{x}(0) = c_1 = \ddot{x}_0 \\
\dddot{x}(0) = 2c_2 = \dddot{x}_0 \\
\ddddot{x}(0) = 6c_3 = \ddddot{x}_0,
\end{cases} \tag{3.4}$$

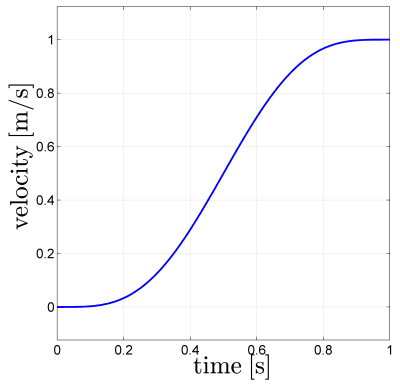
and the final conditions:

$$\begin{cases}
\dot{x}(1) = c_7 + c_6 + c_5 + c_4 + c_3 + c_2 + c_1 + c_0 = \dot{x}_f \\
\ddot{x}(1) = 7c_7 + 6c_6 + 5c_5 + 4c_4 + 3c_3 + 2c_2 + c_1 = 0 \\
\dddot{x}(1) = 42c_7 + 30c_6 + 20c_5 + 12c_4 + 6c_3 + 2c_2 = 0 \\
\ddddot{x}(1) = 210c_7 + 120c_6 + 60c_5 + 24c_4 + 6c_3 = 0.
\end{cases} \tag{3.5}$$

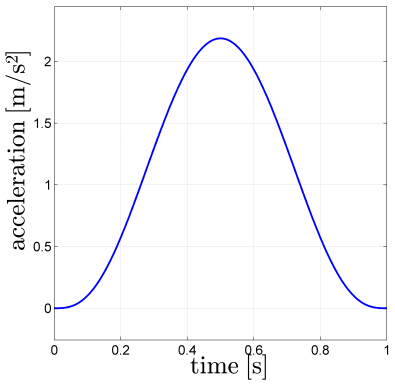
At this point we have eight unknowns c_i, $i = 0, \ldots, 7$, and eight conditions (3.4) and (3.5), so we can build a system of linear homogeneous and non-homogeneous equations:

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & 2 & 0 & 0 \\
0 & 0 & 0 & 0 & 6 & 0 & 0 & 0 \\
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
7 & 6 & 5 & 4 & 3 & 2 & 1 & 0 \\
42 & 30 & 20 & 12 & 6 & 2 & 0 & 0 \\
210 & 120 & 60 & 24 & 6 & 0 & 0 & 0
\end{bmatrix}
\begin{bmatrix} c_7 \\ c_6 \\ c_5 \\ c_4 \\ c_3 \\ c_2 \\ c_1 \\ c_0 \end{bmatrix}
=
\begin{bmatrix} \dot{x}_0 \\ \ddot{x}_0 \\ \dddot{x}_0 \\ \ddddot{x}_0 \\ \dot{x}_f \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{3.6}$$

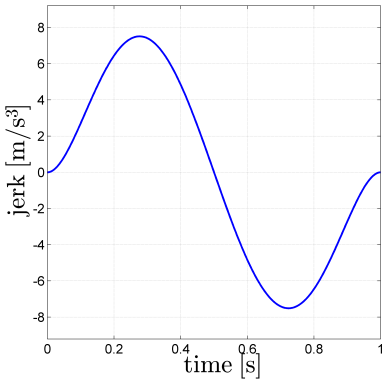
and solve it:



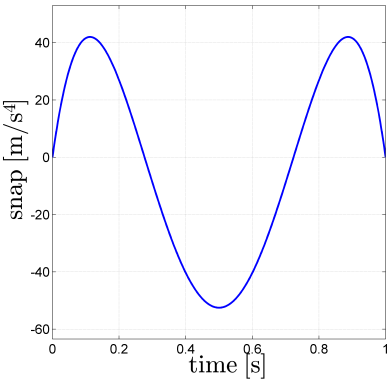
(a) Velocity transition from $\dot{x}_0$ to $\dot{x}_f$.



(b) Acceleration transition.



(c) Jerk transition.



(d) Snap transition.

FIGURE 3.1: An examples of velocity, acceleration, jerk and snap transitions needed to achieve desired velocity $\dot{x}_f$ from initial velocity $\dot{x}_0$.

$$\begin{cases} c_0 = \dot{x}_0 \\ c_1 = \ddot{x}_0 \\ c_2 = \frac{1}{2}\dddot{x}_0 \\ c_3 = \frac{1}{6}\ddddot{x}_0 \\ c_4 = -35\dot{x}_0 - 20\ddot{x}_0 - 5\dddot{x}_0 - \frac{2}{3}\ddddot{x}_0 + 35\dot{x}_f \\ c_5 = 84\dot{x}_0 + 45\ddot{x}_0 + 10\dddot{x}_0 + \ddddot{x}_0 - 84\dot{x}_f \\ c_6 = -70\dot{x}_0 - 36\ddot{x}_0 - \frac{15}{2}\dddot{x}_0 - \frac{2}{3}\ddddot{x}_0 + 70\dot{x}_f \\ c_7 = 20\dot{x}_0 + 10\ddot{x}_0 + 2\dddot{x}_0 + \frac{1}{6}\ddddot{x}_0 - 20\dot{x}_f. \end{cases} \tag{3.7}$$

At the end, by replacing (3.7) in (3.2) and in (3.3), we obtain generic profile functions for the desired velocity, acceleration, jerk and snap, which depend on initial conditions ($\dot{x}_0$, $\ddot{x}_0$, $\dddot{x}_0$, $\ddddot{x}_0$) and desired velocity $\dot{x}_f$. In a similar way, the transition functions for y and z axes can be computed. This method can be also extend for computing the desired velocity ($\dot{\psi}_d$) and accelerations ($\ddot{\psi}_d$) for the yaw orientation.

An example of so obtained transition functions is depicted in Figure 3.1. At the initial time $\tau_0 = 0\text{s}$, the quadrotor is stationary ($\dot{x}_0 = 0\text{m/s}$, $\ddot{x}_0 = 0\text{m/s}^2$, $\dddot{x}_0 = 0\text{m/s}^3$, $\ddddot{x}_0 = 0\text{m/s}^4$) and, at the final time $\tau_f = 1\text{s}$, it reaches the desired velocity ($\dot{x}_f = 1\text{m/s}$).

3.2 Obstacle avoidance

Let consider an Euclidean *workspace* $\mathcal{W} = \mathbb{R}^3$, where the *robot* $\mathcal{B} \subset \mathbb{R}^3$ is free to translate. Let $\mathcal{O}_i \subset \mathcal{W}$, $i = 1, \ldots, p$, be the *obstacles*. Let assume that the geometry and the position of each $\mathcal{O}_i$ is known. The *obstacle avoidance* is the problem of moving $\mathcal{B}$ in $\mathcal{W}$ while avoiding collisions with the obstacles $\mathcal{O}_i$. The obstacle avoidance problem is important for mobile robots.

If the quadrotor can be described by a sphere in $\mathcal{W}$, its configuration can be defined by the Cartesian coordinates of a representative point. Note that the orientation of the sphere is irrelevant for collision checking. An effective scheme for obstacle avoidance is obtained by representing the quadrotor as a point in the *configuration space* $\mathcal{C} = \mathcal{W}$, where the images of the obstacles are also reported. For this reason, it is natural to choose as the generalized coordinates of the quadrotor:

$$\mathbf{q} = \begin{bmatrix} q_1 & q_2 & q_3 \end{bmatrix}^T = \begin{bmatrix} x & y & z \end{bmatrix}^T, \tag{3.8}$$

whose values identify the *configuration* of the robot. Thus, to each configuration $\mathbf{q}$ is associated a point in $\mathcal{C}$ and $\mathcal{C}$ is the set of all configurations that the quadrotor can assume.

In order to find a solution to the obstacle avoidance problem, it is necessary to convert the obstacles from $\mathcal{W}$ in $\mathcal{C}$. Given an obstacle $\mathcal{O}_i$ in $\mathcal{W}$, its image in $\mathcal{C}$ is called $\mathcal{C}$-*obstacle* and is defined as in [12]:

$$\mathcal{CO}_i = \{\mathbf{q} \in \mathcal{C} : \mathcal{B}(\mathbf{q}) \cap \mathcal{O}_i \neq \emptyset\}. \tag{3.9}$$

Thus, $\mathcal{CO}_i$ contains the configurations that cause a collision between the quadrotor $\mathcal{B}$ and the obstacle $\mathcal{O}_i$ in the workspace. A growing procedure is applied to obstacles in $\mathcal{W}$ in order to obtain their image in $\mathcal{C}$. In particular, the boundary of $\mathcal{CO}_i$ is the set of configurations that put the quadrotor in contact with the obstacle $\mathcal{O}_i$. In our case, to build $\mathcal{CO}_i$ it is sufficient to grow $\mathcal{O}_i$ by the radius of the sphere which describes the quadrotor. The union of all $\mathcal{C}$-obstacles

$$\mathcal{CO} = \bigcup_{i=1}^{p} \mathcal{CO}_i \tag{3.10}$$

defines the $\mathcal{C}$-*obstacle region*, while its complement

$$\mathcal{C}_{free} = \mathcal{C} - \mathcal{CO} = \left\{ \mathbf{q} \in \mathcal{C} : \mathcal{B}(\mathbf{q}) \cap \left(\bigcup_{i=1}^{p} \mathcal{O}_i \right) \neq \emptyset \right\} \tag{3.11}$$

is the *free configuration space*, $\mathcal{C}_{free}$ is the subset of $\mathcal{C}$ that do not cause collisions with the obstacles.

3.2.1 Artificial potential fields

The *artificial potential fields* method provides a simple and effective technique for on-line obstacle avoidance applications. This approach uses repulsive potential fields around the obstacles to force the quadrotor move away from them. Essentially in our case, the point that represents the quadrotor in $\mathcal{C}$ moves only under the influence of a *repulsive potential field* U_r from the $\mathcal{C}$-obstacle region. At each robot configuration $\mathbf{q} \in \mathbb{R}^n$, the robot experiences a generalized force equal to the negative gradient of the potential $-\nabla U_r(\mathbf{q})$, which indicates the most promising direction for local motion. In obstacle avoidance, potentials are expressed in the configuration space $\mathcal{C}$ of the quadrotor. Each obstacle $\mathcal{O}_i$ to be avoided is surrounded by repulsive potential functions. These potentials are added to form a composite potential and the quadrotor moves in this field of forces. It is simple to provide a linear control law with constant gain, like the one in (3.1).

The repulsive potential U_r is used to prevent the quadrotor from colliding obstacles as it moves under the influence of the user command. In particular, the idea is to build a barrier potential in the proximity of the $\mathcal{C}$-obstacle region, so as to push away the point that represents the quadrotor in $\mathcal{C}$.

For each component $\mathcal{CO}_i$ define an associated repulsive potential [13]:

$$U_{r,i}(\mathbf{q}) = \begin{cases} \frac{k_r}{\gamma} \left(\frac{1}{\eta_i(\mathbf{q})} - \frac{1}{\eta_0} \right)^{\gamma}, & \text{if } \eta_i(\mathbf{q}) \leq \eta_0 \\ 0, & \text{if } \eta_i(\mathbf{q}) > \eta_0 \end{cases}, \tag{3.12}$$

where $k_r > 0$ is the repulsive gain, γ is the potential slope, $\eta_i(\mathbf{q}) = \|\mathbf{q} - \mathbf{q}_i\|$ is the distance of $\mathbf{q}$ from $\mathcal{CO}_i$ and η_0 is the *range of influence* of the obstacles. An example of a repulsive potential field in the two-dimensional configuration space $\mathcal{C}$ with a circular

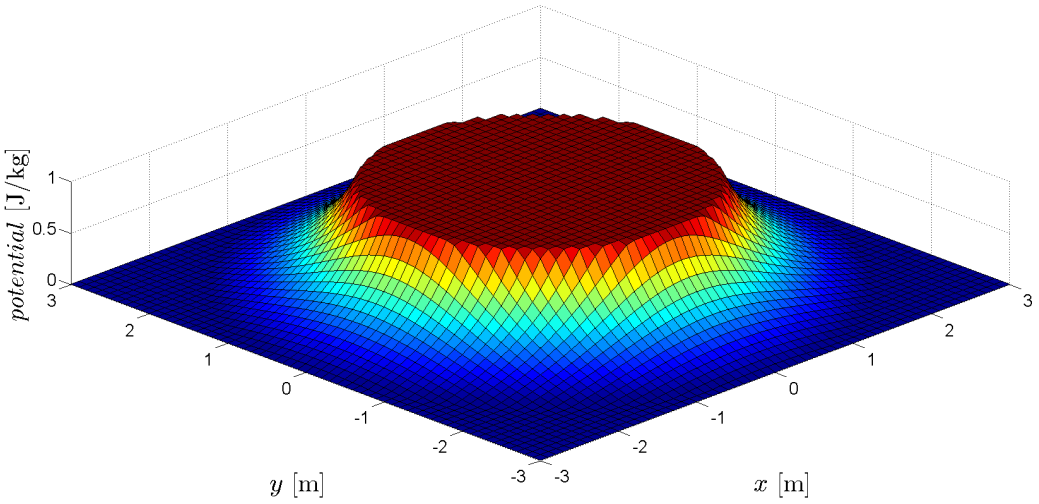


FIGURE 3.2: A repulsive potential field in the two-dimensional configuration space with a circular obstacle.

obstacle is shown in Figure 3.2 (the potential is limited up to 1J/kg). The potential $U_{r,i}$ is zero outside and positive inside the range of influence η_0 and tends to infinity at the boundary of $\mathcal{CO}_i$.

The repulsive force resulting from deriving (3.12) is

$$F_{r,i}(\mathbf{q}) = -\nabla U_{r,i}(\mathbf{q}) = \begin{cases} \frac{k_r}{\eta_i^2(\mathbf{q})} \left(\frac{1}{\eta_i(\mathbf{q})} - \frac{1}{\eta_0} \right)^{\gamma-1} \nabla \eta_i(\mathbf{q}), & \text{if } \eta_i(\mathbf{q}) \leq \eta_0 \\ 0, & \text{if } \eta_i(\mathbf{q}) > \eta_0 \end{cases}. \tag{3.13}$$

The total repulsive force is obtained by adding the individual forces (3.13) associated with the components of $\mathcal{CO}_i$ and normalizing over the number of obstacles:

$$F_r(\mathbf{q}) = \frac{\sum_{i=1}^{p} F_{r,i}(\mathbf{q})}{p}. \tag{3.14}$$

The force field $F_r(\mathbf{q})$ (3.14) is interpreted as the repulsive velocity $\dot{\mathbf{q}}_r$ for the quadrotor [12], by letting

$$\dot{\mathbf{q}}_r = F_r(\mathbf{q}). \tag{3.15}$$

This strategy is fast in executing the motion corrections suggested by the force field F_r and may be considered safer. Using (3.15) guarantees also that in absence of obstacles

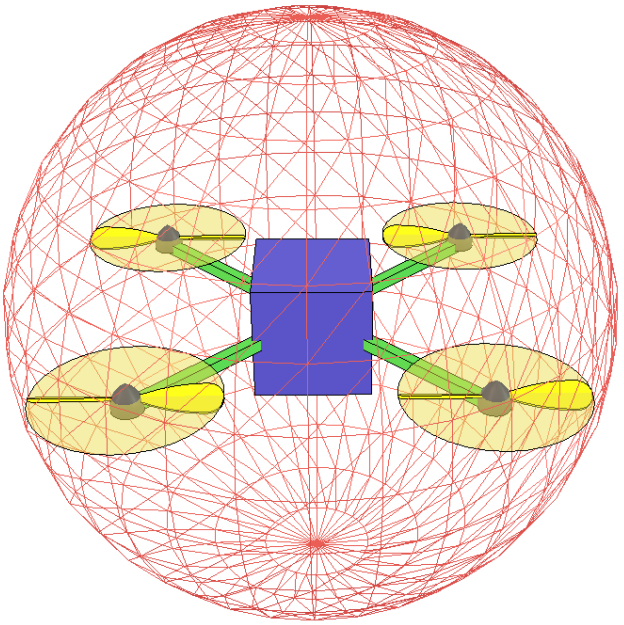


FIGURE 3.3: Bounding sphere surrounding schematic quadrotor structure.

the quadrotor has no velocity. In order to calculate the repulsive acceleration $\ddot{\mathbf{q}}_r$, jerk $\dddot{\mathbf{q}}_r$ and snap $\ddddot{\mathbf{q}}_r$, (3.15) is derived over time [14]:

$$\begin{aligned}
\ddot{\mathbf{q}}_r &= \frac{\partial \dot{\mathbf{q}}_r}{\partial t} = \frac{\partial \dot{\mathbf{q}}_r}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial t} = \nabla \dot{\mathbf{q}}_r \dot{\mathbf{q}} \\
\dddot{\mathbf{q}}_r &= \frac{\partial \ddot{\mathbf{q}}_r}{\partial t} = \frac{\partial \ddot{\mathbf{q}}_r}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial t} = \nabla \ddot{\mathbf{q}}_r \dot{\mathbf{q}} \\
\ddddot{\mathbf{q}}_r &= \frac{\partial \dddot{\mathbf{q}}_r}{\partial t} = \frac{\partial \dddot{\mathbf{q}}_r}{\partial \mathbf{q}} \frac{\partial \mathbf{q}}{\partial t} = \nabla \dddot{\mathbf{q}}_r \dot{\mathbf{q}}.
\end{aligned}$$

3.2.2 Application to quadrotor

The $\mathcal{C}$-obstacle region is build in order to ensure the obstacle avoidance having regard to the overall dimensions of the aircraft, which may be considered contained in a bounding sphere (as in Figure 3.3) of radius

$$R = l + \frac{l_p}{2},$$

in which l_p is the length of the quadrotor propeller. The quadrotor can be considered point-like, if all obstacles are extended using the same bounding sphere of the quadrotor.

As a consequence of this choice, the minimum distance of quadrotor from an obstacle in the configuration space $\mathcal{C}$ depends only on its distance from the position $\begin{bmatrix} x & y & z \end{bmatrix}$ and the following relationship is valid:

$$\eta_i^{'}(\mathbf{q}) = \begin{cases} \eta_i(\mathbf{q}) - R, & \text{if } \eta_i(\mathbf{q}) \geq R \\ 0, & \text{if } \eta_i(\mathbf{q}) < R \end{cases},$$

Another consequence of this choice is that the obstacles do not affect the yaw. So the repulsive force on the yaw is zero under all conditions.

Now, we have to choose how to apply the resulting force $F_r(\mathbf{q})$ generated through artificial potential field to the quadrotor feedback linearizzation. For this purpose we can define a vector of four repulsive inputs $\mathbf{v}_r = \begin{bmatrix} v_{r,1} & v_{r,2} & v_{r,3} & v_{r,4} \end{bmatrix}^T$:

$$\begin{aligned} v_{r,1} &= \dddot{q}_{r,1} + \sum_{i=1}^{3} k_{x,i} q_{r,1}^{(i)} \\ v_{r,2} &= \dddot{q}_{r,2} + \sum_{i=1}^{3} k_{y,i} q_{r,2}^{(i)} \\ v_{r,3} &= \dddot{q}_{r,3} + \sum_{i=1}^{3} k_{z,i} q_{r,3}^{(i)} \\ v_{r,4} &= 0, \end{aligned}$$

where $k_{x,i}$, $k_{y,i}$ and $k_{z,i}$, $i = 1, \ldots, 3$, are the same as in (3.1). The vector of forces then enters in the control as desired snap for the position and as desired angular acceleration of the yaw. At the end, the input law, which commands the linearized system (2.24), is

$$\mathbf{v} = \mathbf{v}_d + \mathbf{v}_r.$$

Chapter 4

Simulation results

Extensive simulations were made considering different parametric uncertainties. Some of the obtained results are presented in the following to illustrate the performance of the proposed controller.

4.1 Numerical simulations

Now, we are going to show some simulations carried out in ®MATLAB about the controller described so far. ®MATLAB is the high-level programming language and interactive environment used by engineers and scientists worldwide. It allows to explore and visualize processes across various disciplines, including signal and image processing, communications and control systems.

The quadrotor intrinsic parameters are reported in Table 4.1. These parameters are chosen close to the ones of real quadrotors. The controller gains for (3.1) are chosen as follows:

Parameter	**Value**	**Unit**
b	10^{-5}	$[\mathrm{N}\cdot\mathrm{s}^2]$
d	10^{-7}	$[\mathrm{N}\cdot\mathrm{m}\cdot\mathrm{s}^2]$
g	9.81	$[\mathrm{m/s}^2]$
I_x	1	$[\mathrm{kg}\cdot\mathrm{m}^2]$
I_y	1	$[\mathrm{kg}\cdot\mathrm{m}^2]$
I_z	1	$[\mathrm{kg}\cdot\mathrm{m}^2]$
l	0.2	[m]
m	1	[kg]

TABLE 4.1: The quadrotor intrinsic parameters.

$$
\begin{aligned}
k_{x,0} &= k_{y,0} = k_{z,0} = 625 \\
k_{x,1} &= k_{y,1} = k_{z,1} = 500 \\
k_{x,2} &= k_{y,2} = k_{z,2} = 150 \\
k_{x,3} &= k_{y,3} = k_{z,3} = 20 \\
k_{\psi,0} &= k_{\psi,1} = 4.
\end{aligned}
$$

The quadrotor is initially in hovering and the initial pose is $x(0) = 0$, $y(0) = 0$, $z(0) = 0$ and $\psi(0) = 0$. So, the quadrotor initial state (2.22) is

$$
\hat{\mathbf{x}}_0 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & m \cdot g & 0 \end{bmatrix}^T.
$$

4.1.1 DFL tests

In this subsection we test the DFL controller (2.23) alone. In the first simple case, the quadrotor follows the circular trajectory. In the second more complex situation, the quadrotor follows the helicoidal trajectory while pointing towards the direction of the movement.

Circular trajectory

In this simple case, the reference trajectory is a circle with unitary radius centred in $(0, 0, 1)$m on xy-plane whose equations are given by

$$
\mathbf{y}_{c,d}(t) = \begin{bmatrix} \cos(t) & \sin(t) & 1 & 0 \end{bmatrix}^T. \tag{4.1}
$$

In order to use the controller in (3.1) we need also the desired velocity, acceleration, jerk and snap, which can be computed deriving (4.3):

$$
\begin{aligned}
\dot{\mathbf{y}}_{c,d}(t) &= \begin{bmatrix} -\sin(t) & \cos(t) & 0 & 0 \end{bmatrix}^T \\
\ddot{\mathbf{y}}_{c,d}(t) &= \begin{bmatrix} -\cos(t) & -\sin(t) & 0 & 0 \end{bmatrix}^T \\
\dddot{\mathbf{y}}_{c,d}(t) &= \begin{bmatrix} \sin(t) & -\cos(t) & 0 & 0 \end{bmatrix}^T \\
\ddddot{\mathbf{y}}_{c,d}(t) &= \begin{bmatrix} \cos(t) & \sin(t) & 0 & 0 \end{bmatrix}^T.
\end{aligned}
$$

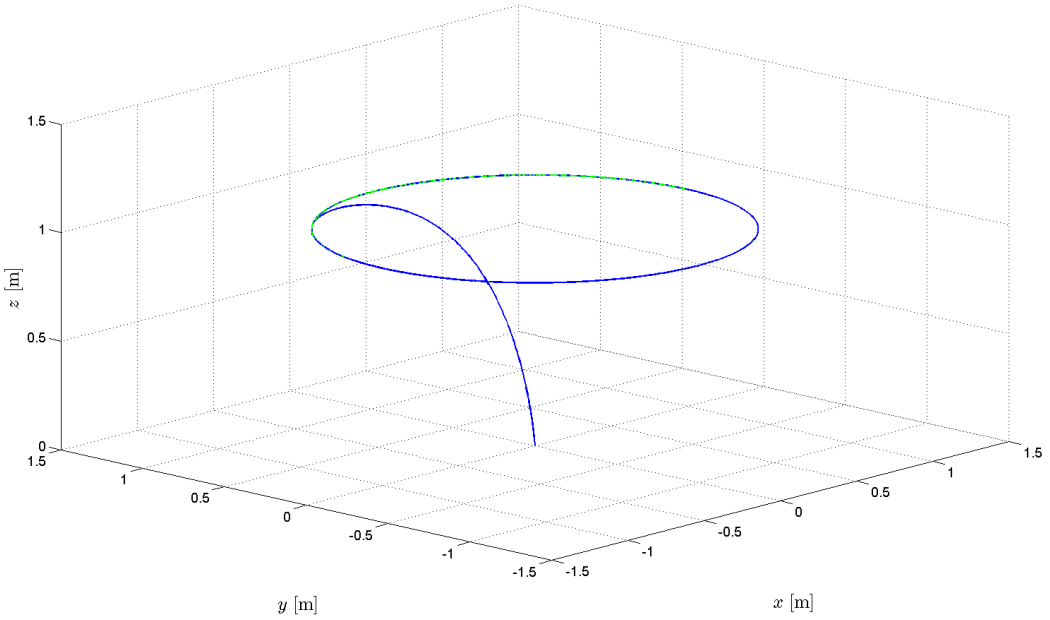


FIGURE 4.1: 3D plot of circle tracking on xy-plane with DFL.

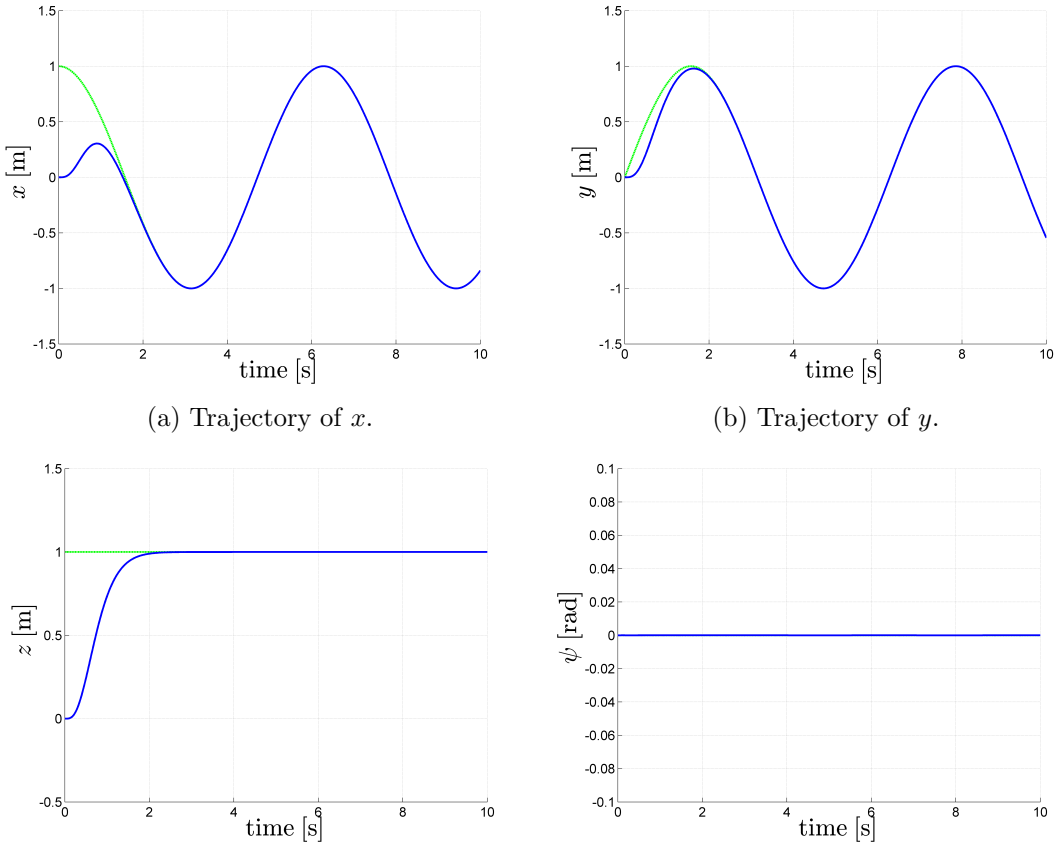
Figure 4.1 shows in 3D space the realized trajectory (in blue) with the desired one (in green). We can observe that the two trajectories overlap perfectly. In Figure 4.2 we can see how x, y, z and ψ converge to the desired values. The trajectories are projected over four controllable outputs: x (Figure 4.2a), y (Figure 4.2b), z (Figure 4.2c) and ψ (Figure 4.2d).

Helicoidal trajectory with variable yaw

In this case, the reference trajectory is a vertical helix centred in $x_h = 0$ and $y_h = 0$. While moving the quadrotor points always towards the direction of the movement. This trajectory is given by

$$\mathbf{y}_{h,d}(t) = \begin{bmatrix} \sin(2t) & \cos(2t) & 1 + \frac{1}{10}t & -2t \end{bmatrix}^T. \tag{4.2}$$

In order to use the controller in (3.1) we need also the desired velocity, acceleration, jerk and snap, which can be computed deriving (4.2):



(a) Trajectory of x.

(b) Trajectory of y.

(c) Trajectory of z.

(d) Trajectory of ψ.

FIGURE 4.2: Tracking of a circle on xy-plane with DFL. Plots of desired (dashed green line) and realized (solid blue line) trajectories.

$$\dot{\mathbf{y}}_{h,d}(t) = \begin{bmatrix} 2\cos(2t) & -2\sin(2t) & \frac{1}{10} & -2 \end{bmatrix}^T$$
$$\ddot{\mathbf{y}}_{h,d}(t) = \begin{bmatrix} -4\sin(2t) & -4\cos(2t) & 0 & 0 \end{bmatrix}^T$$
$$\dddot{\mathbf{y}}_{h,d}(t) = \begin{bmatrix} -8\cos(2t) & 8\sin(2t) & 0 & 0 \end{bmatrix}^T$$
$$\ddddot{\mathbf{y}}_{h,d}(t) = \begin{bmatrix} 16\sin(2t) & 16\cos(2t) & 0 & 0 \end{bmatrix}^T.$$

Figure 4.3 shows in 3D space the realized trajectory (in blue) with the desired one (in green). As we can observe, after the initial transitional part, the quadrotor follows perfectly the desired trajectory. In Figure 4.4 we can see how x, y, z and ψ converge to the desired values. The trajectories are projected over four controllable outputs: x (Figure 4.4a), y (Figure 4.4b), z (Figure 4.4c) and ψ (Figure 4.4d). The yaw orientation ψ is defined between 0 and 2π (cyan dotted line in Figure 4.4d). By the definition of

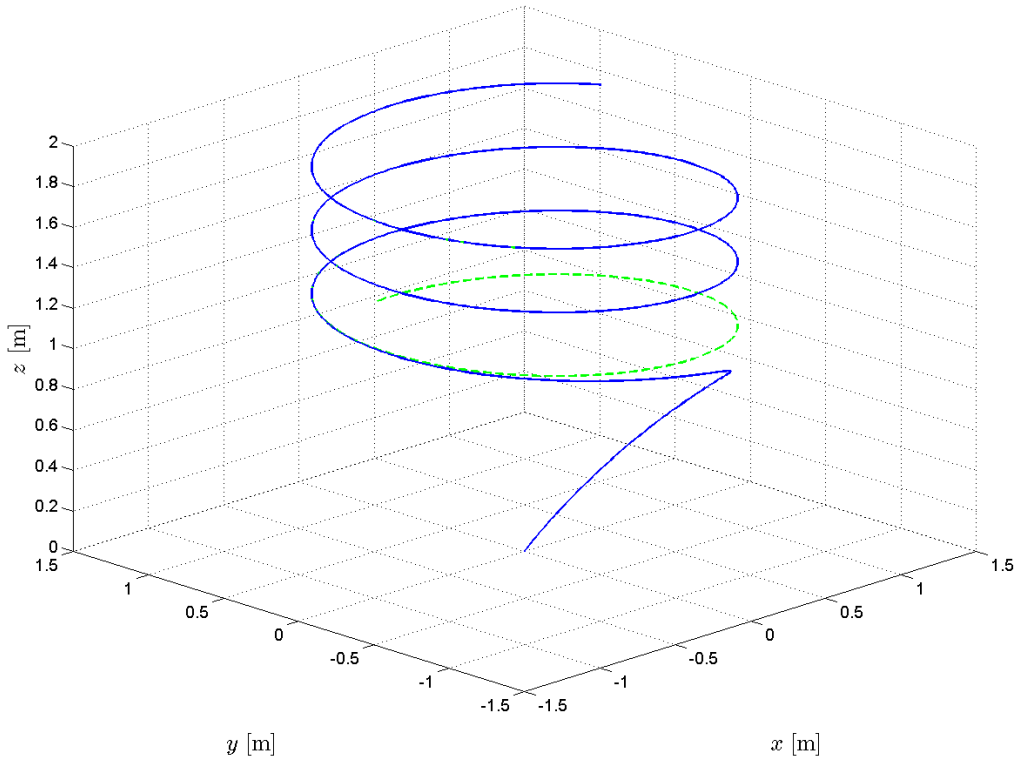


FIGURE 4.3: 3D plot of vertical helix tracking with DFL.

manifold[1], 0 is a neighbourhood of 2π. So after a complete rotation the orientation goes back to 0 or to 2π.

4.1.2 Obstacle avoidance tests

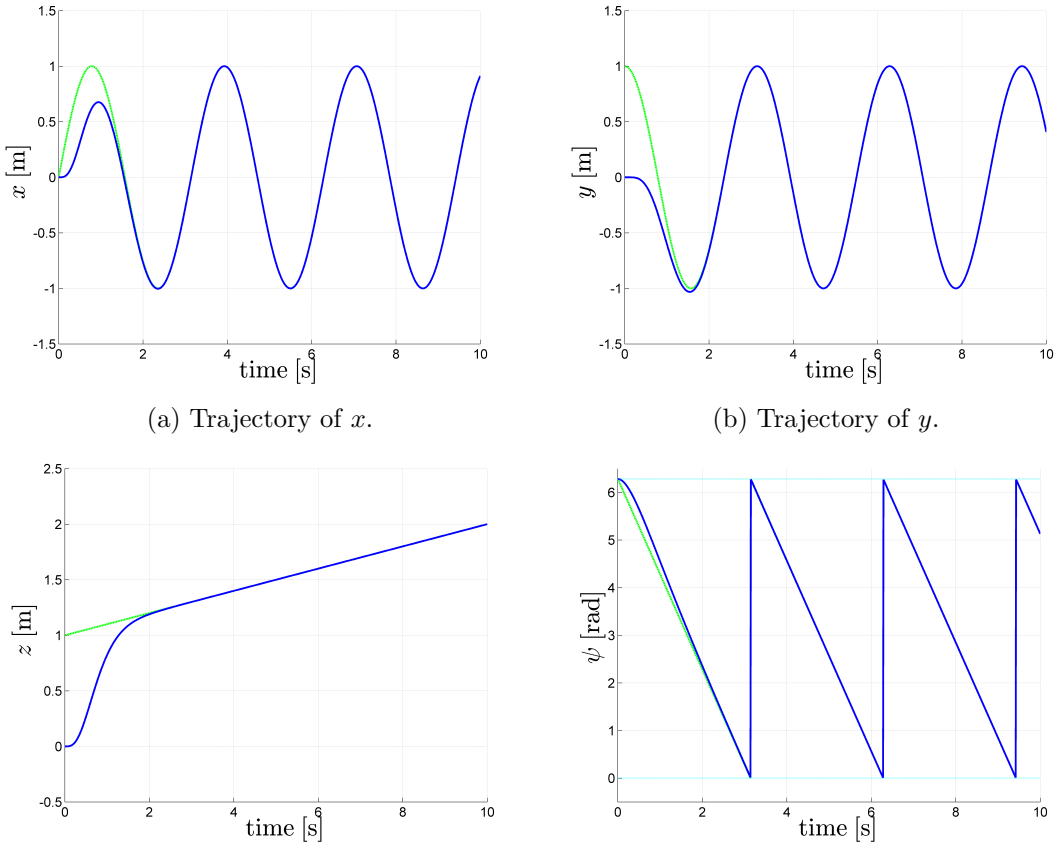
In this subsection we test the behaviour of the quadrotor under the influence of artificial potential fields (3.13). First, we show the case with only one obstacle. After while, the scenario with multiple obstacles and a driven command will be presented.

Single obstacle

In this simple case, the quadrotor workspace $\mathcal{W} \subset \mathbb{R}^2$ and contains only one circular obstacle with radius of 1m and situated in $(1, 1)$m. The quadrotor starts in $(0, 0)$m and has no external commands from the operator.

The artificial potential field related to the obstacle is the one depicted in Figure 3.2. The reaction of the quadrotor to the obstacle (red area) is shown in Figure 4.5. In the

[1] A manifold is a topological space in which each n-dimensional point has a neighbourhood that is *homeomorphic* to the n-dimensional Euclidean space.



(a) Trajectory of x.

(b) Trajectory of y.

(c) Trajectory of z.

(d) Trajectory of ψ.

FIGURE 4.4: Tracking of a vertical helix with DFL. Plots of desired (dashed green line) and actual (solid blue line) trajectories.

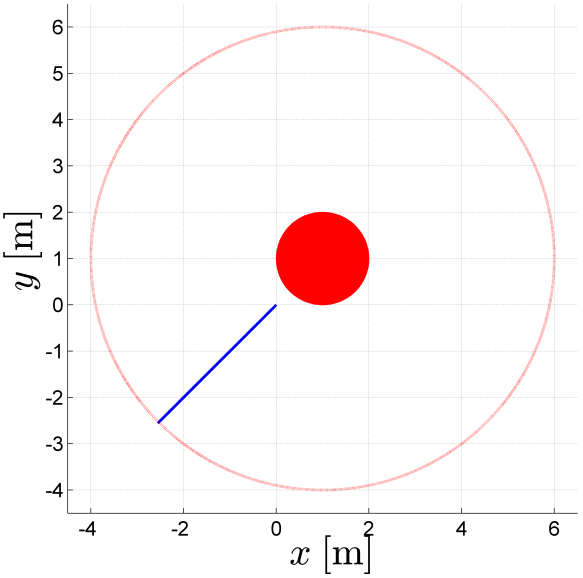


FIGURE 4.5: Reaction of the quadrotor to the obstacle.

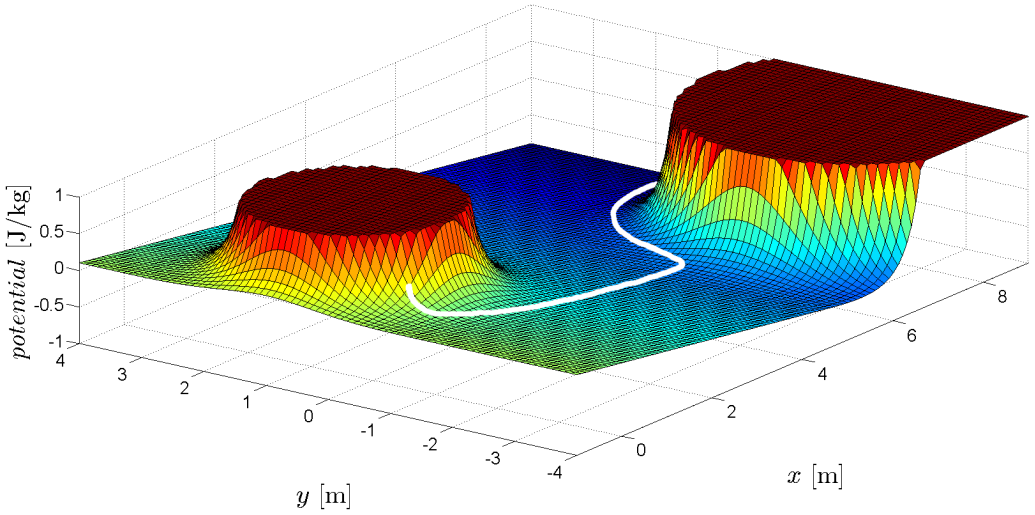


FIGURE 4.6: Artificial potential field related to the obstacles and to the command.

absence of the commands the quadrotor simply moves away from the obstacle, since it goes out of obstacle's range of influence (red dotted curve). The blue line represents the path of the quadrotor connecting initial and final positions.

Multiple obstacles and command

As in the previous case, the quadrotor workspace $\mathcal{W} \subset \mathbb{R}^2$, but contains two circular obstacles: the first one has radius of 1m and situated in $(1, 1)$m, the second one has radius of 2m and situated in $(8, -2)$m. The quadrotor starts in $(0, 0)$m and has the constant positive commands from the operator along x-axis of 1m/s.

The artificial potential field related to the obstacles and to the command is depicted in Figure 4.6. The quadrotor can be imagined as a ball which moves under the action of the gravitational force. The white curve connecting initial and final positions represents the movement of the quadrotor.

The reaction of the quadrotor to the obstacle (red area) is shown in Figure 4.7. In the absence of the commands the quadrotor simply moves away from the obstacle, since it goes out of obstacle's range of influence (red dotted curve). The blue line represents the path of the quadrotor connecting initial and final positions.

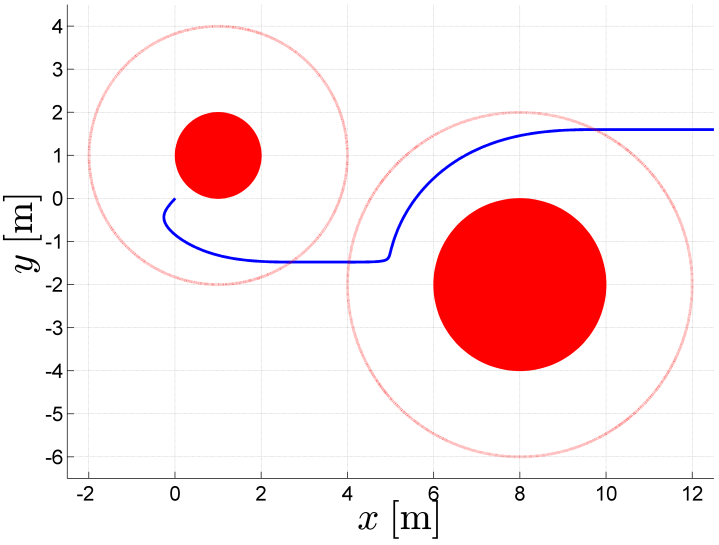


FIGURE 4.7: Reaction of the quadrotor to the obstacle.

4.1.3 DFL with obstacle avoidance tests

Now, we can perform a complete simulation with DFL controller and obstacle avoidance. The desired trajectory is a unitary circle with fixed height centred in $(0, 0, 1)$m on xy-plane. In addition, the quadrotor has to be oriented towards the center of the circle. So, the equations of the desired trajectory are given by:

$$\mathbf{y}_{c,d}(t) = \begin{bmatrix} \cos(2t) & \sin(2t) & 1 & 2t + \pi \end{bmatrix}^T. \tag{4.3}$$

In order to use the controller in (3.1) we need also the desired velocity, acceleration, jerk and snap, which can be computed deriving (4.3):

$$\begin{aligned}
\dot{\mathbf{y}}_{c,d}(t) &= \begin{bmatrix} -2\sin(2t) & 2\cos(2t) & 0 & 2 \end{bmatrix}^T \\
\ddot{\mathbf{y}}_{c,d}(t) &= \begin{bmatrix} -4\cos(2t) & -4\sin(2t) & 0 & 0 \end{bmatrix}^T \\
\dddot{\mathbf{y}}_{c,d}(t) &= \begin{bmatrix} 8\sin(2t) & -8\cos(2t) & 0 & 0 \end{bmatrix}^T \\
\ddddot{\mathbf{y}}_{c,d}(t) &= \begin{bmatrix} 16\cos(2t) & 16\sin(2t) & 0 & 0 \end{bmatrix}^T.
\end{aligned}$$

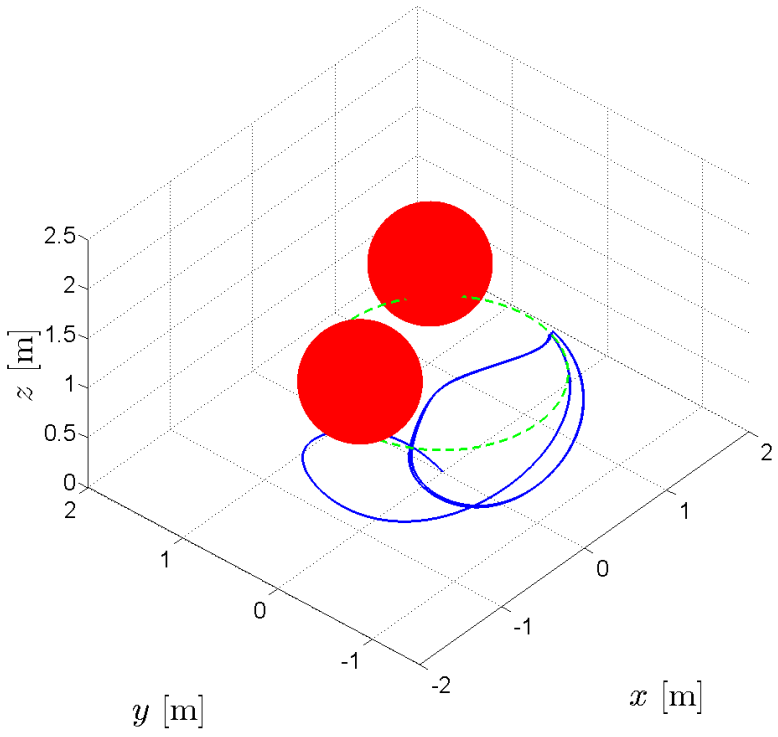


FIGURE 4.8: 3D plot of circle tracking in presence of an obstacle.

Furthermore, in the quadrotor's workspace $\mathcal{W} \subset \mathbb{R}^3$ are present two spherical obstacles with radius of 0.5m and situated in $(-1, 0, 1.5)$m and in $(1, 1, 1)$m. So, the quadrotor's path intersects the obstacles.

Figure 4.8 shows in 3D space the realized trajectory (in blue) with the desired one (in green). We can observe that the two trajectories overlap perfectly due to the presence of the obstacles. The top view of the motion is reported in Figure 4.9. In Figure 4.10 we can see how x, y, z and ψ converge to the desired values. The trajectories are projected over four controllable outputs: x (Figure 4.10a), y (Figure 4.10b), z (Figure 4.10c) and ψ (Figure 4.10d).

4.2 Dynamical simulations

After the development of the numerical tests, the quadrotor was implemented in *Robot Operating System* (ROS) and in GAZEBO. On the one hand, ROS is a flexible framework for writing robot software. It is a collection of tools, libraries, and conventions that aim to simplify the task of creating complex and robust robot behaviour across a wide variety of robotic platforms. On the other hand, Gazebo offers the ability to accurately and efficiently simulate populations of robots in complex indoor and outdoor environments.

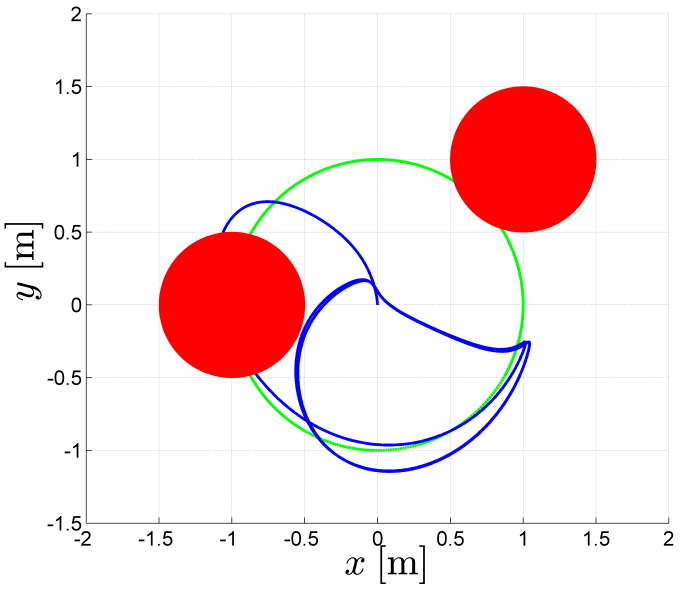
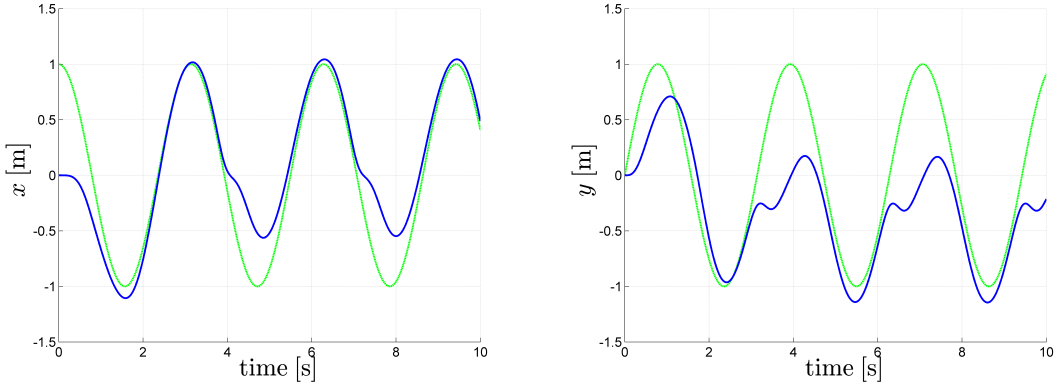
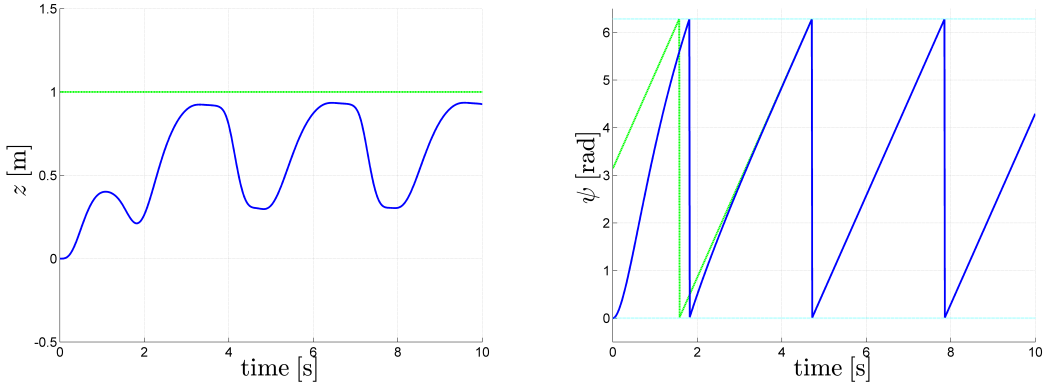


FIGURE 4.9: Top view of circle tracking in presence of an obstacle.



(a) Trajectory of x.

(b) Trajectory of y.



(c) Trajectory of z.

(d) Trajectory of ψ.

FIGURE 4.10: Tracking of a circle in presence of an obstacle. Plots of desired (dashed green line) and realized (solid blue line) trajectories.

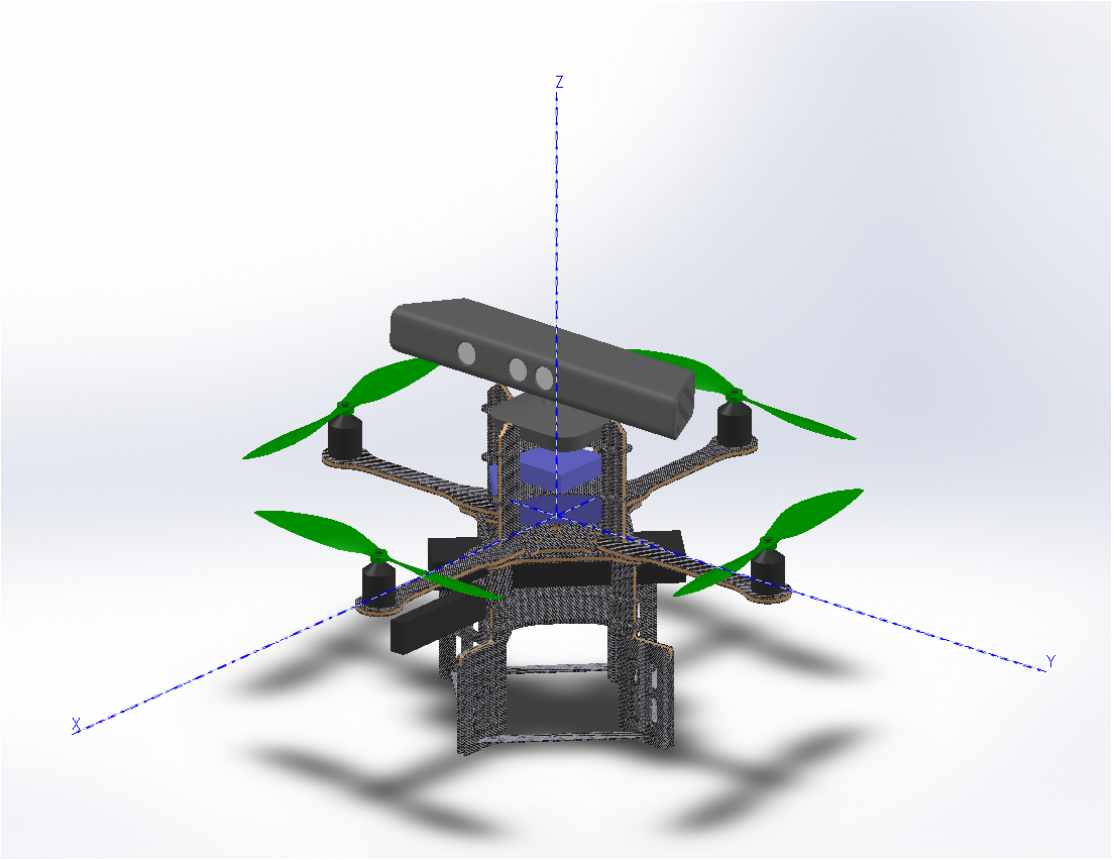


FIGURE 4.11: CAD model of Pelican quadrotor with Kinect depth sensor.

It has a robust physics engine, high-quality graphics, and convenient programmatic and graphical interfaces.

In order to perform dynamical simulations the 3D CAD model of Pelican quadrotor with Kinect depth sensor was built (Figure 4.11). For the dynamical simulation we need to recompute the dynamic parameters of the quadrotor. The quadrotor's body can be decomposed in a set of cuboids and cylinders. The inertia matrix (2.6) for a cuboid of width w_{cub}, height h_{cub}, depth d_{cub} and mass m_{cub} is computed in Appendix C.1 and is given by

$$\mathbf{I}_{cub} = \frac{m_{cub}}{12} \begin{bmatrix} h_{cub}^2 + d_{cub}^2 & 0 & 0 \\ 0 & w_{cub}^2 + d_{cub}^2 & 0 \\ 0 & 0 & w_{cub}^2 + h_{cub}^2 \end{bmatrix}.$$

The inertia matrix (2.6) for a cylinder of radius r_{cyl}, height h_{cyl} and mass m_{cyl} is computed in Appendix C.2 and is given by

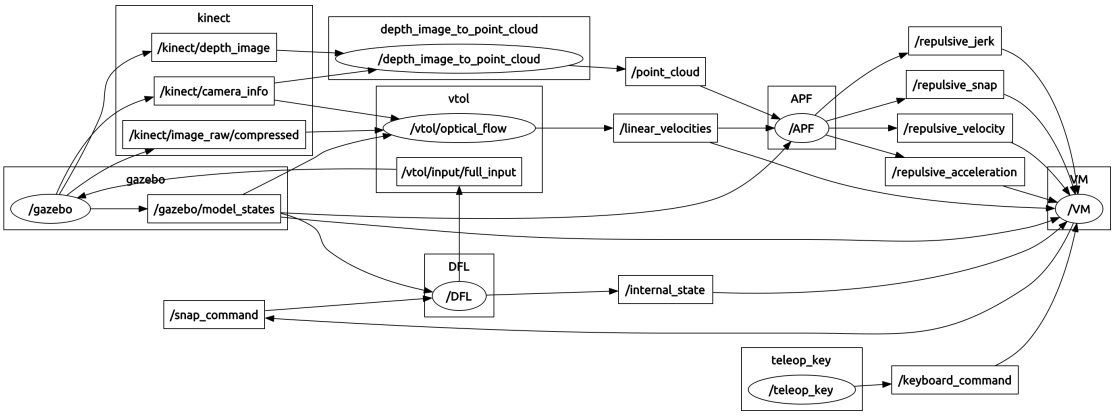


FIGURE 4.12: Block diagram of ROS nodes and ROS topics.

$$\mathbf{I}_{cyl} = \frac{m_{cyl}}{12} \begin{bmatrix} 3r_{cyl}^2 + h_{cyl}^2 & 0 & 0 \\ 0 & 3r_{cyl}^2 + h_{cyl}^2 & 0 \\ 0 & 0 & 6r_{cyl}^2 \end{bmatrix}.$$

In Figure 4.12 is shown the block diagram of ROS nodes and ROS topics created with rqt_graph, which provides a visualization of ROS computation graph. The block labelled GAZEBO simulates the environment, the quadrotor and laser sensor. While the environment and the quadrotor are displayed in the graphical simulator, the data collected by the sensor are published on corresponding topic. The block Feedback Controller, implemented in C++, calculates the motors angular velocities required to perform the command and applies them to the quadrotor in the simulated environment via GAZEBO's API.

In dynamic simulations, as in real situations, we do not use x and y position of the quadrotor. The height we measure with the simulated sonar. The velocities along x-axis and y-axis we get from optical flow, which uses the images from simulated vertical camera. The velocity along z-axis is obtained from the derivative of the height. GAZEBO does not allow to access to the object's acceleration, so this value is computed analytically from the dynamic model (2.9) of quadrotor, as well as its jerk. We can get attitude and angular velocities directly from GAZEBO and we add to them white Gaussian noise. However, in the simulations made in GAZEBO we found that the integration engine creates a large numerical noise. This behaviour is well visible analysing the trend of the angular velocity. In order to obtain the point cloud of the environment, we use a simulated 3D sensor.

4.2.1 Circle tracking

In this subsection we test the DFL controller (2.23). The task of the quadrotor is to follow the circular trajectory. In addition, the quadrotor has to be oriented towards the center of the circle. So, the equations of the desired trajectory are given by:

$$\mathbf{y}_{c,d}(t) = \begin{bmatrix} \cos(t) & \sin(t) & 1 & 2t + \pi \end{bmatrix}^T. \tag{4.4}$$

In order to use the controller in (3.1) we need also the desired velocity, acceleration, jerk and snap, which can be computed deriving (4.4):

$$\begin{aligned}
\dot{\mathbf{y}}_{c,d}(t) &= \begin{bmatrix} -\sin(t) & \cos(t) & 0 & 2 \end{bmatrix}^T \\
\ddot{\mathbf{y}}_{c,d}(t) &= \begin{bmatrix} -\cos(t) & -\sin(t) & 0 & 0 \end{bmatrix}^T \\
\dddot{\mathbf{y}}_{c,d}(t) &= \begin{bmatrix} \sin(t) & -\cos(t) & 0 & 0 \end{bmatrix}^T \\
\ddddot{\mathbf{y}}_{c,d}(t) &= \begin{bmatrix} \cos(t) & \sin(t) & 0 & 0 \end{bmatrix}^T.
\end{aligned}$$

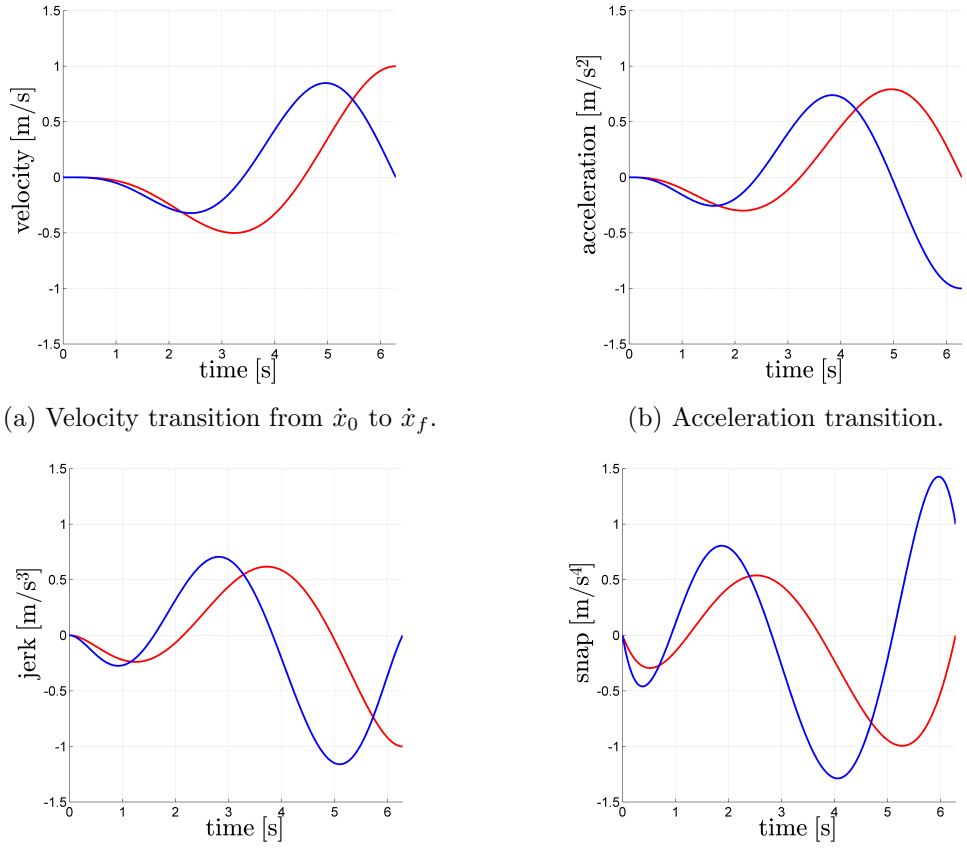
But since we do not use the quadrotor position x and y, in order to correct its trajectory, we have to take into account the initial conditions at $t = 0$ for x-axis and y-axis:

$$\begin{cases} x_{c,d}(0) = 1 \\ \dot{x}_{c,d}(0) = 0 \\ \ddot{x}_{c,d}(0) = -1 \\ \dddot{x}_{c,d}(0) = 0 \\ \ddddot{x}_{c,d}(0) = 1 \end{cases} \quad \text{and} \quad \begin{cases} y_{c,d}(0) = 0 \\ \dot{y}_{c,d}(0) = 1 \\ \ddot{y}_{c,d}(0) = 0 \\ \dddot{y}_{c,d}(0) = -1 \\ \ddddot{y}_{c,d}(0) = 0 \end{cases}.$$

Therefore, we need to reach these conditions before starting to follow the trajectory. To this purpose we use two nonic polynomials[2] as transition functions for x and y:

$$x_{c,d}(\tau) = \sum_{i=0}^{9} c_{x,i}\tau^i \qquad \text{and} \qquad y_{c,d}(\tau) = \sum_{i=0}^{9} c_{y,i}\tau^i,$$

[2] Nonic polynomial is a polynomial that has 9 as the highest exponent of its terms.



(a) Velocity transition from $\dot{x}_0$ to $\dot{x}_f$.

(b) Acceleration transition.

(c) Jerk transition.

(d) Snap transition.

FIGURE 4.13: An examples of profiles of transitory velocities, accelerations, jerks and snaps needed for circle tracking. The red curves are functions on x-axis and the blue curves are functions on y-axis.

in which $c_{x,i}$ and $c_{y,i}$, $i = 0, \ldots, 9$, are coefficients to be adequately chosen and $\tau \in [0, 2\pi]$ is normalized time. We proceed as described in Subsection 3.1.2, in order to find $c_{x,i}$ and $c_{y,i}$, $i = 0, \ldots, 9$. The resulting profiles of transitory functions are depicted in Figure 4.13: velocities (Figure 4.13a), accelerations (Figure 4.13b), jerks (Figure 4.13c) and snaps (Figure 4.13d). Note that at time $\tau = 2\pi$ we have the discontinuities of the second order.

Figure 4.14 shows the realized trajectory on xy-plane (in blue-red) with the desired one (in green). In Figure 4.15 we can see the behaviour of the single components: x (Figure 4.15a), y (Figure 4.15b), z (Figure 4.15c) and ψ (Figure 4.15d). We can note that x and y diverge, it is caused by the absence of reference of x and y positions. On the other side, z remains on its initial and desired value and ψ converges to the desired orientation.

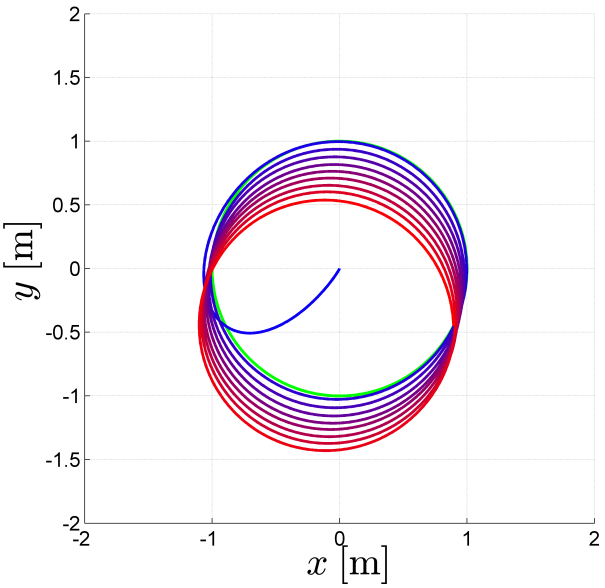
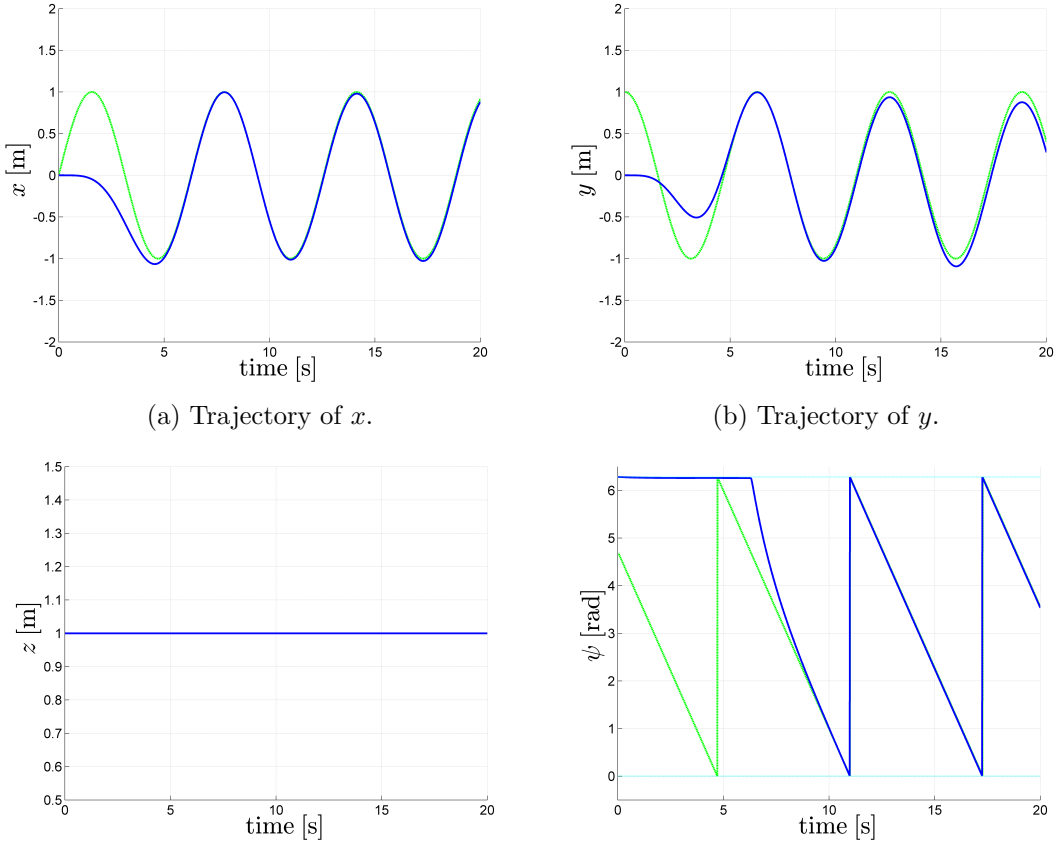


FIGURE 4.14: 3D plot of circle tracking on xy-plane with DFL.



(a) Trajectory of x.

(b) Trajectory of y.

(c) Trajectory of z.

(d) Trajectory of ψ.

FIGURE 4.15: Tracking of a circle on xy-plane with DFL. Plots of desired (dashed green line) and realized (solid blue line) trajectories.

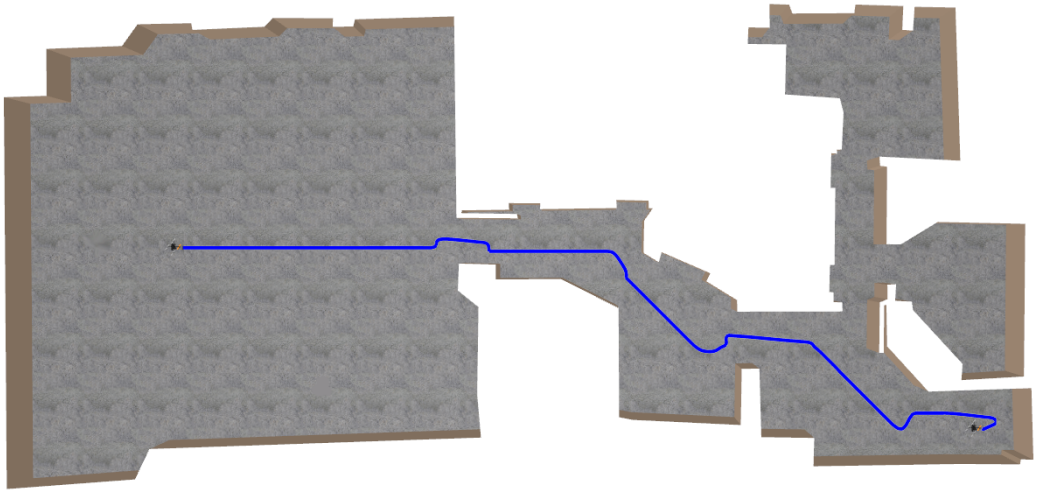


FIGURE 4.16: Top view of a small labyrinth and quadrotor's path (blue curve).

4.2.2 Travelling in labyrinth

Now, we want to make the quadrotor travel in a small labyrinth depicted in Figure 4.16 without hitting walls. By imposing a command from the operator the quadrotor moves in the desired direction avoiding the obstacles. The quadrotors always points toward the direction of the desired motion. This is obtained by setting:

$$\psi_d = \text{atan2}(\dot{y}_d, \dot{x}_d),$$

where atan2 is defined in $(-\pi, \pi]$ and can be expressed as follows:

$$\text{atan2}(y,x) = \begin{cases} \arctan\frac{y}{x}, & \text{if } x > 0 \\ \arctan\frac{y}{x} + \pi, & \text{if } y \geq 0 \text{ and } x < 0 \\ \arctan\frac{y}{x} - \pi, & \text{if } y < 0 \text{ and } x < 0 \\ \frac{\pi}{2}, & \text{if } y > 0 \text{ and } x = 0 \\ -\frac{\pi}{2}, & \text{if } y < 0 \text{ and } x = 0 \\ 0, & \text{if } y = 0 \text{ and } x = 0 \end{cases}.$$

In this simulation, the robot starts in the big left room (as shown in the figure) and successfully reach the right part of the labyrinth.

Chapter 5

Experimental results

In this chapter will be introduced the entire set-up developed in order to perform experiments.

5.1 Hardware system

Entire set-up comprises flying system which represents the quadrotor with its sensors and remote ground station used for controling the quadrotor and for setting the on-board parameters.

FIGURE 5.1: AscTec Pelican quadrotor.

Parameter	**Value**
Size	651×651×188 mm
Engines	4 electrical, brushless motors
Rotor diameter	254 mm
Empty weight	620 g
Maximum payload	650 g
Flight time (with payload)	16 min
Maximum range	1000 m
Maximum airspeed	16 m/s
Maximum climb rate	8 m/s
Maximum thrust	36 N
Wireless communication	XBee: 2,4 GHz, WiFi
Inertial guidance system	AscTec AutoPilot with 1,000 Hz update rate
On-board computer	3rd Generation Intel®Core™i7 processor

TABLE 5.1: Technical data of AscTec Pelican quadrotor.

5.1.1 Quadrotor

Aircraft used to develop the experimental set-up is the Asctec Pelican quadrotor depicted in Figure 5.1. The AscTec Pelican is the packhorse amongst AscTec's Research Line UAVs. This quadrotor offers plenty of space and various interfaces for individual components and payloads. The two-time International Aerial Robotics Competition champion is the key to unlimited experiments. High quality standards in production guarantee reliability and safety of this aerial robot. The technical data of AscTec Pelican quadrotor are reported in Table 5.1.

Asctec Pelican essential characteristics:

- The inertial guidance system provides highest precision through advanced sensor components and two ARM7[1] microprocessors.
- The control unit provides highest flexibility. Latest interfaces simplify the implementation of C-code algorithms.
- The *Low Level Processor* (LLP) ensures a highly stable flight behaviour of the flight system. The LLP is the data controller which processes all sensor data and performs the data fusion of all relevant information with an update rate of 1 kHz.
- The *High Level Processor* (HLP) takes control over the flight system according to the C-code algorithms.

[1] ARM7 is a group of older 32-bit ARM processor cores.

FIGURE 5.2: ASUS Xtion PRO LIVE.

- Safe testing thanks to the Safety Switch function. While testing control commands and manoeuvres, one can simply switch back into safe mode and the AscTec AutoPilot takes back the control.
- The energy-efficient engines work with only 100 Watt per motor.
- The AscTec Pelican is proven and tested as a useful research tool in challenging conditions.
- The possibility of receiving quadrotor status via a serial port makes this platform an excellent base to start developing autonomous behaviours.

5.1.2 RGB-D sensor

The ASUS Xtion PRO LIVE (in Figure 5.2) is the world's first and exclusive professional PC motion sensing development solution. Its multiple sensing functions makes development easier. The Xtion uses infra-red sensors, adaptive depth detection technology, color image sensing and audio stream to capture a real-time image, movement and voice. The Xtion development solution comes with a set of developer tools to make it easier for developers to create their own depth-based applications without the need to write complex programming algorithms. In addition, with Xtion PRO series, developers are offered more options and tools to develop their own applications. The technical data of ASUS Xtion PRO LIVE camera are reported in Table 5.2.

ASUS Xtion PRO LIVE essential characteristics:

- The Xtion PRO LIVE development solution allows developers to track a movement, which makes it ideal for controlling.
- Xtion PRO LIVE enables color (RGB) image sensing. With RGB, Xtion PRO LIVE can capture the image, which is useful for many applications.

Parameter	**Value**
Size	180×35×50 mm
Power consumption	< 2,5 W
Distance of use	0,8 m − 3,5 m
Field of view	horizontal: 58°, vertical: 45°, diagonal: 70°
Sensors	RGB camera, depth camera, 2×microphone
RGB image size	SXGA (1280×1024): 30 fps
Depth image size	VGA (640×480): 30 fps, QVGA (320×240): 60 fps
Platforms	Intel X86, AMD
Interface	USB 2.0, USB 3.0
Programming languages	C++, C#, Java
Operation environment	indoor

TABLE 5.2: Technical data of AscTec Pelican quadrotor.

(a) An example of RGB image. (b) An example of depth map.

FIGURE 5.3: An example of raw data from ASUS Xtion.

- The Xtion PRO LIVE development solution allows developers to apply the latest depth-sensing technology in various applications. The Xtion PRO LIVE development kit is widely open. One can create his own applications more convenient and intuitive.
- The Xtion PRO LIVE has an easy plug and play USB design.
- The Xtion PRO LIVE is OPENNI compatible.

ASUS Xtion provides simultaneously 1280 × 1024 RGB video stream at 30 Hz rate (Figure 5.3a) and a 640 × 480 pixel monochrome intensity coded depth map[2] at 30 Hz (Figure 5.3b). Low cost, reliability and speed of the measurement promises to make Xtion the primary 3D measuring devices in indoor robotics, 3D scene reconstruction

[2] Depth map is an image that contains information relating to the distance of scene objects from a viewpoint.

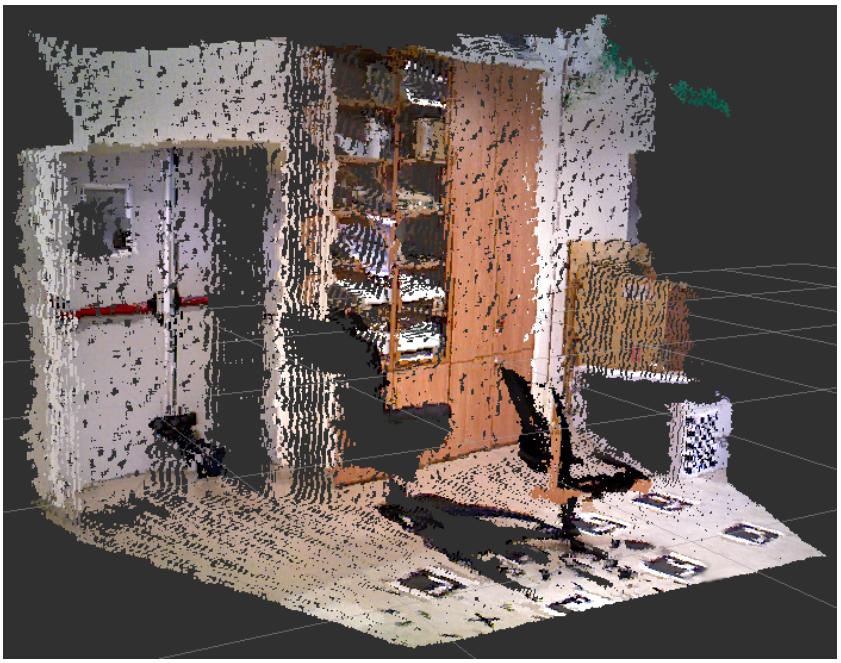


FIGURE 5.4: The point cloud reconstructed from the RGB image in Figure 5.3a and the depth map in Figure 5.3b.

and object recognition. This device connects using USB 2.0 or USB 3.0 interface[3]. The protocol to access the Xtion data is open[4] and a software to read them already exists.

Main raw output of ASUS Xtion is an image that corresponds to the depth in the scene. Rather than providing the actual depth Z, Xtion returns inverse depth. Computation of depth maps can be grouped into passive or active methods. Passive depth sensing tries to infer depth from multiple cameras or images, for example, through stereo correspondence algorithms or optical flow. Active methods usually employ additional physical sensors such as lasers, structured lighting, or infra-red illumination cast on the scene. Xtion uses a form of structured light[5]. Depth sensor consists of an infra-red laser projector combined with a monochrome CMOS sensor. Spatial resolution (X and Y) of the depth sensor at 2 m from the it is 3 mm, while the depth resolution (Z) at the same distance is 10 mm.

In order to obtain a point cloud from the depth map, we use the algorithm described in Appendix D. Figure 5.4 shows the point cloud reconstructed from the RGB image in Figure 5.3a and the depth map in Figure 5.3b. In our application we build the colorless point cloud, thus we do not need the informations from the RGB camera.

[3] Additional current is not required.
[4] OpenNI SDK is bundled.
[5] Developed and patented by PrimeSense.

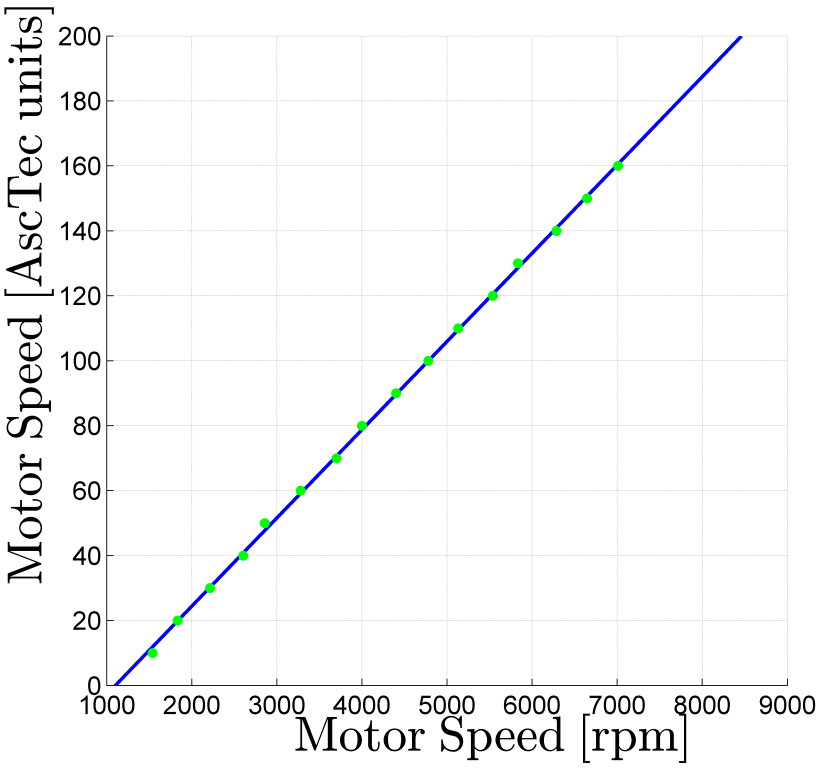


FIGURE 5.5: Correspondences between AscTec motor velocity and real angular velocity and the corresponding regression line.

5.2 Parameters estimation

5.2.1 Speed conversion

AscTec Pelican quadrotor low level controller treats the motors velocity as an positive integer between 0 and 200 (0 is the minimum speed and 200 is the maximum speed). To estimate the conversion factors from the real motor angular velocity ω_{RPM} expressed in rotation per minute to the AscTec motor velocity ω_{AT}, we directly set the motor turning speed in AscTec units and measure the resulting turning speed with a laser tachometer. After a batch of sixteen measurements was collected $\{\langle\omega_{RPM_1}, \omega_{AT_1}\rangle, \ldots, \langle\omega_{RPM_{16}}, \omega_{AT_{16}}\rangle\}$, we noticed that the transformation is linear (shown in Figure 5.5):

$$\omega_{AT} = k_1\omega_{RPM} + k_0,$$

so the linear regression was applied[6]:

[6] A^+ indicates the pseudo-inverse of matrix A.

$$\begin{bmatrix} \tilde{k}_0 \\ \tilde{k}_1 \end{bmatrix} = \begin{bmatrix} 1 & \omega_{RPM_1} \\ \vdots & \vdots \\ 1 & \omega_{RPM_{16}} \end{bmatrix}^{+} \begin{bmatrix} \omega_{AT_1} \\ \vdots \\ \omega_{AT_{16}} \end{bmatrix}.$$

We obtained $k_0 = -29,92$ and $k_1 = 0,027$ and are the same for four motors. Afterwards these values were confirmed by AscTec.

5.2.2 Thrust coefficient

Once we are able to convert the motor angular velocity from AscTec units to rotation per minute, we can proceed with the estimation of the propellers thrust coefficient b. To estimate b we replace the first equation from 2.2 into the seventh, eighth and ninth equations from 2.9 and we obtain

$$\begin{aligned} a_x &= -\frac{1}{m}\left(\cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi\right)\left(b\omega_1^2 + b\omega_2^2 + b\omega_i^2 + b\omega_i^2\right) \\ a_y &= -\frac{1}{m}\left(\cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi\right)\left(b\omega_1^2 + b\omega_2^2 + b\omega_i^2 + b\omega_i^2\right) \\ a_z &= -\frac{1}{m}\cos\phi\cos\theta\left(b\omega_1^2 + b\omega_2^2 + b\omega_i^2 + b\omega_i^2\right) + g. \end{aligned} \tag{5.1}$$

Extracting b from 5.1, we can compute it:

$$b = -m \begin{bmatrix} \left(\cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi\right)\left(\omega_1^2 + \omega_2^2 + \omega_i^2 + \omega_i^2\right) \\ \left(\cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi\right)\left(\omega_1^2 + \omega_2^2 + \omega_i^2 + \omega_i^2\right) \\ \cos\phi\cos\theta\left(\omega_1^2 + \omega_2^2 + \omega_i^2 + \omega_i^2\right) \end{bmatrix}^{+} \begin{bmatrix} a_x \\ a_y \\ a_z - g \end{bmatrix}.$$

We collected a batch of h measurements[7] $\tilde{\omega}_{1i}$, $\tilde{\omega}_{2i}$, $\tilde{\omega}_{3i}$, $\tilde{\omega}_{4i}$, $\bar{\phi}_i$, $\bar{\theta}_i$, $\bar{\psi}_i$, $\bar{a}_{x_i}$, $\bar{a}_{y_i}$ and $\bar{a}_{z_i}$ from the UAV's IMU and Autopilot. Finally, stacking up $3h$ equations we can estimate $\tilde{b}$:

[7] $\tilde{a}$ indicates the estimated value of a; $\bar{a}$ indicates the measured value of a.

$$\tilde{b} = \bar{m} \begin{bmatrix} \left(\cos\bar{\phi}_1 \cos\bar{\psi}_1 \sin\bar{\theta}_1 + \sin\bar{\phi}_1 \sin\bar{\psi}_1\right)\left(\tilde{\omega}_{1_1}^2 + \tilde{\omega}_{2_1}^2 + \tilde{\omega}_{3_1}^2 + \tilde{\omega}_{4_1}^2\right) \\ \left(\cos\bar{\phi}_1 \sin\bar{\psi}_1 \sin\bar{\theta}_1 + \cos\bar{\psi}_1 \sin\bar{\phi}_1\right)\left(\tilde{\omega}_{1_1}^2 + \tilde{\omega}_{2_1}^2 + \tilde{\omega}_{3_1}^2 + \tilde{\omega}_{4_1}^2\right) \\ \cos\bar{\phi}_1 \cos\bar{\theta}_1 \left(\tilde{\omega}_{1_1}^2 + \tilde{\omega}_{2_1}^2 + \tilde{\omega}_{3_1}^2 + \tilde{\omega}_{4_1}^2\right) \\ \vdots \\ \left(\cos\bar{\phi}_h \cos\bar{\psi}_h \sin\bar{\theta}_h + \sin\bar{\phi}_h \sin\bar{\psi}_h\right)\left(\tilde{\omega}_{1_h}^2 + \tilde{\omega}_{2_h}^2 + \tilde{\omega}_{3_h}^2 + \tilde{\omega}_{4_h}^2\right) \\ \left(\cos\bar{\phi}_h \sin\bar{\psi}_h \sin\bar{\theta}_h + \cos\bar{\psi}_h \sin\bar{\phi}_h\right)\left(\tilde{\omega}_{1_h}^2 + \tilde{\omega}_{2_h}^2 + \tilde{\omega}_{3_h}^2 + \tilde{\omega}_{4_h}^2\right) \\ \cos\bar{\phi}_h \cos\bar{\theta}_h \left(\tilde{\omega}_{1_h}^2 + \tilde{\omega}_{2_h}^2 + \tilde{\omega}_{3_h}^2 + \tilde{\omega}_{4_h}^2\right) \end{bmatrix}^+ \begin{bmatrix} \bar{a}_{x_1} \\ \bar{a}_{y_1} \\ \bar{a}_{z_1} + \bar{g} \\ \vdots \\ \bar{a}_{x_h} \\ \bar{a}_{y_h} \\ \bar{a}_{z_h} + \bar{g} \end{bmatrix}.$$

With the help of Matlab we obtain that $\tilde{b} = 2,1 \cdot 10^{-5}\mathrm{N} \cdot \mathrm{s}^2$.

If we want to compute the thrust factors individually for each propeller b_1, b_2, b_3 and b_4, instead of the first equation of 2.2, we have to consider the following one

$$T = b_1\omega_1^2 + b_2\omega_2^2 + b_3\omega_3^2 + b_4\omega_4^2,$$

which gives us

$$\begin{aligned} a_x &= -\frac{1}{m}\left(\cos\phi\cos\psi\sin\theta + \sin\phi\sin\psi\right)\left(b_1\omega_1^2 + b_2\omega_2^2 + b_3\omega_i^2 + b_4\omega_i^2\right) \\ a_y &= -\frac{1}{m}\left(\cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi\right)\left(b_1\omega_1^2 + b_2\omega_2^2 + b_3\omega_i^2 + b_4\omega_i^2\right) \\ a_z &= -\frac{1}{m}\cos\phi\cos\theta\left(b_1\omega_1^2 + b_2\omega_2^2 + b_3\omega_i^2 + b_4\omega_i^2\right) + g, \end{aligned}$$

from which we can extract b_1, b_2, b_3 and b_4 and then estimate $\tilde{b}_1$, $\tilde{b}_2$, $\tilde{b}_3$ and $\tilde{b}_4$ as showed before. Again with the help of Matlab we obtain that $\tilde{b}_1 \approx \tilde{b}_2 \approx \tilde{b}_3 \approx \tilde{b}_4 \approx 2,1 \cdot 10^{-5}\mathrm{N} \cdot \mathrm{s}^2$.

5.2.3 Drag coefficient and moments of inertia

Once we have the thrust coefficient b, we can proceed with the estimation of the quadrotor moments of inertia I_x, I_y, I_z and propellers drag coefficient d [15]. To estimate I_x, I_y, I_z and d we replace the last three equations from 2.2 into the last three equations from 2.9 and obtain

$$\begin{aligned}\dot{p} &= \frac{I_y - I_z}{I_x} qr + \frac{-lb\omega_2^2 + lb\omega_4^2}{I_x} \\ \dot{q} &= \frac{I_z - I_x}{I_y} pr + \frac{-lb\omega_1^2 + lb\omega_3^2}{I_y} \\ \dot{r} &= \frac{I_x - I_y}{I_z} pq + \frac{d\omega_1^2 - d\omega_2^2 + d\omega_3^2 - d\omega_4^2}{I_z}.\end{aligned} \tag{5.2}$$

Extracting I_x, I_y, I_z and d from 5.2 we can write:

$$\begin{bmatrix} I_x \\ I_y \\ I_z \\ d \end{bmatrix} = lb \begin{bmatrix} -\dot{p} & qr & -qr & 0 \\ -pr & -\dot{q} & pr & 0 \\ pq & -pq & -\dot{r} & \omega_1^2 - \omega_2^2 + \omega_3^2 - \omega_4^2 \end{bmatrix}^+ \begin{bmatrix} \omega_2^2 - \omega_4^2 \\ \omega_1^2 - \omega_3^2 \\ 0 \end{bmatrix}$$

We use the batch of h measurements collected from the UAV's IMU and Autopilot for the thrust estimation in Subsection 5.2.2. Finally, stacking up $3h$ equations we can estimate $\tilde{I_x}$, $\tilde{I_y}$, $\tilde{I_z}$ and $\tilde{d}$:

$$\begin{bmatrix} \tilde{I_x} \\ \tilde{I_y} \\ \tilde{I_z} \\ \tilde{d} \end{bmatrix} = \tilde{lb} \begin{bmatrix} -\dot{\tilde{p}}_1 & \bar{q}_1\bar{r}_1 & -\bar{q}_1\bar{r}_1 & 0 \\ -\bar{p}_1\bar{r}_1 & -\dot{\tilde{q}}_1 & \bar{p}_1\bar{r}_1 & 0 \\ \bar{p}_1\bar{q}_1 & -\bar{p}_1\bar{q}_1 & -\dot{\tilde{r}}_1 & \tilde{\omega}_{1_1}^2 - \tilde{\omega}_{2_1}^2 + \tilde{\omega}_{3_1}^2 - \tilde{\omega}_{4_1}^2 \\ \vdots & \vdots & \vdots & \vdots \\ -\dot{\tilde{p}}_n & \bar{q}_n\bar{r}_n & -\bar{q}_n\bar{r}_n & 0 \\ -\bar{p}_n\bar{r}_n & -\dot{\tilde{q}}_n & \bar{p}_n\bar{r}_n & 0 \\ \bar{p}_n\bar{q}_n & -\bar{p}_n\bar{q}_n & -\dot{\tilde{r}}_n & \tilde{\omega}_{1_n}^2 - \tilde{\omega}_{2_n}^2 + \tilde{\omega}_{3_n}^2 - \tilde{\omega}_{4_n}^2 \end{bmatrix}^+ \begin{bmatrix} \tilde{\omega}_{2_1}^2 - \tilde{\omega}_{4_1}^2 \\ \tilde{\omega}_{1_1}^2 - \tilde{\omega}_{3_1}^2 \\ 0 \\ \vdots \\ \tilde{\omega}_{2_n}^2 - \tilde{\omega}_{4_n}^2 \\ \tilde{\omega}_{1_n}^2 - \tilde{\omega}_{3_n}^2 \\ 0 \end{bmatrix}.$$

With the help of Matlab we obtain that $\tilde{I_x} = 1,3 \cdot 10^{-2}\text{kg} \cdot \text{m}^2$, $\tilde{I_y} = 1,2 \cdot 10^{-2}\text{kg} \cdot \text{m}^2$, $\tilde{I_z} = 2,2 \cdot 10^{-2}\text{kg} \cdot \text{m}^2$ and $\tilde{d} = 7,5 \cdot 10^{-7}\text{N} \cdot \text{m} \cdot \text{s}^2$.

If we want to compute the drag factors individually for each propeller d_1, d_2, d_3 and d_4, instead of the last equation of 2.2, we have to consider the following one

$$\tau_\psi = d_1\omega_1^2 - d_2\omega_2^2 + d_3\omega_3^2 - d_4\omega_4^2$$

and therefore we obtain:

$$\begin{aligned}
\dot{p} &= \frac{I_y - I_z}{I_x} qr + \frac{-lb\omega_2^2 + lb\omega_4^2}{I_x} \\
\dot{q} &= \frac{I_z - I_x}{I_y} pr + \frac{-lb\omega_1^2 + lb\omega_3^2}{I_y} \\
\dot{r} &= \frac{I_x - I_y}{I_z} pq + \frac{d_1\omega_1^2 - d_2\omega_2^2 + d_3\omega_3^2 - d_4\omega_4^2}{I_z},
\end{aligned}$$

from which we can extract I_x, I_y, I_z, d_1, d_2, d_3 and d_4 and then estimate $\tilde{I}_x$, $\tilde{I}_y$, $\tilde{I}_z$, $\tilde{d}_1$, $\tilde{d}_2$, $\tilde{d}_3$ and $\tilde{d}_4$ as showed before. Again with the help of Matlab we obtain that the estimated inertia matrices do not change and the drag factors are slightly different $\tilde{d}_1 \approx \tilde{d}_2 \approx \tilde{d}_3 \approx \tilde{d}_4 \approx 7,5 \cdot 10^{-7}\mathrm{N} \cdot \mathrm{m} \cdot \mathrm{s}^2$.

5.3 Data filtering

5.3.1 Median filter

When the quadrotor crosses over a step even small, the sonar mounted on the quadrotor does not receive back the bounced ultrasound signal. As a consequence, it generates spikes. In order to remove these spikes we use the median filter.

We collect the last $h+1$ raw readings $\{\bar{z}_{k-h}, \ldots, \bar{z}_k\}$ from sonar. After while, the estimated quadrotor height $\tilde{z}_k$ at current instant k is

$$\tilde{z}_k = \mathrm{median}(\{\bar{z}_{k-h}, \ldots, \bar{z}_k\}),$$

where median() is a function that computes the median[8] value. This filter is able to eliminate up to $\lfloor \frac{h}{2} \rfloor$ spikes. The computational cost of this filter is $\mathsf{O}(h \log h)$, because we need to sort the array of $h+1$ elements to compute the median.

In Figure 5.6 is shown the row signal from sonar (blue curve) and the filtered one (red curve). The filter uses a window of the last eleven values, so it is able to eliminate up to 5 consecutive spikes. We can observe that the filtered height is not delayed.

[8] The median is the number separating the higher half of a vector from the lower half.

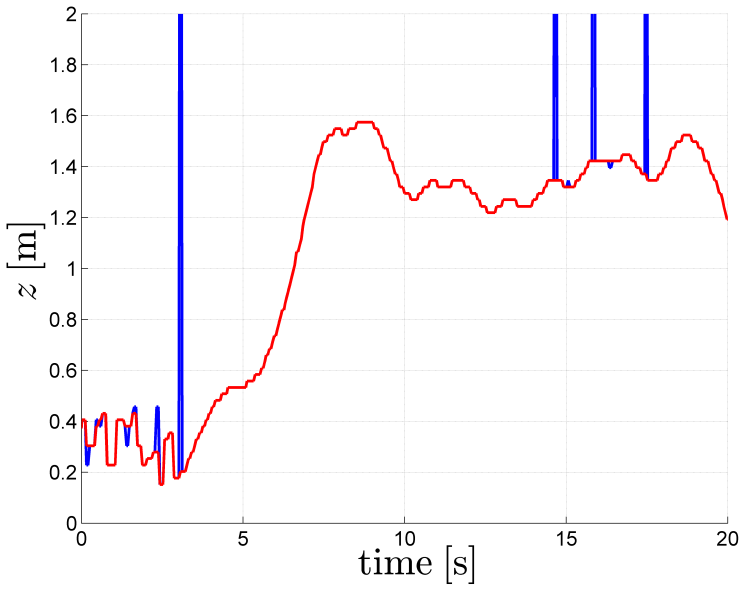


FIGURE 5.6: An example of raw quadrotor height from sonar (blue curve) and filtered value (red curve).

5.3.2 Average filter

The quadrotor angular velocities in body frame p, q and r from on-board IMU are very noisy. In order to use these data we need to remove noise. To this purpose we use the moving average filter.

We collect the last $h+1$ raw readings $\{\bar{p}_{k-h}, \ldots, \bar{p}_k\}$ from the IMU. After while, the estimated angular velocity $\tilde{p}_k$ at current instant k is

$$\tilde{p}_k = \frac{\sum_{i=k-h}^{k} \bar{p}_i}{h+1}.$$

In a very similar way we can compute also $\tilde{q}_k$ and $\tilde{r}_k$. If $\{t_{k-h}, \ldots, t_k\}$ are the times when the measurements were taken, the cut-off frequency of this filter is $\frac{0{,}44294h}{(t_k - t_{k-h})\sqrt{h^2+2h}}$[Hz] and its delay is $\frac{t_k - t_{k-h}}{2}$[s]. The computational cost of this filter is $\mathsf{O}(h)$, due to the sum of $h+1$ elements.

In Figure 5.7 is shown the row signal from IMU (blue curve) and the filtered one (red curve). The filter uses a window of the last five values, so its cut-off frequency is 7,487 Hz and its delay is 20 ms.

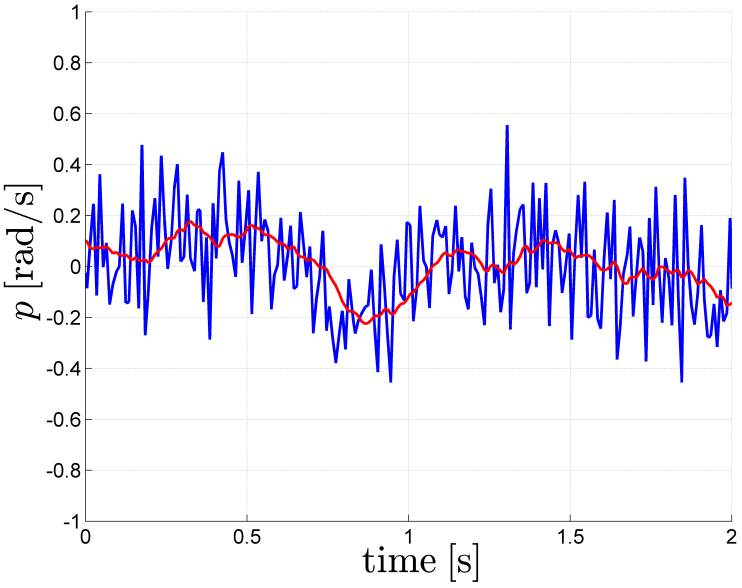


FIGURE 5.7: An example of raw quadrotor angular velocity from IMU (blue curve) and filtered value (red curve).

5.3.3 Polynomial fitting

The quadrotor linear accelerations a_x, a_y and a_z from on-board accelerometers are noisy. In order to use these data and to compute the corresponding jerks j_x, j_y and j_z we use the polynomial fitting.

We collect the last $h+1$ raw readings $\{\bar{a}_{k-h}, \ldots, \bar{a}_k\}$ given by the accelerometers. After while, we try to fit a quadratic polynomial (parabola) $f(t) = c_2 t^2 + c_1 t + c_0$ with collected values $\bar{a}_t$. To this purpose we use a linear regression:

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 1 & k-h & (k-h)^2 \\ \vdots & \vdots & \vdots \\ 1 & k & k^2 \end{bmatrix}^+ \begin{bmatrix} \bar{a}_{k-h} \\ \vdots \\ \bar{a}_k \end{bmatrix}.$$

and we find the polynomial coefficients c_0, c_1 and c_2. Now we can compute the estimated acceleration $\tilde{a}_k$ at current instant k:

$$\tilde{a}_k = f(k) = c_2 k^2 + c_1 k + c_0.$$

If $\{t_{k-h}, \ldots, t_k\}$ are the times when the measurements were taken, the delay of this filter is $\frac{t_k - t_{k-h}}{2}$[s]. But the computational cost of this filter is $\mathsf{O}(h^3)$, due to the pseudo-inverse of $(h+1) \times 3$ matrix. In order to improve the computation, we translate the

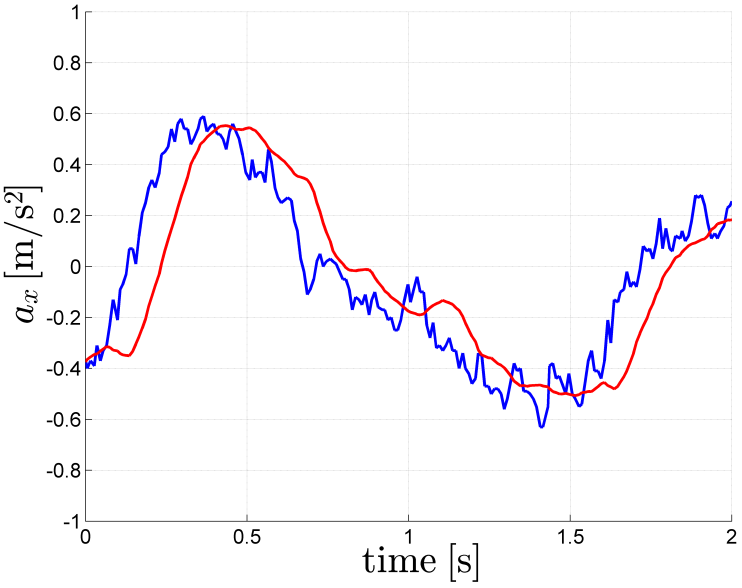


FIGURE 5.8: An example of raw quadrotor acceleration from accelerometer (blue curve) and filtered value (red curve).

current instant k to 0. We can observe that the computation of coefficients c_0, c_1 and c_2 becomes:

$$\begin{bmatrix} c_0 \\ c_1 \\ c_2 \end{bmatrix} = \begin{bmatrix} 1 & -h & h^2 \\ \vdots & \vdots & \vdots \\ 1 & 0 & 0 \end{bmatrix}^{+} \begin{bmatrix} \bar{a}_{(-h)} \\ \vdots \\ \bar{a}_0 \end{bmatrix}.$$

Now the matrix which must be pseudo-inverted is constant. So we have to compute the pseudo-inverse only once. Actually, to compute $\tilde{a}_k$ we need only one coefficient c_0:

$$\tilde{a}_k = f(k)|_{k=0} = c_0.$$

In Figure 5.8 is shown the row signal from accelerometers (blue curve) and the filtered one (red curve). The filter uses a window of the last eleven values, so its delay is 50 ms.

Having the analytical expression of acceleration in the neighbourhood of 0, we can compute the estimated value of jerk in k:

$$\tilde{j}_k = \left.\frac{\partial f(k)}{\partial k}\right|_{k=0} = c_1.$$

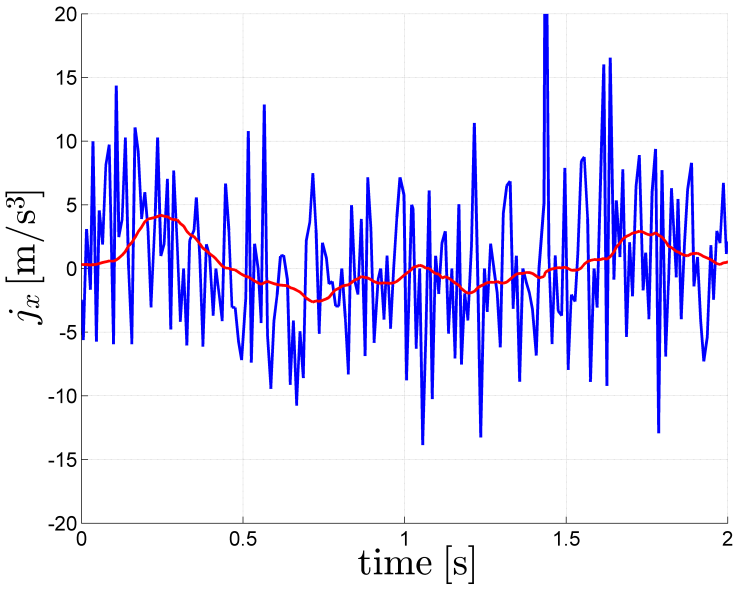


FIGURE 5.9: An example of numerical derivative of jerk (blue curve) and derivation of jerk with polynomial fitting method (red curve).

In Figure 5.9 is shown the numerical derivative of jerk (blue curve) and derivation of jerk with new polynomial fitting method (red curve).

Chapter 6

Conclusions

In this study we presented the solution of ensuring a safe navigation in an unknown environment for a quadrotor vehicle. First of all, we derived a quadrotor model, to which we applied a controller based on dynamic feedback linearization. Hence, the main task was to perform effective obstacle avoidance method. Moreover, we relied on the classical obstacle avoidance technique which uses artificial potential fields. In a first moment, we considered the hypotheses of quasi-stationary flight and we showed that the closed-loop behaviour of the resulting linearized system is not acceptable. Thus, we considered the use of a first-order feed-forward computing suitable inputs for the controller. At the end, we obtained a control framework which ensures the feasibility of the commanded trajectory for the original system. Finally, the proposed control schemes were validated in simulation and on an experimental quadrotor.

The most important limitation of the proposed controller is the requirement to measure all the state. Therefore, a Kalman filter can be implemented for more precise quadrotor state estimation. Moreover, in order to obtain the quadrotor position and linear velocity, the Vicon system can be used. Furthermore, to improve the stability of the UAV, we need better estimation of its intrinsic parameters (inertia matrix, thrust and drag coefficients). Alternatively, we can use well known and widely used geometric controller on $\mathsf{SE}(3)$[16].

Appendix A

Rotation matrix

The rotation of a rigid body in space can be parametrized using three Euler angles. These angles are individually called roll (ϕ), pitch (θ) and yaw (ψ).

Considering a right-hand oriented coordinate system, the three single rotations are described by:

- $\mathbf{R}(x, \phi)$ is the rotation around x-axis by ϕ;
- $\mathbf{R}(y, \theta)$ is the rotation around y-axis by θ;
- $\mathbf{R}(z, \psi)$ is the rotation around z-axis by ψ.

They are represented by:

$$\mathbf{R}(x, \phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix},$$

$$\mathbf{R}(y, \theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

and

$$\mathbf{R}(z, \psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The complete rotation matrix is the product of the previous three successive rotations:

$$\begin{aligned}\mathbf{R}(\phi,\theta,\psi) &= \mathbf{R}(z,\psi)\mathbf{R}(y,\theta)\mathbf{R}(x,\phi) \\ &= \begin{bmatrix} \cos\psi\cos\theta & \cos\psi\sin\phi\sin\theta - \cos\phi\sin\psi & \sin\phi\sin\psi + \cos\phi\cos\psi\sin\theta \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\psi\sin\theta & \cos\phi\sin\psi\sin\theta - \cos\psi\sin\phi \\ -\sin\theta & \cos\theta\sin\phi & \cos\phi\cos\theta \end{bmatrix}.\end{aligned} \tag{A.1}$$

A.1 Derivative of rotation matrix

Let's consider a time-varying *rotation matrix* $\mathbf{R}$ from a static frame $\mathcal{A}$ to a rotating frame $\mathcal{B}$. Let $\mathbf{p}_\mathcal{B}$ be a fixed point in frame $\mathcal{B}$ and $\mathbf{p}_\mathcal{A}$ be the same point in frame $\mathcal{A}$. Then

$$\mathbf{p}_\mathcal{A} = \mathbf{R}\mathbf{p}_\mathcal{B}. \tag{A.2}$$

The time derivative of $\mathbf{p}_\mathcal{A}(t)$ is

$$\dot{\mathbf{p}}_\mathcal{A} = \dot{\mathbf{R}}\mathbf{p}_\mathcal{B}. \tag{A.3}$$

If the vector $\boldsymbol{\omega} = \begin{bmatrix}\omega_x & \omega_y & \omega_z\end{bmatrix}^T$ denotes the *angular velocity* of frame $\mathcal{B}$ with respect to the frame $\mathcal{A}$ at time t, it is known from mechanics that

$$\dot{\mathbf{p}}_\mathcal{A} = \boldsymbol{\omega} \times \mathbf{p}_\mathcal{A} = [\boldsymbol{\omega}]_\times \mathbf{p}_\mathcal{A}, \tag{A.4}$$

in which $[\boldsymbol{\omega}]_\times$ denotes the *skew-symmetric matrix*, given by

$$[\boldsymbol{\omega}]_\times = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{A.5}$$

Combining the (A.2), (A.3) and (A.4) gives

$$\dot{\mathbf{R}}\mathbf{p}_\mathcal{B} = [\boldsymbol{\omega}]_\times \mathbf{p}_\mathcal{A} = [\boldsymbol{\omega}]_\times \mathbf{R}\mathbf{p}_\mathcal{B}. \tag{A.6}$$

Since $\mathbf{p}_{\mathcal{B}}$ is chosen arbitrarily, the above equation holds for all $\mathbf{p}_{\mathcal{B}}$. Hence, (A.6) can be rewritten as

$$\dot{\mathbf{R}} = [\boldsymbol{\omega}]_{\times}\mathbf{R}.$$

Appendix B

Lie derivative

Consider a scalar function $h : D \subset \mathbb{R}^n \to \mathbb{R}$, a vector field $f : D \subset \mathbb{R}^n \to \mathbb{R}^n$ and a vector $\mathbf{x} \in D$. The Lie derivative of h with respect to f, denoted $L_f h$, is given by

$$L_f h(\mathbf{x}) = \frac{\partial h}{\partial \mathbf{x}} f(\mathbf{x}).$$

Given two vector fields $f, g : D \subset \mathbb{R}^n \to \mathbb{R}^n$, we have that

$$L_g L_f h(\mathbf{x}) = L_g[L_f h(\mathbf{x})] = \frac{\partial (L_f h)}{\partial \mathbf{x}} g(\mathbf{x}) \tag{B.1}$$

and in the special case $f = g$ (B.1) becomes

$$L_f L_f h(\mathbf{x}) = L_f^2 h(\mathbf{x}) = \frac{\partial (L_f h)}{\partial \mathbf{x}} f(\mathbf{x}).$$

Appendix C

Inertia matrix

Let $\mathbf{I}$ be the inertia matrix given by

$$\mathbf{I} = \begin{bmatrix} I_{xx} & I_{xy} & I_{xz} \\ I_{yx} & I_{yy} & I_{yz} \\ I_{zx} & I_{zy} & I_{zz} \end{bmatrix}.$$

The quantities I_{xx}, I_{yy} and I_{zz} are called *moments of inertia* with respect to the x, y and z axis, respectively. The moments of inertia are the sums of all the elemental particles masses multiplied by their squared distance from the rotational axis and are given by

$$I_{xx} = \sum_{i=1}^{N} \left(y_i^2 + z_i^2\right) m_i = \int_0^M \left(y^2 + z^2\right) \mathrm{d}m, \tag{C.1}$$

$$I_{yy} = \sum_{i=1}^{N} \left(x_i^2 + z_i^2\right) m_i = \int_0^M \left(x^2 + z^2\right) \mathrm{d}m \tag{C.2}$$

and

$$I_{zz} = \sum_{i=1}^{N} \left(x_i^2 + y_i^2\right) m_i = \int_0^M \left(x^2 + y^2\right) \mathrm{d}m. \tag{C.3}$$

The quantity in the integrand is precisely the square of the distance to the x, y and z axis, respectively. It is also clear, from their expressions, that the moments of inertia are always positive. The quantities I_{xy}, I_{xz}, I_{yx}, I_{yz}, I_{zx} and I_{zy} are called *products of inertia*. They are given by

$$I_{xy} = I_{yx} = -\sum_{i=1}^{N} x_i y_i m_i = -\int_0^M (xy)\,\mathrm{d}m, \tag{C.4}$$

$$I_{xz} = I_{zx} = -\sum_{i=1}^{N} x_i z_i m_i = -\int_0^M (xz)\,\mathrm{d}m,$$

and

$$I_{yz} = I_{zy} = -\sum_{i=1}^{N} y_i z_i m_i = -\int_0^M (yz)\,\mathrm{d}m.$$

C.1 Solid cuboid

Lets compute the inertia matrix $\mathbf{I}_{cub}$ of a solid cuboid. The cuboid length is X along the x-axis, Y along the y-axis, Z along the z-axis and its mass is M.

From the definition of mass M:

$$M = \rho V$$

and from the definition of cuboid volume V

$$V = XYZ,$$

we have

$$M = \rho XYZ, \tag{C.5}$$

where ρ is the cuboid density. The derivative of (C.5) is

$$\mathrm{d}m = \rho \mathrm{d}x\mathrm{d}y\mathrm{d}z. \tag{C.6}$$

Replacing (C.6) in (C.1), one can compute the moment of inertia about the x-axis I_{xx}:

$$\begin{aligned} I_{xx} &= \int_{-\frac{Z}{2}}^{\frac{Z}{2}} \int_{-\frac{Y}{2}}^{\frac{Y}{2}} \int_{-\frac{X}{2}}^{\frac{X}{2}} \left(\left(y^2 + z^2\right) \rho\right) \mathrm{d}x\mathrm{d}y\mathrm{d}z \\ &= \rho \left(\frac{XY^3Z}{12} + \frac{XYZ^3}{12}\right). \end{aligned} \tag{C.7}$$

Inverting (C.5), one obtain ρ:

$$\rho = \frac{M}{XYZ},$$

and replacing it in (C.7) gives

$$I_{xx} = \frac{M}{12}\left(Y^2 + Z^2\right).$$

The other two moments of inertia I_{yy} and I_{zz} can be computed in a similar way by using (C.2) and (C.3). Thus,

$$I_{yy} = \frac{M}{12}\left(X^2 + Z^2\right)$$

and

$$I_{zz} = \frac{M}{12}\left(X^2 + Y^2\right).$$

In order to compute the xy and yx products of inertia, (C.4) is used:

$$I_{xy} = \int_{-\frac{Z}{2}}^{\frac{Z}{2}} \int_{-\frac{Y}{2}}^{\frac{Y}{2}} \int_{-\frac{X}{2}}^{\frac{X}{2}} (xy\rho)\, \mathrm{d}x\mathrm{d}y\mathrm{d}z = 0.$$

Similarly, I_{xz} and I_{yz} are 0.

Finally, the inertia matrix of a cuboid is

$$\mathbf{I}_{cub} = \frac{M}{12}\begin{bmatrix} Y^2 + Z^2 & 0 & 0 \\ 0 & X^2 + Z^2 & 0 \\ 0 & 0 & X^2 + Y^2 \end{bmatrix}. \tag{C.8}$$

Due to the symmetry of the cuboid, its inertia matrix $\mathbf{I}_{cub}$ is diagonal.

C.2 Solid cylinder

Lets compute the inertia matrix $\mathbf{I}_{cyl}$ of a solid cylinder. The cylinder's height is H, its radius is R and its mass is M.

From the definition of mass M:

$$M = \rho V$$

and from the definition of cylinder's volume V

$$V = \pi R^2 H,$$

we have

$$M = \pi \rho R^2 H, \tag{C.9}$$

where ρ is the cylinder density. The derivative of (C.9) is

$$\mathrm{d}m = 2\pi \rho R \mathrm{d}r\mathrm{d}h. \tag{C.10}$$

Replacing (C.10) in (C.1), one can compute the moment of inertia about the x-axis I_{xx}:

$$\begin{aligned} I_{xx} &= \int_{-\frac{H}{2}}^{\frac{H}{2}} \int_0^R \left(\left(r^2 + h^2\right) 2\pi \rho R \right) \mathrm{d}r\mathrm{d}h \\ &= \pi \rho \left(\frac{R^3 H}{4} + \frac{R^2 H^2}{12} \right). \end{aligned} \tag{C.11}$$

Inverting (C.9), one obtain ρ:

$$\rho = \frac{M}{\pi R^2 H},$$

and replacing it in (C.11) gives

$$I_{xx} = \frac{M}{12} \left(3R^2 + H^2\right).$$

The moment of inertia about the y-axis I_{yy} is equal to I_{xx}, because the symmetry of the cylinder. Thus,

$$I_{yy} = \frac{M}{12}\left(3R^2 + H^2\right).$$

In order to compute the moment of inertia about the z-axis I_{zz}, we replace (C.10) in (C.3):

$$\begin{aligned} I_{zz} &= \int_{-\frac{H}{2}}^{\frac{H}{2}} \int_0^R \left(\left(r^2 + h^2\right) 2\pi\rho R\right) \mathrm{d}r\mathrm{d}h \\ &= \pi\rho\left(\frac{R^3 H}{4} + \frac{R^2 H^2}{12}\right). \end{aligned} \tag{C.12}$$

Inverting (C.9), one obtain ρ:

$$\rho = \frac{M}{\pi R^2 H},$$

and replacing it in (C.11) gives

$$I_{xx} = \frac{M}{12}\left(3R^2 + H^2\right).$$

In order to compute the xy and yx products of inertia, (C.4) is used:

$$I_{xy} = \int_{-\frac{Z}{2}}^{\frac{Z}{2}} \int_{-\frac{Y}{2}}^{\frac{Y}{2}} \int_{-\frac{X}{2}}^{\frac{X}{2}} (xy\rho)\,\mathrm{d}x\mathrm{d}y\mathrm{d}z = 0.$$

Similarly, I_{xz} and I_{yz} are 0.

Finally, the inertia matrix of a cylinder is

$$\mathbf{I}_{cyl} = \frac{M}{12}\begin{bmatrix} 3R^2 + H^2 & 0 & 0 \\ 0 & 3R^2 + H^2 & 0 \\ 0 & 0 & 6R^2 \end{bmatrix}. \tag{C.13}$$

Due to the symmetry of the cylinder, its inertia matrix $\mathbf{I}_{cyl}$ is diagonal.

Appendix D

Point reprojection

Consider a projective camera matrix $\mathbf{P}$ given by

$$\mathbf{P} = \begin{bmatrix} \alpha & 0 & x_0 & 0 \\ 0 & \alpha & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix},$$

in which α represent the focal length of the camera in terms of pixel dimensions and $(x_0, y_0)^T$ is the principal point in terms of pixel dimensions. The camera maps a 3D point $(X, Y, Z)^T$ to a 2D point $(x, y)^T$ by the following relation

$$\begin{bmatrix} x \\ y \\ Z \end{bmatrix} = \mathbf{P} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

After the point normalization, we obtain

$$\begin{aligned} x &= \frac{X}{Z}\alpha + x_0 \\ y &= \frac{Y}{Z}\alpha + y_0. \end{aligned} \tag{D.1}$$

Assuming that the intrinsic camera parameters (α, x_0 and y_0) are known and that the depth of the point Z can be estimated, we can compute the position of the point in the space from (D.1):

$$X = \frac{Z}{\alpha}(x - x_0)$$
$$Y = \frac{Z}{\alpha}(y - y_0).$$

So, the coordinates of the reprojected point $(x, y)^T$ are

$$\begin{bmatrix} \frac{Z}{\alpha}(x - x_0) \\ \frac{Z}{\alpha}(y - y_0) \\ Z \end{bmatrix}. \tag{D.2}$$

Bibliography

[1] S. Bouabdallah. *Design and Control of Quadrotors with Application to Autonomous Flying*. PhD thesis, 2007.

[2] R. Mahony, V. Kumar, and P. Corke. Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor. *Robotics & Automation Magazine*, pages 20–32, 2012.

[3] V. Mistler, A. Benallegue, and N.K. M'Sirdi. Exact linearization and noninteracting control of a 4 rotors helicopter via dynamic feedback. In *Robot and Human Interactive Communication*, pages 586–593. IEEE, 2001.

[4] S.B.V. Gomes and J.J.G. Ramos. Airship dynamic modelling for autonomous operation. In *Robotics & Automation*, pages 3462–3467. IEEE, 2001.

[5] H.K. Khalil. *Non-Linear System*. Prentice Hall, 1996.

[6] A. Isidori. *Non-Linear Control System*. Springer, 1995.

[7] S.A. Al-Hiddabi. Quadrotor control using feedback linearization with dynamic extension. In *International Symposium on Mechatronics and its Applications (ISMA09)*, pages 1–3. IEEE, 2009.

[8] D. Mellinger and V. Kumar. Minimum snap trajectory generation and control for quadrotors. In *Robotics and Automation (ICRA)*, pages 2520–2525. IEEE, 2011.

[9] A. Isidori, C.H. Moog, and A. De Luca. A sufficient condition for full linearization via dynamic state feedback. In *Decision and Control*, pages 203–208. IEEE, 1986.

[10] M.-D. Hua, T. Hamel, P. Morin, and C. Samson. A control approach for thrust-propelled underactuated vehicles and its application to vtol drones. *Automatic Control*, pages 1837–1853, 2009.

[11] M.-D. Hua, T. Hamel, P. Morin, and C. Samson. Introduction to feedback control of underactuated vtol vehicles. In *Control Systems*, pages 61–75. IEEE, 2013.

[12] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics Modelling, Planning and Control*. Springer, 2008.

[13] M.G. Park, J.H. Jeon, and M.C. Lee. Obstacle avoidance for mobile robots using artificial potential field approach with simulated annealing. In *Industrial Electronics*, pages 1530–1535. IEEE, 2001.

[14] D. Zhou and M. Schwager. Vector field following for quadrotors using differential flatness. In *Robotics and Automation (ICRA)*, pages 6567–6572. IEEE, 2014.

[15] N. Abas, A. Legowo, and R. Akmeliawati. Parameter identification of an autonomous quadrotor. In *Mechatronics (ICOM)*, pages 1–8. IEEE, 2011.

[16] T. Lee, M. Leok, and N.H. McClamroch. Geometric tracking control of a quadrotor uav on se(3). In *Decision and Control (CDC)*, pages 420–5425. IEEE, 2010.

Подяка

Насамперед, я хотів би подякувати моїм батькам, без яких було б неможливо досягти цієї мети. Також велике дякую моїй сестрі Олені за те, що підтримувала мене увесь цей час. Ще я дякую т. Лялі, Вірі та Іванові за те, що «тримали кулаки» за мій успіх, і всім, хто в мене вірив.

Андрій

Ringraziamenti

Ringrazio il prof. Giuseppe Oriolo per la disponibilità e l'opportunità che mi ha dato nel migliorare e dimostrare le mie capacità attraverso questo lavoro. Vorrei inoltre ringraziare dott. Lorenzo Rosa che con la massima pazienza mi ha sostenuto e mi ha aiutato gentilmente nella ricerca delle soluzioni. Grazie anche a Roberto, Antonio, Andrea, Angela e molti altri che hanno condiviso con me di questo percorso.

Andriy

Acknowledgements

I would like to thank my classmates for the exciting experience we shared together, the stimulating discussions, the sleepless nights we were working together before deadlines, and all the fun we have had in the last three years.

Andriy