3

Mobile Robot Vehicles

Consider a mechanical system whose configuration q is described by a vector of generalized coordinates and assume that the configuration space $\mathcal{C}$ (i.e., the space of all possible robot configurations) coincides with $\mathbb{R}^n$. The motion of the system that is represented by the evolution of q over time may be subject to constraints that can be classified under various criteria. For example, they may be expressed as equalities and inequalities (respectively, *bilateral* or *unilateral* constraints), and they may depend explicitly on time or not (*rheonomic* or *scleronomic* constraints). In this chapter, only bilateral scleronomic constraints will be considered.
Constraints that can be put in the form

$$h_i(q) = 0, \qquad i = 1, \ldots, k < n \tag{3.1}$$

are called *holonomic* (or *integrable*). We assume that the functions $h_i : C \to \mathbb{R}$ are of class C^∞ (smooth) and independent. The effect of holonomic constraints is to reduce the space of accessible configurations to a subset of $\mathcal{C}$ with dimension $n - k$. A mechanical system for which all the constraints can be expressed in the form (3.1) is called *holonomic*.
In the presence of holonomic constraints, the implicit function theorem (Dini's theorem) can be used in pronciple to solve the equations (3.1) by expressing k generalized coordinates as a function of the remaining $n - k$, so as to eliminate them from the formulation of the problem. However, in general this procedure is only valid locally, and may introduce singularities. A convenient alternative is to replace the original generalized coordinates with a reduced set of $n - k$ new coordinates that are directly defined on the accessible subspace, in such a way that the available degrees of freedom (DOFs) are effectively characterized. The mobility of the reduced system thus obtained is completely equivalent to that of the original mechanism.
Holonomic constraints are generally the result of interconnections between the various bodies of the system (see prismatic and revolute joints).
Constraints that involve generalized coordinates and velocities

$$a_i(q, \dot{q}) = 0, \qquad i = 1, \ldots, k < n$$

are called *kinematic*. They constrain the instantaneous admissible motion of the mechanical system by reducing the set of generalized velocities that can be attained at each configuration. Kinematic constraints are generally expressed in *Pfaffian form*, i.e., they are linear in the generalized velocities:

$$a_i^T(q)\dot{q} = 0, \qquad i = 1, \ldots, k < n \tag{3.2}$$

or, in matrix form,

$$A^T(q)\dot{q} = 0. \tag{3.3}$$

Vectors $a_i : \mathcal{C} \to \mathbb{R}^n$ are assumed to be smooth as well as linearly independent.
Clearly, the existence of k holonomic constraints (3.1) implies that of an equal number of kinematic constraints:

$$\frac{dh_i(q)}{dt} = \frac{\partial h_i(q)}{\partial q}\dot{q} = 0, \qquad i = 1, \dots k.$$

However the converse is not true in general. A system of kinematic constraints in the form (3.3) may or may not be integrable to the form (3.1). In the negative case, the kinematic constraints are said to be *nonholonomic* (or *non-integrable*). A mechanical system that is subject to at least one such constraint is called *nonholonomic*.
Nonholonomic constraints reduce the mobility of the mechanical system in a completely different way with the respect to holonomic constraints. To appreaciate this fact, consider a Pfaffian constraint

$$a^T(q)\dot{q} = 0. \tag{3.4}$$

If the constraint is holonomic, it can be integrated and written as

$$h(q) = c,$$

where $\partial h/\partial q = \gamma(q)a^T(q)$, with $\gamma(q) \neq 0$ an *integrating factor* and c an integration constant.
Therefore, there is a loss of *accessibility* in the configuration space, because the motion of the mechanical system in $\mathcal{C}$ is confine to a particular *level surface* of the scalar function h. This surface, which depends on the initial configuration q_0 through the value of $h(q_0) = c$, has dimension $n - 1$.
Assume instead that the constraint (3.4) is nonholonomic. In this case, generalized velocities are indeed constrained to belong to a subspace of dimension $n - 1$, i.e., the null space of matrix $a^T(q)$. Nevertheless, the fact that the constraint is non-integrable means that there is no loss of accessibility in $\mathcal{C}$ for the system. In other words, while the number of degrees of freedom (DOFs) decreases to $n-1$ due to the constraint, the number of generalized coordinates cannot be reduced, not even locally.
The conclusion just drawn for the case of a single constraint is general. An n-dimensional mechanical system subject to k nonholonomic constraints can access its whole configuration space $\mathcal{C}$, although at any configuration its generalized velocities must belong to an $(n - k)$-dimensional subspace.
The following is a classical example of nonholonomic mechanical system, that clarifies the above concepts.

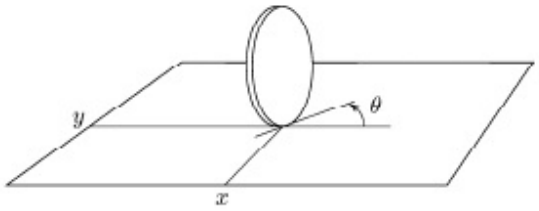

Figure 3.1: Generalized coordinates for a disk rolling on a plane.

Example 3.1 Consider a disk that rolls without slipping on the horizontal plane, while keeping its sagittal plane (i.e., the plane that contains the disk) in the vertical direction (see Figure 3.1). Its configuration is described by three generalized coordinates: the Cartesian coordinates (x, y) of the contact point with the ground, measured in a fixed reference frame, and the angle θ characterizing the orientation of the disk with the respect to x axis. The configuration vector is therefore $q = [x \;\; y \;\; \theta]^T$.

The *pure rolling* constraint for the disk is expressed in the Pfaffian form as

$$\dot{x}\sin\theta - \dot{y}\cos\theta = [\sin\theta \quad -\cos\theta \quad 0]\dot{q} = 0. \tag{3.5}$$

and entails that, in absence of slipping, the velocity of the contact point has zero component in the direction orthogonal to the sagittal plane. The angular velocity of the disk around the vertical axis instead is uncontrained.
Constraint (3.5) is nonholonomic, because it implies no loss of accessibility in the configuration space of the disk. To substantiate this claim, consider that the disk can be driven from any initial configuration $q_i = [x_i \;\; y_i \;\; \theta_i]^T$ to any final configuration $q_f = [x_f \;\; y_f \;\; \theta_f]^T$ through the following sequence of movements that do not violate constraint (3.5):

1. rotate the disk around its vertical axis so as to reach the orientation θ_v for which the *sagittal axis* (i.e., the intersection of the sagittal plane and the horizontal plane) goes through the final contact point (x_f, y_f);
2. roll the disk on the plane at a constant orientation θ_v, until the contact point reaches the final position (x_f, y_f);
3. rotate again the disk around its vertical axis to change the orientation θ_v to θ_f.

3.1 Bicycle model

A commonly used model for a four-wheeled car-like is the bicycle model shown in Figure 3.2 .

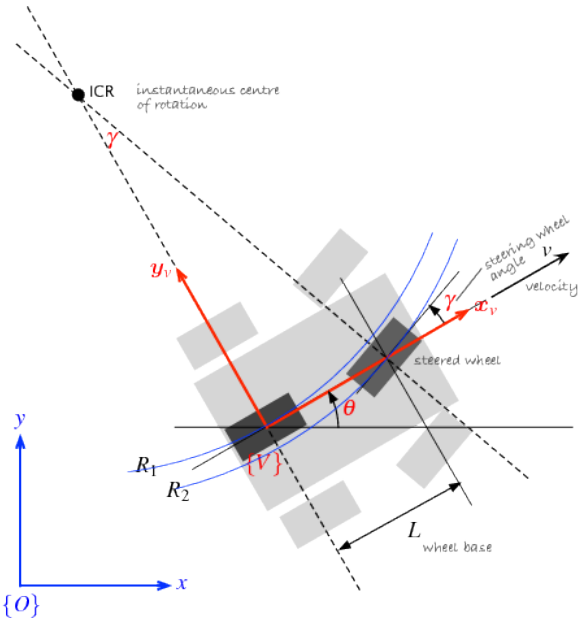

Figure 3.2: Bicycle model of a car. The car is shown in light grey, and the bicycle approximation is dark grey.The vehicle's coordinate frame is shown in red, and the world coordinate frame in blue. The steering wheel angle is γ and the velocity of the back wheel, in the x-direction, is v. The two wheel axes are extended as dashed lines and intersect at the Instantaneous Centre of Rotation (ICR) and the distance from the ICR to the back and front wheels is R_1 and R_2 respectively.

The rear wheel is fixed and the front one can rotate. The configuration of the vehicle is represented by (x, y, θ), where (x, y) are the coordinates of the center of the back wheel with respect to the frame $\{O\}$ and θ is the orientation with respect to the x-axis of the frame $\{O\}$.The vehicle's velocity is by definition v in the vehicle's x-direction, and zero in the y-direction since the wheels cannot slip sideways. In the vehicle frame V this is

$$\begin{aligned} v_{\dot{x}} &= v \\ v_{\dot{y}} &= 0. \end{aligned} \tag{3.6}$$

The angular velocity of the vehicle is $\dot{\theta} = \frac{v}{R_1}$, where R_1 is the distance between the back wheel and the Instantaneous Centre of Rotation (ICR). By simple geometry the turning radius is $R_1 = \frac{L}{tan\gamma}$, where L is the wheel base and γ is the steering angle. Note that $R_2 > R_1$ which means the front wheel must follow a longer path and therefore rotate more quickly than the back wheel. When a four-wheeled vehicle goes around a corner the two steered wheels follow circular paths of different radius and therefore the angles of the steered wheels γ_L and γ_R should be very slightly different. We are now ready to state the bicycle model:

$$\begin{cases} \dot{x} = v\cos(\theta) \\ \dot{y} = v\sin(\theta) \\ \dot{\theta} = \frac{v}{L} tan(\gamma). \end{cases} \tag{3.7}$$

If we rearrange the first two equations of (3.7) we get the non-holonomic constraint, which is $\dot{y}\cos(\theta) - \dot{x}\sin(\theta) = 0$. This equation cannot be integrated to form a relationship between x, y and θ. Note that it is not possible to change the vehicle orientation when $v = 0$ because $\dot{\theta}$ depends linearly from v. This make sense because as we know from driving we must be moving in order to turn.

3.2 Bicycle Control

Consider the problem of moving the vehicle to a point (x^*, y^*). The main idea is to control the robot'velocity to be proportional to its distance from the goal. We use a proportional controller to regulate the velocity

$$v = K_v\sqrt{(x^* - x)^2 + (y^* - y)^2}. \tag{3.8}$$

The objective angle is not assigned and is setted to $\theta^* = atan2((y^* - y), (x^* - x))$. Again we use a proportional controller to steer the robot

$$\gamma = K_h(\theta^* \ominus \theta). \tag{3.9}$$

Note that the operator $\ominus$ computes the angular difference to keep the angle between $[-\pi, \pi]$. Instead of moving to a point we might wish to follow a path defined as some locus on the x-y plane. The path might be generated by a motion planner, as discussed in the previous chapter. The problem is very similar to the control problem that we tackled before, except this time the point is moving. We wish to maintain a given distance d^* behind the pursuit point. The error can be formulated in the following way

$$e = \sqrt{(x^* - x)^2 + (y^* - y)^2} - d^* \tag{3.10}$$

that is the input of a PI controller which control the velocity

$$v = K_v e + K_i \int edt. \tag{3.11}$$

To steer the robot we exploit the same controller presented to move the robot to a point, given by equaton (3.9).

3.3 Quadcopter Model

In this section we will create a model for a quadcopter flying vehicle such as shown in Figure 3.3.

Figure 3.3: SERL Quadrotor Research Platform. Credits: SERL

Compared to fixed wing aircraft they are highly manoeuvrable and can be flown safely indoors which makes them well suited for laboratory or hobbyist use. Compared to conventional helicopters, with the large main rotor and tail rotor, the quadcopter is easier to fly, does not have the complex swash plate mechanism and is easier to model and control.
The notation for the quadcopter model is shown in Figure 3.4.

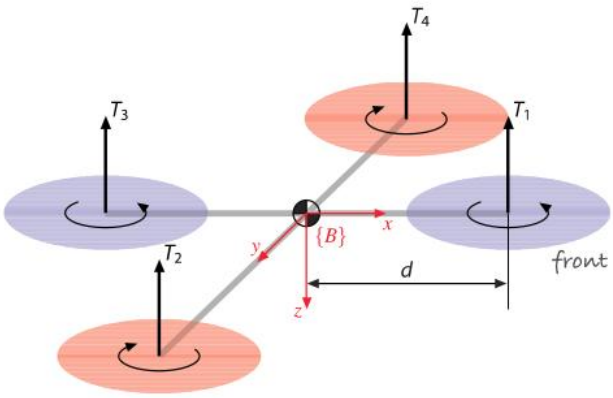

Figure 3.4: Quad-rotor notation showing the four rotors, their thrust vectors and directions of rotation. The body-fixed frame $\{B\}$ is attache to the vehicle and has its origin at the vehicle's centre of mass. Rotors 1 and 3 rotate counte-clockwise (viewed from above) while rotors 2 and 4 rotate clockwise.

The body-fixed coordinate frame $\{B\}$ has its z-axis downward following the aerospace convention. The quadcopter has four rotors, labelled 1 to 4, mounted at the end of each cross arm. The rotors are driven by electric motors powered by electronic speed controllers. Some low-cost quadcopters use small motors and reduction gearing to achieve sufficient torque. The rotor speed is ω_i and the thrust is an upward vector in the robot'negative z-direction

$$T_i = b\omega_i^2, \quad i = 1, \ldots, 4. \tag{3.12}$$

The coefficient $b > 0$ is the lift constant that depends on the air density, the geometry and the

number of the blades. The translation dynamic in the world coordinates is given by

$$m\dot{v} = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} - R_B^0 \begin{bmatrix} 0 \\ 0 \\ T \end{bmatrix} \tag{3.13}$$

where v is the velocity of the quadcopter in the world frame, g is gravitational acceleration, m is the mass of the robot and $T = \sum_{i=1}^{4} T_i$ is the total upward thrust. The first vector is the gravity force , the second is the total thrust rotated in the world coordinate frame. Pairwise differences in rotor thrusts cause the vehicle to rotate. The torque about the x-axis, the rolling torque, is

$$\tau_x = dT_4 - dT_2 \tag{3.14}$$

where d is the distance from the motor to the center of mass. If we substitute equation (3.12) in the previous we get

$$\tau_x = db(w_4^2 - w_2^2). \tag{3.15}$$

Similarly, the pitching torque is

$$\tau_y = db(w_1^2 - w_3^2). \tag{3.16}$$

The torque applied to each propeller by the motor is opposed by aerodynamic drag

$$Q_i = k\omega_i^2 \tag{3.17}$$

where k is a factor similar to b. This torque exerts a reaction torque on the airframe which acts to rotate the airframe about the propeller shaft in the opposite direction to its rotation. The total reaction torque about the z-axis is

$$\tau_z = Q_1 - Q_2 + Q_3 - Q_4 = k(\omega_1^2 + \omega_3^2 - \omega_2^2 - \omega_4^2) \tag{3.18}$$

where the signs are different because the rotors are rotating in different directions. The rotational acceleration of the airframe is given by the Euler' rotation equations

$$J\dot{\omega} = -\omega \times J\omega + \Gamma \tag{3.19}$$

where J is the inertia matrix, ω the angular velocity vector with respect to the vehicle frame and Γ is the torque applied to the airframe, i.e. $\Gamma = [\tau_x,\ \tau_y,\ \tau_z]^T$. The forward dynamics is given by equations (3.13) and (3.19). The forces and the torques on the airframe are related to the rotors' speed in the following way

$$\begin{bmatrix} T \\ \Gamma \end{bmatrix} = \begin{bmatrix} -b & -b & -b & -b \\ 0 & -db & 0 & db \\ db & 0 & -db & 0 \\ k & -k & k & -k \end{bmatrix} \begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{bmatrix} = A \begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{bmatrix} \tag{3.20}$$

The matrix A is rank full if b, k and $d > 0$. If this assumptions hold the matrix A can be inverted

$$\begin{bmatrix} w_1^2 \\ w_2^2 \\ w_3^2 \\ w_4^2 \end{bmatrix} = A^{-1} \begin{bmatrix} \tau_x \\ \tau_y \\ \tau_z \\ \Gamma \end{bmatrix} \tag{3.21}$$

3.4 Quadcopter Control

Our goal is to control the position of the quadrotor in the space. To control the vehicle we will employ a nested control structure. The innermost loop uses a proportional and derivative controller to compute the required pitching torque on the airframe

$$\tau_y = K_{\tau_y,p}\left(\theta_p^* - \theta_p\right) + K_{\tau_y,d}\left(\dot{\theta}_p^* - \dot{\theta}_p\right). \tag{3.22}$$

Usually $K_{\tau_y,p}$ and $K_{\tau_y,d}$ are computed by classical control design approaches based on an approximate dynamic model and then tuned. The actual vehicle orientation (roll, pich and yaw angles) are estimated by an inertial navigation system, then the rotor required speed are computed using equation (3.21). The set point $\dot{\theta}_p^*$ is commonly setted to zero, while θ_p^* is computed using in intermediate control loop. The following calculation will provide the set point for the pitch angle. Consider a coordinate frame $\{V\}$ attached to the vehicle and with the same origin as $\{B\}$ but with its x- and y-axes parallel to the ground. To move the vehicle in the ${}^V x$ direction (x direction with respect to the frame $\{V\}$), we need to pitch the nose down which generate a force

$$f = R_y(\theta_p)\begin{bmatrix}0\\0\\T\end{bmatrix} = \begin{bmatrix}T\sin\theta_p\\0\\T\cos\theta_p\end{bmatrix}. \tag{3.23}$$

The component on the x-axis can be approximated by

$$f_x = T\sin\theta_p \approx T\theta_p. \tag{3.24}$$

We can control the velocity in this direction using a proportional controller as

$$f_x^* = mK_f\left({}^V v_x^* - {}^V v_x\right), \tag{3.25}$$

where K_f is the proportional gain. Combining the approximation with the last equation we get

$$\theta_p^* = \frac{m}{T}K_f\left({}^V v_x^* - {}^V v_x\right), \tag{3.26}$$

which is the set point we were looking for. Now we need to find out a velocity set-point. If the position of the vehicle in the xy-plane of the world frame is $p \in \mathbb{R}^2$ then the desired velocity is given by the proportional control law

$$v^* = K_p\left(p^* - p\right), \tag{3.27}$$

based on the error between the desired and actual position. We only need to rotate the velocity v^* in the frame $\{V\}$ to obtain the set-point ${}^V v_x^*$

$${}^V v = R_0^V(\theta_y)v. \tag{3.28}$$

To move the vehicle on the y-axis we can use the same control structure. We would also like to control the yaw angle, and this can be done again using a PD controller

$$\tau_z = K_{\tau_z,p}\left(\theta_y^* - \theta_y\right) + K_{\tau_z,d}\left(\dot{\theta}_y^* - \dot{\theta}_y\right), \tag{3.29}$$

where also this time $\dot{\theta}_y^*$ is ignored since it is typically small. Finally we want to control the altitude, and this is done exploiting another PD controller

$$T = K_{T,p}\left(z^* - z\right) + K_{T,d}\left(\dot{z}^* - \dot{z}\right) + \omega_0, \tag{3.30}$$

where $w_0 = \sqrt{\frac{mg}{4b}}$ is the rotor speed necessary to generate a thrust equal to the weight of the vehicle. This is an example of feedforward controller, used here to counter effect the effect of gravity.