Planning and control of aerial grasping with a quadrotor UAV

Facoltà di Ingegneria dell'Informazione Informatica e Statistica
Corso di Laurea Magistrale in Ingegneria Elettronica
Cattedra di Robotica mobile e autonoma

Candidato
Riccardo Spica
n° matricola
1148899

Relatore
Prof. Giuseppe Oriolo

Correlatore
dott. Antonio Franchi
dott. Paolo Robuffo Giordano

a.a. 2011/2012

Experience is what you get
when you didn't get what you wanted

Randy Pausch

Abstract

An Unmanned Aerial Vehicle (UAV) is a flying machine which does not carry a human operator and is either autonomous or remotely piloted. UAVs are very promising robotic systems for their potential to achieve many applications with reduced cost/danger associated with the absence of human pilots on board. This is particularly true for the so called Miniature UAVs (MAVs), i.e. autonomous flying vehicles of narrow size, typically small enough to be man-portable. These devices usually have low payloads and, consequently, low on-board computational capabilities and poor sensor equipment. On the other hand, their small size makes it possible to employ them in indoor environments. An example of this kind of platform is the quadrotor, a particular kind of rotor-craft propelled by four fixed-pitch rotors disposed on the vertices of a square and independently actuated. By combining rotor speeds in different ways the quadrotor can move in any direction of the space and also rotate around the yaw axis.

In this thesis we deal with the problem of developing a motion planning algorithm and a suitable controller that allow a quadrotor, equipped with a gripper, to perform the grasping of a moving target object in a minimum time.

Possible applications of this study are in search and rescue operations, good transportation and aerial manipulation.

First a mathematical formulation of the problem is derived. Then a possible solution is presented, that is based on the application of the Pontryagin's minimum principle. Since this approach is not easy to handle from a numerical point of view, a different strategy is proposed that is based on the composition of multiple sub-trajectories to obtain the complete grasping trajectory. A controller is also described that is able to track the trajectories generated by the developed motion planning algorithm. The performances of the proposed method are finally tested using realistic physical simulations.

Sommario

Un velivolo senza pilota (UAV) è una macchina volante priva di operatore umano a bordo, pilotata da remoto o completamente autonoma. I velivoli senza pilota sono sistemi robotici molto promettenti. Essi possono essere impiegati efficacemente in diverse applicazioni, riducendo i costi e i pericoli normalmente associati alla presenza di piloti a bordo. Ciò è particolarmente vero per i velivoli senza pilota di dimensioni ridotte, tipicamente abbastanza piccoli da poter essere trasportati a mano. Questi dispositivi hanno solitamente ridotte capacità di carico e, di conseguenza, scarse capacità computazionali ed equipaggiamento sensoristico. Grazie alle loro ridotte dimensioni, tuttavia, essi possono venire impiegati anche in ambienti chiusi. Un esempio di questo tipo è il quadrirotore: un particolare velivolo ad ala rotante dotato di quattro rotori ad asse fisso disposti ai vertici di un quadrato e attuati in modo indipendente. Combinando opportunamente le velocità dei singoli rotori il quadrirotore può muoversi in tutte le direzioni dello spazio e ruotare intorno al suo asse verticale.

In questa tesi si affronta il problema dello sviluppo di un pianificatore di traiettorie e di un controllore che permettano ad un quadrirotore, equipaggiato con una pinza a bordo, di afferrare oggetti in moto in un tempo minimo.

Possibili applicazioni di questo studio vanno dalle operazioni di ricerca e salvataggio al trasposto di merci e alla manipolazione aerea.

Dopo aver derivato una formulazione matematica del problema, viene proposta una prima soluzione basata sul principio del minimo di Pontryagin. Poiché tale strategia non si presta ad una soluzione numerica, viene proposto un approccio alternativo basato sulla composizione di diverse sotto-traiettorie per l'ottenimento della traiettoria di grasping completa. Viene inoltre descritto un controllore in grado di eseguire effettivamente le traierrorie generate dall'algoritmo di pianificazione sviluppato. Infine le prestazioni del metodo proposto vengono dimostrate con l'ausilio di un simulatore fisico realistico.

Acknowledgements

I want to thank Dr. Antonio Franchi and Dr. Paolo Robuffo Giordano for having competently assisted me during the preparation of this thesis at the Max Planck Institute for Biological Cybernetics. Despite all the commitments that absorb the life of a researcher, they have never denied their help to me.

I also thank Prof. Dr. Giuseppe Oriolo for having supervised my work.

Many thanks to the Max Planck Institute for Biological Cybernetics and, in particular, to Prof. Dr. Heinrich H. Bülthoff for having given me the opportunity to live this experience, and for having provided for all the tools I needed to pursue this study.

I want to thank Martin Riedel and Johannes Lächele for their indispensable help with the implementation of the physical simulations. An acknowledgement is also due to all the colleagues of the Human-Robot Interaction Group for having warmly welcomed me into their nice working environment and for the invaluable small hints that made it possible the accomplishment of these results.

Finally I want to thank my parents, my sister and my friends, for their unconditional support and love.

Contents

Chapter 1

Introduction

An Unmanned Aerial Vehicle (UAV) is a flying machine which does not carry a human operator and is either autonomous or remotely piloted. UAVs are very promising robotic systems for their potential to achieve many powerful applications with reduced cost/danger associated with the absence of human pilots on board. Such autonomous vehicles are mostly employed in military applications where they are used both for passive surveillance and for carrying and launching rockets, thus reducing the cost of the missions and the risks for human pilots. UAVs are also employed in civil applications concerning environment surveillance, traffic control, pesticide-spraying, landscape survey, ad-hoc communication network and autonomous exploration of dangerous and inhospitable areas. The diffusion of this kind of devices in the field of entertainment is also remarkable.

The reason why there has been great interest for UAVs in the last years is that they are both cheap and versatile. This is particularly true for the so called Miniature UAVs (MAVs) or Small UAVs (SUAVs), i.e. autonomous flying vehicles of narrow size, typically small enough to be man-portable. These devices usually have low payloads and, consequently, low on-board computational capabilities and poor sensor equipment. On the other hand, their small size makes it possible to employ them in indoor environments.

An example of this kind of platform is the quadrotor, also called quadcopter or quadrocopter (see fig. 1.1). This is a particular kind of rotor-craft propelled by four fixed-pitch rotors disposed on the vertices of a square and independently actuated. By combining rotor speeds in different ways the quadrotor can move in any direction of the space and also rotate around its vertical axis. When all rotors are spinning at the same angular velocity, they generate a thrust that, together with the gravity force, moves the robot vertically. Translational acceleration is achieved by maintaining a non-zero pitch and/or roll angle. To compensate the anti-torque generated by the air drag on the rotating blades, the propellers do not

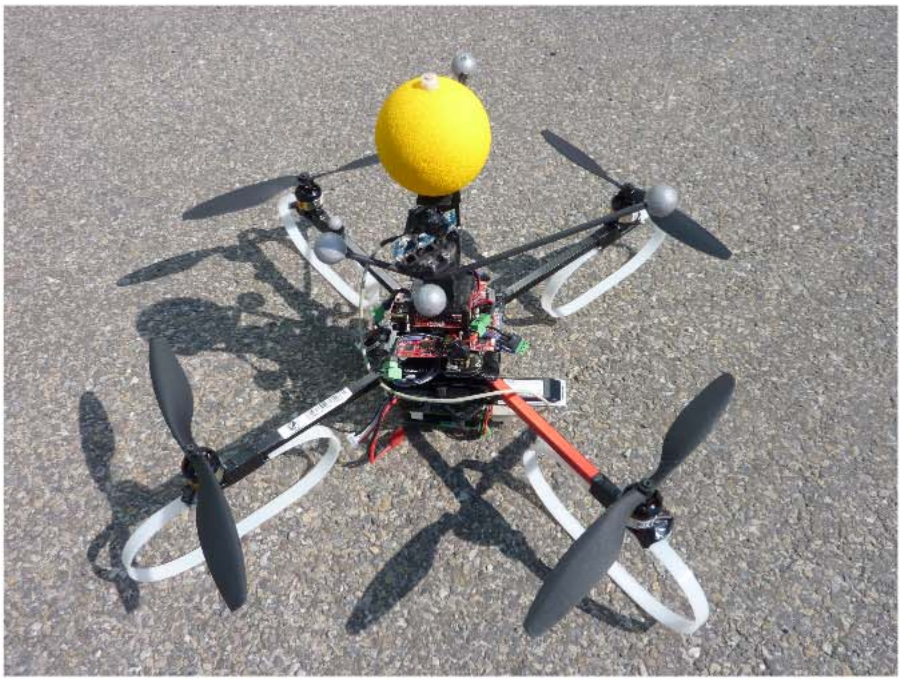


Figure 1.1: The Microkopter robotic system

turn all in the same verse: each propeller, instead, rotates in the same direction of the opposite one and in the opposite direction of the two adjacent ones. Each pair of blades rotating in the same direction also controls one axis, either roll or pitch one. Torques about the roll or pitch axes are indeed achieved by unbalancing the velocities of rotors belonging to the same pair. Finally torques about the yaw axis are obtained by unbalancing the velocity of one co-rotating couple with respect to the other one, thus unbalancing the compensation of the anti-torque.

The absence of complex mechanical linkages to vary the rotor blade pitch angle as they spin, makes this system much simpler and robust with respect to standard helicopters, especially in the case of small size vehicles. Moreover the relatively large displacement between opposite rotors makes it possible to generate high rotational torques and thus to perform complex and agile maneuvers as it has been demonstrated in [1, 2, 3].

The use of four rotors instead of one, allows each of them to have a smaller diameter than the equivalent helicopter rotor. The blades will consequently have less kinetic energy during flight, thus reducing the damage in case of collisions with an obstacle. This makes it safer to navigate in narrow and cluttered environments.

In this thesis we deal with the problem of using a quadrotor helicopter to perform the grasping of a moving target object. The vehicle is assumed to be equipped with a pseudo-claw/gripper that can grasp a target object only if some constraints on the position and velocity of the gripper with respect to the target are satisfied. The grasping must also be accomplished in a minimum time. Finally we take into accout realistic limitations on the quadrotor control authority (upper/lower bounds on the propeller speeds) and the need of a finite time for the gripper in order to successfully perform the grasping task.

Possible applications of this study are in search and rescue operations, good transportation and aerial manipulation.

Consider, for example, the case of a man that must be saved from drowning in a river. In this case the trajectory of the target is only partially known from the river path and water velocity and it is not possible to stop his motion. Moreover it is clear that the grasping must take place in a minimum time.

Another possible application is that in which one wants to transfer a shipment parcel from a mean of transport to another one without stopping neither of them. Also in this case we can assume to have a partial knowledge of the target trajectory before grasping, given the path that the target carriers are following (roads, railways, maritime courses, etc...) and some information about the range of achievable velocities. The duration of the transfer might affect the cost of the operation and then its minimization might be an important goal. A quadrotor could also be used to autonomously move parts in assembly operations as it has been shown [4]. Note that the problem of grasping an object and that of placing it in a specified position are equivalent. In this case time constraints might be due to the necessity to interact with other machines, e.g. in assembly lines.

Finally quadrotors have also demonstrated to be able to perform cooperative aerial manipulation and transportation tasks (see [5, 6, 7, 8]). In this context it may be useful for a quadrotor to have the possibility to grasp an object which is being hold in air by one or more other vehicles, already performing some manipulation tasks. It is also possible that a quadrotor wants to change the grasp configuration by leaving the object and grasping it again in a different way. In these cases it is also very important to consider what the robot does soon after the grasping has been accomplished because it clearly influences the motion of the other robots.

Different motion planning algorithms for the quadrotor, without a focus on grasping tasks, have been proposed in the literature. In [9, 10, 11] the trajectories are parameterized using polynomials or splines and the optimal values of the parameters are obtained by resorting to numerical methods. In [12, 3, 13] the Pontryagin's minimum principle is applied to a simplified model of the quadrotor. All the above planning methods assume hovering (i.e. parallel to the ground) initial and final configurations of the robot, thus not allowing for an arbitrary selection of the full-3D orientation of the quadrotor.

An approach to overcome this limitation is presented in [1] where the quadrotor, after having reached a launch configuration, switches to an attitude-only controller in order to attain the desired orientation. Nevertheless this approach relies on heuristics and require a learning experimental phase to tune the controller parameters accounting for errors in the dynamic model and noise. A similar strategy is adopted in [14] for perching on a vertical surface. Also in [15] acrobatic maneuvers with multiple flips are obtained using a learning strategy for a low-order first-principles 2D model of the quadrotor.

As for quadrotor grasping applications, several previous works have also explored possible solutions. In [8] a gripping mechanism is developed and a control algorithm is presented to estimate and compensate the effects of the payload on the system dynamics. In [16], the authors discuss the issues involved in automatic assembly and construction of structures using flying vehicles. However, in both these works, the target is assumed to be still and the grasping is assumed to be performed in a hovering condition. The planning process is then reduced to a point-to-point motion without specific optimality considerations.

Compared to these previous works, the main contributions of this thesis can be listed as follows:

- a time-optimal planning strategy is applied in order to minimize the duration of the grasping trajectory;
- the quadrotor is allowed to attain generic, also non-hovering, states during the grasping phase in order to enrich the search space for the optimization;
- the case of a target moving at constant velocity during the grasping is considered.

This is also complemented by the presence of additional 'real-world' constraints such as limited actuation capabilities and the need of a finite time to actually lock the gripper and perform the grasp.

Part of the thesis is also aimed at developing a control algorithm that is able to track the generated trajectories and at the implementation of the proposed method in a realistic physical simulator.

The thesis is organized as follows:

- in chapter 2 the dynamical model of the robot is derived using Newton-Euler approach;
- in chapter 3 the grasping problem is introduced and a mathematical formulation is provided;

- in chapter 4 an important property of the system is introduced, that is the differential flatness. Some considerations about the controllability of the system are also provided;
- in chapter 5 a strategy is proposed for solving the grasping problem by using the Pontryagin's minimum principle;
- in chapter 6 an alternative approach is described that is based on the composition of different sub-trajectories;
- in chapter 7 some planning results obtained by applying the latter strategy are presented;
- in chapter 8 a controller is described that can be used to track the generated trajectories;
- in chapter 9 the implementation on the physical simulator SwarmSimX is described;
- in chapter 10 some conclusion are drown and possible further developments are proposed.

Chapter 2

Dynamic model of the quadrotor

Before addressing the actual grasping problem we need to derive the dynamical model of the quadrotor. To this purpose we adopt the Newton-Euler approach. We start by introducing two reference frames: the world inertia frame W and the robot frame B, whose origin O_B corresponds to the robot center of mass (see fig. 2.1).

In our notation we indicate with a right subscript the frame to which a quantity is referred and with a left superscript the frame in which a quantity is expressed. For instance the expression ${}^{W}r_B$ will indicate the position of the frame B expressed in the frame W. Whenever the left superscript is absent, quantities are assumed to be expressed in the world inertial frame W, then r_B will be used instead of ${}^{W}r_B$.

The configuration of the quadrotor can then be described in terms of the position and orientation of the frame B with respect to the frame W. Its state, denoted by χ, will also comprise information on the derivatives of these quantities, i.e. the

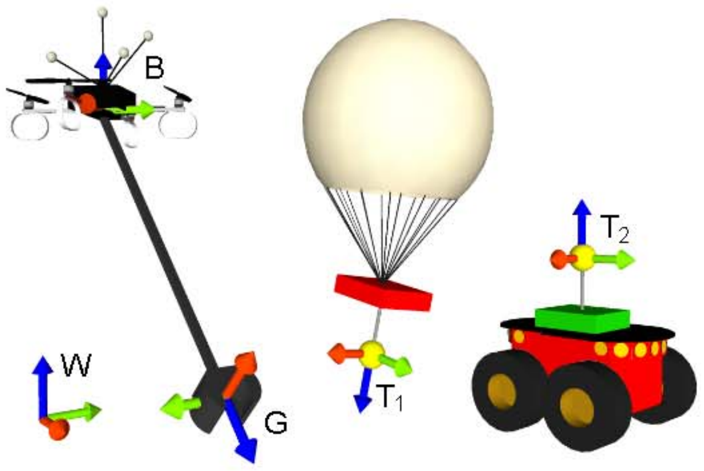


Figure 2.1: Reference frames

linear and angular velocity of the frame B with respect to the frame W, namely

$$\chi = \begin{bmatrix} r_B \\ \dot{r}_B \\ {}^W R_B \\ {}^B\omega_{BW} \end{bmatrix} \in X = SE(3) \times se(3).$$

In some cases, for more clearance, we will also use the roll, pitch and yaw (RPY) angles to represent the orientation of the robot. The rotation matrix corresponding to a given RPY configuration is given by:

$$\begin{aligned} {}^W R_B(\varphi, \vartheta, \psi) &= R_z(\psi) R_y(\vartheta) R_x(\varphi) \\ &= \begin{bmatrix} c_\psi c_\vartheta & c_\psi s_\vartheta s_\varphi - s_\psi c_\varphi & c_\psi s_\vartheta c_\varphi + s_\psi s_\varphi \\ s_\psi c_\vartheta & s_\psi s_\vartheta s_\varphi + c_\psi c_\varphi & s_\psi s_\vartheta c_\varphi - c_\psi s_\varphi \\ -s_\vartheta & c_\vartheta s_\varphi & c_\vartheta c_\varphi \end{bmatrix}. \end{aligned} \tag{2.1}$$

It is also immediate to verify that the inverse transformation is given by:

$$\vartheta = \operatorname{atan2}\left(-r_{31}, \pm\sqrt{r_{32}^2 + r_{33}^2}\right) \tag{2.2a}$$

$$\varphi = \operatorname{atan2}(r_{32}, r_{33}) \tag{2.2b}$$

$$\psi = \operatorname{atan2}(r_{21}, r_{11}), \tag{2.2c}$$

where r_{ij} indicates the component on the i-th row and j-th column of ${}^W R_B$. The transformation has a singularity of representation for $\cos(\vartheta) = 0$.

As known, the derivative of the rotation matrix is given by:

$${}^W \dot{R}_B = {}^W R_B \, {}^B\Omega_{BW}, \tag{2.3}$$

where ${}^B\Omega_{BW}$ is the skew-symmetric matrix built with the components of ${}^B\omega_{BW}$. More specifically, assuming that

$${}^B\omega_{BW} = \begin{bmatrix} p \\ q \\ r \end{bmatrix}, \tag{2.4}$$

we have

$${}^B\Omega_{BW} = \begin{bmatrix} 0 & -r & q \\ r & 0 & -p \\ -q & p & 0 \end{bmatrix}.$$

The map that relates ${}^B\omega_{BW}$ to the corresponding skew-symmetric matrix ${}^B\Omega_{BW}$ is often called hat-map. Its inverse typically takes the name of vee-map. The angular velocity of the robot is also related to the vector of roll, pitch and yaw angles derivatives, indeed

$$
{}^{B}\omega_{BW} = R_x(\varphi)^T \begin{pmatrix} \dot{\varphi} \\ 0 \\ 0 \end{pmatrix} + R_x(\varphi)^T R_y(\vartheta)^T \begin{pmatrix} 0 \\ \dot{\vartheta} \\ 0 \end{pmatrix} + R_x(\varphi)^T R_y(\vartheta)^T R_z(\psi)^T \begin{pmatrix} 0 \\ 0 \\ \dot{\psi} \end{pmatrix},
$$

then

$$
{}^{B}\omega_{BW} = T(\vartheta, \varphi) \begin{pmatrix} \dot{\varphi} \\ \dot{\vartheta} \\ \dot{\psi} \end{pmatrix},
$$

where

$$
T(\vartheta, \varphi) = \begin{pmatrix} 1 & 0 & -s_\vartheta \\ 0 & c_\varphi & c_\vartheta s_\varphi \\ 0 & -s_\varphi & c_\vartheta c_\varphi \end{pmatrix}.
$$

Since

$$
\det(T(\vartheta, \varphi)) = \cos(\vartheta),
$$

the above relation is invertible out of the singularities of representation and its inverse is

$$
\begin{pmatrix} \dot{\varphi} \\ \dot{\vartheta} \\ \dot{\psi} \end{pmatrix} = \begin{pmatrix} 1 & s_\varphi t_\vartheta & c_\varphi t_\vartheta \\ 0 & c_\varphi & -s_\varphi \\ 0 & s_\varphi / c_\vartheta & c_\varphi / c_\vartheta \end{pmatrix} {}^{B}\omega_{BW}. \tag{2.5}
$$

As it is well known (see e.g. [17]), each of the four propellers produces a force of modulus F_i along z_B and a torque of modulus M_i about z_B. Both are proportional to the square of the motor rotational speed ω_i:

$$
\begin{aligned} F_i &= k_F \, \omega_i^2 \\ M_i &= k_M \, \omega_i^2. \end{aligned}
$$

k_F and k_M are the thrust and drug factors respectively. They are both positive and their value depends on the shape of the propellers.

Motor dynamics are relatively fast and hence negligible with respect to the rigid body dynamics, then we can consider the rotor velocities ω_i as the control inputs of the system. We also introduce the following input transformation:

$$
u = \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{pmatrix} = \begin{pmatrix} k_F & k_F & k_F & k_F \\ 0 & k_F l & 0 & -k_F l \\ -k_F l & 0 & k_F l & 0 \\ k_M & -k_M & k_M & -k_M \end{pmatrix} \begin{pmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{pmatrix} = A\,\tilde{u}, \tag{2.6}
$$

where l is the distance between the rotor axes of rotation and the geometric center of the quadrotor. The matrix A has always maximum rank, then the transformation is also invertible. The transformed input vector comprises the total thrust force u_1 along z_B and the torques u_2, u_3 and u_4 around x_B, y_B and z_B respectively.

Since we want the robot to perform the grasping in the minimum time, it would be reasonable to consider the aerodynamic effects, which become consistent when the robot reaches high velocities. Dynamic models and controllers dealing with aerodynamic disturbances have been developed in e.g. [18, 19, 2, 20]. However these effects are rather complex to manage and they are typically only modeled in an empirical fashion. For these reasons we will neglect any aerodynamic effect, entrusting the control action for their compensation.

Having said that, the forces acting on the system are the gravity force directed along $z_W = e_3$ (being e_i the i-th column of the identity matrix) and the total thrust force generated by the propellers and directed along z_B. We also assume that the robot center of mass is coincident with its geometric center, where the total thrust is applied. With these assumptions, the translational dynamics of the system is given by the following Newton's equation:

$$m\ddot{r}_B = -mge_3 + u_1 z_B, \tag{2.7}$$

where m is the total mass of the robot.

The angular acceleration is governed by the Euler's equation. Since the gravity force is applied to the robot center of mass, the only torque acting on the system is the one generated by the propellers, hence

$$J\,{}^B\dot{\omega}_{BW} + {}^B\omega_{BW} \times J\,{}^B\omega_{BW} = \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix}, \tag{2.8}$$

where J is the constant inertia matrix referenced to the center of mass of the robot and expressed in the frame B. If the robot is assumed to have a perfect cylindrical symmetry with respect to the axis z_B, the inertia matrix is also diagonal and two of its eigenvalues, namely J_{xx} and J_{yy}, are equal. This makes it possible, if desired, to neglect the gyroscopic term ${}^B\omega_{BW} \times J\,{}^B\omega_{BW}$, without introducing large modeling errors (see for example [1]).

Chapter 3

Problem formulation

In this section we provide a mathematical formulation of the aerial grasping problem. The quadrotor is initially in the given state χ_i. It has to move from this state, to grasp the target and to reach a final assigned state χ_f, in a minimum time. The trajectory of the target object is assumed to be fully known a-priori. Moreover we do not optimize the point along the target trajectory in which the grasping is performed. Instead we only specify the position and velocity that the target will have when the grasping takes place. If the target is being transported by an active carrier (such as a mobile robot or a human driven truck, and so on) we can imagine that the quadrotor agrees with the carrier on a meeting point in a preliminary phase. Since the instant in which the quadrotor will actually reach the meeting point is a result of the optimization, we also assume that the carrier is able to adapt its motion to the above result in such a way that it reaches the meeting point together with the quadrotor.

The problem can be formulated as that of constructing a feasible trajectory

$$\chi(t)^*: [t_i, t_f] \to X$$

for the state of the quadrotor, such that the robot fulfills the given task.

To satisfy the given initial and final conditions we will impose

$$\chi(t_i)^* = \chi_i$$
$$\chi(t_f)^* = \chi_f.$$

As we already said, the quadrotor gripper can grasp the target only if some geometric and kinematic constraints are satisfied. Assuming that the target is grasped at the time instant t_g, we will then impose

$$\chi(t_g)^* \in X_g,$$

where X_g is the set of states for which the grasping is admissible, i.e. such that:

- the position of the center of grasping is equal to that of the target;
- the velocity of the center of grasping with respect to the target is zero;

To mathematically specify these constraints we need to introduce two more reference frames (see again fig. 2.1). The frame G attached to the gripper has a fixed position ${}^{B}r_{G}$ and an orientation ${}^{B}R_{G}$ with respect to the body frame, then:

$$r_G = r_B + {}^{W}R_B\,{}^{B}r_G \tag{3.1a}$$

$$\dot{r}_G = \dot{r}_B + {}^{W}R_B \; {}^{B}\omega_{BW} \times {}^{B}r_G \; . \tag{3.1b}$$

The frame T attached to the target has a position r_T and an orientation ${}^{W}R_T$ with respect to the world inertial frame. We assume that the target has a spherical symmetry so that we can neglect its orientation. The above conditions then translate to the following set of equations:

$$r_G(t_g) = r_B(t_g) + {}^{W}R_B(t_g)\,{}^{B}r_G = r_T(t_g) \tag{3.2}$$

$$\dot{r}_G(t_g) = \dot{r}_B(t_g) + {}^{W}R_B(t_g)\;{}^{B}\omega_{BW}(t_g) \times {}^{B}r_G = \dot{r}_T(t_g) \, . \tag{3.3}$$

From eq. (3.2) we obtain

$${}^{W}R_B(t_g)\,{}^{B}r_G = r_T(t_g) - r_B(t_g) \, ,$$

and substituting this in eq. (3.3) provides

$$\dot{r}_B(t_g) = \dot{r}_T(t_g) + {}^{W}R_B(t_g)\,{}^{B}\omega_{BW}(t_g) \times (r_B(t_g) - r_T(t_g)) \, , \tag{3.4}$$

which simply means that the robot must instantaneously rotate around the target for $t = t_g$. In practice the gripper will require a finite time, namely $T_g^{min} > 0$, to actually grab the object. It is therefore much more realistic to plan a trajectory during which the robot rotates around the target for a possibly small but non infinitesimal time, namely $T_g \geq T_g^{min}$. The grasping constraint then becomes:

$$\chi(t)^* \in X_g \quad \forall t \in \; t_g, t_g + T_g^{min}$$

The robot is finally subject to dynamic constraints due to the limited authority of its actuators. These constraints act on the minimum and maximum values of propeller rotational speed

$$\omega_i \in [\underline{\omega}, \overline{\omega}] \, , \tag{3.5}$$

defining an hypercube in the space of admissible motor velocities

$$\tilde{u} \in \tilde{U} \subset \mathbb{R}^4 = [\underline{\omega}, \overline{\omega}] \times [\underline{\omega}, \overline{\omega}] \times [\underline{\omega}, \overline{\omega}] \times [\underline{\omega}, \overline{\omega}] \, .$$

This hypercube is then mapped into the space of admissible transformed inputs through eq. (2.6)

$$u \in U = \quad u \in \mathbb{R}^4 : u = A\,\tilde{u} \text{ where } \tilde{u} \in \tilde{U} \quad .$$

Note that, since the propellers can not change their direction of rotation during the motion, we have to assume $\underline{\omega} \geq 0$ and obviously also $u_1 \geq 0$.

The instant of grasping t_g is not fixed in principle by the problem and can be considered as a variable to be optimized. The same is true for the final time t_f. Since we want to do the grasping in a minimum time we end up with the following optimization problem:

$$\begin{aligned}
&\min_{\{\chi(t)^*, t_g, t_f\}} \; t_f \\
\text{s.t. } & \chi(t_i)^* = \chi_i \\
& \chi(t_f)^* = \chi_f \\
& \chi(t)^* \in X_g \quad \forall t \in \; t_g, t_g + T_g^{min} \\
& \chi(t)^* \in X \\
& u \in U \\
& t_g > 0 \\
& t_f > t_g,
\end{aligned}$$

Chapter 4

Differential flatness

Now we demonstrate an important property of the quadrotor system: the differential flatness. This property was first introduced in the early 1990s (see [21]) and makes it possible to express the complete state of a system as well as the value of its control inputs, as functions of a certain number of its outputs (which take the name of flat outputs) and of a finite number of their time derivatives. Differential flatness has been widely exploited in the literature of motion planning because it allows to move the trajectory generation problem to the space of the flat outputs in which it is easier, in general, to deal with possible geometric constraints. In [22] a comparison is proposed between differential flatness and dynamic feedback linearization in motion planning. Indeed the two properties are equivalent in the sense that any feedback linearizable system is also differentially flat and vice versa as it is demonstrated in [23]. Moreover the feedback linearizing outputs and the flat outputs of a system coincide.

Thanks to this analogy it is well known in the literature that the quadrotor is differentially flat and its flat outputs are the three components of the position vector and the yaw angle (see, e.g., [24]). We can then define a flat output vector as:

$$\sigma = \begin{bmatrix} x & y & z & \psi \end{bmatrix}^T .$$

Trajectory planning algorithms for the quadrotor based on the flatness property have already been proposed in other papers such as [25, 9, 10, 26, 11]. The following description retrace the mathematical passages of [11]. With respect to that work we adopted a more widely used definition of the roll, pitch and yaw angles, defined in the sequence X-Y'-Z" as commented in chapter 2. We also corrected the computation of the last components of the angular velocity and angular acceleration.

4.1 Flat transformation

In this section we assume that a sufficiently smooth[1] trajectory $\sigma(t)$ is defined in the space of flat outputs and we will show that the state of the system and the control inputs can be written in terms of $\sigma(t)$ and its time derivatives. In the following we will drop the notation of time dependencies for the sake of compactness.

The position and velocity of the quadrotor center of mass are simply the first three terms of σ and $\dot{\sigma}$. Similarly we can compute the acceleration $\ddot{r}_B$, the jerk $\dot{a}_B$ and the snap $\ddot{a}_B$. From the fourth components of σ, $\dot{\sigma}$ and $\ddot{\sigma}$ we can instead obtain the yaw angle ψ and its derivatives $\dot{\psi}$ and $\ddot{\psi}$.

Defining

$$t := \ddot{r}_B + ge_3,$$

and considering that u_1 is always positive, from eq. (2.7) we obtain the direction of the robot vertical axis

$$z_B = \frac{t}{t}, \tag{4.1}$$

and also the total thrust

$$u_1 = m \quad t \quad . \tag{4.2}$$

Given the yaw angle $\psi = \sigma_4$ we can define the vector:

$$y_C := R_z(\psi)\, e_2 = \quad -\sin\psi \quad \cos\psi \quad 0 \quad ^T$$

and from eq. (2.1) it is easy to verify that:

$$y_C \times z_B = \cos\varphi x_B.$$

Provided that $\cos\varphi > 0$, we are then able to compute x_B as

$$x_B = \frac{y_C \times z_B}{y_C \times z_B} := \frac{\tilde{x}_B}{\tilde{x}_B}.$$

The last axis of the frame B is simply given by

$$y_B = z_B \times x_B,$$

and the rotation matrix describing the full 3D orientation of the robot is

$$^W R_B = \quad x_B \quad y_B \quad z_B \quad ^T.$$

Now we take the first derivative of eq. (2.7):

$$m\dot{a}_B = \dot{u}_1 z_B + \omega_{BW} \times u_1 z_B. \tag{4.3}$$

[1]The exact meaning of this will be clear later.

Projecting this equation along z_B we obtain:

$$\dot{u}_1 = m z_B^T \dot{a}_B. \tag{4.4}$$

We can now substitute $\dot{u}_1$ and u_1 back in eq. (4.3) getting

$$\begin{aligned} \omega_{BW} \times z_B &= \frac{1}{t} \quad \dot{a}_B - \quad z_B^T \dot{a}_B \quad z_B \\ &= \frac{1}{t} \quad I - z_B z_B^T \quad \dot{a}_B := h. \end{aligned} \tag{4.5}$$

We assumed in eq. (2.4) that ω_{BW} has components p, q, and r in the body frame, i.e.

$$\omega_{BW} = p x_B + q y_B + r z_B,$$

then

$$h = (p x_B + q y_B + r z_B) \times z_B = -p y_B + q x_B,$$

and hence

$$\begin{aligned} p &= -h^T y_B, \\ q &= h^T x_B. \end{aligned} \tag{4.6}$$

The third component r is found by considering that from eq. (2.3)

$$r = y_B^T \dot{x}_B,$$

and

$$\dot{x}_B = \quad I - x_B x_B^T \quad \frac{\dot{\tilde{x}}_B}{\tilde{x}_B},$$

then, since $y_B^T x_B = 0$, we can conclude that

$$\begin{aligned} r = y_B^T \frac{\dot{\tilde{x}}_B}{\tilde{x}_B} &= \frac{y_B^T}{\tilde{x}_B} \quad -x_C \times \dot{\psi} z_B + y_C \times h \\ &= \frac{1}{\tilde{x}_B} \quad x_C^T \quad y_B \times \dot{\psi} z_B \quad - y_C^T (y_B \times h) \\ &= \frac{1}{\tilde{x}_B} \quad x_C^T x_B \dot{\psi} + y_C^T z_B q \quad . \end{aligned} \tag{4.7}$$

Once we know the values of p, q and r we are able to compute ω_{BW} as:

$$\omega_{BW} = {}^W R_B \quad \begin{matrix} p \\ q \\ r \end{matrix} \quad .$$

To calculate the angular acceleration ${}^B\dot{\omega}_{BW}$ we operate in the same way. By deriving eq. (4.3) with respect to time we obtain:

$$m \ddot{a}_B = \ddot{u}_1 z_B + 2\omega_{BW} \times \dot{u}_1 z_B + \dot{\omega}_{BW} \times u_1 z_B + \omega_{BW} \times (\omega_{BW} \times u_1 z_B). \tag{4.8}$$

Projecting this equation along z_B we have:

$$m z_B^T \ddot{a}_B = \ddot{u}_1 + z_B^T [\omega_{BW} \times (\omega_{BW} \times u_1 z_B)],$$

from which we can isolate $\ddot{u}_1$:

$$\ddot{u}_1 = m z_B^T \quad \ddot{a}_B - \omega_{BW} \times \quad \omega_{BW} \times \frac{u_1}{m} z_B \quad . \tag{4.9}$$

Substituting $\ddot{u}_1$ back in eq. (4.8) and putting

$$\delta := \ddot{a}_B - \omega_{BW} \times \quad \omega_{BW} \times \frac{u_1}{m} z_B \quad = \ddot{a}_B - \omega_{BW} \times \quad t \quad h, \tag{4.10}$$

we obtain

$$\begin{aligned} \dot{\omega}_{BW} \times z_B &= \frac{1}{u_1} \quad m\delta - m \quad z_B^T \delta \quad z_B - 2\omega_{BW} \times \dot{u}_1 z_B \\ &= \frac{1}{t} \quad I - z_B z_B^T \quad \delta - 2 \quad z_B^T \dot{a}_B \quad h \quad := l. \end{aligned} \tag{4.11}$$

Now assuming that ${}^B\dot{\omega}_{BW} = \quad m \quad n \quad o \quad ^T$, and hence

$$\dot{\omega}_{BW} = m x_B + n y_B + o z_B,$$

it is easy to verify that

$$\begin{aligned} m &= -l^T y_B \\ n &= l^T x_B. \end{aligned} \tag{4.12}$$

The third component o is found by taking the derivative of eq. (4.7)

$$\begin{aligned} o = &\frac{1}{\tilde{x}_B} \quad y_C^T x_B \dot{\psi}^2 + x_C^T \dot{x}_B \dot{\psi} + x_C^T x_B \ddot{\psi} - x_C^T z_B q \dot{\psi} + y_C^T h r + y_C^T z_B n \\ &- \frac{\tilde{x}_B^T \dot{\tilde{x}}_B}{\tilde{x}_B \;^3} \quad x_C^T x_B \dot{\psi} + y_C^T z_B q \\ = &\frac{1}{\tilde{x}_B} \quad x_C^T \dot{x}_B \dot{\psi} + x_C^T x_B \ddot{\psi} - x_C^T z_B q \dot{\psi} + y_C^T h r + y_C^T z_B n - x_B^T \dot{\tilde{x}}_B r \quad . \end{aligned}$$

Since

$$\begin{aligned} y_C^T h &= -\omega_{BW}^T (y_C \times z_B) = -\omega_{BW}^T \tilde{x}_B = - \quad \tilde{x}_B \quad p \\ x_B^T \dot{\tilde{x}}_B &= x_C^T \quad x_B \times z_B \dot{\psi} \quad - y_C^T (x_B \times h) = y_C^T z_B p - x_C^T y_B \dot{\psi}, \end{aligned}$$

then

$$o = \frac{1}{\tilde{x}_B} \quad x_C^T \dot{x}_B \dot{\psi} + x_C^T x_B \ddot{\psi} + x_C^T y_B r \dot{\psi} - x_C^T z_B q \dot{\psi} + y_C^T z_B (n - pr) \quad - pq.$$

Moreover from eq. (2.3) we obtain

$$x_B^T \dot{x}_B = 0$$
$$y_B^T \dot{x}_B = r$$
$$z_B^T \dot{x}_B = -q,$$

then

$$x_C^T \dot{x}_B = x_C^T (y_B r - z_B q)$$

and we conclude that

$$o = \frac{1}{\tilde{x}_B} \; 2 \; x_C^T y_B r - x_C^T z_B q \;\; \dot{\psi} + x_C^T x_B \ddot{\psi} + y_C^T z_B (n - pr) \;\; - pq. \qquad (4.13)$$

Finally from eq. (2.8) we compute the remaining inputs (u_2, u_3, u_4).

In order to guarantee the continuity of the state we will require that σ_1, σ_2 and σ_3 have continuous derivatives at least up to the third order, while for σ_4 it is sufficient to require continuity up to the first order. If we also wanted the continuity of the inputs, then we would also require $\sigma_1^{(4)}$, $\sigma_2^{(4)}$, $\sigma_3^{(4)}$ and $\sigma_4^{(2)}$ to be continuous.

4.2 Reachability analysis

Differential flatness is a powerful property in control theory being strictly related to the concept of controllability and dynamic feedback linearization. Indeed any flat system is also controllable while the converse is not true (see [21]). Nevertheless in the study of controllability in general the presence of input limitations is not taken into account.

In this section we analyze how the presence of input boundaries affects the controllability of the system. We show that the quadrotor can always move safely (i.e. satisfying actuators constraints) from any initial value of the flat outputs to any other one, provided that the derivatives of the flat output trajectory are small enough. In order to prove this we study the behave of the inputs when the derivatives of the flat outputs tend to zero in norm.

We start from the thrust input. From eq. (4.2) we can write

$$u_1 = m \;\; t \;\; = m \;\; \ddot{r}_B + g e_3 \;\; \leq m(\;\; \ddot{r}_B \;\; + g),$$

then

$$\lim_{\|\ddot{r}_B\| \to 0} u_1 = mg$$

and, provided that $u_1 = mg$ is a feasible thrust, it is always possible to properly limit u_1 by operating on $\ddot{r}_B$.

Now we consider the torque inputs. From eq. (2.8) we have

$$
\begin{aligned}
u_{2:4} &= J\ {}^{B}\dot{\omega}_{BW} + {}^{B}\omega_{BW} \times J\ {}^{B}\omega_{BW} \\
&\le J\ {}^{B}\dot{\omega}_{BW} + {}^{B}\omega_{BW} \times J\ {}^{B}\omega_{BW} \\
&\le J\ {}^{B}\dot{\omega}_{BW} + {}^{B}\omega_{BW}\ \ J\ {}^{B}\omega_{BW} \\
&= J\ \ {}^{B}\dot{\omega}_{BW} + {}^{B}\omega_{BW}\ {}^{2}
\end{aligned}
$$

then $u_{2:4}$ tends to zero for ${}^{B}\dot{\omega}_{BW}$ and ${}^{B}\omega_{BW}$ going to zero.

From eq. (4.6) and eq. (4.7) we get

$$
\begin{aligned}
{}^{B}\omega_{BW}\ {}^{2} &= h^{T}y_{B}\ {}^{2} + h^{T}x_{B}\ {}^{2} + \frac{x_{C}^{T}x_{B}\dot{\psi} + y_{C}^{T}z_{B}h^{T}x_{B}}{\tilde{x}_{B}}\ {}^{2} \\
&\le 2\ h\ {}^{2} + \frac{\dot{\psi}\ {}^{2} + h\ {}^{2}}{\tilde{x}_{B}\ {}^{2}}.
\end{aligned}
$$

It is easy to see that if $\ddot{r}_{B}$ tends to zero then z_{B} tends to z_{W} and becomes orthogonal to y_{C}. As a consequence $\tilde{x}_{B}$ becomes close to 1 and we can conclude that ${}^{B}\omega_{BW}$ tends to zero if h and $\dot{\psi}$ do. From eq. (4.5)

$$
h = \frac{1}{t}\ \ I - z_{B}z_{B}^{T}\ \dot{a}_{B} \le \frac{\dot{a}_{B}}{t}.
$$

For $\ddot{r}_{B}$ g, $t \approx g$ and h tends to zero for $\dot{a}_{B}$ going to zero.

From eq. (4.12) and eq. (4.13) we get

$$
\begin{aligned}
{}^{B}\dot{\omega}_{BW}\ {}^{2} &= l^{T}y_{B}\ {}^{2} + l^{T}x_{B}\ {}^{2} \\
&+ \frac{2\ x_{C}^{T}y_{B}r - x_{C}^{T}z_{B}q\ \dot{\psi} + x_{C}^{T}x_{B}\ddot{\psi} + y_{C}^{T}z_{B}\ \ l^{T}x_{B} - pr}{\tilde{x}_{B}} - pq\ {}^{2} \\
&\le 2\ l\ {}^{2} + \frac{2\ |r|^{2} + |q|^{2}\ \ \dot{\psi}\ {}^{2} + \ddot{\psi}\ {}^{2} + l\ {}^{2} + |pr|^{2}}{\tilde{x}_{B}\ {}^{2}} + |pq|^{2}.
\end{aligned}
$$

As we already commented $\tilde{x}_{B}$ tends to 1 for $\ddot{r}_{B}$ going to zero, then ${}^{B}\dot{\omega}_{BW}$ tends to zero if ${}^{B}\omega_{BW}$, l and $\ddot{\psi}$ do. From eq. (4.11)

$$
\begin{aligned}
l &= \frac{1}{t}\ \ I - z_{B}z_{B}^{T}\ \delta - 2\ z_{B}^{T}\dot{a}_{B}\ h \\
&\le \frac{1}{t}(\ \delta\ + 2\ \dot{a}_{B}\ \ h\ ) = \frac{\delta}{t} + 2\ \frac{\dot{a}_{B}}{t}\ {}^{2}.
\end{aligned}
$$

We have already demonstrated that the last term goes to zero for $\ddot{r}_B$ and $\dot{a}_B$ going to zero. Moreover from eq. (4.10) we conclude that

$$\begin{aligned} \frac{\delta}{t} &= \frac{1}{t} \left(\ddot{a}_B - \omega_{BW} \times \; t \;\; h \right) \\ &\leq \frac{\ddot{a}_B}{t} + \;\; \omega_{BW} \;\; h \; = \frac{\ddot{a}_B}{t} + \;\; {}^{B}\omega_{BW} \;\; h \; . \end{aligned}$$

Since $t \approx g$ for $\ddot{r}_B$ $\;\; g$, then the first term tends to zero for $\ddot{a}_B$ going to zero. Finally, for previous computations, also the second term goes to zero when $\dot{\psi}$ and $\dot{a}_B$ go to zero.

We have demonstrated that if the derivatives of the flat outputs tend to zero in norm, the robot tends to remain in an almost hovering condition and the input vector tends to

$$u = \begin{matrix} mg \\ 0 \\ 0 \\ 0 \end{matrix} := u_{min}, \tag{4.14}$$

thus, provided that this value is admissible, the trajectory can be performed. Note that obviously it is not necessary to require that the velocity of the center of mass $\dot{r}_B$ tends to zero.

Now assume that we have planned a rest-to-rest trajectory $\sigma(t)$ with initial and final values of the derivatives of the flat output equal to zero, and that this trajectory violates the input constraints. If we scale the trajectory in time such that

$$\hat{\sigma}(t) = \sigma \;\; \frac{t}{\lambda}$$

then the k-th derivative of the flat output scales by a factor λ^k, still maintaining the boundary conditions satisfied (as they were both zero). While the geometric path followed by the quadrotor center of mass remains the same, the orientation of the robot gets closer to hovering and the input vector tends to its minimum value u_{min}. As a consequence the trajectory will eventually become feasible for a value of λ sufficiently large. This property has been exploited in [11] where a post-processing time scaling of the trajectory is used to satisfy input constraints assuming zero initial and final values of the flat output derivatives (i.e. only rest-to-rest motions are considered).

If the initial and/or final states of the robot are not hovering states, then the corresponding values of the flat output derivatives are not zero and the previous reasoning is not valid anymore. Indeed a simple uniform scaling of all the derivatives would also modify their values at the initial and final points, thus affecting the satisfaction of the boundary conditions.

Nevertheless it is possible to provide a qualitative demonstration that the robot can actually move from any state to any other, provided that the input value u_{min} is an interior point of the set of admissible inputs, i.e. provided that there exists an open set centered at u_{min} that is contained in U.

To prove this let's first demonstrate that it is always possible to reach from any initial state, the zero state, i.e. the state in which the position, velocity and angular velocity are zero and the orientation of the robot is described by the identity matrix.

Thanks to the cascade structure of the system, the rotational dynamics influences the translational one (since it determines the direction of the thrust), but the converse is not true. We can then operate in two steps: first we regulate the rotational part of the dynamics to the desired value, then we also regulate the translational part while maintaining the former unaltered.

Since the rotational dynamics is fully actuated and, by assumption, both positive and negative values (possibly close to zero) of the torques are admissible, the robot can always control its attitude so that it is parallel to the ground and its angular velocity is zero. In this condition, since u_{min} is an interior point of U, and given the controllability of the rotational dynamics, the robot can compensate the gravity and make use of a residual thrust (or of a not completely compensated gravity) to apply forces in any direction, remaining arbitrarily close to the hovering condition. The translational dynamics then becomes a fully actuated double integrator with both positive and negative admissible inputs and the robot can always reach the zero state in a finite time (see [27]).

Now we want to demonstrate that, starting from the zero state, the robot can reach any other state. Again, since the rotational dynamics is fully actuated and with the same assumptions as before, there always exists a trajectory for the inputs that regulates the rotational part of the state to the desired value in a arbitrarily long but finite time T. Lets call this trajectory $u(t)^*$. Now assume that, starting from the zero state and applying $u^*(t)$ for the time T, the translational part of the state would reach the value

$$\chi_{t,f} = \begin{matrix} r_{B,f} \\ \dot{r}_{B,f} \end{matrix} ,$$

and that the desired value of the translational part of the final state is instead

$$\chi_{t,d} = \begin{matrix} r_{B,d} \\ \dot{r}_{B,d} \end{matrix} .$$

We can compute a launch state whose translational part is

$$\chi_{t,l} = \chi_{t,d} - \chi_{t,f}$$

and whose rotational part has an identity rotation matrix and zero angular velocity. This is the state from which, applying the input $u(t)^*$, we would reach the desired final state $\chi_{t,d}$ after a time T. Again, since u_{min} is an interior point of U, the robot can alway reach the launching state from any initial state and then, applying the input $u(t)^*$, it can evolve to the desired final state $\chi_{t,d}$.

To reach the launching state we can adopt a similar strategy as before in the sense that:

- we compute an input trajectory that brings the robot from the zero state to a state in which the velocity, orientation and angular velocity are the same as in $\chi_{t,l}$, but the position is in general different;
- we compute a launch state in which all the velocities are zero and the orientation is parallel to the ground;
- we go to the launch state via a rest-to-rest motion (always feasible for previous reasonings) and then we apply the previously derived inputs until we reach the launching state $\chi_{t,l}$.

Since the robot can reach the zero state from any initial state and vice versa, then it can also move between two arbitrary states, possibly passing through the zero state.

4.3 Inverse flat transformation

Now we deal with the inverse problem of the one studied of section 4.1: we are given the state of the robot in terms of r_B, $\dot{r}_B$, ${}^W R_B$ and ${}^B\omega_{BW}$ and possibly the input vector u and we want to compute the value of the flat outputs and their derivatives.

The position vector and its derivative are simply contained in the state and can immediately be extracted from it.

Using the equations reported at the beginning of chapter 2 we can conpute the roll, pitch and yaw angles and their derivatives from the state components ${}^W R_B$ and ${}^B\omega_{BW}$.

The linear acceleration is given by eq. (2.7).

$$\ddot{r}_B = \frac{u_1}{m} z_B - g e_3$$

If the thrust is fixed then $\ddot{r}_B$ is univoquelly defined, otherwise any value satisfying the equation

$$I - z_B z_B^T \;\; \ddot{r}_B = -g \;\; I - z_B z_B^T \;\; e_3$$

is valid. In general we can split $\ddot{r}_B$ into its components orthogonal and parallel to the local axis z_B

$$\ddot{r}_B = -g \;\; I - z_B z_B^T \;\; e_3 + \;\; u_1 - g z_B^T e_3 \;\; z_B,$$

and it is clear that the latter component can be chosen at will (inside an admissible interval) since it is controlled by the total thrust input u_1.

Once we know $\ddot{r}_B$, we define

$$t = \ddot{r}_B + g e_3,$$

and we compute the component of $\dot{a}_B$ orthogonal to z_B from eq. (4.5) solving

$$I - z_B z_B^T \;\; \dot{a}_B = \;\; t \;\; {}^W R_B \, {}^B\omega_{BW} \times z_B.$$

The minimum norm solution is

$$\dot{a}_B = \;\; t \;\; {}^W R_B \, {}^B\omega_{BW} \times z_B. \qquad (4.15)$$

If $\dot{u}_1$ is fixed, then we must add to $\dot{a}_B$ a component along the z_B axis such that eq. (4.4) is satisfied, i.e.

$$\dot{a}_B = \;\; t \;\; {}^W R_B \, {}^B\omega_{BW} \times z_B + \frac{\dot{u}_1}{m} z_B$$

Assuming that the torque inputs u_2, u_3 and u_4 are known we can compute the angular acceleration in the body frame from eq. (2.8)

$${}^B\dot{\omega}_{BW} = J^{-1} \;\; \begin{matrix} u_2 \\ u_3 \\ u_4 \end{matrix} \;\; - {}^B\omega_{BW} \times J \, {}^B\omega_{BW}$$

and using eq. (4.13) we compute $\ddot{\psi}$ as

$$\ddot{\psi} = \frac{(o + pq) \;\; \tilde{x}_B \;\; - \;\; 2 \;\; x_C^T y_B r - x_C^T z_B q \;\; \dot{\psi} + y_C^T z_B (n - pr)}{x_C^T x_B}$$

where n and o are the last two components of ${}^B\dot{\omega}_{BW}$.

We also compute the component of δ orthogonal to z_B by solving the system

$$I - z_B z_B^T \;\; \delta = \;\; t \;\; \dot{\omega}_{BW} + 2 \;\; z_B^T \dot{a}_B \;\; \omega_{BW} \;\; \times z_B.$$

derived from eq. (4.11). Again the minimum norm solution is:

$$\delta = \;\; t \;\; \dot{\omega}_{BW} + 2 \;\; z_B^T \dot{a}_B \;\; \omega_{BW} \;\; \times z_B.$$

If $\dot{u}_1$ is not fixed and we chose $\dot{a}_B$ according to eq. (4.15), then the above equations simplify to

$$I - z_B z_B^T \quad \delta = \dot{\omega}_{BW} \times \quad t \quad z_B,$$

and

$$\delta = \dot{\omega}_{BW} \times \quad t \quad z_B.$$

It is easy to demonstrate that if also $\ddot{u}_1$ is given, then in order to satisfy eq. (4.9) we have to add to δ a component along z_B, i.e.

$$\delta = \quad t \quad \dot{\omega}_{BW} + 2 \quad z_B^T \dot{a}_B \quad \omega_{BW} \quad \times z_B + \frac{\ddot{u}_1}{m} z_B.$$

Finally, inverting eq. (4.10), we obtain

$$\ddot{a}_{B,d} = \delta + \omega_{BW} \times (\omega_{BW} \times \quad t \quad z_B) .$$

Chapter 5

Planning based on Pontryagin's minimum principle

As we have already said in chapter 1, the problem of generating optimal trajectories for a quadrotor UAV has been addressed in [12, 3, 13] using Pontryagin's minimum principle. This optimal control theory principle has been formulated by Lev Semenovich Pontryagin in 1956 and defines a necessary, but not sufficient, condition for optimality of a system trajectory. Consider a general dynamic system described by a differential equation in the form:

$$\dot{\chi}(t) = f(\chi(t), u(t))$$

and assume that we want to find a trajectory for the state and the input

$$\begin{aligned} \chi(t)^*: &\quad [0, T] \to X \\ u(t)^*: &\quad [0, T] \to U \end{aligned}$$

that minimizes the cost function

$$V = h(\chi(T)^*) + \int_0^T g(\chi(t)^*, u(t)^*)\, dt,$$

subject to

$$\begin{aligned} \dot{\chi}(0)^* &= \chi_i \\ \dot{\chi}(T)^* &= \chi_f . \end{aligned}$$

Neglecting the time dependencies, we define the Hamiltonian of the system as

$$H(\chi, u, p) = g(\chi, u) + p^T f(\chi, u),$$

where the vector p is also called the costate vector and plays a similar role to the Lagrange multipliers.

The Pontryagin's principle states that if $u(t)^*$ is an optimal trajectory for the inputs and $\chi(t)^*$ is the corresponding optimal trajectory for the state, then the following conditions hold

$$\begin{aligned}\dot{\chi}(t)^* &= f(\chi(t)^*, u(t)^*)\\ \dot{\chi}(0)^* &= \chi_i\\ \dot{\chi}(T)^* &= \chi_f\\ \dot{p}(t) &= -\nabla_\chi H(\chi(t)^*, u(t)^*, p(t)).\end{aligned}$$

and for all $t \in [0, T]$

$$\dot{u}(t)^* = \underset{u \in U}{\operatorname{argmin}} \{H(\chi(t)^*, u, p(t))\}.$$

Moreover if the total time T is not fixed by the problem, the following condition also holds true

$$H(\chi(t)^*, u(t)^*, p(t)) \equiv 0.$$

For a detaild explanation of the Pontryagin's principle, refer to e.g. [27].

We tried to apply the Pontryagin's minimum principle to a simplified two-dimensional version of the grasping problem in which we assume that the robot and the target move in the same vertical plane and that the yaw angle of the robot is fixed. With this assumptions we can consider a simpler planar model of the quadrotor as it has been done by the authors of [3, 13]. We have taken the inspiration for the following description from these articles but, in contrast with them, we assume to control the pitch/roll acceleration instead of the corresponding velocities.

In the next sections we describe the simplified 2D model of the quadrotor and the two dimensional version of the constraints introduced in chapter 3. We then apply the Pontryagin's minimum principle to the resulting optimization problem.

5.1 Two dimensional model of the quadrotor

In this section we derive a two dimensional model of the quadrotor dynamics. To this purpose we assume that the gyroscopic effect is negligible and that the yaw angle is constant and is such that the axis y_B is orthogonal to the plane in which the robot and the target are moving. Projecting eq. (2.7) on this plane and eq. (2.8) on the axis orthogonal to this plane, we easily obtain the following two dimensional model of the quadrotor:

$$
\begin{aligned}
\ddot{x} &= \frac{u_1}{m}\sin\vartheta \\
\ddot{z} &= \frac{u_1}{m}\cos\vartheta - g \\
\ddot{\vartheta} &= \frac{u_2}{J_{xx}}
\end{aligned}
$$

with state vector:

$$\chi = \quad x \quad \dot{x} \quad z \quad \dot{z} \quad \vartheta \quad \dot{\vartheta} \quad ^T$$

and input vector:

$$u = \quad u_1 \quad u_2 \quad ^T$$

The input constraints specified by the problem act on the single thrusts generated by each propeller, that are coupled in eq. (2.6) to generate the total thrust and the torque around y_B. Nevertheless we can assume that it is possible to determine conservative boundaries on the thrust and torque that always result in admissible values of the single propeller rotational speeds. Having said that we assume

$$
\begin{aligned}
u_1 &\in \underline{u_1}, \overline{u_1} \\
u_2 &\in [-\overline{\alpha}, \overline{\alpha}] .
\end{aligned}
$$

We try to follow the same strategy adopted in [3]. In particular we try to reduce the number of parameters by using a scaled version of the above dynamic model. Given the state transformation

$$
\begin{aligned}
\hat{x} &= \frac{\overline{\alpha}}{J_{xx}g}x \\
\hat{z} &= \frac{\overline{\alpha}}{J_{xx}g}z,
\end{aligned}
$$

the input transformation

$$
\begin{aligned}
u_T &= \frac{u_1}{mg} \in \underline{u_T}, \overline{u_T} \\
u_R &= \frac{\alpha}{\overline{\alpha}} \in [-1, +1] ,
\end{aligned}
$$

and the transformed time

$$\hat{t} = \sqrt{\frac{\overline{\alpha}}{J_{xx}}}\,t,$$

we have

$$
\begin{aligned}
\ddot{\hat{x}} &= u_T \sin\vartheta \\
\ddot{\hat{z}} &= u_T \cos\vartheta - 1 \\
\ddot{\hat{\vartheta}} &= u_R
\end{aligned}
$$

where the derivatives are taken with respect to the transformed time.

Neglecting the hat notation the system state dynamics can be written as:

$$\dot{\chi} = \begin{bmatrix} \dot{x} \\ u_T \sin\vartheta \\ \dot{z} \\ u_T \cos\vartheta - 1 \\ \dot{\vartheta} \\ u_R \end{bmatrix} \tag{5.1}$$

The dimensionless model contains two model parameters, namely the lower and upper limit of the collective thrust input ($\underline{u_T}$ and $\overline{u_T}$).

5.2 Two dimensional constraints

The constraints for the grasping can also be easily transformed in their two dimensional version obtaining

$$\begin{bmatrix} x_G \\ z_G \end{bmatrix} = \begin{bmatrix} x \\ z \end{bmatrix} + \begin{bmatrix} \cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta \end{bmatrix} \begin{bmatrix} {}^B x_G \\ {}^B z_G \end{bmatrix} = \begin{bmatrix} x_T \\ z_T \end{bmatrix} \tag{5.2}$$

$$\begin{bmatrix} \dot{x}_G \\ \dot{z}_G \end{bmatrix} = \begin{bmatrix} \dot{x} \\ \dot{z} \end{bmatrix} + \begin{bmatrix} \cos\vartheta & -\sin\vartheta \\ \sin\vartheta & \cos\vartheta \end{bmatrix} \begin{bmatrix} 0 & -\dot{\vartheta} \\ \dot{\vartheta} & 0 \end{bmatrix} \begin{bmatrix} {}^B x_G \\ {}^B z_G \end{bmatrix} = \begin{bmatrix} \dot{x}_T \\ \dot{z}_T \end{bmatrix} \tag{5.3}$$

from which, as for eq. (3.4), it follows that

$$\begin{bmatrix} \dot{x}_G \\ \dot{z}_G \end{bmatrix} = \begin{bmatrix} \dot{x}_T \\ \dot{z}_T \end{bmatrix} + \begin{bmatrix} 0 & -\dot{\vartheta} \\ \dot{\vartheta} & 0 \end{bmatrix} \begin{bmatrix} x - x_T \\ z - z_T \end{bmatrix}.$$

In order to put the problem in the form described at the beginning of this chapter we assume that the final state has been fixed to any of the states that are admissible for the grasping, i.e. that satisfy eqs. (5.2) and (5.3).

5.3 Pontryagin's minimum principle

Since in our case we only want to minimize the total time needed to complete the trajectory we have

$$V = \int_0^T dt,$$

that is

$$h(\chi^*(T)) = 0$$
$$g(\chi(t)^*, u(t)^*) \equiv 1,$$

and then we define the Hamiltonian as

$$\begin{aligned} H(\chi, u, p) &= 1 + p^T f(\chi, u) \\ &= 1 + p_1 \dot{x} + p_2 u_T s_\vartheta + p_3 \dot{z} + p_4 (u_T c_\vartheta - 1) + p_5 \dot{\vartheta} + p_6 u_R \end{aligned} \tag{5.4}$$

and we impose

$$\dot{p} = -\nabla_\chi H(\chi^*, u^*, p). \tag{5.5}$$

Since the terminal time for the maneuver is not fixed, we also have to impose

$$H(\chi^*, u^*, p) \equiv 0. \tag{5.6}$$

Equation (5.5) brings to:

$$\begin{aligned} \dot{p}_1 &= 0 && \Rightarrow p_1 = c_1 \\ \dot{p}_2 &= -p_1 && \Rightarrow p_2 = c_2 - c_1 t \\ \dot{p}_3 &= 0 && \Rightarrow p_3 = c_3 \\ \dot{p}_4 &= -p_3 && \Rightarrow p_4 = c_4 - c_3 t. \end{aligned}$$

where the constants $c = (c_1, c_2, c_3, c_4)$ remain to be determined. Note that these constants actually correspond to the initial conditions of the costate variables

$$\begin{aligned} c_1 &= p_1(0) \\ c_2 &= p_2(0) \\ c_3 &= p_3(0) \\ c_4 &= p_4(0) \end{aligned}$$

The last two elements of the costate vector p are given by

$$\begin{aligned} \dot{p}_5 &= -p_2 u_T \cos\vartheta + p_4 u_T \sin\vartheta \\ \dot{p}_6 &= -p_5 \end{aligned}$$

and their initial conditions have to be determined too.

Since the control inputs do not appear in the same summand in the Hamiltonian, the lattert can be minimized separately for u_R and u_T.

Optimal control input u_R^*

For the rotational control input the minimization of eq. (5.4) brings to:

$$u_R^* = \underset{u_R \in [-1,+1]}{\operatorname{argmin}} \{p_6 u_R\} \tag{5.7}$$

If we define a switching function of u_R as

$$\Sigma_R = p_6, \tag{5.8}$$

then the resulting control law is

$$u_R^* = \begin{cases} +1 & \Sigma_R < 0 \\ u_{R,sing}^* & \Sigma_R = 0 \\ -1 & \Sigma_R > 0 \end{cases}. \tag{5.9}$$

If Σ_R is zero for a nontrivial interval of time, then eq. (5.7) is insufficient to determine u_R^*. In these intervals, which are called singular arcs, u_R^* is determined using the condition that Σ_R remains zero, which implies that also $\dot{\Sigma}_R$ and $\ddot{\Sigma}_R$ vanish, i.e.

$$\dot{\Sigma}_R = -p_5 = 0,$$

and

$$\begin{aligned} \ddot{\Sigma}_R &= p_2 u_T^* \cos\vartheta^* - p_4 u_T^* \sin\vartheta^* \\ &= (c_2 - c_1 t)\, u_T^* \cos\vartheta^* - (c_4 - c_3 t)\, u_T^* \sin\vartheta^* = 0. \end{aligned} \tag{5.10}$$

From eq. (5.10) we conclude that:

$$\vartheta^* = \arctan \frac{p_2}{p_4} = \arctan \frac{c_2 - c_1 t}{c_4 - c_3 t} \tag{5.11}$$

Differentiating eq. (5.11) twice w.r.t. time we obtain

$$\dot{\vartheta}^* = \frac{c_2 c_3 - c_1 c_4}{(c_2 - c_1 t)^2 + (c_4 - c_3 t)^2}$$

and

$$u_{R,sing}^* = \frac{2(c_2 c_3 - c_1 c_4)\,[(c_2 - c_1 t)\, c_1 + (c_4 - c_3 t)\, c_3]}{\left((c_2 - c_1 t)^2 + (c_4 - c_3 t)^2\right)^2}.$$

Optimal control input u_T^*

In order to compute the optimal thrust input u_T^* we have to solve the minimization problem:

$$u_T^* = \underset{u_T \in [\underline{u_T}, \overline{u_T}]}{\operatorname{argmin}} \{(p_2 \sin\vartheta^* + p_4 \cos\vartheta^*)\, u_T\}. \tag{5.12}$$

As for the rotational input, we define a switching function for the thrust input as:

$$\begin{aligned} \Sigma_T &= p_2 \sin\vartheta^* + p_4 \cos\vartheta^* \\ &= (c_2 - c_1 t) \sin\vartheta^* + (c_4 - c_3 t) \cos\vartheta^*. \end{aligned} \tag{5.13}$$

We will now demonstrate that no singular arcs exist for Σ_T, then the resulting control law is:

$$u_T^* = \begin{cases} \overline{u_T} & \Sigma_T \leq 0 \\ \underline{u_T} & \Sigma_T > 0 \end{cases}. \tag{5.14}$$

For a singular arc to exist, Σ_T must be zero for a nontrivial interval of time. Setting Σ_T to zero yields:

$$\vartheta^* = \arctan\left(-\frac{p_4}{p_2}\right) = \arctan\left(\frac{c_3 t - c_4}{c_2 - c_1 t}\right). \tag{5.15}$$

The angle ϑ^* is determined by the rotational control input u_R^*. If u_R^* is in a non-singular interval, then ϑ^* is a parabolic function of time ($\ddot{\vartheta}^* =$ cost) and eq. (5.15) can not be satisfied over a nontrivial time interval. If, instead, u_R^* is singular, ϑ^* is given by eq. (5.11) and then for u_T^* to be singular too, eq. (5.11) and eq. (5.15) must be equal:

$$\arctan\left(\frac{c_2 - c_1 t}{c_4 - c_3 t}\right) = \arctan\left(\frac{c_3 t - c_4}{c_2 - c_1 t}\right)$$

Taking the tangent of both sides we obtain:

$$(c_2 - c_1 t)^2 + (c_4 - c_3 t)^2 = 0,$$

which can not be true for a nontrivial time unless for the trivial case $c = (0, 0, 0, 0)$. Therefore we conclude that the thrust input u_T^* does not contain singular arcs.

Augmented system

Since only the second order derivative of the switching function Σ_R is known, we introduce an augmented system that contains the quadrotor state, the switching function Σ_R and its first derivative $\dot{\Sigma}_R$:

$$\chi_a = \begin{bmatrix} \chi^* & \Sigma_R & \dot{\Sigma}_R \end{bmatrix}. \tag{5.16}$$

The evolution of this system is governed by the following first order differential equation:

$$\dot{\chi}_a = f_a(t, \chi_a) = \begin{bmatrix} \dot{x}^* \\ u_T^* \sin\vartheta^* \\ \dot{z}^* \\ u_T^* \cos\vartheta^* - 1 \\ \dot{\vartheta}^* \\ u_R^* \\ \dot{\Sigma}_R \\ (c_2 - c_1 t)\, u_T^* \cos\vartheta^* + (c_3 t - c_4)\, u_T^* \sin\vartheta^* \end{bmatrix} \tag{5.17}$$

where the control inputs u_R^* and u_T^* are given by the control laws of eq. (5.9) and eq. (5.14). A quadrotor maneuver from χ_i to X_G that satisfies the minimum principle solves the boundary value problem (BVP)

$$\dot{\chi}_a = f_a(t, \chi_a) \tag{5.18}$$

$$\chi(0) = \chi_i \tag{5.19}$$

$$\chi(t_g) \in X_G. \tag{5.20}$$

This problem has seven unknowns: the final time T, the four constants c and the initial value of the switching function $\Sigma_R(0)$ and of its first time derivative $\dot{\Sigma}_R(0)$. These last two unknowns must also satisfy the condition eq. (5.6). In particular, assuming that the state, costate and input trajectories are known, and considering eq. (5.4) and eq. (5.6) we can write

$$\Sigma_R = \frac{1 + p_1\dot{x} + p_2 u_T \sin\vartheta + p_3\dot{z} + p_4(u_T\cos\vartheta - 1) + p_5\dot{\vartheta}}{-u_R}$$

and, being $p_5 = -\dot{\Sigma}_R$,

$$\begin{aligned}\Sigma_R &= \frac{1 + p_1\dot{x} + p_2 u_T \sin\vartheta + p_3\dot{z} + p_4(u_T\cos\vartheta - 1) - \dot{\Sigma}_R\dot{\vartheta}}{-u_R}\\ &= \frac{1 + c_1\dot{x} + (c_2 - c_1 t)u_T \sin\vartheta + c_3\dot{z} + (c_4 - c_3 t)(u_T\cos\vartheta - 1) - \dot{\Sigma}_R\dot{\vartheta}}{-u_R}\end{aligned} \tag{5.21}$$

We tried different numerical algorithms to solve this problem but none of them was able to find an optimal solution. Indeed these algorithms are in general very sensible to the initial guesses of the unknowns and, apart from the total time T, these guesses do not have any physical meaning and are then hard to determine. Moreover even if we found a solution to this problem, the resulting trajectory would only satisfy a necessary condition for optimality. Finally we would still need to generalize the method to the original three-dimensional problem in which the grasping state is only partially fixed and the conditions for grasping have to be maintained for a finite time. For all these reasons we decided to abandon this strategy and to try a different solution that is described in the next chapter.

Chapter 6

Compound trajectory generation

A different approach to solve the grasping problem is to split it into different subproblems. The complete grasping trajectory can indeed be obtained as a composition of three sub-trajectories:

- an approach transfer trajectory bringing the robot from the starting state χ_i to the state χ_g^i in which the robot reaches the target, for $t = t_g$;
- a gripper-closure trajectory maintaining the grasping conditions satisfied for a finite time interval $T_g > T_g^{min}$, while bringing the robot from the state χ_g^i to the state χ_g^f;
- a leaving transfer trajectory bringing the robot from the final state along of the gripper-closure trajectory, namely χ_g^f, to the desired final state χ_f, for $t = t_f$.

Each of these sub-trajectories must be chosen according to an optimality criteria, i.e. with the aim of minimizing the duration of the complete trajectory.

In this chapter we describe each of the above steps separately. We start by introducing two possible classes of gripper-closure trajectories satisfying eqs. (3.2) and (3.3). First we consider the case of a fixed target and then we extend our results to the case of a target moving at constant velocity during the grasping. We will finally show how the flatness property can be exploited to generate the two transfer trajectories needed to connect the gripper closure trajectory to the initial and final states.

In the following we denote with (α, β, ρ) the polar coordinates (i.e., azimuth, zenith distance, and radius) of the (constant) vector ${}^B r_G$ defining the position of the center of grasping O_G in the frame B.

6.1 Horizontal circular paths

A possible trajectory satisfying the constraints defined by eqs. (3.2) and (3.3) is that in which the robot rotates at constant velocity around an axis parallel to z_W and passing through O_T. In these conditions the robot acts as a mass attached to a fixed point through a wire, rotating around the axis of gravity. The only difference is that here the reaction force of the wire must be substituted by the thrust force generated by the propellers.

To analyze this class of trajectories it is convenient to consider the reference frame R, obtained by translating the world inertial frame to the position of the target and rotating it around the axis z_W by an angle $\zeta(t) = \Omega_\zeta(t - t_g) + \zeta_g$ where Ω_ζ and ζ_g are constant:

$$^{W}R_R = R_z(\Omega_\zeta(t - t_i) + \zeta_g).$$

In this frame the velocity of the robot must be zero and its position and orientation must be fixed. We assume, without loss of generality, that the robot center of mass, i.e. the point O_B, always lies on the plane defined by x_R and z_R, i.e.

$$(r_B - r_T)^T y_R = 0.$$

The forces acting on the robot, as shown in figure fig. 6.1, are the gravity force, the thrust produced by the actuators and the (fictitious) centrifugal force, due to the fact that the reference frame is rotating.

In order for the acceleration to be null in the non inertial frame we have to impose the equilibrium of the forces acting on the system. Since there is no force acting on the direction of y_R, in order to guarantee that the robot has a constant tangential velocity along the circular trajectory the component of the thrust vector along y_R must be zero, i.e.

$$z_B^T y_R = 0.$$

Moreover the following relations involving the thrust input must hold true

$$-u_1 \cos(\xi_g + \beta) = mg, \tag{6.1a}$$

$$u_1 \sin(\xi_g + \beta) = m\Omega_\zeta^2 \rho \sin\xi_g. \tag{6.1b}$$

Given a desired value of ξ_g, we immediately compute the value of the thrust force that is necessary to compensate the gravity action using eq. (6.1a)

$$u_1 = -\frac{mg}{\cos(\xi_g + \beta)}, \tag{6.2}$$

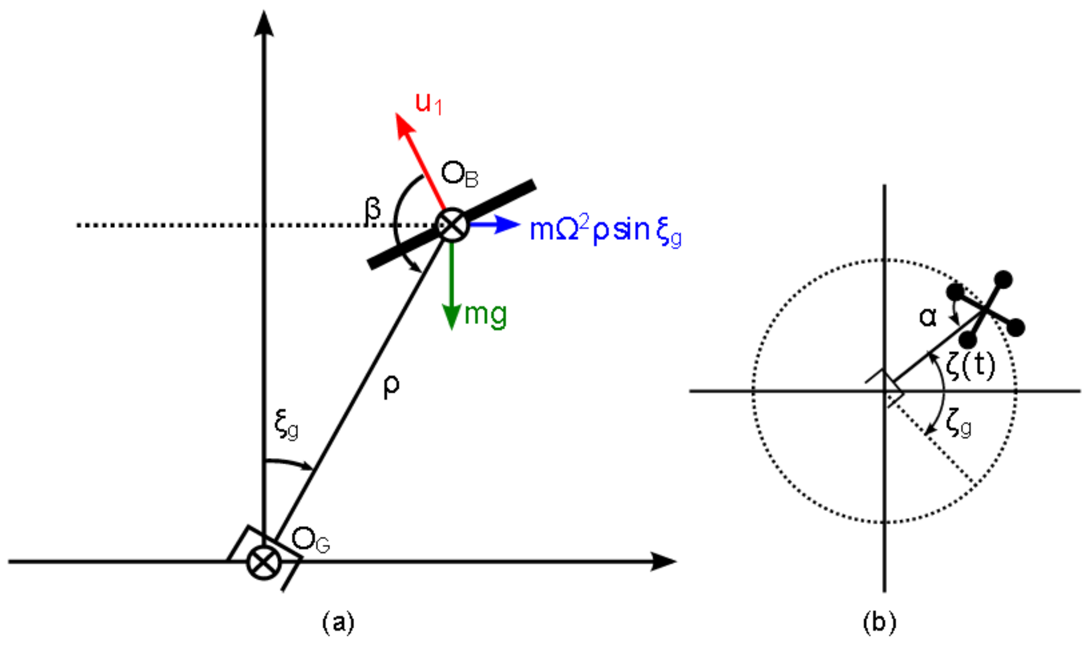


Figure 6.1: Side (a) and top (b) views of the horizontal gripper-closure sub-trajectory, with forces and relevant parameters. Note that ξ_g is constant in this case, while $\zeta(t)$ is a time-varying quantity starting from the initial value $\zeta(t_g) = \zeta_g$

then we obtain the angular velocity using eq. (6.1b)

$$\Omega_\zeta = \pm\sqrt{\frac{-g\tan(\xi_g + \beta)}{\rho\sin\xi_g}}. \tag{6.3}$$

The position and orientation of the robot in the rotating must be fixed in such a way that O_G is positioned on the target, i.e. in the origin of the rotating frame. This result in the following condition on the position of the robot center of mass O_B:

$${}^{R}r_B = \begin{matrix} \rho\sin\xi_g \\ 0 \\ \rho\cos\xi_g \end{matrix} . \tag{6.4}$$

Concerning the orientation, the robot must first be rotated around z_R by and angle $(\pi - \alpha)$ and then around y_R by an angle $(\xi_g + \beta - \pi)$. This is described by the following rotation matrix:

$${}^{R}R_B = R_y(\xi_g + \beta - \pi)\, R_z(\pi - \alpha). \tag{6.5}$$

By considering the simple rigid transformation between the rotating frame and the world frame we can compute the trajectory of the robot state:

$$\begin{aligned} r_B &= r_T + {}^W R_R \, {}^R r_B \\ \dot{r}_B &= \omega_{RW} \times {}^W R_R \, {}^R r_B \\ {}^W R_B &= {}^W R_R \, {}^R R_B \\ {}^B \omega_{BW} &= {}^R R_B^T \, {}^R \omega_{RW}. \end{aligned}$$

In particular the orientation of the robot is given by

$${}^W R_B = {}^W R_R \, {}^R R_B = R_z(\zeta) \, R_y(\xi_g + \beta - \pi) \, R_z(\pi - \alpha)$$

and applying eq. (2.2c) we can compute the yaw angle along the trajectory

$$\psi = \text{atan2} \; c_\alpha s_\zeta c_{\xi_g+\beta} + s_\alpha c_\zeta, c_\alpha c_\zeta c_{\xi_g+\beta} - s_\alpha s_\zeta \; . \tag{6.7}$$

The angular velocity in the local frame is given by:

$${}^B \omega_{BW} = {}^R R_B^T \, {}^R \omega_{RW} = R_z(\alpha - \pi) \, R_y(\pi - \xi_g - \beta) \begin{matrix} 0 \\ 0 \\ \Omega_\zeta \end{matrix} = \Omega_\zeta \begin{matrix} -c_\alpha s_{\xi_g+\beta} \\ -s_\alpha s_{\xi_g+\beta} \\ -c_{\xi_g+\beta} \end{matrix}$$

and differentiating eq. (6.7) w.r.t. time we can easily verify that

$$\dot{\psi} = \dot{\zeta} = \Omega_\zeta.$$

We can then simplify the calculation of ψ by considering that

$$\psi = \Omega_\zeta (t - t_g) + \psi_g,$$

where obviously

$$\psi_g = \text{atan2} \; c_\alpha s_{\zeta_g} c_{\xi_g+\beta} + s_\alpha c_{\zeta_g}, c_\alpha c_{\zeta_g} c_{\xi_g+\beta} - s_\alpha s_{\zeta_g} \; .$$

The angular acceleration is obviously null

$${}^B \dot{\omega}_{BW} = 0,$$

then the torque inputs, assuming that the inertia matrix is diagonal, are

$$\begin{matrix} u_2 \\ u_3 \\ u_4 \end{matrix} = {}^B \omega_{BW} \times J \, {}^B \omega_{BW} = \frac{\Omega_\zeta^2}{2} \begin{matrix} -s_\alpha s_{2\xi_g+2\beta} (J_{yy} - J_{zz}) \\ c_\alpha s_{2\xi_g+2\beta} (J_{xx} - J_{zz}) \\ -s_{2\alpha} s_{\xi_g+\beta}^2 (J_{xx} - J_{yy}) \end{matrix}$$

and are constant over time. If the quadrotor is symmetric, then

$$J_{xx} = J_{yy},$$

and the inputs further simplify to

$$\begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} = \frac{\Omega_\zeta^2}{2} s_{2\xi_g + 2\beta} \left(J_{xx} - J_{zz} \right) \begin{bmatrix} -s_\alpha \\ c_\alpha \\ 0 \end{bmatrix} . \tag{6.8}$$

By inverting eq. (2.6) we also obtain the actual inputs

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \begin{bmatrix} -\frac{mg}{4k_F\, c_{\xi_g+\beta}} - \frac{\Omega_\zeta^2}{4}\left(\frac{s_{2\alpha}\, s^2_{\xi_g+\beta}\left(J_{xx}-J_{yy}\right)}{2k_M} + \frac{c_\alpha\, s_{2\xi_g+2\beta}\left(J_{xx}-J_{zz}\right)}{k_F\, l}\right) \\ -\frac{mg}{4k_F\, c_{\xi_g+\beta}} + \frac{\Omega_\zeta^2}{4}\left(\frac{s_{2\alpha}\, s^2_{\xi_g+\beta}\left(J_{xx}-J_{yy}\right)}{2k_M} - \frac{s_\alpha\, s_{2\xi_g+2\beta}\left(J_{yy}-J_{zz}\right)}{k_F\, l}\right) \\ -\frac{mg}{4k_F\, c_{\xi_g+\beta}} - \frac{\Omega_\zeta^2}{4}\left(\frac{s_{2\alpha}\, s^2_{\xi_g+\beta}\left(J_{xx}-J_{yy}\right)}{2k_M} - \frac{c_\alpha\, s_{2\xi_g+2\beta}\left(J_{xx}-J_{zz}\right)}{k_F\, l}\right) \\ -\frac{mg}{4k_F\, c_{\xi_g+\beta}} + \frac{\Omega_\zeta^2}{4}\left(\frac{s_{2\alpha}\, s^2_{\xi_g+\beta}\left(J_{xx}-J_{yy}\right)}{2k_M} + \frac{s_\alpha\, s_{2\xi_g+2\beta}\left(J_{yy}-J_{zz}\right)}{k_F\, l}\right) \end{bmatrix} ,$$

that simplify to

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \begin{bmatrix} -\frac{mg}{4k_F\, c_{\xi_g+\beta}} - \frac{\Omega_\zeta^2}{4}\frac{c_\alpha\, s_{2\xi_g+2\beta}\left(J_{xx}-J_{zz}\right)}{k_F\, l} \\ -\frac{mg}{4k_F\, c_{\xi_g+\beta}} - \frac{\Omega_\zeta^2}{4}\frac{s_\alpha\, s_{2\xi_g+2\beta}\left(J_{xx}-J_{zz}\right)}{k_F\, l} \\ -\frac{mg}{4k_F\, c_{\xi_g+\beta}} + \frac{\Omega_\zeta^2}{4}\frac{c_\alpha\, s_{2\xi_g+2\beta}\left(J_{xx}-J_{zz}\right)}{k_F\, l} \\ -\frac{mg}{4k_F\, c_{\xi_g+\beta}} + \frac{\Omega_\zeta^2}{4}\frac{s_\alpha\, s_{2\xi_g+2\beta}\left(J_{xx}-J_{zz}\right)}{k_F\, l} \end{bmatrix} \tag{6.9}$$

if the quadrotor is assumed to be symmetric.

The quantities needed to compute the state and the inputs using the flat transformation are:

$$\begin{aligned} r_{B,t} &= r_T + R_z\left(\Omega_\zeta\left(t - t_g\right) + \zeta_g\right)\, {}^R r_B \\ \dot{r}_{B,t} &= \omega_{RW} \times \left(r_{B,t} - r_T\right) \\ \ddot{r}_{B,t} &= \omega_{RW} \times \dot{r}_{B,t} \\ \dot{a}_{B,t} &= \omega_{RW} \times \ddot{r}_{B,t} \\ \ddot{a}_{B,t} &= \omega_{RW} \times \dot{a}_{B,t} \\ \psi_t &= \Omega_\zeta\left(t - t_g\right) + \psi_g^i \\ \dot{\psi}_t &= \Omega_\zeta \\ \ddot{\psi}_t &= 0. \end{aligned}$$

Before describing the other possible class of gipper-closure trajectories it is worth to analyze how the position of the gripper with respect to the quadrotor (defined by the parameters α, β and ρ) influences the possible horizontal circular trajectories.

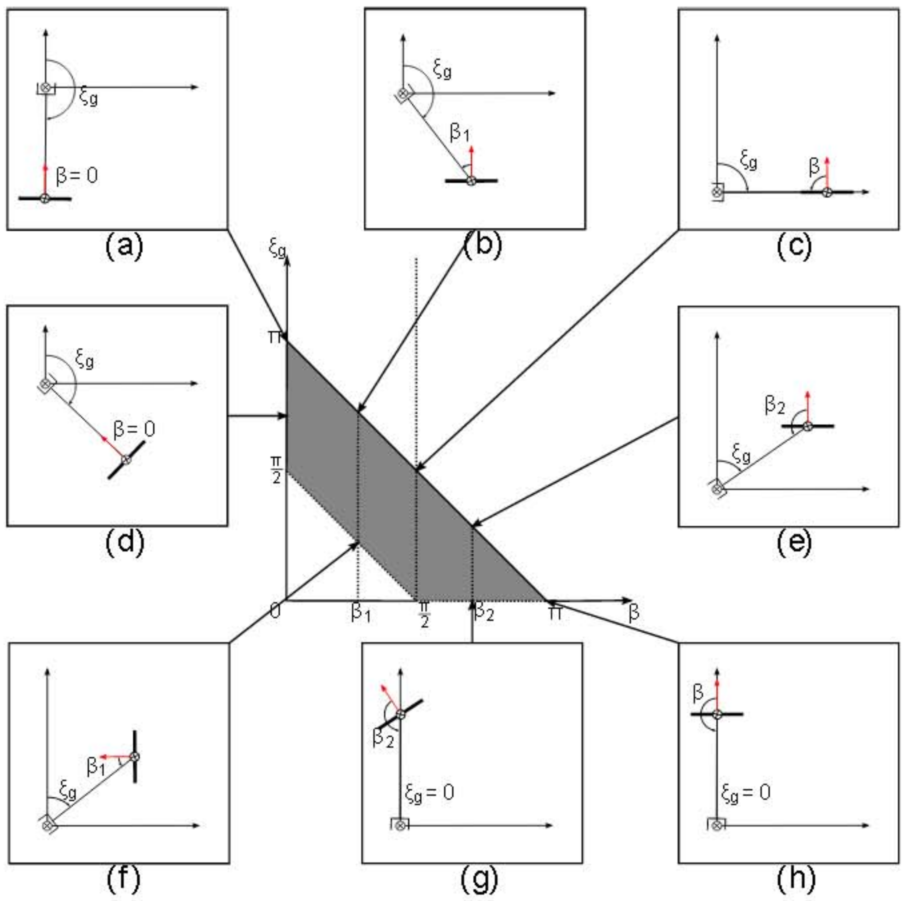


Figure 6.2: Admissible values for ξ_g depending on β.

The azimuth α determines the way in which the control effort is distributed among the motors in order to generate the needed torques. From eq. (6.9) it is clear that if $\alpha = k\frac{\pi}{2}$ only one pair of motors is used to generate the necessary torque. To guarantee that every motor is contributing in the same way and then to reduce the effort required to each of them it is more convenient to chose

$$\alpha = (2k + 1)\frac{\pi}{4}.$$

The parameter β has a strong influence on the determination of the admissible values of $\xi_g \in [0, \pi]$. Indeed from eqs. (6.2) and (6.3), we note that we must impose $\cos(\xi_g + \beta) < 0$ for having a positive thrust, and $\tan(\xi_g + \beta) \leq 0$ for Ω_ζ to be real (note that $\sin \xi_g \geq 0$). Therefore the only admissible interval for $\xi_g + \beta$ is $(\frac{\pi}{2}, \pi]$, implying

$$\xi_g \in \max\ 0, \frac{\pi}{2} - \beta\ , \pi - \beta\ = (\xi_{g,inf}, \xi_{g,max}]. \qquad (6.10)$$

The resulting admissible set of ξ_g and β is represented by the gray shaded area in fig. 6.2.

The case $\xi_g = \xi_{g,max}$ corresponds to hovering trajectories (fig. 6.2(a-c)(e)(h)), i.e. to trajectories for which $\Omega_\zeta = 0$ and O_B is in a fixed position with respect to the target. A singularity arises when $\beta = (0, \pi)$, since Ω_ζ becomes undefined in this case (fig. 6.2 (a)(h)).

Now consider separately the two cases in which the gripper is above or below the quadrotor horizontal plane, i.e., $\beta \in [0, \pi/2)$ or $\beta \in [\pi/2, \pi]$, respectively.

The first case yields $\xi_{g,inf} = \pi/2 - \beta > 0$ making the choice $\xi_g = \xi_{g,inf}$ unfeasible (fig. 6.2(f)), as it would imply a perfectly vertical quadrotor with the total thrust applied in a direction orthogonal to that of the gravity (in fact, eq. (6.2) becomes singular in this case).

The second case yields $\xi_{g,inf} = 0$ and $\xi_g = \xi_{g,inf}$, corresponds to the situation in which O_B is exactly above the target. If $\beta = \pi$, this second case is not feasible because, as it is clear from eq. (6.3), it would require an infinite Ω_ζ (fig. 6.2(g)). On the other hand, if $\beta = \pi$ then Ω_ζ becomes indefinite, thus corresponding to another hovering case (fig. 6.2(h)).

It is also clear that, in general, values of β that are too close to π reduce the range of admissible values of ξ_g, and then the number of possible grasping trajectories. In particular, if $\beta = \pi$, it is easy to verify that the sole admissible solutions reduce to the hovering ones (fig. 6.2(h)).

It is also important to consider that a zenith distance β close to $\pi/2$ might invalidate the assumption of having an almost symmetric inertia matrix, and then the possibility of neglecting the weight of the target.

In the same fashion, a large value of ρ (the gripper length) also increases the inertia introduced by the target after performing the grasping. Additionally, the length ρ also determines the angular velocity Ω_ζ for a given value of ξ_g, and thus the amount of necessary torques as it can be seen from eqs. (6.3) and (6.8).

Concerning additional possible geometric constraints of the problem, values of β larger or smaller than $\pi/2$ are clearly most suited for targets lying on the ground plane or hanging from the ceiling, respectively.

6.2 Vertical circular paths

We now analyze a different class of trajectories in which the robot center of mass rotates around an axis orthogonal to z_W and passing through O_T. In this case it is convenient to consider a rotating frame R having the origin on the target, the axis y_R coincident with the axis of rotation of O_B, and the axis z_R pointing toward

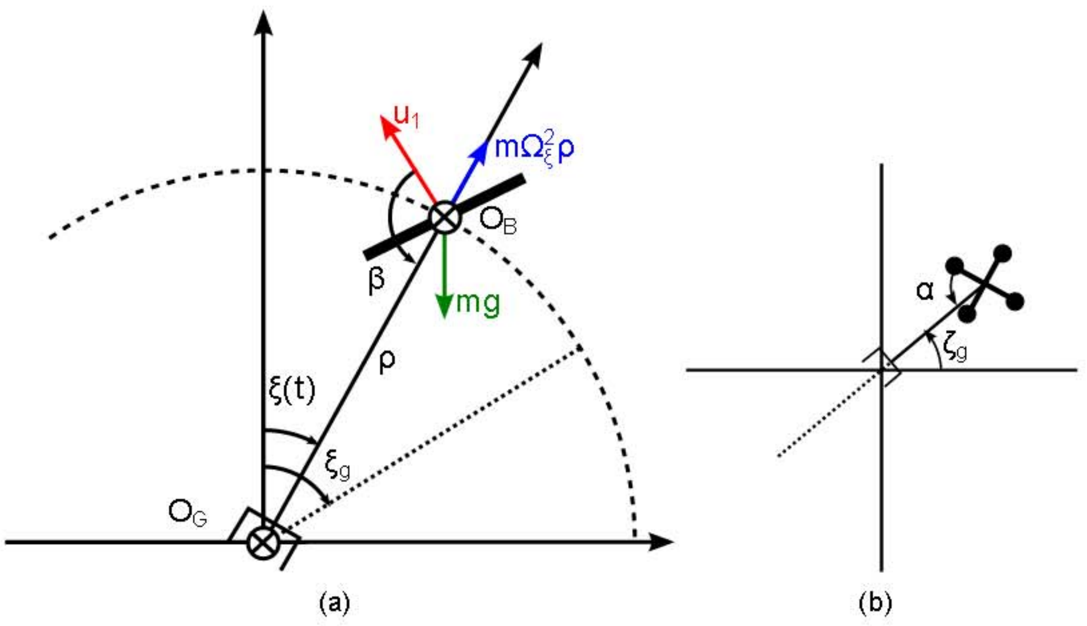


Figure 6.3: Side (a) and top (b) views of the vertical gripper-closure sub-trajectory at constant angular velocity, with forces and relevant parameters. Note that, contrarily to the horizontal case, here ζ_g is a constant parameter, while $\xi(t)$ changes over time with initial value $\xi(t_g) = \xi_g$.

O_B. The orientation of this frame with respect to the world frame is given by:

$$^{W}R_R = R_z(\zeta_g)\, R_y(\xi(t))$$

Let's first consider the case of a constant tangential velocity along the trajectory. In this case we have

$$\xi(t) = \Omega_\xi (t - t_g) + \xi_g.$$

The scheme of the forces acting on the system is represented in fig. 6.3. As we can see the main difference with respect to the previous case lies in the direction of the fictitious centrifugal force which remains constant in norm.

As we did for the previous case we must impose that the robot does not move in the rotating frame. Again we assume that O_B always lies on the plane defined by x_R and z_R, i.e.

$$(r_B - r_T)^T y_R = 0.$$

In order to guarantee that the robot does not leave this plane, the component of the thrust vector along y_R must be zero, i.e.

$$z_B^T y_R = 0.$$

The equilibrium of forces is given by:

$$\begin{aligned} u_1 \cos\beta &= m\Omega_\xi^2\rho - mg\cos(\Omega_\xi(t - t_g) + \xi_g) \\ u_1 \sin\beta &= mg\sin(\Omega_\xi(t - t_g) + \xi_g). \end{aligned}$$

Multiplying the first equation by $\sin\beta$ and the second one by $\cos\beta$ and then subtracting we get

$$0 = m\Omega_\xi^2\rho\sin\beta - mg\sin(\Omega_\xi(t - t_g) + \xi_g - \beta),$$

which cannot be satisfied for a nontrivial time unless

$$\begin{aligned} \Omega_\xi &= 0 \\ \xi_g &= \beta + k\pi, \end{aligned}$$

which corresponds to the hovering case. This means that it is impossible to perform this trajectory at constant non-zero angular velocity. More in general we will have a non-zero angular acceleration of the rotating frame which implies the presence of another tangential fictitious force as reported in fig. 6.4. In this case we have

$$u_1 \cos\beta = m\dot{\xi}^2\rho - mg\cos\xi \tag{6.11a}$$

$$u_1 \sin\beta = mg\sin\xi - m\ddot{\xi}\rho. \tag{6.11b}$$

If we multiply eq. (6.11a) by $\sin\beta$ and eq. (6.11b) by $\cos\beta$, and then we subtract the resulting equations, we obtain the nonlinear differential equation governing the evolution of $\xi(t)$.

$$m\rho\ \ \dot{\xi}^2\sin\beta + \ddot{\xi}\cos\beta\ \ - mg\sin(\xi + \beta) = 0. \tag{6.12}$$

We can put it in the form of a first order differential equation introducing

$$\boldsymbol{\xi} = \begin{matrix} \xi \\ \dot{\xi} \end{matrix} \rightarrow \dot{\boldsymbol{\xi}} = \begin{matrix} \xi_2 \\ \frac{g\sin(\xi_1 + \beta) - \rho\xi_2^2\sin\beta}{\rho\cos\beta} \end{matrix}.$$

If instead we multiply eq. (6.11a) by $\cos\beta$ and eq. (6.11b) by $\sin\beta$ and we sum we get an expression of the thrust input:

$$u_1 = m\rho\ \ \dot{\xi}^2\cos\beta - \ddot{\xi}\sin\beta\ \ - mg\cos(\xi + \beta).$$

Also in this case the position and orientation of the robot in the rotating frame must be chosen such that the gripper is always in the origin of the frame. The coordinates of the robot in the rotating frame are simply given by

$${}^R r_B = \begin{matrix} 0 \\ 0 \\ \rho \end{matrix}.$$

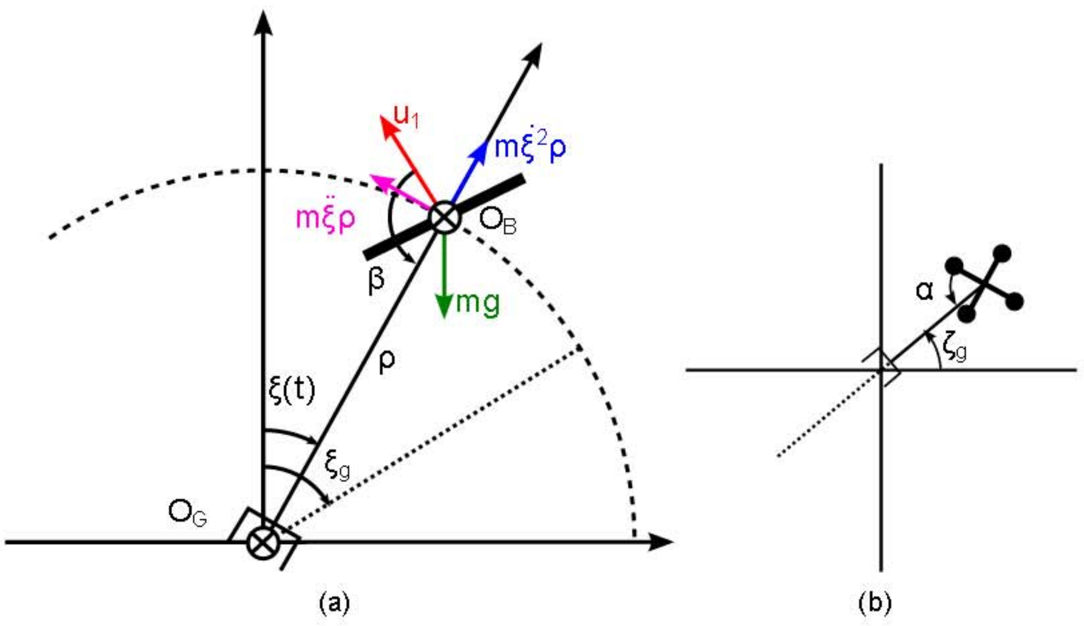


Figure 6.4: Side (a) and top (b) views of the vertical gripper-closure sub-trajectory at non-constant angular velocity, with forces and relevant parameters. Note that, contrarily to the horizontal case, here ζ_g is a constant parameter, while $\xi(t)$ changes over time with initial value $\xi(t_g) = \xi_g$.

Concerning the orientation, the robot must first be rotated around z_R by an angle $(\pi - \alpha)$ and then around y_R by an angle $(\beta - \pi)$. This is described by the following rotation matrix:

$$ {}^{R}R_B = R_y(\beta - \pi)\, R_z(\pi - \alpha)\,. \tag{6.13}$$

As we did before, by considering the simple rigid transformation between the rotating frame and the world frame we can compute the trajectory of the robot state as follows:

$$\begin{aligned} r_B &= r_T + {}^{W}R_R\, {}^{R}r_B \\ \dot{r}_B &= \omega_{RW} \times {}^{W}R_R\, {}^{R}r_B \\ {}^{W}R_B &= {}^{W}R_R\, {}^{R}R_B \\ {}^{B}\omega_{BW} &= {}^{R}R_B^{T}\, {}^{R}\omega_{RW}\,. \end{aligned}$$

The orientation of the robot is given by

$$ {}^{W}R_B = {}^{W}R_R\, {}^{R}R_B = R_z(\zeta_g)\, R_y(\xi + \beta - \pi)\, R_z(\pi - \alpha)$$

and applying eq. (2.2c) we can compute the yaw angle along the trajectory

$$\psi = \mathrm{atan2}\;\; c_\alpha s_{\zeta_g} c_{\xi+\beta} + s_\alpha c_{\zeta_g},\, c_\alpha c_{\zeta_g} c_{\xi+\beta} - s_\alpha s_{\zeta_g}\;\; . \tag{6.15}$$

The angular velocity in the local frame is given by

$$ {}^{B}\omega_{BW} = {}^{R}R_{B}^{T}\,{}^{R}\omega_{RW} = R_{z}(\alpha-\pi)\,R_{y}(\pi-\beta)\begin{matrix} 0 \\ \dot{\xi} \\ 0 \end{matrix} = \dot{\xi}\begin{matrix} s_{\alpha} \\ -c_{\alpha} \\ 0 \end{matrix} , $$

while the yaw rate can be obtained by derivation of eq. (6.15)

$$ \dot{\psi} = \frac{c_{\alpha}\dot{\xi}}{\tilde{s}_{\psi}^{2}+\tilde{c}_{\psi}^{2}} s_{\xi+\beta} \quad c_{\zeta_g}\tilde{s}_{\psi} - s_{\zeta_g}\tilde{c}_{\psi} \quad . \tag{6.16} $$

where $\tilde{s}_{\psi}$ and $\tilde{c}_{\psi}$ are the first and last argument in eq. (6.15).

Similarly the angular acceleration is given by

$$ {}^{B}\dot{\omega}_{BW} = {}^{R}R_{B}^{T}\,{}^{R}\dot{\omega}_{RW} = R_{z}(\alpha-\pi)\,R_{y}(\pi-\beta)\begin{matrix} 0 \\ \ddot{\xi} \\ 0 \end{matrix} = \ddot{\xi}\begin{matrix} s_{\alpha} \\ -c_{\alpha} \\ 0 \end{matrix} , $$

and differentiating eq. (6.16) we obtain the yaw acceleration

$$ \ddot{\psi} = \frac{c_{\alpha}}{\tilde{s}_{\psi}^{2}+\tilde{c}_{\psi}^{2}} \quad \ddot{\xi}s_{\xi+\beta} + \dot{\xi}^{2}c_{\xi+\beta} \quad c_{\zeta_g}\tilde{s}_{\psi} - s_{\zeta_g}\tilde{c}_{\psi} \; + 2\dot{\xi}\dot{\psi}s_{\xi+\beta} \; s_{\zeta_g}\tilde{s}_{\psi} + c_{\zeta_g}\tilde{c}_{\psi} \quad . \tag{6.17} $$

Assuming again that the robot inertia tensor is symmetric, the torque inputs are given by

$$ \begin{matrix} u_{2} \\ u_{3} \\ u_{4} \end{matrix} = J\,{}^{B}\dot{\omega}_{BW} + {}^{B}\omega_{BW} \times J\,{}^{B}\omega_{BW} = \begin{matrix} \ddot{\xi}s_{\alpha}J_{xx} \\ -\ddot{\xi}c_{\alpha}J_{yy} \\ \frac{\dot{\xi}^{2}}{2}s_{2\alpha}(J_{xx}-J_{yy}) \end{matrix} , $$

and in case of perfect symmetry the gyroscopic effect disappears and they simplify to the following expression

$$ \begin{matrix} u_{2} \\ u_{3} \\ u_{4} \end{matrix} = \ddot{\xi}J_{xx}\begin{matrix} s_{\alpha} \\ -c_{\alpha} \\ 0 \end{matrix} . \tag{6.18} $$

The corresponding squared motors velocities are

$$ \begin{matrix} \omega_{1}^{2} \\ \omega_{2}^{2} \\ \omega_{3}^{2} \\ \omega_{4}^{2} \end{matrix} = \begin{matrix} \frac{u_{1}(\xi)}{4k_{F}} + \frac{\ddot{\xi}c_{\alpha}J_{yy}}{2k_{F}l} + \frac{\dot{\xi}^{2}s_{2\alpha}(J_{xx}-J_{yy})}{8k_{M}} \\ \frac{u_{1}(\xi)}{4k_{F}} + \frac{\ddot{\xi}s_{\alpha}J_{xx}}{2k_{F}l} - \frac{\dot{\xi}^{2}s_{2\alpha}(J_{xx}-J_{yy})}{8k_{M}} \\ \frac{u_{1}(\xi)}{4k_{F}} - \frac{\ddot{\xi}c_{\alpha}J_{yy}}{2k_{F}l} + \frac{\dot{\xi}^{2}s_{2\alpha}(J_{xx}-J_{yy})}{8k_{M}} \\ \frac{u_{1}(\xi)}{4k_{F}} - \frac{\ddot{\xi}s_{\alpha}J_{xx}}{2k_{F}l} - \frac{\dot{\xi}^{2}s_{2\alpha}(J_{xx}-J_{yy})}{8k_{M}} \end{matrix} , $$

in the general case and

$$\begin{bmatrix} \omega_1^2 \\ \omega_2^2 \\ \omega_3^2 \\ \omega_4^2 \end{bmatrix} = \begin{bmatrix} \frac{u_1(\xi)}{4k_F} + \frac{\ddot{\xi} c_\alpha J_{xx}}{2k_F l} \\ \frac{u_1(\xi)}{4k_F} + \frac{\ddot{\xi} s_\alpha J_{xx}}{2k_F l} \\ \frac{u_1(\xi)}{4k_F} - \frac{\ddot{\xi} c_\alpha J_{xx}}{2k_F l} \\ \frac{u_1(\xi)}{4k_F} - \frac{\ddot{\xi} s_\alpha J_{xx}}{2k_F l} \end{bmatrix},$$

in the symmetric one.

It is easy to verify that the quantities needed to compute the state and the inputs using the flat transformation are:

$$\begin{aligned}
r_{B,t} &= r_T + {}^W R_R(\xi)\, {}^R r_B \\
\dot{r}_{B,t} &= \omega_{RW} \times (r_{B,t} - r_T) \\
\ddot{r}_{B,t} &= \dot{\omega}_{RW} \times (r_{B,t} - r_T) + \omega_{RW} \times \dot{r}_{B,t} \\
\dot{a}_{B,t} &= \ddot{\omega}_{RW} \times (r_{B,t} - r_T) + 2\dot{\omega}_{RW} \times \dot{r}_{B,t} + \omega_{RW} \times \ddot{r}_{B,t} \\
\ddot{a}_{B,t} &= \dot{\alpha}_{RW} \times (r_{B,t} - r_T) + 3\ddot{\omega}_{RW} \times \dot{r}_{B,t} + 3\dot{\omega}_{RW} \times \ddot{r}_{B,t} + \omega_{RW} \times \dot{a}_{B,t} \\
\psi_t &= \operatorname{atan2}\left(c_\alpha s_{\zeta_g} c_{\xi+\beta} + s_\alpha c_{\zeta_g},\, c_\alpha c_{\zeta_g} c_{\xi+\beta} - s_\alpha s_{\zeta_g}\right) \\
\dot{\psi}_t &= \frac{c_\alpha \dot{\xi}}{\tilde{s}_\psi^2 + \tilde{c}_\psi^2} s_{\xi+\beta} \left(c_{\zeta_g} \tilde{s}_\psi - s_{\zeta_g} \tilde{c}_\psi\right) \\
\ddot{\psi}_t &= \frac{c_\alpha}{\tilde{s}_\psi^2 + \tilde{c}_\psi^2} \left[\left(\ddot{\xi} s_{\xi+\beta} + \dot{\xi}^2 c_{\xi+\beta}\right)\left(c_{\zeta_g} \tilde{s}_\psi - s_{\zeta_g} \tilde{c}_\psi\right) + 2\dot{\xi}\dot{\psi} s_{\xi+\beta}\left(s_{\zeta_g} \tilde{s}_\psi + c_{\zeta_g} \tilde{c}_\psi\right)\right],
\end{aligned}$$

where

$$\begin{aligned}
\omega_{RW} &= R_z(\zeta_g) \begin{bmatrix} 0 & \dot{\xi} & 0 \end{bmatrix}^T \\
\dot{\omega}_{RW} &= R_z(\zeta_g) \begin{bmatrix} 0 & \ddot{\xi} & 0 \end{bmatrix}^T \\
\ddot{\omega}_{RW} &= R_z(\zeta_g) \begin{bmatrix} 0 & \dddot{\xi} & 0 \end{bmatrix}^T \\
\dot{\alpha}_{RW} &= R_z(\zeta_g) \begin{bmatrix} 0 & \ddddot{\xi} & 0 \end{bmatrix}^T.
\end{aligned}$$

Once we have obtained ξ and $\dot{\xi}$ solving the differential equation, we are able to compute the higher order derivatives, indeed

$$\begin{aligned}
\ddot{\xi} &= \frac{g \sin(\xi + \beta) - \rho \dot{\xi}^2 \sin\beta}{\rho \cos\beta} \\
\dddot{\xi} &= \frac{g \cos(\xi + \beta) - 2\rho \ddot{\xi} \sin\beta}{\rho \cos\beta} \dot{\xi} \\
\ddddot{\xi} &= \frac{g\left(\cos(\xi + \beta)\ddot{\xi} - \sin(\xi + \beta)\dot{\xi}^2\right) - 2\rho \sin\beta \left(\dddot{\xi}\dot{\xi} + \ddot{\xi}^2\right)}{\rho \cos\beta}.
\end{aligned}$$

Concerning the position of the gripper with respect to the robot, the comments we made in the previous paragraph regarding the parameter α are still valid. In

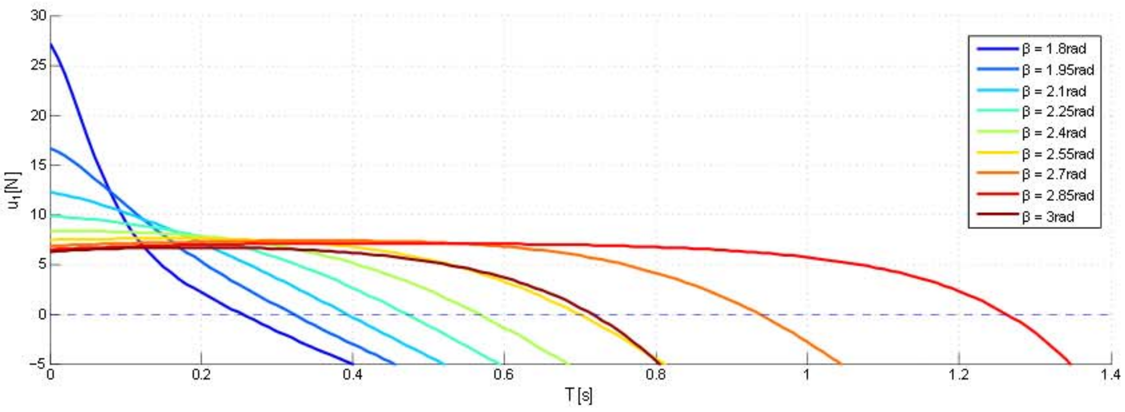


Figure 6.5: Total thrust trend for different values of β.

particular from eq. (6.18) it is clear that if $\alpha = k\frac{\pi}{2}$ only one pair of motors is used to generate the necessary torque. To guarantee that every motor is contributing in the same way and then to reduce the effort required to each of them it is more convenient to chose

$$\alpha = (2k + 1)\,\frac{\pi}{4}.$$

To guarantee that this trajectory is actually feasible, u_1 must lie within its admissible range and, in particular, it must be positive. Differently from the previous case, here in general the input u_1 can remain admissible only for a finite time interval. The length of this interval depends both on the parameters β and ρ and on the initial value of ξ and $\dot{\xi}$, namely ξ_g and $\dot{\xi}_g$. An explicit expression of this relationship is hard to deduce since the algebraic solution of eq. (6.12) is not known. However we can get a qualitative overview by considering a numerical example. In fig. 6.5 we report the time profile of the thrust input obtained by numerical integration of eq. (6.12) for a fixed ξ_g, $\dot{\xi}_g$ but for different values of β. There seems to be an optimal value of β, resulting in the maximum duration of the trajectory, around 2.7 rad. Nevertheless the optimum changes depending on the initial value of ξ and then there does not exist a unique best choice for β.

The comments we made about the influence of the target weight for β close to $\pi/2$ and large values of ρ remain valid.

6.3 Translating circular paths

The above results can easily be extended to the case of a target moving at constant velocity or, more in general, if it is possible to assume that the velocity of the target remains almost constant during the duration T_g of the gripper-closure trajectory. In this case, indeed, the real and fictitious forces acting on the system remain the same and we just need to properly translate the trajectory:

$$\begin{aligned} r_B &= r_T + {}^W R_R \, {}^R r_B && (6.19a) \\ \dot{r}_B &= \dot{r}_T + \omega_{RW} \times (r_B - r_T) && (6.19b) \\ {}^W R_B &= {}^W R_R \, {}^R R_B && (6.19c) \\ {}^B\omega_{BW} &= {}^R R_B^T \, {}^R\omega_{RW}. && (6.19d) \end{aligned}$$

The flat outputs and their derivatives become:

$$\begin{aligned} r_{B,t} &= r_T + R_z(\Omega_\zeta(t - t_g) + \zeta_g) \, {}^R r_B \\ \dot{r}_{B,t} &= \dot{r}_T + \omega_{RW} \times (r_{B,t} - r_T) \\ \ddot{r}_{B,t} &= \omega_{RW} \times (\dot{r}_{B,t} - \dot{r}_T) \\ \dot{a}_{B,t} &= \omega_{RW} \times \ddot{r}_{B,t} \\ \ddot{a}_{B,t} &= \omega_{RW} \times \dot{a}_{B,t} \\ \psi_t &= \Omega_\zeta(t - t_g) + \psi_g \\ \dot{\psi}_t &= \Omega_\zeta \\ \ddot{\psi}_t &= 0. \end{aligned}$$

in the first case and

$$\begin{aligned} r_{B,t} &= r_T + {}^W R_R(\xi) \, {}^R r_B \\ \dot{r}_{B,t} &= \dot{r}_T + \omega_{RW} \times (r_{B,t} - r_T) \\ \ddot{r}_{B,t} &= \dot{\omega}_{RW} \times (r_{B,t} - r_T) + \omega_{RW} \times (\dot{r}_{B,t} - \dot{r}_T) \\ \dot{a}_{B,t} &= \ddot{\omega}_{RW} \times (r_{B,t} - r_T) + 2\dot{\omega}_{RW} \times (\dot{r}_{B,t} - \dot{r}_T) + \omega_{RW} \times \ddot{r}_{B,t} \\ \ddot{a}_{B,t} &= \dot{\alpha}_{RW} \times (r_{B,t} - r_T) + 3\ddot{\omega}_{RW} \times (\dot{r}_{B,t} - \dot{r}_T) + 3\dot{\omega}_{RW} \times \ddot{r}_{B,t} + \omega_{RW} \times \dot{a}_{B,t} \\ \psi_t &= \operatorname{atan2} \; c_\alpha s_{\zeta_g} c_{\xi+\beta} + s_\alpha c_{\zeta_g}, c_\alpha c_{\zeta_g} c_{\xi+\beta} - s_\alpha s_{\zeta_g} \\ \dot{\psi}_t &= \frac{c_\alpha \dot{\xi}}{\tilde{s}_\psi^2 + \tilde{c}_\psi^2} s_{\xi+\beta} \;\; c_{\zeta_g} \tilde{s}_\psi - s_{\zeta_g} \tilde{c}_\psi \\ \ddot{\psi}_t &= \frac{c_\alpha}{\tilde{s}_\psi^2 + \tilde{c}_\psi^2} \;\; \ddot{\xi} s_{\xi+\beta} + \dot{\xi}^2 c_{\xi+\beta} \;\; (c_\zeta \tilde{s}_\psi - s_\zeta \tilde{c}_\psi) + 2\dot{\xi}\dot{\psi} s_{\xi+\beta} (s_\zeta \tilde{s}_\psi + c_\zeta \tilde{c}_\psi) \;\; , \end{aligned}$$

in the second one.

6.4 Selection of the gripper-closure trajectory

At this stage we can assume that, once a particular gripper-closure trajectory and its duration T_g have been chosen, a method exists to generate the two transfer trajectories that connect it to the initial and final robot states in the minimum times T_i and T_f respectively. If this is true, then the total duration of the grasping depends only on the choice of the gripper-closure sub-trajectory. The variables to be optimized are then the gripper-closure trajectory parameters, i.e.:

1. the class (either horizontal or vertical) of the gripper-closure trajectory;
2. the values of the parameters identifying a trajectory within the chosen class, that is the values of ζ_g, ξ_g and sign Ω_ζ for the horizontal trajectory and those of ζ_g, ξ_g and $\dot{\xi}_g$ for the vertical one;
3. the amount of time T_g during which the chosen gripper-closure trajectory is being followed.

The selection of the optimal parameters can be done, for a given class, using a local optimization routine that is briefly described in algorithm 1. Here χ_g^f and χ_g^f denote the state of the robot at the beginning and at the end of the gripper closure trajectory (i.e. for $t = t_g$ and $t = t_g + T_g$) respectively. By means of the expression terminating conditions we want to indicate a series of checks that are performed at the end of each iteration. These checks are strictly related to the particular optimization algorithm that is being adopted and concern, in general, the number of iterations, the derivatives of the constraint and objective functions, the distance between the current guess and the previous one, and so on.

At the beginning of the computation a class type is chosen together with the sing of Ω_ζ (for the first class of gripper-closure trajectories). This is done in order to avoid the introduction of binary variables that are not well managed by local optimization algorithms. The optimization is then repeated for the remaining class and Ω_ζ sign. Finally the best solution is selected.

6.5 Transfer trajectories generation

In this section we deal with the problem of generating two transfer trajectories connecting a chosen gripper-closure trajectory to the initial and final states of the robot in a minimum time. The resulting algorithm is used in lines 3 and 4 of algorithm 1.

Thanks to the flatness property it is possible to move the trajectory planning problem from the control space to the output space. Since the flat transformation is invertible, as it has been shown in section 4.3, we can transform the conditions on the initial and final states in equivalent conditions on the flat outputs and their derivatives that we indicate with

$$
\begin{aligned}
\varsigma_i &= \quad r_{B,i}, \;\; \dot{r}_{B,i}, \;\; \ddot{r}_{B,i}, \;\; \dot{a}_{B,i}, \;\; \psi_i, \;\; \dot{\psi}_i \\
\varsigma_f &= \quad r_{B,f}, \;\; \dot{r}_{B,f}, \;\; \ddot{r}_{B,f}, \;\; \dot{a}_{B,f}, \;\; \psi_f, \;\; \dot{\psi}_f
\end{aligned}
\qquad (6.20)
$$

Algorithm 1 Compound trajectory optimization

```
Require: Gripper-closure trajectory class, $\chi_i$, $\chi_f$, $r_T$, $\dot{r}_T$
Require: Initial guesses for $\zeta_g$, $\xi_g$, $T_g$ (and possibly $\dot{\xi}_g$).
 1: loop
 2:     Compute the gripper-closure trajectory for $\zeta_g$, $\xi_g$, $T_g$, $(\dot{\xi}_g)$
 3:     Compute a minimum-time initial transfer trajectory from $\chi_i$ to $\chi^i_g$, denote
        its duration with $T_i$ (algorithm 2);
 4:     Compute a minimum-time final transfer trajectory from $\chi^f_g$ to $\chi_f$, denote
        its duration with $T_f$ (algorithm 2);
 5:     Compute the constraint function, i.e., the maximum and minimum values of
        the inputs along the compound trajectory;
 6:     Compute the objective function $T_i + T_g + T_f$;
 7:     if Terminating conditions are satisfied then
 8:         return $\zeta_g$, $\xi_g$, $T_g$, $(\dot{\xi}_g)$
 9:     end if
10:     Select new guesses for $\zeta_g$, $\xi_g$, $T_g$, $(\dot{\xi}_g)$;
11: end loop
```

The values of the flat outputs and their derivatives at the beginning and at the end of the gripper closure trajectory, namely

$$\begin{aligned} \varsigma^i_g &= \quad r^i_{B,g}, \quad \dot{r}^i_{B,g}, \quad \ddot{r}^i_{B,g}, \quad \dot{a}^i_{B,g}, \quad \psi^i_g, \quad \dot{\psi}^i_g \\ \varsigma^f_g &= \quad r^f_{B,g}, \quad \dot{r}^f_{B,g}, \quad \ddot{r}^f_{B,g}, \quad \dot{a}^f_{B,g}, \quad \psi^f_g, \quad \dot{\psi}^f_g \end{aligned} \tag{6.21}$$

can also be easily computed. The problem can then be solved by constructing a trajectory for the flat outputs going from ς_i (ς^f_g) to ς^i_g (ς_f) in a minimum time T_i (T_f) and satisfying eq. (3.5). Note that, as we commented in section 4.3, the components of $\ddot{r}_B$ and $\dot{a}_B$ along the axis z_B are not related to the state but to the thrust input and its derivative. In principle they could then be set at will, within an admissible interval. By imposing the conditions on the whole vectors $\ddot{r}_B$ and $\dot{a}_B$ we are implicitly also guaranteeing the continuity of the thrust input up to the first order of derivation.

In the following we indicate with ς_1 and ς_2 the generic boundary conditions and with t_1 and t_2 the initial and final time. Note that t_1 is fixed: for the first connection it corresponds to t_i which is specified by the original problem; for the second connection it corresponds to $t_g + T_g$ where T_g is fixed by the parent optimization in which this algorithm is nested and t_g is the result of the first transfer trajectory optimization. In both cases the quantity that must be optimized is the final time t_2 or, equivalently, the total duration $T = t_2 - t_1$.

First of all we need to reduce the dimension of the problem to a finite amount, such that it can be then solved using a numerical optimization algorithm. This is done by introducing a parametrization of the flat output trajectories. Such a parametrization consists in expressing the function as a linear combination of a certain number of predefined basis functions

$$f(s) = \sum_{j=1}^{m} p_j B_j(s) . \qquad (6.22)$$

Different parametrization techniques have been proposed in the literature which are characterized by the use of different basis functions. Among the possible solutions we can cite sinusoids, polynomials, Chebyshev polynomials, Laguerre polynomials and Taylor series expansion polynomials. A comparison of these techniques can be found in [9, 28]. Once the basis functions $B_j(s)$ have been selected, the curve is univocally defined by choosing the set of coefficients p_j, according to the conditions imposed by the problem and, possibly, to a certain optimality criteria.

When a high number of basis functions is desired in order to satisfy multiple conditions still leaving some room for optimization, polynomial functions are not a good choice. Indeed to increase the number of parameters in a polynomial we need to increase its degree. Nevertheless a polynomial of degree n can have up to $n - 1$ stationary points (in fact its derivative is a polynomial of degree $n - 1$) and then the number of oscillations grows when n gets bigger.

In these cases it is preferable to use piecewise polynomial curves, also called splines. These functions are obtained as a composition of a certain number of polynomials, each of whom is defined in a limited sub-domain of the overall function domain. The advantage of this solution is that we can increase the number of curve coefficients by increasing the number of polynomial components, while maintaining a low degree of the single polynomials. In particular a spline is said to be of degree n if it is composed by polynomials of degree n. Given the complete function scalar domain

$$D = [s_1, s_m]$$

and a domain partition

$$[s_1, s_2, \ldots, s_{m-1}, s_m]$$

the spline is defined as

$$f(s) = \begin{cases} q_1(s), & \text{for } s \in [s_1, s_2) \\ q_2(s), & \text{for } s \in [s_2, s_3) \\ & \vdots \\ q_{m-1}(s), & \text{for } s \in [s_{m-1}, s_m], \end{cases}$$

where

$$q_j(s) = a_{j0} + a_{j1}(s - s_{j-1}) + \cdots + a_{jn}(s - s_{j-1})^n .$$

The points s_k defining the domain partition are usually called break points. The coefficients a_{jk} must be determined according to the conditions imposed by the specific problem. Typically these conditions affect the values of the function and its derivatives at some points

$$s^{(i)}(s_l) = s_l^i,$$

where in general the points s_l do not coincide with the break points. Assuming that $s_l \in [s_{k-1}, s_k)$ the above conditions will be satisfied if and only if

$$q_{k-1}^{(i)}(s_l) = s_l^i.$$

Note that the spline is infinitely differentiable everywhere (as every polynomial function) except for the break points. To guarantee a desired level of continuity at the break points we must impose appropriate conditions in the form

$$\lim_{t \to s_k^-} s^{(i)}(s) = \lim_{t \to s_k^+} s^{(i)}(s),$$

that is

$$q_{k-1}^{(i)}(s_k) = q_k^{(i)}(s_k).$$

Since the spline is linear in the coefficients, it is easy to verify that the above conditions are linear too and lead to a linear system of equations.

The splines we presented so far are in the so-called piecewise polynomial form or pp-form. A more efficient way to represent splines is in the so-called Basic form or B-form. Also in this case the function, which usually takes the name of B-spline, is defined as a linear combination of basic functions as expressed in eq. (6.22). The computational efficiency of this representation comes from the fact that the basis functions can be computed using a fast and numerically stable recursive algorithm also known as de Boor's algorithm. Letting $s = [s_1, s_2, \ldots, s_{n_{knot}}]$ be a vector of ordered real numbers, not necessarily distinct, called knots, the j-th B-spline of degree n (or equivalently of order $n+1$) is defined as

$$B_j^0(s) = \begin{cases} 1, & \text{for } s \in [s_j, s_{j+1}) \\ 0, & \text{otherwise} \end{cases}$$

$$B_j^n(s) = \frac{s - s_j}{s_{j+n} - s_j} B_j^{n-1}(s) - \frac{s - s_{j+n+1}}{s_{j+n+1} - s_{j+1}} B_{j+1}^{n-1}(s)$$

The B-spline representation can also be easily be extended to the multidimensional case by introducing vector coefficients such that

$$f(s) = \sum_{j=1}^{m} p_j B_j^n(s).$$

The coefficients p_j are called control points.

It is possible to demonstrate the following properties of the B-splines:

Property 1: $B_j^n(s)$ is a piecewise polynomial of degree n;

Property 2: $B_j^n(s)$ has a compact support, i.e. it is equal to zero outside the interval $[s_j, s_{j+n+1}]$;

Property 3: the B-spline basis functions define a partition of the unity, i.e.

$$\sum_{j=1}^{m} B_j^n(s) = 1 \quad \forall s \in [s_1, s_{n_{knot}}];$$

Property 4: in every knot span $[s_i, s_{i+1}]$ at most $n+1$ basis functions are not null, namely $B_{i-n}^n, \ldots B_i^n$. Moreover the change of a control point p_j only modifies the spline in the interval $[s_j, s_{j+n+1}]$.

Property 5: the B-spline is invariant under affine transformations (translation, rotation or scaling) of its control points;

Property 6: the B-spline can be scaled or translated in time by scaling or translating the knot vector. The derivatives will scale or translate accordingly, in particular if $\hat{s} = \lambda s$ then $\hat{s}^{(i)}(t) = \frac{s^{(i)}(t)}{\lambda^i}$;

Property 7: the B-spline always lies within the so called control polygon which is the convex hull of the spline control points. The segments joining consecutive control points also represent a piecewise linear approximation of the curve. In general the lower is the degree of the spline, the better is the linear approximation (if the degree of the spline is one then it actually coincides with the approximation). Moreover the value of the spline at its endpoints is the same as the first and last control points, i.e. $f(s_0) = p_0$ and $s(s_{n_{knots}}) = p_m$;

Property 8: the B-spline is of class C^∞ in the interior of every knot span and it is of class C^{n-k} in a knot of multiplicity k;

Property 9: the number of knots $n_{knots} + 1$ is related to the number m of control points and to the degree n of the spline by $n_{knots} = m + n$.

Property 10: the derivative of a B-spline is also a B-spline of lower degree. Indeed

$$f^{(i)}(s) = \sum_{j=1}^{m} p_j B_j^{n(i)}(s)$$

and it is possible to efficiently compute the i^th order derivative of the basis functions in terms of the basis functions of degree $n-i$ defined on the same knot vector u

$$B_j^{n(i)}(s) = \frac{n!}{(n-i)!} \sum_{k=0}^{i} a_{i,k} B_{j+1}^{n-i}(s)$$

where the coefficients $a_{i,k}$ are defined in a recursive way

$$
\begin{aligned}
a_{0,0} &= 1 \\
a_{i,0} &= \frac{a_{i-1,0}}{s_{j+n-i+1} - s_j} \\
a_{i,k} &= \frac{a_{i-1,k} - a_{i-1,k-1}}{s_{j+n-i+k+1} - s_{j+k}}, \quad \text{for } k = 1, \ldots, i-1 \\
a_{i,i} &= \frac{-a_{i-1,i-1}}{s_{j+n+1} - s_{j+i}}.
\end{aligned}
$$

Thanks to all this properties, B-splines have been widely used in different applications such as computer graphics, data interpolation and trajectory planning. For an exhaustive description of the B-splines and their properties see [28], from which we took the above introduction, or other specific books such as [29, 30]. Trajectory planners for quadrotors based on piecewise polynomials were also proposed in other papers (see for example [1, 10, 26, 9]). Nevertheless these planners cannot be used in our application because they only deal with rest-to-rest motions.

To define a curve in B-form it is necessary to specify:

- the degree n of the spline;
- the knot vector s;
- the control points p_j.

As we already said, to guarantee the continuity of the state, the position must be continuous up to the third order of derivation while the yaw angle must be continuous up to the first order. To keep the degree of the spline as low as possible we use two different splines: one for the position vector and another (scalar) one for the yaw angle. The parameter s can be directly equal to the time and we can chose the knot vector so that all the internal nodes have multiplicity 1. Thanks to property 8, this choice guarantees the maximum possible order of continuity, namely $n - 1$, in the internal knots. We then have to chose $n_{r_B} = 4$ for the position and $n_\psi = 2$ for the yaw angle.

The number m of control points obviously depends on the number of conditions that we want to impose to the spline and on the redundancy we want to keep for further optimization. For each of the two connecting trajectories we must satisfy boundary conditions determined by the initial/final state and by the continuity of the junction with the gripper-closure trajectory. Also in this case the continuity of the state is guaranteed by the continuity of the position up to the third order of derivation and of the yaw angle up to the first order of derivation. This results in a total amount of eight conditions on the position spline and four conditions on the yaw spline (see eqs. (6.20) and (6.21)). Therefore, in order to satisfy these conditions, we need at least eight control points for the position ($m_{r_B} = 8$) and four control points for the yaw ($m_\psi = 4$). If we choose these values we end up with two square linear systems in the control points that can be conveniently written in a matrix form

$$A_{r_B} P_{r_B} = B_{r_B},$$
$$A_\psi p_\psi = b_\psi,$$

where the system variables are

$$P_{r_B} = \begin{matrix} p^T_{r_B,1} \\ p^T_{r_B,2} \\ \vdots \\ p^T_{r_B,8} \end{matrix} \quad , \quad p_\psi = \begin{matrix} p_{\psi,1} \\ p_{\psi,2} \\ \vdots \\ p_{\psi,4} \end{matrix} \quad .$$

The coefficients matrices A_{r_B} and A_ψ contain the values of the B-spline basis functions and their derivatives at the initial and final times:

$$A_{r_B} = \begin{matrix} B^4_{r_B,1}(t_1), & B^4_{r_B,2}(t_1), & \dots, & B^4_{r_B,8}(t_1) \\ B^{4(1)}_{r_B,1}(t_1), & B^{4(1)}_{r_B,2}(t_1), & \dots, & B^{4(1)}_{r_B,8}(t_1) \\ B^{4(2)}_{r_B,1}(t_1), & B^{4(2)}_{r_B,2}(t_1), & \dots, & B^{4(2)}_{r_B,8}(t_1) \\ B^{4(3)}_{r_B,1}(t_1), & B^{4(3)}_{r_B,2}(t_1), & \dots, & B^{4(3)}_{r_B,8}(t_1) \\ B^4_{r_B,1}(t_2), & B^4_{r_B,2}(t_2), & \dots, & B^4_{r_B,8}(t_2) \\ B^{4(1)}_{r_B,1}(t_2), & B^{4(1)}_{r_B,2}(t_2), & \dots, & B^{4(1)}_{r_B,8}(t_2) \\ B^{4(2)}_{r_B,1}(t_2), & B^{4(2)}_{r_B,2}(t_2), & \dots, & B^{4(2)}_{r_B,8}(t_2) \\ B^{4(3)}_{r_B,1}(t_2), & B^{4(3)}_{r_B,2}(t_2), & \dots, & B^{4(3)}_{r_B,8}(t_2), \end{matrix}$$

and

$$A_\psi = \begin{matrix} B^2_{\psi,1}(t_1), & B^2_{\psi,2}(t_1), & \dots, & B^2_{\psi,4}(t_1) \\ B^{2(1)}_{\psi,1}(t_1), & B^{2(1)}_{\psi,2}(t_1), & \dots, & B^{2(1)}_{\psi,4}(t_1) \\ B^2_{\psi,1}(t_2), & B^2_{\psi,2}(t_2), & \dots, & B^2_{\psi,4}(t_2) \\ B^{2(1)}_{\psi,1}(t_2), & B^{2(1)}_{\psi,2}(t_2), & \dots, & B^{2(1)}_{\psi,4}(t_2) \end{matrix}$$

Algorithm 2 Transfer trajectory generation

```
Require: $\varsigma_1$, $\varsigma_2$
Require: Initial guess for T (transfer duration).
1: loop
2:    Compute the interpolating B-spline for time T;
3:    Compute the constraint function, i.e., the maximum and minimum values of
      the inputs along the transfer trajectory;
4:    Compute the objective function T;
5:    if Conditions for terminating are satisfied then
6:       return T
7:    end if
8:    Select new guess for T;
9: end loop
```

Finally the known terms are determined by the boundary conditions ς_1 and ς_2:

$$B_{r_B} = \begin{bmatrix} r^T_{B,1} \\ \dot{r}^T_{B,i} \\ \ddot{r}^T_{B,1} \\ \dot{a}^T_{B,1} \\ r^T_{B,2} \\ \dot{r}^T_{B,2} \\ \ddot{r}^T_{B,2} \\ \dot{a}^T_{B,2} \end{bmatrix}, \quad b_\psi = \begin{bmatrix} \psi_1 \\ \dot{\psi}_1 \\ \psi_2 \\ \dot{\psi}_2 \end{bmatrix}.$$

The system has a unique solution, provided that $t_1 = t_2$ and that the knots are properly chosen. Moreover thanks to property 6 the curve is fully defined once $T = t_2 - t_1$ has been fixed. This can be done in an optimal way: the time T must be the minimum time such that the resulting trajectory is feasible, that is such that the quadrotor can follow the resulting trajectory without violating the limits on the propeller rotational speed. The problem can be then put in the form:

$$\min_{\{T\}} \quad T \tag{6.23a}$$

$$\text{s.t.} \quad \omega_i(t) \in [\underline{\omega}, \overline{\omega}] \quad \forall t \in [t_1, t_2], i = 1, \ldots, 4 \tag{6.23b}$$

$$T > 0 \tag{6.23c}$$

Also for this problem we used a local optimization method. The procedure is described in algorithm 2.

6.6 Considerations on the existence of solutions

In section 4.2 we have demonstrated that the robot can move from any state to any other, provided that the limits on the propeller rotational speeds are not too strict.

As a consequence, whatever are the initial and final states and the chosen (feasible) gripper-closure trajectory, there always exist two feasible transfer trajectories that link them in a compound trajectory. Nevertheless with the introduction of the B-spline parametrization, we have reduced the search space so that it may not contain these feasible transfer trajectories. Moreover even when our search space contains some feasible solutions, it may not contain the optimal one in the sense that we might still find a better solution if we enlarged the search space.

Let us consider the problem of generating a single rest-to-rest trajectory, i.e. a trajectory between two hovering states. In this simple case the desired values of all the derivatives of the flat outputs at the beginning and at the end of the trajectory are obviously zero. Now assume that a trajectory has been found, using B-spline interpolation, that goes from the initial state to the final one in a time T. Also assume that after computing the value of the inputs along the trajectory we realize that these are not feasible. Now if we scale the knot vector by a factor $\lambda > 1$ such that $\hat{s} = \lambda s$ then, thanks to property 6, the derivatives of the spline will uniformly scale by a factor λ^i. The new spline will still satisfy the boundary conditions because the scaling will keep the initial and final values of the derivatives at zero as they were. Moreover since the derivatives scale, for the results of section 4.2, the robot will tend to perform the trajectory in a quasi hovering condition with the inputs close to their minimum value $u_{min} = (mg, 0, 0, 0)^T$. As a consequence the trajectory will eventually become feasible, provided that λ is big enough and that u_{min} is an interior point of the set U of admissible inputs. In this case it is then always possible to find a feasible B-spline trajectory. This property has been exploited in [11] to generate feasible point-to-point trajectories in an efficient way.

In the more general case in which we have non zero initial and/or final values of the flat output derivatives, this property does not hold anymore. Indeed if we scaled the knot vector, the derivatives would change everywhere and also at the initial and final instants. As a consequence the boundary conditions would no longer be satisfied. If we want to complete the trajectory in a longer time interval we need to repeat the interpolation with the new knot vector, i.e. with different coefficient matrices A_{r_B} and A_{ψ}. The interpolation will lead to a new trajectory which, in general, is completely different from the previous one apart from the initial and final points. The new trajectory does not necessarily have smaller derivatives of the flat outputs and then it does not necessarily imply a reduction of the input effort. As a consequence the existence of a feasible solution is no longer guaranteed.

To convince ourselves about this fact we consider a numerical example: we try to connect a fixed hovering initial state of the robot to the start point of a non hovering ($\xi_g < \pi - \beta$) horizontal circular trajectory. We let the time T change from an initial value T_0 to a final one T_1 and we observe how the maximum and minimum values of the inputs correspondingly change. More in details we start from hover at $\sigma_i = (0, -10, 2, 0)^T$ and we want to connect to the horizontal circular trajectory corresponding to $\zeta_g = 0$ and $\xi_g = 0.5$, resulting in the following boundary conditions:

$$\begin{aligned}
r_{B,f} &= \begin{pmatrix} 0.3894 & 0 & 0.9211 \end{pmatrix}^T \\
\dot{r}_{B,f} &= \begin{pmatrix} 0 & 1.2449 & 0 \end{pmatrix}^T \\
\ddot{r}_{B,f} &= \begin{pmatrix} -3.9798 & 0 & 0 \end{pmatrix}^T \\
\dot{a}_{B,f} &= \begin{pmatrix} 0 & -12.7227 & 0 \end{pmatrix}^T \\
\psi_f &= 2.3181 \\
\dot{\psi}_f &= 3.1968.
\end{aligned}$$

The time T is made vary from 4s to 28s. The values of minimum and maximum thrust and torque inputs along the resulting trajectories are depicted in fig. 6.6. As we see while the minimum thrust is almost constant, the maximum grows with T. Concerning the torques, the first two seem almost constant, but for the third one the minimum decreases with T while the maximum grows. If we look at the maximum and minimum values of the thrusts generated by the single propellers (fig. 6.7) we notice that all the quantities show a stationary point (either a minimum or a maximum) and only some of them eventually enter the admissible interval, denoted by dashed blue lines. The reason for that is made clear by figs. 6.8 and 6.9 where we can see that an increase of T cause bigger deviations of the trajectories from the straight line joining the initial and final positions.

6.7 Introduction of further degrees of freedom

The probability of finding a feasible solution, and in general the quality of the sub-optimal solution found, is obviously proportional to the dimension of the search space. It is indeed a general property of the B-splines, that of being able to approximate any smooth curve when the number of control points go to infinity. In the algorithm described in section 6.5 the only variable which is kept free for the optimization is the interpolating time T. As soon as this is specified the trajectory is fully determined. This happens because we decided to use the minimum possible amount of control points needed to satisfy the given boundary conditions. As we already commented, one of the advantages of using B-splines (and piecewise polynomials in general) for the interpolation is that we can easily introduce more

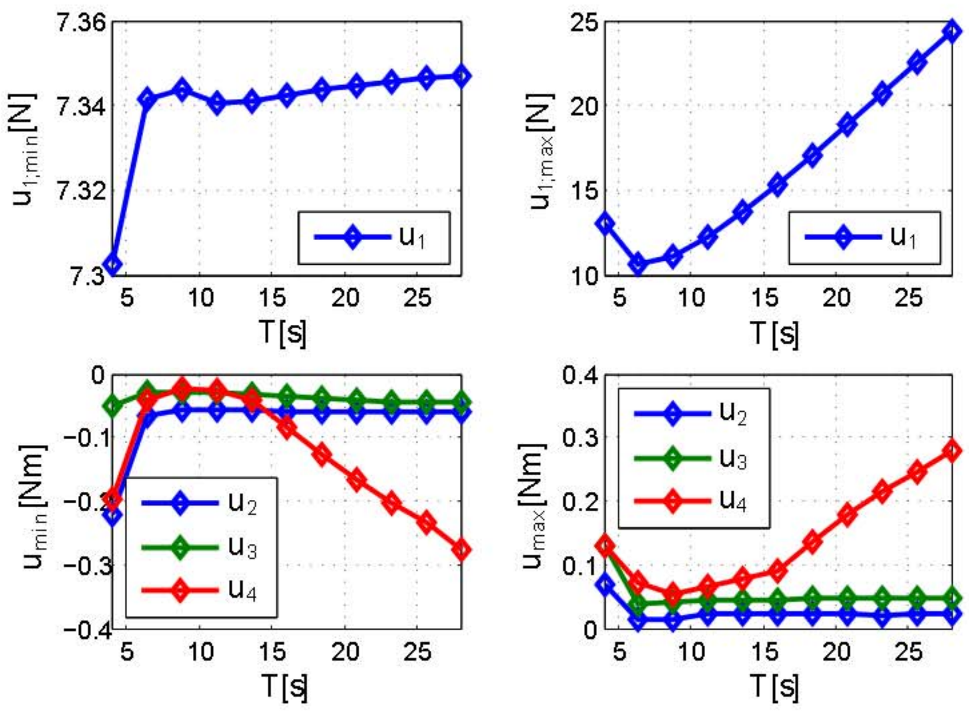


Figure 6.6: Minimum and maximum values of the thrust/torque inputs for different total times T

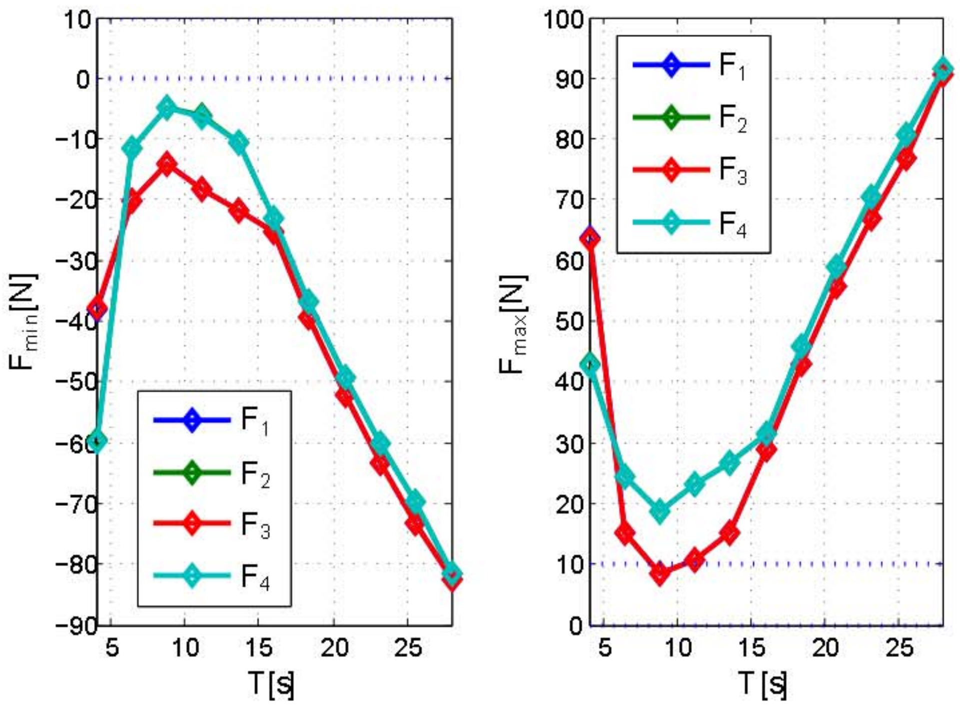


Figure 6.7: Minimum and maximum values of the single thrusts for different total times T. Note that the four traces are superimposed in pairs.

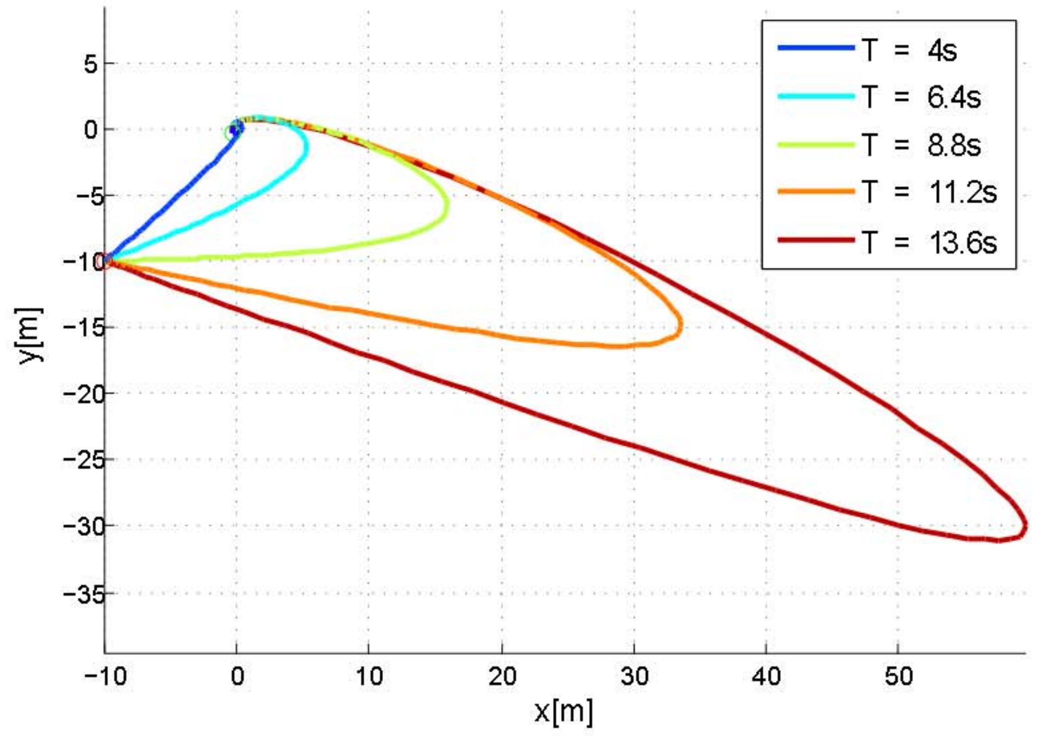


Figure 6.8: XY trajectories for different total times T

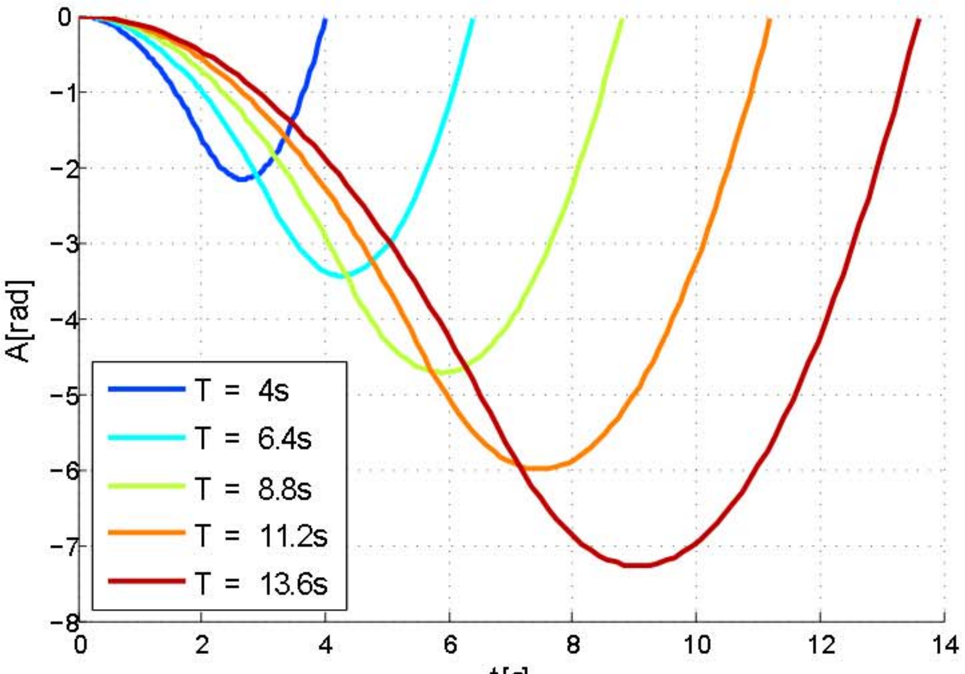


Figure 6.9: Yaw angle trajectories for different total times T

degrees of freedom (i.e. more control points) without increasing the degree of the single polynomials. In this way we can increase the dimension of the search space at will. On one hand we can exploit these redundant degrees of freedom to get a better solution but, on the other hand, the problem will obviously become more complex to handle from a numerical point of view. We will still end up with a linear system of the form

$$A_{r_B} P_{r_B} = B_{r_B},$$
$$A_\psi p_\psi = b_\psi.$$

but if the number of control points is greater than the number of conditions, then the matrices A_{r_B} and A_ψ are wide matrices instead of square ones, i.e. they have more columns than rows. Moreover, thanks to property 4 the central columns of the coefficient matrices are null. The corresponding control points are then undefined and can be moved freely without affecting the satisfaction of the boundary conditions. In particular we can position these redundant control points in such a way that the control effort of the propellers, for a given interpolation time T, is minimized. In practice we have to nest another optimization routine in the previous two. Also this optimization requires the specification of an objective function and, possibly, of some constraints. We tried two different approaches that will be described in the following sections.

Minimum snap trajectories

In this case we used a similar strategy to the one adopted in [11] for fixing the redundant control points: since the torque control inputs are proportional to the position snap (fourth derivative) and to the yaw acceleration, we try to minimize both by using an objective function of the form:

$$V(P_{r_B}, p_\psi) = \frac{1}{2} \int_{t_1}^{t_2} \left(c_1 \left\| \ddot{a}_B(t) \right\|^2 + c_2 \ddot{\psi}^2(t) \right) dt$$

where c_1 and c_2 are two positive scalar weights. Note that for the linearity of the integral operation we can minimize the above optimization function by separately minimizing the integral of the fist and second term of the sum. As a consequence the weights have no effect on the final solution. Moreover since

$$\left\| \ddot{a}_B(t) \right\|^2 = \ddot{a}^2_{B,x}(t) + \ddot{a}^2_{B,y}(t) + \ddot{a}^2_{B,z}(t)$$

we can also separately optimize each scalar component of the snap.

Lets consider for simplicity the minimization of the yaw angle acceleration. We have

$$\psi(t) = \sum_{j=1}^{m} p_{\psi,j} B^2_{\psi,j}(t),$$

and

$$\ddot{\psi}(t) = \sum_{j=1}^{m} p_{\psi,j} B_{\psi,j}^{2(2)}(t),$$

then the objective function can be written as

$$V_\psi(p_\psi) = \frac{1}{2}\int_{t_1}^{t_2} \ddot{\psi}^2(t)\,dt = \frac{1}{2}\int_{t_1}^{t_2}\left(\sum_{j=1}^{m} p_{\psi,j} B_{\psi,j}^{2(2)}(t)\right)^2 dt$$
$$= \frac{1}{2}\sum_{j=1}^{m} p_{\psi,j}^2 \int_{t_1}^{t_2}\left(B_{\psi,j}^{2(2)}(t)\right)^2 dt + \sum_{j\neq k} p_{\psi,j}\, p_{\psi,k}\int_{t_1}^{t_2} B_{\psi,j}^{2(2)}(t)\, B_{\psi,k}^{2(2)}(t)\, dt$$

and we can put it in a matrix form as

$$V_\psi(p_\psi) = \frac{1}{2} p_\psi^T Q_\psi p_\psi$$

where the matrix Q_ψ is symmetric and positive definite

$$Q_\psi = \int_{t_1}^{t_2} \begin{bmatrix} \left(B_{\psi,1}^{2(2)}(t)\right)^2 & \dots & B_{\psi,1}^{2(2)}(t)\, B_{\psi,m_\psi}^{2(2)}(t) \\ & \ddots & \vdots \\ & & \left(B_{\psi,m_\psi}^{2(2)}(t)\right)^2 \end{bmatrix} dt.$$

We can repeat the same process also for the snap obtaining an objective function of the form

$$V_{r_B}(P_{r_B}) = \frac{1}{2} P_{r_B}^T Q_{r_B} P_{r_B}$$

where

$$Q_{r_B} = \int_{t_1}^{t_2} \begin{bmatrix} \left(B_{r_B,1}^{4(4)}(t)\right)^2 & \dots & B_{r_B,1}^{4(4)}(t)\, B_{r_B,m_{r_B}}^{4(4)}(t) \\ & \ddots & \vdots \\ & & \left(B_{r_B,m_{r_B}}^{4(4)}(t)\right)^2 \end{bmatrix} dt.$$

In both cases, considering also the interpolation conditions, we obtain a linear quadratic optimization problem in the general form

$$\min_x \ \frac{1}{2} x^T Q x$$
$$\text{s.t. } Ax = c.$$

The problem is obviously convex and then it always has a global optimum. Moreover it can be solved numerically using very efficient algorithms. Nevertheless the use of this technique did not induce a significant reduction of the minimum interpolation time, but did noticeably increase the computational time needed to compute the trajectory.

Direct inputs minimization

The previous technique has the significant advantage that the resulting optimization problem is convex and can be efficiently solved using numerical methods. On the other way the inputs of the system do not depend only on the minimized quantities but also on lower order derivatives of the flat outputs.

Another possibility would be to directly reduce the inputs by solving the following optimization problem

$$\min_{\{P_{r_B}, p_\psi\}} \quad \max \quad \max_{\{t,i\}} \{\omega_i(t)\} - \overline{\omega}, \underline{\omega} - \min_{\{t,i\}} \{\omega_i(t)\} \quad .$$

We tried to solve this optimization using the previously described one to generate an initial guess for the control points. Nevertheless we were not able to demonstrate the existence of a solution for this problem and the introduction of another optimization step caused a further increase of the necessary computational resources, such that the algorithm was prematurely terminated due to a lack of system memory.

These numerical difficulties suggested us to limit the number of control points at its smallest amount.

Chapter 7

Planner implementation and examples

In this chapter we provide some details about the implementation of the above described algorithm and we present some numerical results.

For the general optimization (algorithm 1) a Sequential Quadratic Programming (SQP) algorithm has been used. This algorithm consists of solving a sequence of optimization sub-problems, each of whom optimizes a quadratic approximation of the objective function, subject to a linearization of the constraints. More in details, for a general optimization problem of the form

$$\begin{aligned} \min_{\{x\}} \quad & f(x) \\ \text{s.t.} \quad & g(x) \leq 0 \\ & h(x) = 0, \end{aligned}$$

we define the Lagrangian as

$$L(x, \lambda, \mu) = f(x) + \lambda^T b(x) + \mu^T c(x) \tag{7.1}$$

where λ and μ are the Lagrange multipliers. At algorithm iteration k, the search direction d_k for the new guess is obtained as the solution of the following linear quadratic problem:

$$\begin{aligned} \min_{\{d_k\}} \quad & L(x_k, \lambda_k, \mu_k) + \nabla L(x_k, \lambda_k, \mu_k)^T d_k + \frac{1}{2} d_k^T \nabla^2_{xx} L(x_k, \lambda_k, \mu_k) d_k \\ \text{s.t.} \quad & g(x_k) + \nabla g(x_k)^T d_k \leq 0 \\ & h(x_k) + \nabla h(x_k)^T d_k = 0. \end{aligned}$$

The new guess is then computed as

$$\begin{matrix} x_{k+1} \\ \lambda_{k+1} \\ \mu_{k+1} \end{matrix} = \begin{matrix} x_k \\ \lambda_k \\ \mu_k \end{matrix} + \beta d_k .$$

where β is an appropriately chosen positive parameter. In particular in this thesis we used an implementation of the SQP provided by the Matlab® Optimization Toolbox through the function fmincon. For more information about the LQR algorithm see [31]. For details about the implementation please refer to Matlab® documentation.

As we already said, the class of circular trajectories to be used is fixed, at the beginning of the computation, together with the sing of Ω_ζ (for the first class of grasping trajectories). This has been done in order to avoid the introduction of binary variables which are not well managed by local optimization algorithms. However the optimization can be easily repeated for all the possible choices and then one can select the best solution.

Concerning the transfer trajectories, for the sake of efficiency of computation, we transformed eq. (6.23) in an unconstrained optimization problem. The constraints defined by eq. (6.23b) can indeed be written in the form

$$\max_{\{t,i\}}\{\omega_i(t)\} \leq \overline{\omega}$$
$$\min_{\{t,i\}}\{\omega_i(t)\} \geq \underline{\omega}$$

or, equivalently,

$$\max \quad \max_{\{t,i\}}\{\omega_i(t)\} - \overline{\omega}, \underline{\omega} - \min_{\{t,i\}}\{\omega_i(t)\} \quad \leq 0. \tag{7.2}$$

Since the objective function does not have any stationary point, the optimal solution must lie on the boundary of the admissible set, i.e. it must satisfy eq. (7.2) with the equal sign. We can then try to solve the problem in an alternative unconstrained form

$$\min_{\{T\}} \quad \max \quad \max_{\{t,i\}}\{\omega_i(t)\} - \overline{\omega}, \underline{\omega} - \min_{\{t,i\}}\{\omega_i(t)\} \quad ^2 .$$

The optimization has been solved using a numerical local algorithm. Since the problem has been transformed in an unconstrained optimization, we can use the Nelder-Mead simplex direct search solver available in the Matlab® function fminsearch. This algorithm does not require the knowledge of any derivative of the optimization function and can also recover from Not a Number intermediate solutions. The latter property is necessary because we return a NaN whenever the generated guess for the interpolation time T is negative. We adopted this strategy to guarantee that eq. (6.23c) is satisfied.

Note that this unconstrained problem is not equivalent to the previous one. Indeed the solution of this problem might correspond to a positive value of the optimization function. If this happens the final solution is either not optimal (if $\max \ \max_{\{t,i\}}\{\omega_i(t)\} - \overline{\omega}, \underline{\omega} - \min_{\{t,i\}}\{\omega_i(t)\} \ < 0$) or not admissible (when $\max \ \max_{\{t,i\}}\{\omega_i(t)\} - \overline{\omega}, \underline{\omega} - \min_{\{t,i\}}\{\omega_i(t)\} \ > 0$). In the latter case the solution must obviously be rejected. To take care of this aspect we pass the value of eq. (7.2) (before squaring) to the parent optimization where it is still being considered as a hard constraint.

The initial guess for T needed by the numerical solver has always been chosen proportional to the euclidean distance between the initial and final position, through an heuristic constant v

$$T = \frac{1}{v} \ r_{B,1} - r_{B,2} \ .$$

Since an explicit expression of the dependence of eq. (7.2) on T is not available, for a given T we compute the value of the inputs along the resulting trajectory with a fixed time resolution using the flat transformation described in chapter 4. Then we search the maximum and minimum values that the inputs assume. A remark must be done concerning the computation of the actual inputs from the force/torques inputs using eq. (2.6). The above transformation can indeed be written in terms of the forces generated by the single propellers

$$u = \begin{matrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{matrix} = \begin{matrix} 1 & 1 & 1 & 1 \\ 0 & l & 0 & -l \\ -l & 0 & l & 0 \\ \frac{k_M}{k_F} & -\frac{k_M}{k_F} & \frac{k_M}{k_F} & -\frac{k_M}{k_F} \end{matrix} \begin{matrix} F_1 \\ F_2 \\ F_3 \\ F_4 \end{matrix} \ .$$

This second form has been preferred to the previous one because an approximate measure of the maximum and minimum admissible values of the single propeller thrust for a real quadrotor was available.

Finally, for the computation, derivation and integration of the B-splines, we used the Curve Fitting Toolbox in Matlab®.

We will now present some trajectories that have been obtained using the previously described algorithm. The robot parameters that have been used for the planning are collected in table 7.1.

Parameter	Value
m	$0.74906\,\mathrm{kg}$
J	$\begin{bmatrix} 1.008\times10^{-2} & -1.679\times10^{-4} & -2.443\times10^{-10} \\ -1.679\times10^{-4} & 1.027\times10^{-2} & -4.740\times10^{-6} \\ -2.443\times10^{-10} & -4.740\times10^{-6} & 1.946\times10^{-2} \end{bmatrix}\,\mathrm{kg\,m^2}$
α	$\frac{\pi}{4}$ rad
β	$\frac{3}{4}\pi$ rad
${}^{B}r_G$	$1\,\mathrm{m}$
l	$25\,\mathrm{cm}$
k_M / k_F	$3.5\,\mathrm{mm}$
$\overline{F}_i$	$10\,\mathrm{N}$
$\underline{F}_i$	$0\,\mathrm{N}$
$T_{g_{min}}$	$0.1\,\mathrm{s}$

Table 7.1: Physical robot parameters used for planning and simulation

7.1 Case of a fixed target

In the first simulation that we present the robot is initially in a hovering state with:

$$r_{B,i} = \begin{bmatrix} 0 & 4 & 3 \end{bmatrix}^T$$

and is required to reach another hovering state with

$$r_{B,f} = \begin{bmatrix} 0 & -4 & 3 \end{bmatrix}^T$$

after grasping a target that is positioned in

$$r_T = \begin{bmatrix} 0 & 0 & 0.478 \end{bmatrix}^T$$

and has zero velocity. The initial and final yaw angles are equal and have been chosen in such a way that at the beginning of the simulation the gripper points toward the position of the target ($\psi_i = -\frac{3}{4}\pi$).

We started the optimization from an initial guess in which the robot stops in a position close to the target and remains still with respect to the target for the whole time needed by the gripper to close. More in detail the initial guesses for the optimization variables have been chosen as follows

$$\zeta_g = \frac{\pi}{2}, \quad \xi_g = \pi - \beta, \quad \dot{\xi}_g = 0, \quad T_g = T_{g_{min}}$$

As we have already commented, the existence of a solution in this case is always guaranteed. The optimization returned a solution equal to the initial guess both for the vertical and for the horizontal gripper-closure trajectories. Since in these conditions the two classes of gripper-closure trajectories coincide, also the

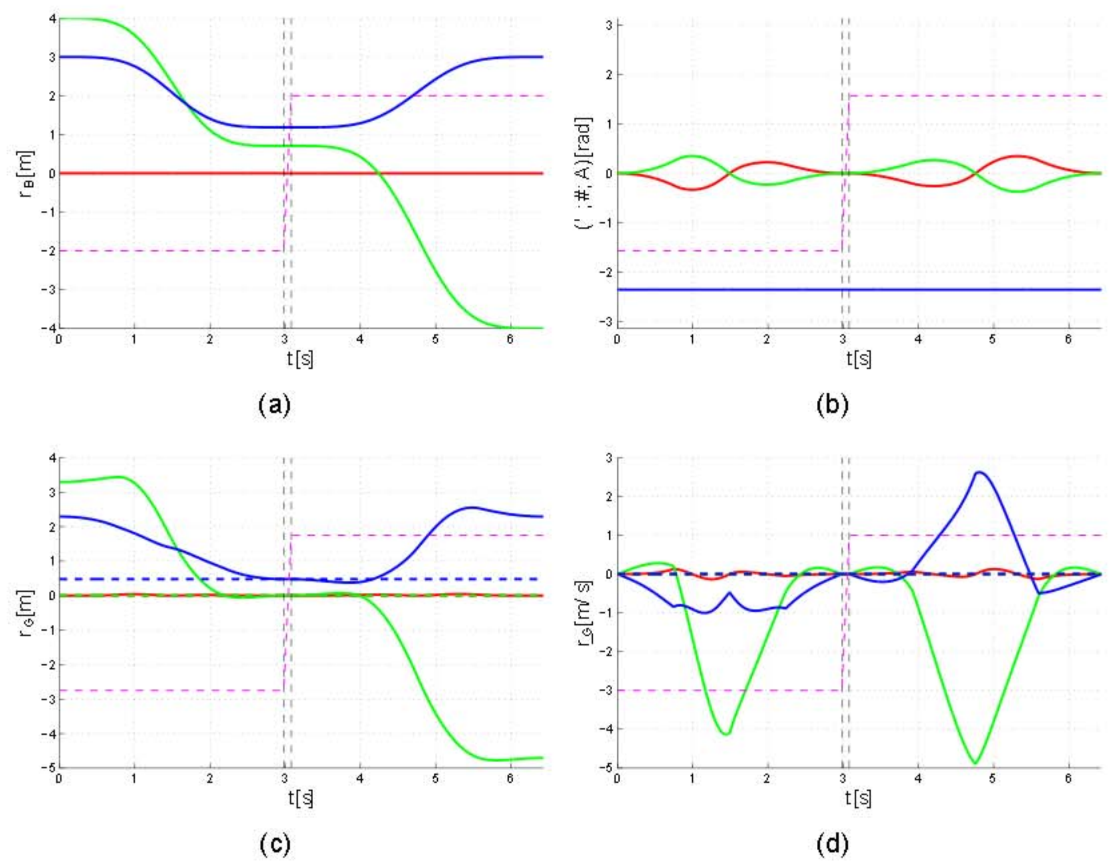


Figure 7.1: Grasping of a fixed target using a hovering gripper-closure trajectory with the first set of initial and final robot states: position (a) and velocity (c) of O_G (continuous line) and target (dashed line); position (b) and orientation (d) of the robot. A magenta dashed line is superimposed to every plot, indicating the gripper state: from completely open (lower value) to completely closed (upper value).

compound trajectories are the same. The results are reported in fig. 7.1. We can see that, during the grasping interval T_g, the gripper is actually positioned on the target. Note that in this situation, in which both the robot and the target are in a fixed position with respect to the world frame, the problem reduces to a simple point-to-point motion planning and is not particularly interesting for our study.

7.2 Case of a moving target

We repeated the same simulation as before, but with a target moving along the y_W axis at a constant velocity of 1 m/s. Also in this case the optimization did not modify the initial guess and the two classes of gripper-closure trajectories produced the same result that is represented in fig. 7.2. We can notice that, since the target is moving toward the desired final position of the robot, the quadrotor maintains a higher average velocity and the total time needed to complete the grasping is

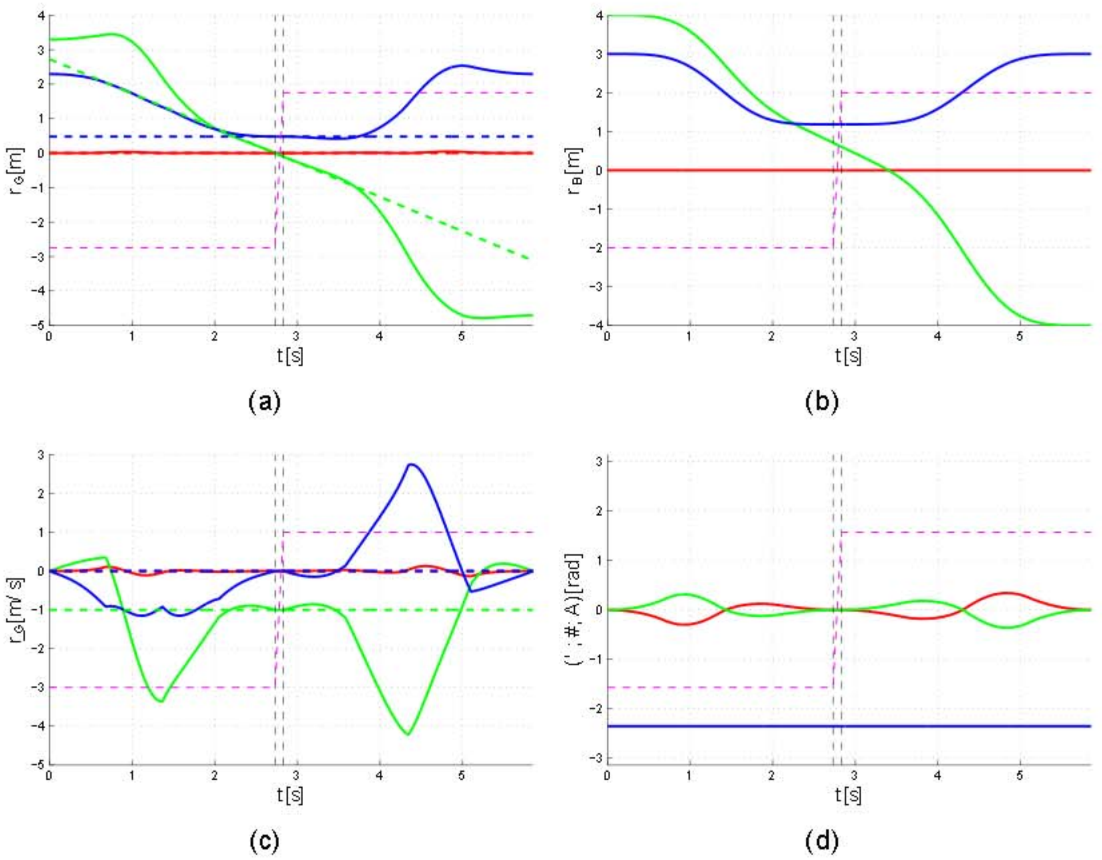


Figure 7.2: Grasping of a moving target using a hovering gripper-closure trajectory with the first set of initial and final robot states: position (a) and velocity (c) of O_G (continuous line) and target (dashed line); position (b) and orientation (d) of the robot. A magenta dashed line is superimposed to every plot, indicating the gripper state: from completely open (lower value) to completely closed (upper value).

consequently shorter.

7.3 Influence of the initial guesses

In order to better understand the nature of the optimization problem, we repeated the same simulations as before, but providing a different initial guess to the optimization algorithm. We maintained the same guesses for ζ_g and T_g, but we changed the values of ξ_g and $\dot{\xi}_g$ so that in the resulting gripper-closure trajectory the robot has a non zero velocity with respect to the target and then it is actually following a circular path. In particular we used $\xi_g = \pi - \beta - 0.3$ for the horizontal gripper-closure trajectory and $\xi_g = \pi - \beta - 0.1$ and $\dot{\xi}_g = -0.1$ for the vertical one. Even in this case the optimization returned values of the optimization variables that are very close to the initial guess. Nevertheless, while for the vertical gripper-closure trajectory this resulted in a slight shortening of the total duration of the grasping,

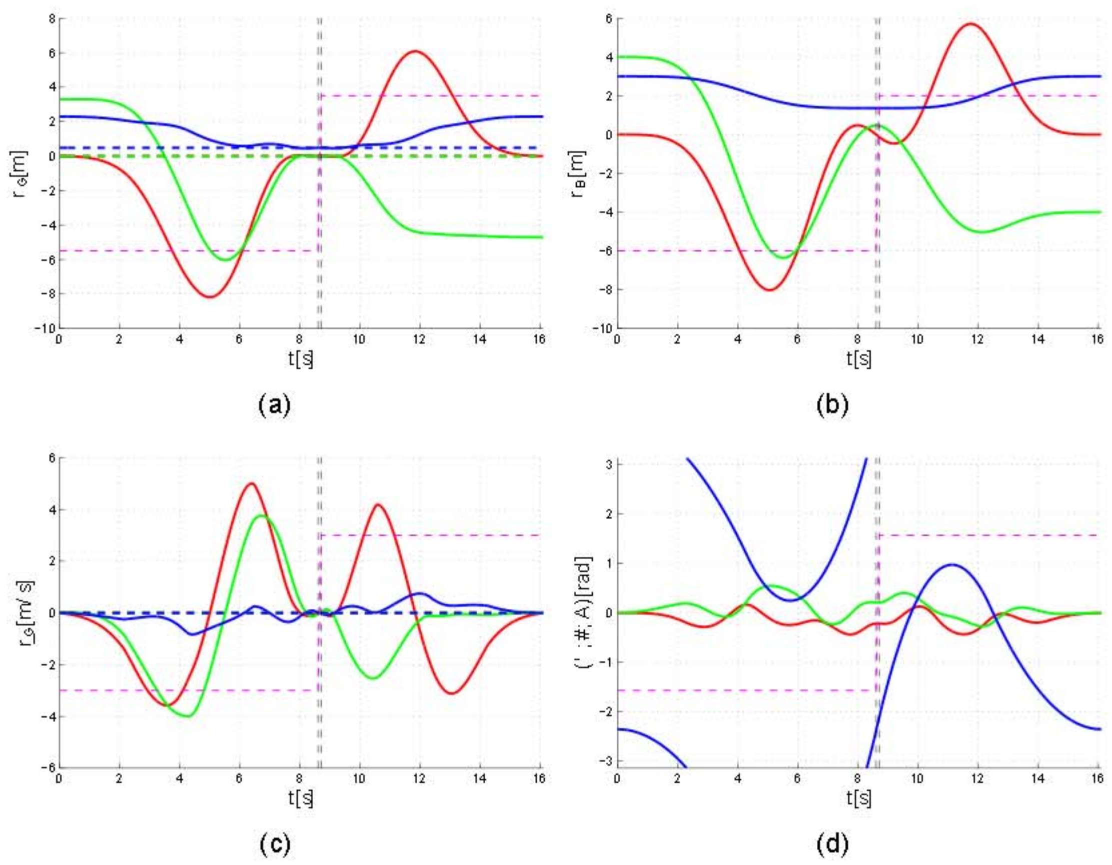


Figure 7.3: Grasping of a fixed target using a non-hovering horizontal gripper-closure trajectory: position (a) and velocity (c) of O_G (continuous line) and target (dashed line); position (b) and orientation (d) of the robot. A magenta dashed line is superimposed to every plot, indicating the gripper state: from completely open (lower value) to completely closed (upper value).

for the horizontal gripper-closure trajectory the total time grew noticeably (see figs. 7.3 to 7.6). This demonstrates that the numerical optimization gets stuck in local minima and is not able to find the actual optimum.

In this case, since the robot is actually following a circular path during the gripper-closure phase, the two trajectories are different as we can see from the time profile of the roll, pitch and yaw angles in figs. 7.3 to 7.6(d). We can notice that, during the grasping, in the case of a horizontal gripper-closure trajectory, the roll and pitch angles remain constant and the yaw changes at constant velocity. In the case of the vertical gripper-closure trajectory, instead, the yaw remains constant during the grasping, while the roll and pitch angles change.

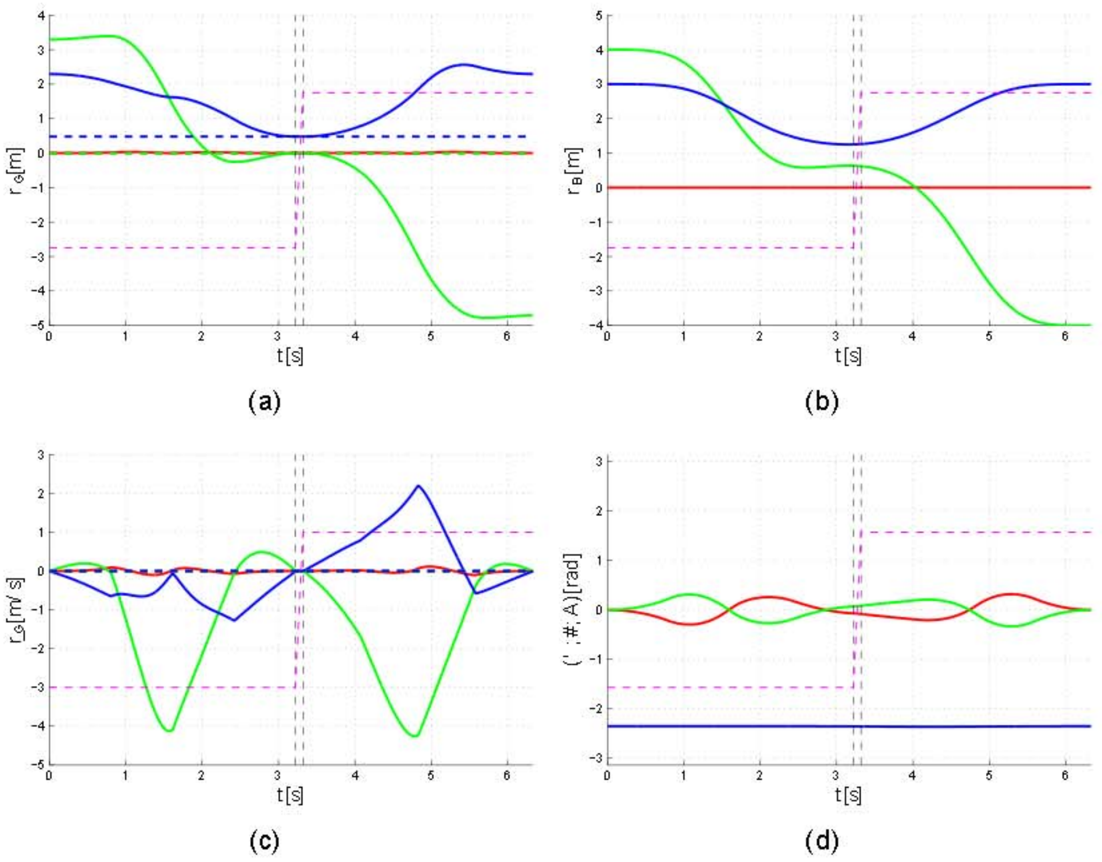


Figure 7.4: Grasping of a fixed target using a non-hovering vertical gripper-closure trajectory: position (a) and velocity (c) of O_G (continuous line) and target (dashed line); position (b) and orientation (d) of the robot. A magenta dashed line is superimposed to every plot, indicating the gripper state: from completely open (lower value) to completely closed (upper value).

7.4 Influence of the initial/final states

The same kind of simulations have been repeated for different values of the initial and final robot states. The robot always starts and ends its motion in a still state but now we have

$$r_{B,i} = \begin{matrix} -1 & 4 & 3 \end{matrix}^T, \quad \psi_i = 0$$

and

$$r_{B,f} = \begin{matrix} 3 & -4 & 3 \end{matrix}^T, \quad \psi_f = \frac{\pi}{2}.$$

The target is positioned as before, possibly with a velocity of 1 m/s along y_W.

Again the local optimization has proven inadequate to find the best solution, always returning results that are very close to the initial guesses. Also in this case when the grasping is performed with the robot standing still with respect to the target, the two classes of gripper-closure trajectories are perfectly equivalent. Moreover, when also the target is fixed with respect to the world frame, the problem

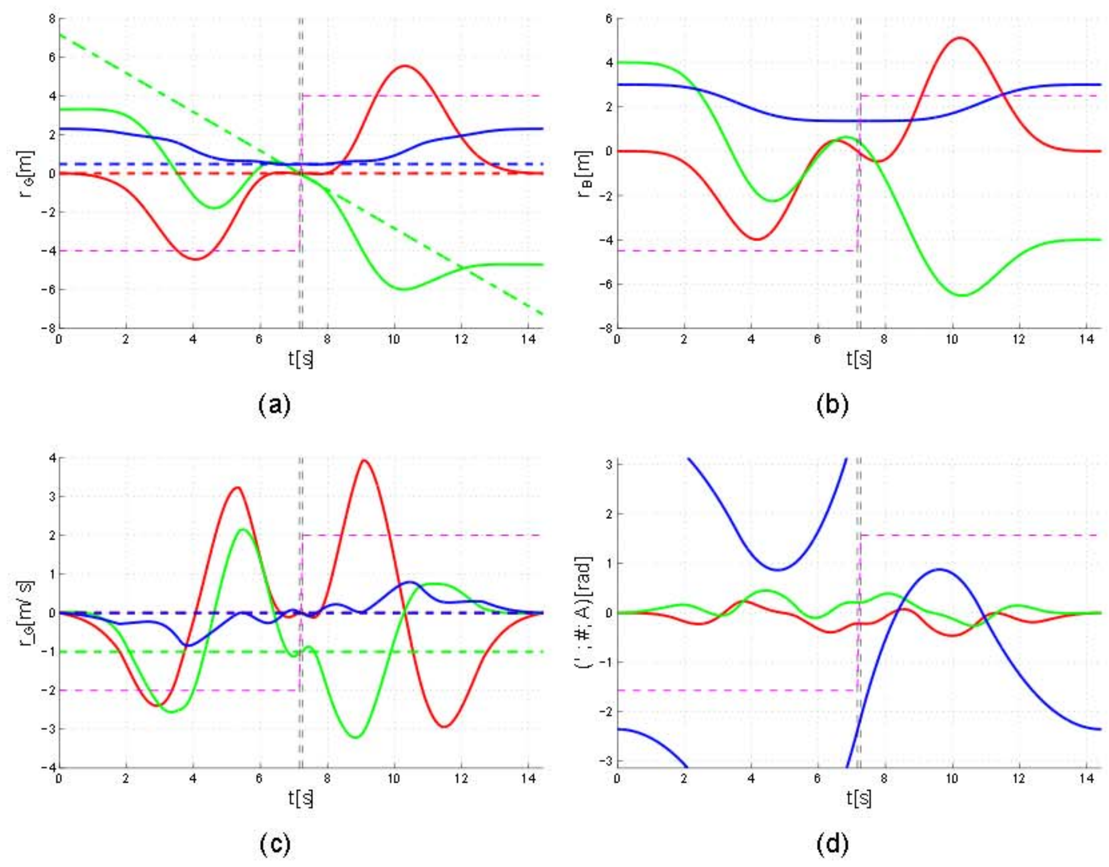


Figure 7.5: Grasping of a moving target using a non-hovering horizontal gripper-closure trajectory: position (a) and velocity (c) of O_G (continuous line) and target (dashed line); position (b) and orientation (d) of the robot. A magenta dashed line is superimposed to every plot, indicating the gripper state: from completely open (lower value) to completely closed (upper value).

reduces to a simple point to point motion planning and the results obtained for different initial and final states are not particularly interesting.

It is instead interesting to mention that in this case the use of a non hovering initial guess resulted in a better solution both for the vertical and for the horizontal gripper-closure trajectories. Moreover the use of a horizontal gripper-closure trajectory resulted in a better solution with respect to the vertical one. The results obtained with non hovering initial guesses and for a moving target are represented in figs. 7.7 and 7.8.

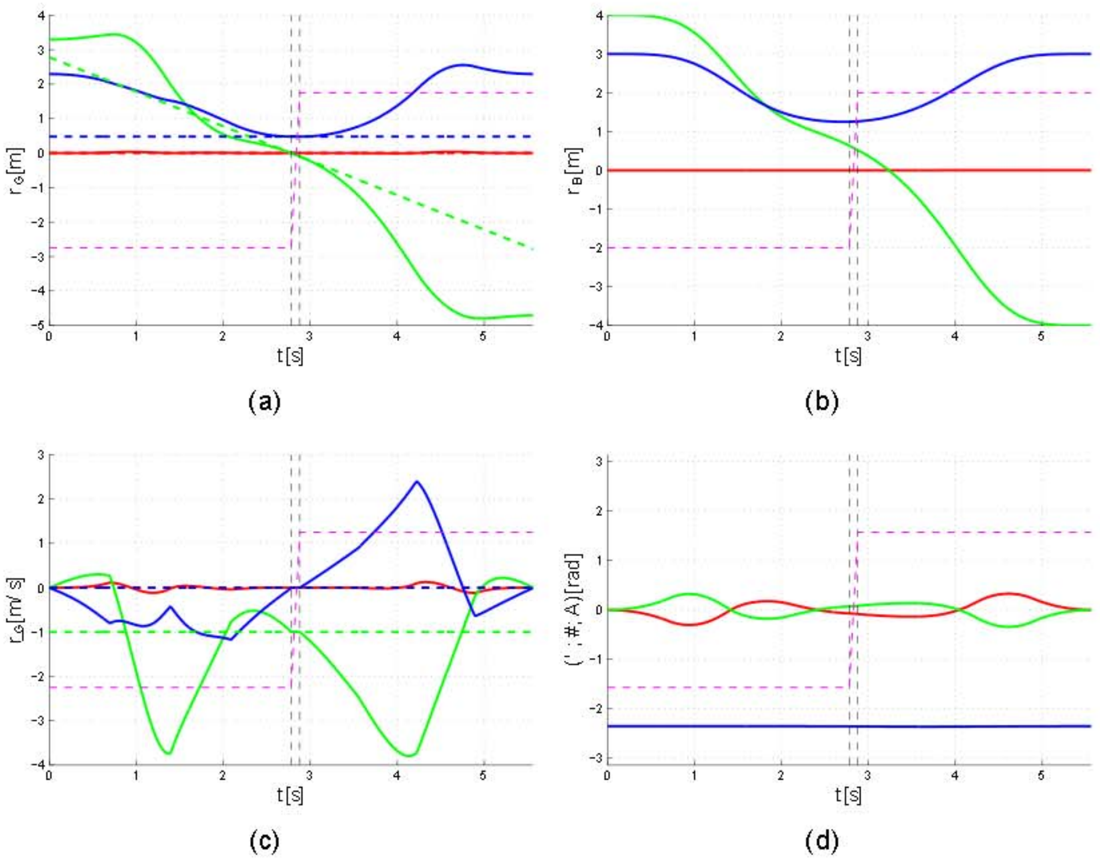
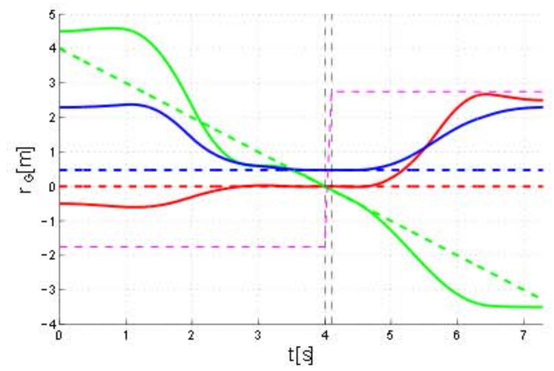


Figure 7.6: Grasping of a moving target using a non-hovering vertical gripper-closure trajectory: position (a) and velocity (c) of O_G (continuous line) and target (dashed line); position (b) and orientation (d) of the robot. A magenta dashed line is superimposed to every plot, indicating the gripper state: from completely open (lower value) to completely closed (upper value).

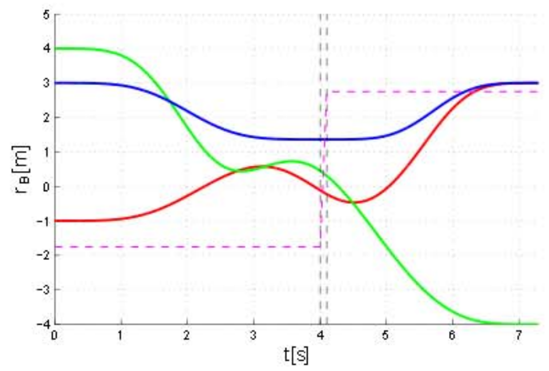
7.5 Concatenation of compound grasping trajectories

So far we have always referred to the situation in which the robot has to grasp an object which is located somewhere in the world frame. Nevertheless this problem is perfectly equivalent to that in which the robot is initially carrying an object and has to place it in a desired position. Not only the same planner can be used, but also multiple trajectories of the two types (either pick or place) can be concatenated by considering the final state of a trajectory as the initial state of the following one.

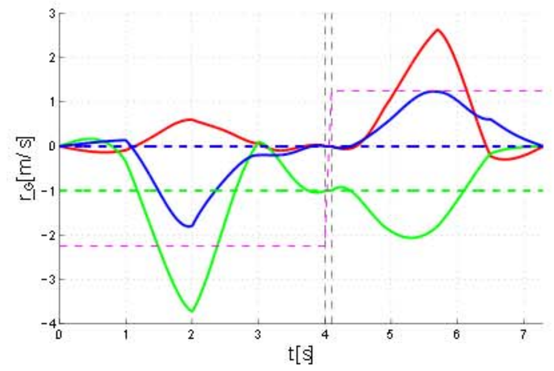
In fig. 7.9 we present the result of the concatenation of two pick& place operations using four compound grasping trajectories. It is easy to notice that in each grasping (placing) phase, delimited by blue vertical lines, the gripper follows the same trajectory of one of the target present in the scene.



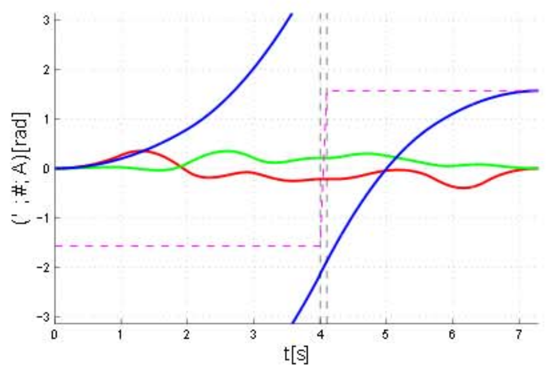
(a)



(b)



(c)



(d)

Figure 7.7: Grasping of a moving target using a non-hovering horizontal gripper-closure trajectory for the second set of initial and final robot conditions: position (a) and velocity (c) of O_G (continuous line) and target (dashed line); position (b) and orientation (d) of the robot.

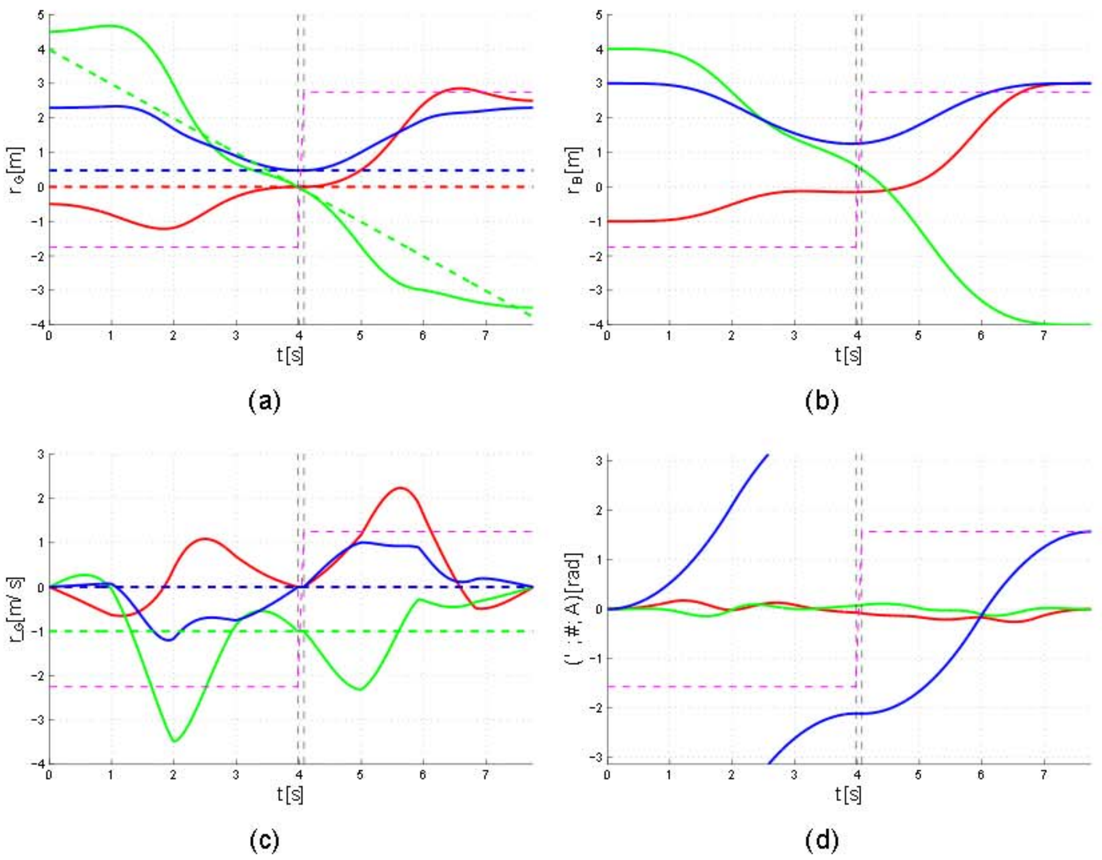
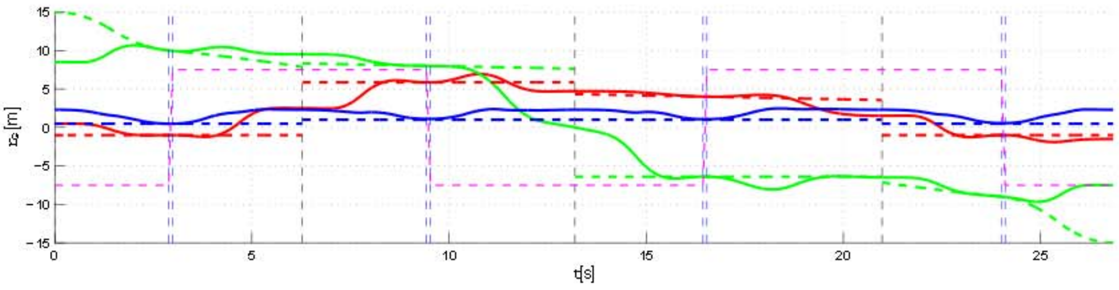
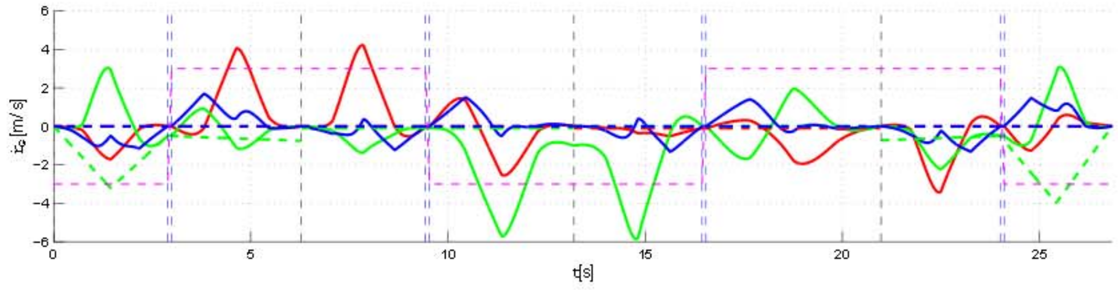


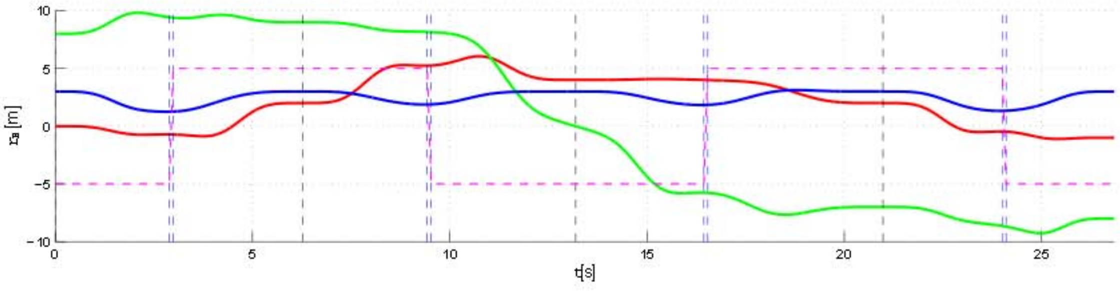
Figure 7.8: Grasping of a moving target using a non-hovering vertical gripper-closure trajectory for the second set of initial and final robot conditions: position (a) and velocity (c) of O_G (continuous line) and target (dashed line); position (b) and orientation (d) of the robot. A magenta dashed line is superimposed to every plot, indicating the gripper state: from completely open (lower value) to completely closed (upper value).



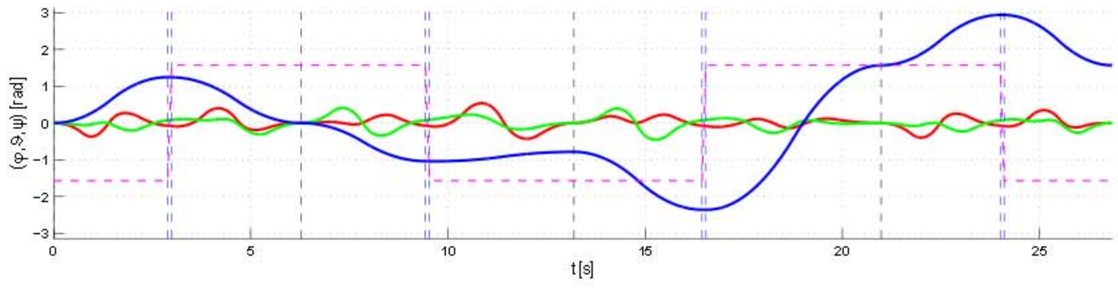
(a)



(b)



(c)



(d)

Figure 7.9: Multiple pick&place operations: position (a) and velocity (b) of O_G (continuous line) and target (dashed line); position (c) and orientation (d) of the robot. A magenta dashed line is superimposed to every plot, indicating the gripper state: from completely open (lower value) to completely closed (upper value).

Chapter 8

Control in SE(3)

The problem of controlling the dynamics of a quadrotor has been addressed for a long time in the literature. Many different controller have been proposed, based on feedback linearization (see [24]), Linear Quadratic Regulators (see [9]), Back-stepping techniques (see [32]), sliding mode control (see again [32]), model approximations in quasi hovering conditions (see [17]) just to cite a few of them.

Since in our application the quadrotor will be asked to perform agile maneuvers, we have to deal with significant divergences of the roll and pitch angles from the hovering condition. Therefore we need a controller that is developed in SE(3) and has almost global convergence properties. A controller with this features has been developed and proved to be stable in [33]. The performances of a simplified version of this controller have also been proven experimentally through the implementation on a real robot in [11].

Given a desired trajectory for the flat outputs and their derivatives (indicated with a subscript t in the following) and the current state of the robot, we compute the position and velocity errors as

$$\begin{aligned} e_p &= r_{B,t} - r_B \\ e_v &= \dot{r}_{B,t} - \dot{r}_B, \end{aligned}$$

and we define a desired force vector containing a feed forward term, a PD action and a gravity cancellation term

$$f_d = m\ddot{r}_{B,t} + K_p e_p + K_v e_v + mge_3, \tag{8.1}$$

where K_p and K_v are two positive scalar matrices. f_d is a force that, if it were applied to the quadrotor center of mass, would make the position and velocity error converge to zero asymptotically. We can also define a desired acceleration $\ddot{r}_{B,d}$ such that

$$f_d = m(\ddot{r}_{B,d} + ge_3) = mt_d, \tag{8.2}$$

then obviously

$$\ddot{r}_{B,d} = \ddot{r}_{B,t} + \frac{1}{m}K_p e_p + \frac{1}{m}K_v e_v := \ddot{r}_{B,t} + \hat{K}_p e_p + \hat{K}_v e_v. \quad (8.3)$$

Due to the dynamics of the robot, we can only apply forces along the current direction of the local vertical axis z_B. Therefore we operate in two steps: i) we apply the component of f_d along z_B directly, using the total thrust input u_1; ii) we use the torque inputs u_2, u_3, u_4, to orient the robot in such a way that the vertical axis z_B is parallel to f_d and the yaw is the one specified by the trajectory. First we project the desired force vector onto the current direction of the body frame z_B axis in order to compute the first input

$$u_1 = f_d^T z_B = f_d^T \, {}^W R_B e_3. \quad (8.4)$$

To determine the remaining inputs we assume that $f_d = 0$ and we compute the desired direction of the z_B axis, i.e. the direction of the desired force, as

$$z_{B,d} = \frac{f_d}{f_d}. \quad (8.5)$$

The other columns of the matrix describing the desired attitude of the quadrotor are computed following the steps of chapter 4. We start by defining

$$y_{C,d} = \begin{bmatrix} -\sin\psi_t & \cos\psi_t & 0 \end{bmatrix}^T,$$

and then we compute

$$x_{B,d} = \frac{y_{C,d} \times z_{B,d}}{y_{C,d} \times z_{B,d}} := \frac{\tilde{x}_{B,d}}{\tilde{x}_{B,d}},$$

and

$$y_{B,d} = z_{B,d} \times x_{B,d}.$$

The desired value of ${}^W R_B$, denoted by ${}^W R_{B,d}$, is then given by

$${}^W R_{B,d} = \begin{bmatrix} x_{B,d} & y_{B,d} & z_{B,d} \end{bmatrix}$$

and we can define an orientation error in SO(3) as

$$e_R = \frac{1}{2}\left({}^W R_B^T \, {}^W R_{B,d} - {}^W R_{B,d}^T \, {}^W R_B \right)^\vee,$$

where the $\vee$ symbol indicates the vee-map that relates a skew-symmetric matrix to the corresponding vector (see chapter 2).

To compute the desired angular velocity consider that, since the structure of the equations remains the same, we can retrace the same steps of chapter 4, substituting to each quantity the corresponding desired value. We therefore define

$$h_d = \frac{1}{t_d}\left(I - z_{B,d} z_{B,d}^T \right) \dot{a}_{B,d}$$

and we compute the desired angular velocity ${}^{B,d}\omega_{BW,d}$ as

$$
{}^{B,d}\omega_{BW,d} = \begin{matrix} -h_d \cdot y_{B,d} \\ h_d \cdot x_{B,d} \\ \frac{x_{C,d}^T x_{B,d} \dot{\psi}_t + y_{C,d}^T z_{B,d} q_d}{\tilde{x}_{B,d}} \end{matrix} .
$$

Differentiating eq. (8.3) with respect to time we obtain

$$
\dot{a}_{B,d} = \dot{a}_{B,t} + \hat{K}_p e_v + \hat{K}_v e_a, \tag{8.6}
$$

where e_a is the acceleration error

$$
e_a = \ddot{r}_{B,t} - \ddot{r}_B .
$$

Considering the system dynamics described by eq. (2.7) we conclude that

$$
e_a = \ddot{r}_{B,t} + g e_3 - \frac{u_1}{m} z_B .
$$

Since ${}^{B}\omega_{BW}$ and ${}^{B,d}\omega_{BW,d}$ belong to different spaces, we cannot compare them directly, but we first have to rotate them to the same reference frame. Therefore we rotate ${}^{B,d}\omega_{BW,d}$ first to the world frame and then to the body frame. The resulting tracking error for the angular velocity is defined as

$$
e_\omega = {}^{W}R_B^T \, {}^{W}R_{B,d} \, {}^{B,d}\omega_{BW,d} - {}^{B}\omega_{BW} .
$$

We operate in a similar fashion for the desired angular acceleration. We start putting

$$
\delta_d = \ddot{a}_{B,d} - \omega_{BW,d} \times \quad t_d \quad h_d ,
$$

then we compute

$$
l_d = \frac{1}{t_d} \quad I - z_{B,d} z_{B,d}^T \quad \delta_d - 2 (z_{B,d} \cdot \dot{a}_{B,d}) \, h_d \quad ,
$$

and we finally obtain

$$
{}^{B,d}\dot{\omega}_{BW,d} = \begin{matrix} -l_d \cdot y_{B,d} \\ l_d \cdot x_{B,d} \\ \frac{2 \left(x_{C,d}^T y_{B,d} r_d - x_{C,d}^T z_{B,d} q_d \right) \dot{\psi}_t + x_{C,d}^T x_{B,d} \ddot{\psi}_t + y_{C,d}^T z_{B,d} (n_d - p_d r_d)}{\|\tilde{x}_B\|} - p_d q_d \end{matrix} .
$$

Differentiating eq. (8.6) with respect to time we obtain

$$
\ddot{a}_{B,d} = \ddot{a}_{B,t} + \hat{K}_p e_a + \hat{K}_v e_j
$$

where e_j is the jerk error that, considering eq. (4.3), is given by

$$\begin{aligned} e_j &= \dot{a}_{B,t} - \dot{a}_B \\ &= \dot{a}_{B,t} - \frac{\dot{u}_1}{m} z_B - \frac{u_1}{m} \omega_{BW} \times z_B \\ &= \dot{a}_{B,t} - [\dot{a}_{B,d} \cdot z_B + t_d \cdot (\omega_{BW} \times z_B)] z_B - (t_d \cdot z_B)\, \omega_{BW} \times z_B . \end{aligned}$$

To conclude we compute the remaining control inputs as:

$$\begin{aligned} \begin{bmatrix} u_2 \\ u_3 \\ u_4 \end{bmatrix} &= K_R e_R + K_\omega e_\omega + {}^B\omega_{BW} \times J\, {}^B\omega_{BW} \\ &\quad + J \left({}^B R_W^T\, {}^W R_{B,d}\, {}^{B,d}\dot{\omega}_{BW,d} - {}^B\Omega_{BW}\, {}^W R_B^T\, {}^W R_{B,d}\, {}^{B,d}\omega_{BW,d} \right) \end{aligned} \tag{8.7}$$

where K_R and K_ω are two positive scalar matrices. Note that this control law contains a PD action, a cancellation of the gyroscopic force and a feed forward term.

The demonstration of the stability of this controller is contained in [33] and it is not reported in this thesis. Intuitively, the proof is based on the cascade structure of the quadrotor dynamics. Indeed it is possible to demonstrate that the attitude controller described by eq. (8.7) has a large basin of attraction. Provided that, it is straightforward to demonstrate that the trajectory tracking controller defined by eqs. (8.2) and (8.4) is stable when the attitude tracking error is zero.

Chapter 9

Physical simulations

As we already commented the proposed algorithm for the grasping trajectory generation has been entirely implemented in Matlab®. All the optimizations have been done using the routines available from Matlab® Optimization Toolbox and for the computation of the B-spline basis functions as well as for the evaluation of the B-splines and of their integrals, we used the Matlab® Curve Fitting Toolbox.

The computation of a complete trajectory takes around two minutes, that is too long to be performed on line. Therefore, as stated in chapter 3, we only considered the case of full a-priori knowledge of the target trajectory. The grasping trajectory is computed off-line and no on-line adaption is performed. The result of the planning phase is an array containing the value of the planned trajectory with a fixed time resolution of 10 ms. This constitutes the input for the trajectory controller of the quadrotor.

The controller described in chapter 8 has been implemented in Simulink®. In this process care has been taken to organize the code in such a way that it resembles as much as possible a hypothetical real implementation. This would make it easier, if desired, to model non-idealities that possibly affect the real system (communication delays, noise, . . .) and also to implement the controller on a real system. Before describing the Simulink® model we then have to introduce a possible hardware implementation.

The quadrotors available at the Max Planck Institute for Biological Cybernetics are produced by MikroKopter (see [34]). One of them is shown in fig. 1.1. The size of the robot (i.e. the distance between opposite motors) is about 40 cm and the weight is around 750 g.

The Institute also has a Vicon motion capture system ([35]) at disposal. This system is able to determine the position and orientation of the quadrotor by tracking some markers attached to the robot, using a set of cameras and sophisticated vision algorithms. The system has an accuracy of less than 1cm.

An acceptable estimate of the linear velocity of the robot can be obtained by numerical differentiation of the position provided by the Vicon. The angular velocity obtained by differentiation of the orientation is, instead, too noisy to be used. A much better measure of this quantity is made available by an inertia measurement unit (IMU) mounted on board of the quadrotor.

The quadrotor is also equipped with an on-board micro-controller that generates setting points for the low-level brush-less motor controllers. This can wirelessly communicate with a remote base station using an Xbee® module, connected to a serial port of the micro-controller. The computational power of the on-board micro-controller is limited, but part of the computation can be carried on in the base station (a Linux based PC), and sent to the quadrotor through the Xbee® communication channel. The remote desktop can also be used to generate and store the reference trajectories and as an interface with the Vicon to receive the state measurements at a high rate. It is also possible to equip the robot with a high performance Atom board directly connected to the serial port of the micro-controller, but this significantly increases the robot payload, thus reducing its agility.

Given this hardware setup, the control algorithm must be distributed in such a way that:

- it minimizes the computation effort for the on-board micro-controller;
- it reduces the amount of data which is being transmitted through the Xbee® wireless connection;
- it uses the best possible sensor measurements, i.e. the position, orientation and linear velocity provided by the Vicon tracking system and the angular velocity measured from the on-board IMU;
- it uses the reference trajectory computed and stored by the remote computer.

From the equations of chapter 8, we can notice that for the computation of the first input (i.e. the total thrust) we need the reference trajectory and the position, velocity and orientation of the robot. This information is provided by the Vicon and is available in the remote high performance computer. The first input can then be computed here and sent to the on-board micro-controller via the wireless communication.

The torque input computation also requires the angular velocity of the robot. More in details the first term in eq. (8.7) requires position velocity and orientation of the robot and then can be computed remotely. The second term also requires