By using the input transformation

$$\boldsymbol{\tau} = \boldsymbol{M}\boldsymbol{u} = \begin{bmatrix} m & 0 \\ 0 & I \end{bmatrix} \boldsymbol{u},$$

the second-order kinematic model is obtained as

$$\dot{\xi} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_2$$

with the state vector $\xi = [\,x \quad y \quad \theta \quad v \quad \omega\,]^T \in \mathbb{R}^5$.

11.5 Planning

As with manipulators, the problem of planning a trajectory for a mobile robot can be broken down in finding a *path* and defining a *timing law* on the path. However, if the mobile robot is subject to nonholonomic constraints, the first of these two subproblems becomes more difficult than in the case of manipulators. In fact, in addition to meeting the boundary conditions (interpolation of the assigned points and continuity of the desired degree) the path must also satisfy the nonholonomic constraints at all points.

11.5.1 Path and Timing Law

Assume that one wants to plan a trajectory $\boldsymbol{q}(t)$, for $t \in [t_i, t_f]$, that leads a mobile robot from an initial configuration $\boldsymbol{q}(t_i) = \boldsymbol{q}_i$ to a final configuration $\boldsymbol{q}(t_f) = \boldsymbol{q}_f$ in the absence of obstacles. The trajectory $\boldsymbol{q}(t)$ can be broken down into a geometric path $\boldsymbol{q}(s)$, with $d\boldsymbol{q}(s)/ds \neq 0$ for any value of s, and a timing law $s = s(t)$, with the parameter s varying between $s(t_i) = s_i$ and $s(t_f) = s_f$ in a monotonic fashion, i.e., with $\dot{s}(t) \geq 0$, for $t \in [t_i, t_f]$. A possible choice for s is the arc length along the path; in this case, it would be $s_i = 0$ and $s_f = L$, where L is the length of the path.

The above space-time separation implies that

$$\dot{\boldsymbol{q}} = \frac{d\boldsymbol{q}}{dt} = \frac{d\boldsymbol{q}}{ds}\,\dot{s} = \boldsymbol{q}'\dot{s},$$

where the prime symbol denotes differentiation with respect to s. The generalized velocity vector is then obtained as the product of the vector $\boldsymbol{q}'$, which is directed as the tangent to the path in configuration space, by the scalar $\dot{s}$, that varies its modulus. Note that the vector $[\,x' \quad y'\,]^T \in \mathbb{R}^2$ is directed as

the tangent to the Cartesian path, and has unit norm if s is the cartesian arc length (see Sect. 5.3.1).

Nonholonomic constraints of the form (11.3) can then be rewritten as

$$\boldsymbol{A}(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{A}(\boldsymbol{q})\boldsymbol{q}'\dot{s} = \boldsymbol{0}.$$

If $\dot{s}(t) > 0$, for $t \in [t_i, t_f]$, one has

$$\boldsymbol{A}(\boldsymbol{q})\boldsymbol{q}' = \boldsymbol{0}. \tag{11.40}$$

This condition, that must be verified at all points by the tangent vector on the configuration space path, characterizes the notion of *geometric* path admissibility induced by the kinematic constraint (11.3) that actually affects generalized velocities. Similar to what has been done for trajectories in Sect. 11.2, geometrically admissible paths can be explicitly defined as the solutions of the nonlinear system

$$\boldsymbol{q}' = \boldsymbol{G}(\boldsymbol{q})\widetilde{\boldsymbol{u}}, \tag{11.41}$$

where $\widetilde{\boldsymbol{u}}$ is a vector of *geometric* inputs that are related to the velocity inputs $\boldsymbol{u}$ by the relationship $\boldsymbol{u}(t) = \widetilde{\boldsymbol{u}}(s)\dot{s}(t)$. Once the geometric inputs $\widetilde{\boldsymbol{u}}(s)$ are assigned for $s \in [s_i, s_f]$, the path of the robot in configuration space is uniquely determined. The choice of a timing law $s = s(t)$, for $t \in [t_i, t_f]$, will then identify a particular trajectory along this path.

For example, in the case of a mobile robot with unicycle-like kinematics, the pure rolling constraint (11.6) entails the following condition for geometric admissibility of the path:

$$[\,\sin\theta \quad -\cos\theta \quad 0\,]\,\boldsymbol{q}' = x'\sin\theta - y'\cos\theta = 0,$$

that simply expresses the fact that the tangent to the Cartesian path must be aligned with the robot sagittal axis. As a consequence, a path whose tangent is discontinuous (e.g., a broken line) is *not* admissible, unless the unicycle is allowed to stop at discontinuity points by setting $\dot{s} = 0$ for the time necessary to rotate on the spot so as to align with the new tangent.

Geometrically admissible paths for the unicycle are the solutions of the system

$$\begin{aligned} x' &= \widetilde{v}\cos\theta \\ y' &= \widetilde{v}\sin\theta \\ \theta' &= \widetilde{\omega}, \end{aligned} \tag{11.42}$$

where $\widetilde{v}, \widetilde{\omega}$ are related to v, ω by

$$v(t) = \widetilde{v}(s)\dot{s}(t) \tag{11.43}$$

$$\omega(t) = \widetilde{\omega}(s)\dot{s}(t). \tag{11.44}$$

11.5.2 Flat Outputs

Many kinematic models of mobile robots, including the unicycle and the bicycle, exhibit a property known as *differential flatness*, that is particularly relevant in planning problems. A nonlinear dynamic system $\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{G}(\boldsymbol{x})\boldsymbol{u}$ is *differentially flat* if there exists a set of outputs $\boldsymbol{y}$, called *flat* outputs, such that the state $\boldsymbol{x}$ and the control inputs $\boldsymbol{u}$ can be expressed algebraically as a function of $\boldsymbol{y}$ and its time derivatives up to a certain order:

$$\begin{aligned} \boldsymbol{x} &= \boldsymbol{x}(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}}, \ldots, \boldsymbol{y}^{(r)}) \\ \boldsymbol{u} &= \boldsymbol{u}(\boldsymbol{y}, \dot{\boldsymbol{y}}, \ddot{\boldsymbol{y}}, \ldots, \boldsymbol{y}^{(r)}). \end{aligned}$$

As a consequence, once an output trajectory is assigned for $\boldsymbol{y}$, the associated trajectory of the state $\boldsymbol{x}$ and history of control inputs $\boldsymbol{u}$ are uniquely determined.

In the unicycle and bicycle cases, the Cartesian coordinates are indeed flat outputs. In the following, this property is established for the unicycle. This can be done with reference to either the kinematic model (11.13) or the geometric model (11.42). For simplicity, refer to the latter. Its first two equations imply that, given a Cartesian path $(x(s), y(s))$, the associated state trajectory is $\boldsymbol{q}(s) = [\, x(s) \quad y(s) \quad \theta(s) \,]^T$ where

$$\theta(s) = \text{Atan2}\,(y'(s), x'(s)) + k\pi \qquad k = 0, 1. \tag{11.45}$$

The two possible choices for k account for the fact that the same Cartesian path may be followed moving forward ($k = 0$) or backward ($k = 1$). If the initial orientation of the robot is assigned, only one of the choices for k is correct. The geometric inputs that drive the robot along the Cartesian path are easily obtained from (11.42),(11.45) as

$$\widetilde{v}(s) = \pm\sqrt{(x'(s))^2 + (y'(s))^2} \tag{11.46}$$

$$\widetilde{\omega}(s) = \frac{y''(s)x'(s) - x''(s)y'(s)}{(x'(s))^2 + (y'(s))^2}. \tag{11.47}$$

These equations deserve some comments:

- The choice of the sign of $\widetilde{v}(s)$ depends on the type of motion (forward or backward).
- If $x'(\bar{s}\,) = y'(\bar{s}\,) = 0$ for some $\bar{s} \in [s_i, s_f]$, one has $\widetilde{v}(\bar{s}\,) = 0$. This happens, for example, in correspondence of cusps (motion inversions) in the Cartesian path. In these points, Eq. (11.45) does not define the orientation, that can however be derived by continuity, i.e., as the limit of its right-hand side for $s \to \bar{s}^{-}$. Similar arguments can be repeated for the steering velocity $\widetilde{\omega}$ given by (11.47).
- The possibility of reconstructing θ and $\widetilde{\omega}$ is lost when the Cartesian trajectory degenerates to a point, because in this case it is $x'(s) = y'(s) = 0$ identically.

It is interesting to note that for driftless dynamic systems — like the kinematic models of mobile robots — differential flatness is a necessary and sufficient condition for transformability in the chained form introduced in Sect. 11.3. In particular, it is easy to prove that the flat outputs of a $(2, n)$ chained form are z_1 and z_n, from which it is possible to compute all the other state variables as well as the associated control inputs. For example, in the case of the $(2, 3)$ chained form (11.25) it is

$$z_2 = \frac{\dot{z}_3}{\dot{z}_1} \qquad v_1 = \dot{z}_1 \qquad v_2 = \frac{\dot{z}_1 \ddot{z}_3 - \ddot{z}_1 \dot{z}_3}{\dot{z}_1^2}.$$

Note that z_2 and v_2 can be actually reconstructed only if $\dot{z}_1(t) \neq 0$, for $t \in [t_i, t_f]$.

11.5.3 Path Planning

Whenever a mobile robot admits a set of flat outputs $\boldsymbol{y}$, these may be exploited to solve planning problems efficiently. In fact, one may use any interpolation scheme to plan the path of $\boldsymbol{y}$ in such a way as to satisfy the appropriate boundary conditions. The evolution of the other configuration variables, together with the associated control inputs, can then be computed algebraically from $\boldsymbol{y}(s)$. The resulting configuration space path will *automatically* satisfy the nonholonomic constraints (11.40).

In particular, consider the problem of planning a path that leads a unicycle from an initial configuration $\boldsymbol{q}(s_i) = \boldsymbol{q}_i = [\, x_i \quad y_i \quad \theta_i \,]^T$ to a final configuration $\boldsymbol{q}(s_f) = \boldsymbol{q}_f = [\, x_f \quad y_f \quad \theta_f \,]^T$.

Planning via Cartesian polynomials

As mentioned above, the problem can be solved by interpolating the initial values x_i, y_i and the final values x_f, y_f of the flat outputs x, y. Letting $s_i = 0$ and $s_f = 1$, one may use the following cubic polynomials:

$$\begin{aligned} x(s) &= s^3 x_f - (s-1)^3 x_i + \alpha_x s^2 (s-1) + \beta_x s (s-1)^2 \\ y(s) &= s^3 y_f - (s-1)^3 y_i + \alpha_y s^2 (s-1) + \beta_y s (s-1)^2, \end{aligned}$$

that automatically satisfy the boundary conditions on x, y. The orientation at each point being related to x', y' by (11.45), it is also necessary to impose the additional boundary conditions

$$\begin{aligned} x'(0) &= k_i \cos\theta_i & x'(1) &= k_f \cos\theta_f \\ y'(0) &= k_i \sin\theta_i & y'(1) &= k_f \sin\theta_f, \end{aligned}$$

where $k_i \neq 0$, $k_f \neq 0$ are free parameters that must however have the same sign. This condition is necessary to guarantee that the unicycle arrives in $\boldsymbol{q}_f$ with the same kind of motion (forward or backward) with which it leaves $\boldsymbol{q}_i$;

in fact, since $x(s)$ and $y(s)$ are cubic polynomials, the Cartesian path does not contain motion inversions in general.

For example, by letting $k_i = k_f = k > 0$, one obtains

$$\begin{bmatrix} \alpha_x \\ \alpha_y \end{bmatrix} = \begin{bmatrix} k\cos\theta_f - 3x_f \\ k\sin\theta_f - 3y_f \end{bmatrix} \qquad \begin{bmatrix} \beta_x \\ \beta_y \end{bmatrix} = \begin{bmatrix} k\cos\theta_i + 3x_i \\ k\sin\theta_i + 3y_i \end{bmatrix}.$$

The choice of k_i and k_f has a precise influence on the obtained path. In fact, by using (11.46) it is easy to verify that

$$\widetilde{v}(0) = k_i \qquad \widetilde{v}(1) = k_f.$$

The evolution of the robot orientation along the path and the associated geometric inputs can then be computed by using Eqs. (11.45) and (11.46), (11.47), respectively.

Planning via the chained form

Another technique, which can be immediately generalized to other kinematic models of mobile robots (e.g., the bicycle), is planning the path in the chained form coordinates $\boldsymbol{z}$. To this end, it is first necessary to compute the initial and final values $\boldsymbol{z}_i$ and $\boldsymbol{z}_f$ that correspond to $\boldsymbol{q}_i$ and $\boldsymbol{q}_f$, by using the change of coordinates (11.23). It is then sufficient to interpolate the initial and final values of z_1 and z_3 (the flat outputs) with the appropriate boundary conditions on the remaining variable $z_2 = z_3'/z_1'$.

Again, it is possible to adopt a cubic polynomial to solve the problem. As an alternative, one may use polynomials of different degree for x and y in order to reduce the number of unknown coefficients to be computed. For example, under the assumption $z_{1,i} \neq z_{1,f}$, consider the following interpolation scheme:

$$\begin{aligned} z_1(s) &= z_{1,f}s - (s-1)z_{1,i} \\ z_3(s) &= s^3 z_{3,f} - (s-1)^3 z_{3,i} + \alpha_3 s^2(s-1) + \beta_3 s(s-1)^2, \end{aligned}$$

with $s \in [0,1]$. Note that $z_1'(s)$ is constant and equal to $z_{1,f} - z_{1,i} \neq 0$. The unknowns α_3, β_3 must be determined by imposing the boundary conditions on z_2:

$$\frac{z_3'(0)}{z_1'(0)} = z_{2i} \qquad \frac{z_3'(1)}{z_1'(1)} = z_{2f},$$

from which

$$\begin{aligned} \alpha_3 &= z_{2,f}(z_{1,f} - z_{1,i}) - 3z_{3,f} \\ \beta_3 &= z_{2,i}(z_{1,f} - z_{1,i}) + 3z_{3,i}. \end{aligned}$$

This scheme cannot be directly applied when $z_{1,i} = z_{1,f}$, i.e., when $\theta_i = \theta_f$. To handle this singular case, one may introduce a *via point* $\boldsymbol{q}_v = [\, x_v \quad y_v \quad \theta_v \,]^T$

such that $\theta_v \neq \theta_i$, and solve the original planning problem using two consecutive paths, the first from $\boldsymbol{q}_i$ to $\boldsymbol{q}_v$ and the second from $\boldsymbol{q}_v$ to $\boldsymbol{q}_f$. Another possibility, which avoids the introduction of the via point, is to let $z_{1,f} = z_{1,i} + 2\pi$ (i.e., to replace θ_f with $\theta_f + 2\pi$); this obviously corresponds to the same final configuration of the unicycle. With the resulting manoeuvre, the robot will reach its destination while performing a complete rotation of orientation along the path.

Once the path has been planned for the chained form, the path $\boldsymbol{q}(s)$ in the original coordinates and the associated geometric inputs $\widetilde{\boldsymbol{u}}(s)$ are reconstructed by inverting the change of coordinates (11.23) and of inputs (11.24), respectively.

Planning via parameterized inputs

A conceptually different approach to path planning consists of writing the inputs — rather than the path — in parameterized form, and computing the value of the parameters so as to drive the robot from $\boldsymbol{q}_i$ to $\boldsymbol{q}_f$. Again, it is convenient to work on the chained form, whose equations are easily integrable in closed form under appropriate inputs. For the sake of generality, refer to the $(2, n)$ chained form (11.20), whose geometric version is

$$\begin{aligned} z_1' &= \widetilde{v}_1 \\ z_2' &= \widetilde{v}_2 \\ z_3' &= z_2 \widetilde{v}_1 \\ &\vdots \\ z_n' &= z_{n-1} \widetilde{v}_1. \end{aligned}$$

Let the geometric input be chosen as

$$\widetilde{v}_1 = \mathrm{sgn}(\Delta) \tag{11.48}$$

$$\widetilde{v}_2 = c_0 + c_1 s + \ldots + c_{n-2} s^{n-2}, \tag{11.49}$$

with $\Delta = z_{1,f} - z_{1,i}$ and $s \in [s_i, s_f] = [0, |\Delta|]$. Parameters $c_0, \ldots, c_{n-2}$ must be chosen so as to give $\boldsymbol{z}(s_f) = \boldsymbol{z}_f$. It is possible to verify that such condition is expressed as a linear system of equations

$$\boldsymbol{D}(\Delta) \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_{n-2} \end{bmatrix} = \boldsymbol{d}(\boldsymbol{z}_i, \boldsymbol{z}_f, \Delta) \tag{11.50}$$

where matrix $\boldsymbol{D}(\Delta)$ is invertible if $\Delta \neq 0$. For example, in the case of the $(2, 3)$ chained form, one obtains

$$\boldsymbol{D} = \begin{bmatrix} |\Delta| & \dfrac{\Delta^2}{2} \\ \mathrm{sgn}(\Delta)\dfrac{\Delta^2}{2} & \dfrac{\Delta^3}{6} \end{bmatrix} \quad \boldsymbol{d} = \begin{bmatrix} z_{2,f} - z_{2,i} \\ z_{3,f} - z_{3,i} - z_{2,i}\Delta \end{bmatrix}. \tag{11.51}$$

If $z_{1,i} = z_{1,f}$ a singular case is met that can be handled as before.

Both $\boldsymbol{z}(s)$ and $\widetilde{\boldsymbol{v}}(s)$ must then be converted to $\boldsymbol{q}(s)$ and $\widetilde{\boldsymbol{u}}(s)$ using the inverse coordinate and input transformations that apply to the specific case.

The above method does not make explicit use of the flat outputs, but relies on the closed-form integrability of the chained form, whose existence, as already mentioned, is equivalent to differential flatness. Note also that in this planning scheme, as in the previous two, parameter s does not represent the arc length on the path.

Other classes of parameterized inputs that can be used in place of (11.48), (11.49) are sinusoidal and piecewise constant functions.

Numerical results

For illustration, some numerical results of the planning methods so far described are now presented. The considered vehicle is a unicycle that must perform various 'parking' manoeuvres.

Two typical paths produced by the planner that uses cubic Cartesian polynomials are shown in Fig. 11.5. As already noticed, the unicycle never inverts its motion, that is forward in these two manoeuvres because $k = 5 > 0$. For $k < 0$, the manoeuvres would have been performed in backward motion with different paths.

In Fig. 11.6, the same planner is used to solve a *parallel parking* problem, in which the difference between the initial and the final configuration of the unicycle is a pure displacement in the direction orthogonal to the sagittal axis. Note how the path changes as $k_i = k_f = k$ is changed; in particular, an increase in k leads to elongated 'take-off' (from $\boldsymbol{q}_i$) and 'landing' (on $\boldsymbol{q}_f$) phases.

Figure 11.7 refers to the case in which $\boldsymbol{q}_i$ and $\boldsymbol{q}_f$ differ only for the value of θ (a pure reorientation); the unicycle leaves the initial position and follows a path that leads back to it with the correct orientation. This behaviour is to be expected, because with this planner a Cartesian path of nonzero length is needed to achieve any kind of reconfiguration.

To allow a comparison, the same planning problems have also been solved with the method based on the use of parameterized inputs in conjunction with the chained form.

Figure 11.8 shows the path obtained with this planner for the same two parking problems of Fig. 11.5. While in the first case the obtained manoeuvre is similar to the one obtained before, in the second the path contains a cusp, corresponding to a motion inversion. In fact, in view of its nature, this planner can only generate paths along which the robot orientation stays between its initial value θ_i and its final value θ_f.

In Fig. 11.9, two different solutions produced by this planner are reported for the parallel parking problem of Fig. 11.6. The singularity due to $\theta_i = \theta_f$ has been solved in two different ways: by adding a via point $\boldsymbol{q}_v$, and redefining θ_f as $\theta_f = \theta_i + 2\pi$. Note that in the latter case the path produced by the

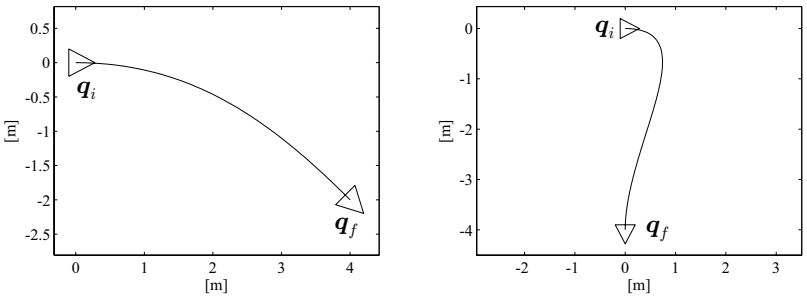

Fig. 11.5. Two parking manoeuvres planned via cubic Cartesian polynomials; in both cases $k = 5$ has been used

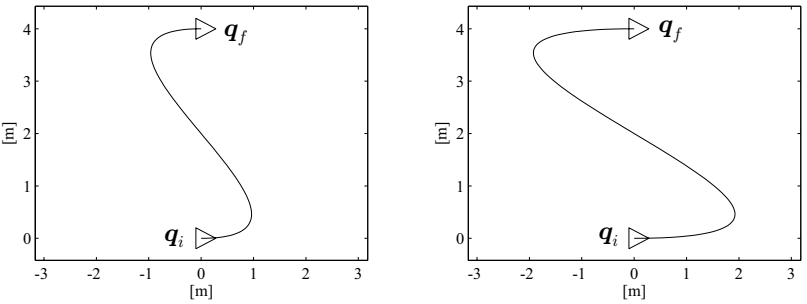

Fig. 11.6. Planning a parallel parking manoeuvre via cubic Cartesian polynomials; *left*: with $k = 10$, *right*: with $k = 20$

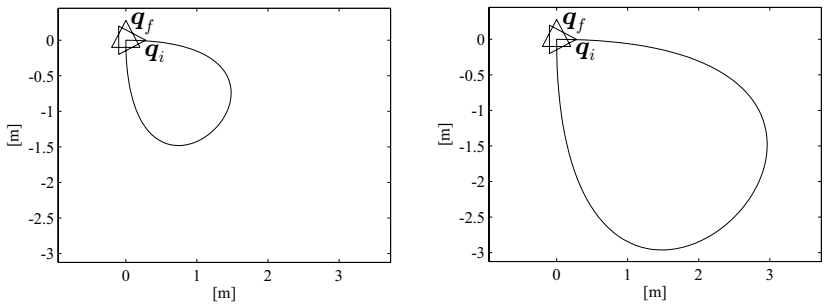

Fig. 11.7. Planning a pure reorientation manoeuvre via cubic Cartesian polynomials; *left*: with $k = 10$, *right*: with $k = 20$

planner leads the robot to the destination with a complete rotation of its orientation.

Finally, the same pure reorientation manoeuvre of Fig. 11.7 has been considered in Fig. 11.10. The path on the left has been obtained as outlined before, i.e., exploiting the transformations of coordinates (11.23) and of in-

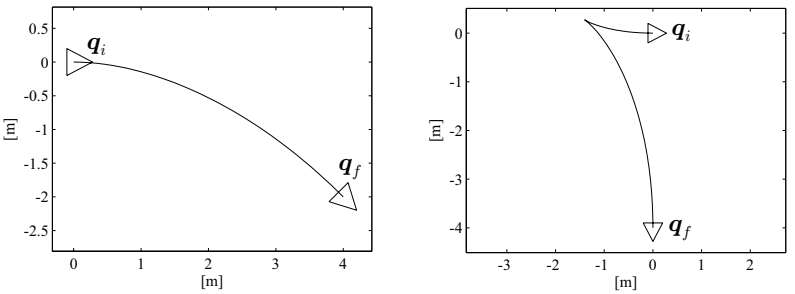

Fig. 11.8. Two parking manoeuvres planned via the chained form

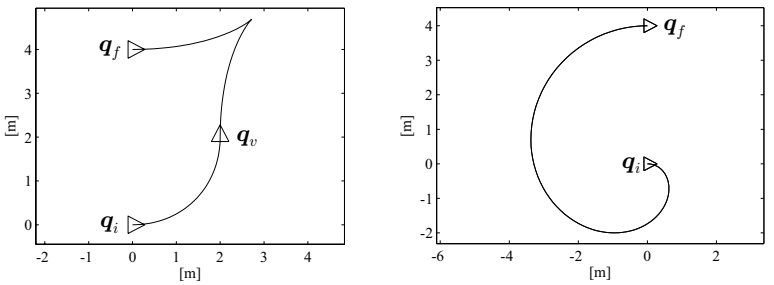

Fig. 11.9. Planning a parallel parking manoeuvre via the chained form; *left*: adding a via point $\boldsymbol{q}_v$, *right*: letting $\theta_f = \theta_i + 2\pi$

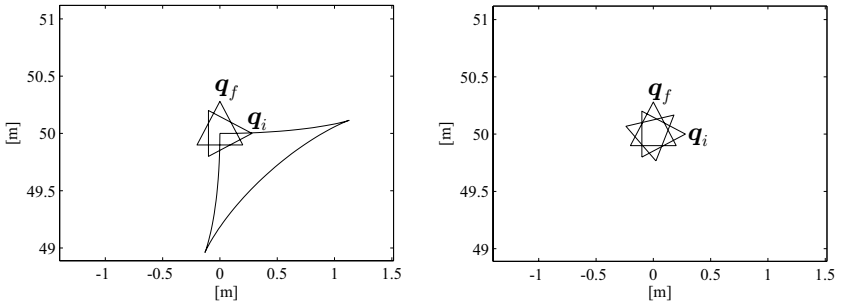

Fig. 11.10. Planning a pure reorientation manoeuvre via the chained form; *left*: with the coordinate transformation (11.23), *right*: with the coordinate transformation (11.52)

puts (11.24) to put the system in chained form, and then using the parameterized inputs (11.48), (11.49). As in the previous case, the required reorientation is realized by a Cartesian path. This is a consequence of the structure of (11.23), for which $\theta_i \neq \theta_f$ implies in general $z_{2,i} \neq z_{2,f}$ and $z_{3,i} \neq z_{3,f}$, even when $x_i = x_f$, $y_i = y_f$.

The manoeuvre on the right of Fig. 11.10, which is a rotation on the spot, was achieved by using a different change of coordinates to put the system in (2,3) chained form. In particular, the following transformation has been used

$$\begin{aligned} z_1 &= \theta - \theta_f \\ z_2 &= (x - x_i)\cos\theta + (y - y_i)\sin\theta \\ z_3 &= (x - x_i)\sin\theta - (y - y_i)\cos\theta, \end{aligned} \tag{11.52}$$

which places the origin of the (z_2, z_3) reference frame in correspondence of the initial Cartesian position of the unicycle. With this choice one has $z_{2,i} = z_{2,f}$ and $z_{3,i} = z_{3,f}$ for a pure reorientation, and thus the manoeuvre is efficiently obtained as a simple rotation.

As a matter of fact, using (11.52) in place of (11.23) is always recommended. In fact, the analysis of (11.51) shows that in general the magnitude of the coefficients of $\widetilde{v}_2$ — and therefore, the length of the obtained path — depends not only on the amount of reconfiguration required for z_2 and z_3, but also on the value of $z_{2,i}$ itself. The adoption of (11.52), which implies $z_{2,i} = 0$, makes the size of the manoeuvre invariant with respect to the Cartesian position of the unicycle.

11.5.4 Trajectory Planning

Once a path $\boldsymbol{q}(s)$, $s \in [s_i, s_f]$, has been determined, it is possible to choose a timing law $s = s(t)$ with which the robot should follow it. In this respect, considerations similar to those of Sect. 5.3.1 apply. For example, if the velocity inputs of the unicycle are subject to bounds of the form[8]

$$|v(t)| \leq v_{\max} \qquad |\omega(t)| \leq \omega_{\max} \qquad \forall t, \tag{11.53}$$

it is necessary to verify whether the velocities along the planned trajectory are admissible. In the negative case, it is possible to slow down the timing law via *uniform scaling*. To this end, it is convenient to rewrite the timing law by replacing t with the normalized time variable $\tau = t/T$, with $T = t_f - t_i$. From (11.43), (11.44) one has

$$v(t) = \widetilde{v}(s)\frac{ds}{d\tau}\frac{d\tau}{dt} = \widetilde{v}(s)\frac{ds}{d\tau}\frac{1}{T} \tag{11.54}$$

$$\omega(t) = \widetilde{\omega}(s)\frac{ds}{d\tau}\frac{d\tau}{dt} = \widetilde{\omega}(s)\frac{ds}{d\tau}\frac{1}{T}, \tag{11.55}$$

and therefore is is sufficient to increase T (i.e., the duration of the trajectory) to reduce uniformly v and ω, so as to stay within the given bounds.

[8] For a differential drive unicycle, the actual bounds affect the wheel angular speeds ω_L and ω_R. Through Eqs. (11.14), these bounds can be mapped to constraints on v and ω (see Problem 11.9).

It is also possible to plan directly a trajectory without separating the geometric path from the timing law. To this end, all the techniques presented before can be used with the time variable t directly in place of the path parameter s. A drawback of this approach is that the duration $t_f - t_i = s_f - s_i$ of the trajectory is fixed, and uniform scaling cannot be used to satisfy bounds on the velocity inputs. In fact, an increase (or decrease) of $t_f - t_i$ would modify the geometric path associated with the planned trajectory.

11.5.5 Optimal Trajectories

The planning techniques so far presented can be used to compute trajectories that lead the robot from an initial configuration $\boldsymbol{q}_i$ to a final configuration $\boldsymbol{q}_f$ while complying with the nonholonomic constraints and, possibly, bounds on the velocity inputs. Often, other requirements are added, such as limiting the path curvature, avoiding workspace obstacles or reducing energy consumption. In general, these are integrated in the design procedure as the optimization of a suitable cost criterion along the trajectory. For example, the previous objectives will be respectively formulated as the minimization of the maximum curvature, the maximization of the minimum distance between the robot and the obstacles, or the minimization of the total energy needed by the mobile robot to follow the path.

A simple technique for attacking the optimal planning problem consists of *over-parameterizing* the adopted interpolation scheme, so as to pursue the optimization — typically, via numerical techniques — of the cost criterion by appropriately choosing the redundant parameters. Clearly, the obtained trajectory will be optimal only with respect to the set of trajectories that can be generated by the chosen scheme, and will be a *suboptimal* solution for the original planning problem; this may or may not lead to the fulfilment of the original specifications. For example, the planning scheme based on cubic Cartesian polynomials contains two free parameters (k_i and k_f), that may be chosen so as to maximize the minimum distance along the path between the unicycle and certain obstacles. However, depending on the placement of the obstacles with respect to $\boldsymbol{q}_i$ and $\boldsymbol{q}_f$, a collision-free path (i.e., a path for which the above distance is always positive) may or may not[9] exist within the chosen family of cubic polynomials.

A more systematic approach to the problem relies on the use of *optimal control* theory. The basic problem considered in this discipline is in fact the determination of a control law that transfers a dynamic system between two assigned states so as to minimize a chosen cost functional along the trajectory. A powerful tool for solving this problem is the *Pontryagin minimum principle* that provides *necessary* conditions for optimality. By exploiting these conditions in conjunction with the analysis of the specific characteristics of the

[9] The complexity of the problem of planning collision-free motions in the presence of obstacles is such that specific solution techniques are needed; these are presented in Chap. 12.

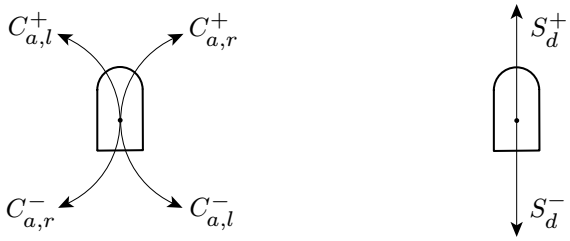

Fig. 11.11. The elementary arcs that constitute the trajectories of the sufficient family for the minimum-time planning problem for the unicycle

considered problem, it is often possible to identify a reduced set of candidate trajectories, also referred to as a *sufficient family*, among which there is certainly the desired optimal solution (if it exists).

In any case, each optimal planning problem must be formulated in the appropriate context. When minimizing curvature or avoiding static obstacles, the timing law part of the trajectory is irrelevant, and the problem can be solved by planning a path for the geometric model (11.41). If the cost criterion depends on the path as well as on the timing law, it is necessary to plan directly on the kinematic model (11.10). A particularly important example of the latter situation is met when minimum-time trajectories are sought in the presence of bounds on the velocity inputs.

Minimum-time trajectories

Consider the problem of transferring the unicycle (11.13) from the initial configuration $\boldsymbol{q}_i$ to the final configuration $\boldsymbol{q}_f$ while minimizing the functional

$$J = t_f - t_i = \int_{t_i}^{t_f} dt,$$

under the assumption that the driving and steering velocity inputs v and ω are bounded as in (11.53). By combining the conditions provided by the minimum principle with geometrical arguments, it is possible to determine a sufficient family for the solution to this problem. This family consists of trajectories obtained by concatenating *elementary arcs* of two kinds only:

- arcs of circle of variable length, covered with velocities $v(t) = \pm v_{\max}$ and $\omega(t) = \pm\omega_{\max}$ (the radius of the circle is always $v_{\max}/\omega_{\max}$);
- line segments of variable length, covered with velocities $v(t) = \pm v_{\max}$ and $\omega(t) = 0$.

These elementary arcs are shown in Fig. 11.11, where a compact notation is also defined for identifying them. In particular, C_a and S_d indicate an arc of circle of duration a and a line segment of duration d, respectively (in the

particular case $v_{\max} = 1$, a and d are also the lengths of these arcs). The superscript indicates forward (+) o'r backward (−) motion, while for circular arcs the second subscript indicates a rotation in the clockwise (r) or counterclockwise (l) direction. With this notation, and considering for simplicity the case $v_{\max} = 1$ and $\omega_{\max} = 1$, the trajectories of the sufficient family (also called *Reeds–Shepp curves*) can be classified in the following nine *groups*:

$$
\begin{array}{lll}
\text{I} & C_a|C_b|C_e & a \geq 0, b \geq 0, e \geq 0, a+b+e \leq \pi \\
\text{II} & C_a|C_bC_e & 0 \leq a \leq b, 0 \leq e \leq b, 0 \leq b \leq \pi/2 \\
\text{III} & C_aC_b|C_e & 0 \leq a \leq b, 0 \leq e \leq b, 0 \leq b \leq \pi/2 \\
\text{IV} & C_aC_b|C_bC_e & 0 \leq a \leq b, 0 \leq e \leq b, 0 \leq b \leq \pi/2 \\
\text{V} & C_a|C_bC_b|C_e & 0 \leq a \leq b, 0 \leq e \leq b, 0 \leq b \leq \pi/2 \\
\text{VI} & C_a|C_{\pi/2}S_eC_{\pi/2}|C_b & 0 \leq a \leq \pi/2, 0 \leq b \leq \pi/2, e \geq 0 \\
\text{VII} & C_a|C_{\pi/2}S_eC_b & 0 \leq a \leq \pi, 0 \leq b \leq \pi/2, e \geq 0 \\
\text{VIII} & C_aS_eC_{\pi/2}|C_b & 0 \leq a \leq \pi/2, 0 \leq b \leq \pi, e \geq 0 \\
\text{IX} & C_aS_eC_b & 0 \leq a \leq \pi/2, 0 \leq b \leq \pi/2, e \geq 0,
\end{array}
\tag{11.56}
$$

where the symbol "|" between two elementary arcs indicates the presence of a cusp (motion inversion) on the path. Each group contains trajectories consisting of a *sequence* of no more than five elementary arcs; the duration of circular arcs is bounded by either $\pi/2$ or π, while the duration of line segments depends on the Cartesian distance between $\boldsymbol{q}_i$ and $\boldsymbol{q}_f$. The number of possible sequences produced by each group is finite; they are obtained by instantiating the elementary arcs depending on the direction of motion and of rotation. For example, it is easy to show that group IX generates eight sequence types, each corresponding to a trajectory entirely covered in forward or backward motion:

$$
\begin{array}{l}
C^+_{a,r}S^+_eC^+_{a,r},\ C^+_{a,r}S^+_eC^+_{a,l},\ C^+_{a,l}S^+_eC^+_{a,r},\ C^+_{a,l}S^+_eC^+_{a,l} \\
C^-_{a,r}S^-_eC^-_{a,r},\ C^-_{a,r}S^-_eC^-_{a,l},\ C^-_{a,l}S^-_eC^-_{a,r},\ C^-_{a,l}S^-_eC^-_{a,l}.
\end{array}
$$

By this kind of argument, it may be verified that the above groups generate a total number of 48 different sequences.

In practice, one may use an exhaustive algorithm to identify the minimum-time trajectory that leads the unicycle from $\boldsymbol{q}_i$ to $\boldsymbol{q}_f$:

- Determine all the trajectories belonging to the sufficient family that join $\boldsymbol{q}_i$ and $\boldsymbol{q}_f$.
- Compute the value of the cost criterion $t_f - t_i$ along these trajectories, and choose the one to which the minimum value is associated.

The first is clearly the most difficult step. Essentially, for each of the aforementioned 48 sequences it is necessary to identify — if it exists — the corresponding trajectory going from $\boldsymbol{q}_i$ to $\boldsymbol{q}_f$, and to compute the duration of the associated elementary arcs. To this end, for each sequence, it is possibly

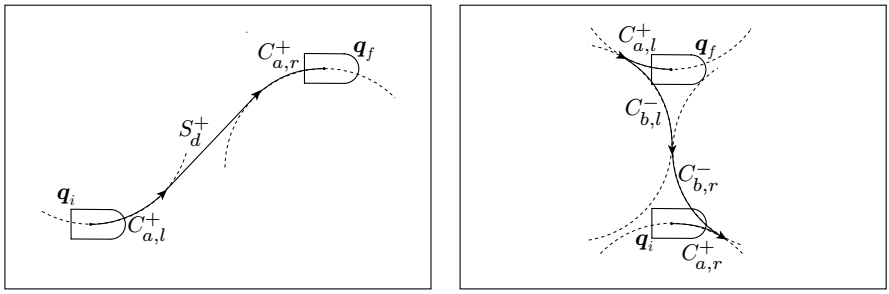

Fig. 11.12. Two examples of minimum-time trajectories for the unicycle

to express and invert in closed form the relationship between the duration of the arcs and the obtained change in configuration. By doing so, the first step can also be completed very quickly.

Figure 11.12 shows two examples of minimum-time trajectories for the unicycle. The first (on the left) belongs to group IX and contains three elementary arcs without inversions of motion. The second (on the right) is a group V trajectory consisting of four arcs of circle, along which there are two motion inversions.

11.6 Motion Control

The motion control problem for wheeled mobile robots is generally formulated with reference to the kinematic model (11.10), i.e., by assuming that the control inputs determine directly the generalized velocities $\dot{\boldsymbol{q}}$. For example, in the case of the unicycle (11.13) and of the bicycle (11.18) or (11.19) this means that the inputs are the driving and steering velocity inputs v and ω. There are essentially two reasons for taking this simplifying assumption. First, as already seen in Sect. 11.4, under suitable assumptions it is possible to cancel the dynamic effects via state feedback, so that the control problem is actually transferred to the second-order kinematic model, and from the latter to the first-order kinematic model. Second, in the majority of mobile robots it is not possible to command directly the wheel torques, because there are *low-level* control loops that are integrated in the hardware or software architecture. These loops accept as input a reference value for the wheel angular speed, that is reproduced as accurately as possible by standard regulation actions (e.g., PID controllers). In this situation, the actual inputs available for high-level controls are precisely the reference velocities.

In this section, a unicycle-like vehicle is again considered, although some of the presented control schemes may be extended to other kinds of mobile robots. Two basic control problems, illustrated in Fig. 11.13, will be considered for system (11.13):

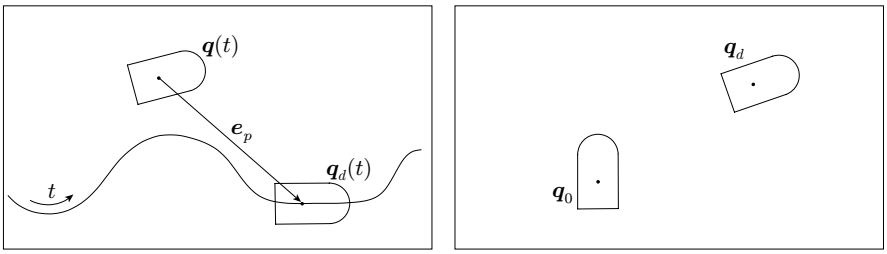

Fig. 11.13. Control problems for a unicycle; *left*: trajectory tracking, *right*: posture regulation

- *Trajectory tracking*: the robot must asymptotically track a desired Cartesian trajectory $(x_d(t), y_d(t))$, starting from an initial configuration $\boldsymbol{q}_0 = [\,x_0 \quad y_0 \quad \theta_0\,]^T$ that may or may not be 'matched' with the trajectory.
- *Posture regulation*: the robot must asymptotically reach a given posture, i.e., a desired configuration $\boldsymbol{q}_d$, starting from an initial configuration $\boldsymbol{q}_0$.

From a practical point of view, the most relevant of these problems is certainly the first. This is because, unlike industrial manipulators, mobile robots must invariably operate in unstructured workspaces containing obstacles. Clearly, forcing the robot to move along (or close to) a trajectory planned in advance considerably reduces the risk of collisions. On the other hand, a preliminary planning step is not required when posture regulation is performed, but the Cartesian trajectory along which the robot approaches $\boldsymbol{q}_d$ cannot be specified by the user.

11.6.1 Trajectory Tracking

For the tracking problem to be soluble, it is necessary that the desired Cartesian trajectory $(x_d(t), y_d(t))$ is admissible for the kinematic model (11.13), i.e., it must satisfy the equations

$$\begin{aligned} \dot{x}_d &= v_d \cos\theta_d \\ \dot{y}_d &= v_d \sin\theta_d \\ \dot{\theta}_d &= \omega_d \end{aligned} \tag{11.57}$$

for some choice of the reference inputs v_d and ω_d. For example, this is certainly true if the trajectory has been produced using one of the planning schemes of the previous section. In any case, as seen in Sect. 11.5.2, since the unicycle coordinates x and y are flat outputs ,the orientation along the desired trajectory $(x_d(t), y_d(t))$ can be computed as

$$\theta_d(t) = \mathrm{Atan2}\,(\dot{y}_d(t), \dot{x}(t)_d) + k\pi \qquad k = 0, 1, \tag{11.58}$$

as well as the reference inputs

$$v_d(t) = \pm\sqrt{\dot{x}_d^2(t) + \dot{y}_d^2(t)} \tag{11.59}$$

$$\omega_d(t) = \frac{\ddot{y}_d(t)\dot{x}_d(t) - \ddot{x}_d(t)\dot{y}_d(t)}{\dot{x}_d^2(t) + \dot{y}_d^2(t)}. \tag{11.60}$$

Note that (11.58) and (11.59), (11.60), correspond respectively to (11.45) and (11.46), (11.47) with $s = t$. In the following, it is assumed that the value of k in (11.58) — and correspondingly, the sign of v_d in (11.59) — has been chosen.

By comparing the desired state $\boldsymbol{q}_d(t) = [\,x_d(t) \quad y_d(t) \quad \theta_d(t)\,]^T$ with the current measured state $\boldsymbol{q}(t) = [\,x(t) \quad y(t) \quad \theta(t)\,]^T$ it is possible to compute an error vector that can be fed to the controller. However, rather than using directly the difference between $\boldsymbol{q}_d$ and $\boldsymbol{q}$, it is convenient to define the tracking error as

$$\boldsymbol{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d - x \\ y_d - y \\ \theta_d - \theta \end{bmatrix}.$$

The positional part of $\boldsymbol{e}$ is the Cartesian error $\boldsymbol{e}_p = [\,x_d - x \quad y_d - y\,]^T$ expressed in a reference frame aligned with the current orientation θ of the robot (see Fig. 11.13). By differentiating $\boldsymbol{e}$ with respect to time, and using (11.13) and (11.57), one easily finds

$$\begin{aligned} \dot{e}_1 &= v_d \cos e_3 - v + e_2\,\omega \\ \dot{e}_2 &= v_d \sin e_3 - e_1\,\omega \\ \dot{e}_3 &= \omega_d - \omega. \end{aligned} \tag{11.61}$$

Using the input transformation

$$v = v_d \cos e_3 - u_1 \tag{11.62}$$

$$\omega = \omega_d - u_2, \tag{11.63}$$

which is clearly invertible, the following expression is obtained for the tracking error dynamics:

$$\dot{\boldsymbol{e}} = \begin{bmatrix} 0 & \omega_d & 0 \\ -\omega_d & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{e} + \begin{bmatrix} 0 \\ \sin e_3 \\ 0 \end{bmatrix} v_d + \begin{bmatrix} 1 & -e_2 \\ 0 & e_1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{11.64}$$

Note that the first term of this dynamics is linear, while the second and third are nonlinear. Moreover, the first and second terms are in general time-varying, due to the presence of the reference inputs $v_d(t)$ and $\omega_d(t)$.

Control based on approximate linearization

The simplest approach to designing a tracking controller consists of using the approximate linearization of the error dynamics around the reference trajectory, on which clearly $\boldsymbol{e} = \boldsymbol{0}$. This approximation, whose accuracy increases as the tracking error $\boldsymbol{e}$ decreases, is obtained from (11.64) simply setting $\sin e_3 = e_3$ and evaluating the input matrix on the trajectory. The result is

$$\dot{\boldsymbol{e}} = \begin{bmatrix} 0 & \omega_d & 0 \\ -\omega_d & 0 & v_d \\ 0 & 0 & 0 \end{bmatrix} \boldsymbol{e} + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}. \tag{11.65}$$

Note that the approximate system is still time-varying. Consider now the linear feedback

$$u_1 = -k_1\, e_1 \tag{11.66}$$
$$u_2 = -k_2\, e_2 - k_3\, e_3 \tag{11.67}$$

that leads to the following closed-loop linearized dynamics:

$$\dot{\boldsymbol{e}} = \boldsymbol{A}(t)\, \boldsymbol{e} = \begin{bmatrix} -k_1 & \omega_d & 0 \\ -\omega_d & 0 & v_d \\ 0 & -k_2 & -k_3 \end{bmatrix} \boldsymbol{e}. \tag{11.68}$$

The characteristic polynomial of matrix $\boldsymbol{A}$ is

$$p(\lambda) = \lambda(\lambda + k_1)(\lambda + k_3) + \omega_d^2(\lambda + k_3) + v_d k_2(\lambda + k_1).$$

At this point, it is sufficient to let

$$k_1 = k_3 = 2\zeta a \qquad k_2 = \frac{a^2 - \omega_d^2}{v_d}, \tag{11.69}$$

with $\zeta \in (0, 1)$ and $a > 0$, to obtain

$$p(\lambda) = (\lambda + 2\zeta a)(\lambda^2 + 2\zeta a\lambda + a^2).$$

The closed-loop linearized error dynamics is then characterized by three constant eigenvalues: one real negative eigenvalue in $-2\zeta a$ and a pair of complex eigenvalues with negative real part, damping coefficient ζ and natural frequency a. However, in view of its time-varying nature, there is no guarantee that system (11.68) is asymptotically stable.

A notable exception is when v_d and ω_d are constant, as in the case of circular or rectilinear trajectories. In fact, the linearized system (11.68) is then time-invariant and therefore asymptotically stable with the choice of gains in (11.69). Hence, by using the control law (11.66), (11.67) with the same gains, the origin of the original error system (11.64) is also asymptotically stable, although this result is not guaranteed to hold globally. For sufficiently

small initial errors, the unicycle will certainly converge to the desired Cartesian trajectory (either circular or rectilinear).

In general, the feedback controller (11.66), (11.67) is linear but also time-varying in view of the expression of k_2 in (11.69). The actual velocity inputs v and ω should be reconstructed from u_1 and u_2 through (11.62), (11.63). In particular, it is easy to verify that v and ω tend to coincide with the reference inputs v_d and ω_d (i.e., they reduce to a pure feedforward action) as the tracking error $\boldsymbol{e}$ vanishes.

Finally, note that k_2 in (11.69) diverges when v_d goes to zero, i.e., when the reference Cartesian trajectory tends to stop. Therefore, the above control scheme can only be used for *persistent* Cartesian trajectories, i.e., trajectories such that $|v_d(t)| \geq \bar{v} > 0, \forall t \geq 0$. This also means that motion inversions (from forward to backward motion, or vice versa) on the reference trajectory are not allowed.

Nonlinear control

Consider again the exact expression (11.64) of the tracking error dynamics, now rewritten for convenience in the 'mixed' form:

$$\begin{aligned} \dot{e}_1 &= e_2\,\omega + u_1 \\ \dot{e}_2 &= v_d \sin e_3 - e_1\omega \\ \dot{e}_3 &= u_2, \end{aligned} \tag{11.70}$$

and the following nonlinear version of the control law (11.66), (11.67):

$$u_1 = -k_1(v_d, \omega_d)\, e_1 \tag{11.71}$$

$$u_2 = -k_2\, v_d\, \frac{\sin e_3}{e_3}\, e_2 - k_3(v_d, \omega_d)\, e_3, \tag{11.72}$$

where $k_1(\cdot,\cdot) > 0$ and $k_3(\cdot,\cdot) > 0$ are bounded functions with bounded derivatives, and $k_2 > 0$ is constant. If the reference inputs v_d and ω_d are also bounded with bounded derivatives, and they do not both converge to zero, the tracking error $\boldsymbol{e}$ converges to zero globally, i.e., for any initial condition.

A sketch is now given of the proof of this result. Consider the closed-loop error dynamics

$$\begin{aligned} \dot{e}_1 &= e_2\,\omega - k_1(v_d, \omega_d)\, e_1 \\ \dot{e}_2 &= v_d \sin e_3 - e_1\,\omega \\ \dot{e}_3 &= -k_2\, v_d \frac{\sin e_3}{e_3}\, e_2 - k_3(v_d, \omega_d)\, e_3, \end{aligned} \tag{11.73}$$

and the candidate Lyapunov function

$$V = \frac{k_2}{2}\left(e_1^2 + e_2^2\right) + \frac{e_3^2}{2},$$

whose time derivative along the system trajectories

$$\dot{V} = -k_1(v_d, \omega_d)k_2\, e_1^2 - k_3(v_d, \omega_d)e_3^2$$

is negative semi-definite. This means that V (which is bounded below) tends to a limit value, and also that the norm of $\boldsymbol{e}$ is bounded. As system (11.73) is time-varying, it is not possible to use La Salle theorem to gain further insight. However, under the above assumptions, one may verify that $\ddot{V}$ is limited, and therefore $\dot{V}$ is uniformly continuous. Hence, Barbalat lemma (see Sect. C.3) may be used to conclude that $\dot{V}$ tends to zero, i.e., that e_1 and e_3 tend to zero. From this and the system equations it is possible to prove that

$$\lim_{t\to\infty} (v_d^2 + \omega_d^2)e_2^2 = 0,$$

and thus e_2 tends to zero as well, provided that at least one of the reference inputs is persistent.

Again, the actual velocity inputs v and ω must be computed from u_1 and u_2 using (11.62), (11.63). Note that the control law (11.71), (11.72) requires the persistency of the state trajectory $\boldsymbol{q}(t)$, but not of the Cartesian trajectory. In particular, the reference velocity $v_d(t)$ can converge to zero as long as $\omega_d(t)$ does not, and vice versa. For example, this controller can be used to track a Cartesian trajectory that degenerates to a simple rotation on the spot.

Input/output linearization

A well-known systematic approach to the design of trajectory tracking controllers is based on input/output linearization via feedback (see Sect. 8.5.2). In the case of the unicycle, consider the following outputs:

$$\begin{aligned} y_1 &= x + b\cos\theta \\ y_2 &= y + b\sin\theta, \end{aligned}$$

with $b \neq 0$. They represent the Cartesian coordinates of a point B located along the sagittal axis of the unicycle at a distance $|b|$ from the contact point of the wheel with the ground (see Fig. 11.3). In particular, B is 'ahead' of the contact point if b is positive and 'behind' if it is negative.

The time derivatives of y_1 and y_2 are

$$\begin{bmatrix} \dot{y}_1 \\ \dot{y}_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & -b\sin\theta \\ \sin\theta & b\cos\theta \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \boldsymbol{T}(\theta) \begin{bmatrix} v \\ \omega \end{bmatrix}. \tag{11.74}$$

Matrix $\boldsymbol{T}(\theta)$ has determinant b, and is therefore invertible under the assumption that $b \neq 0$. It is then sufficient to use the following input transformation

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \boldsymbol{T}^{-1}(\theta) \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta/b & \cos\theta/b \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

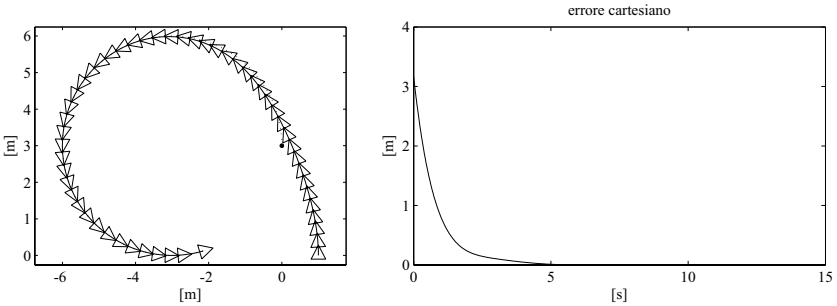

Fig. 11.14. Tracking a circular reference trajectory (*dotted*) with the controller based on approximate linearization; *left*: Cartesian motion of the unicycle, *right*: time evolution of the norm of the Cartesian error $\boldsymbol{e}_p$

to put the equations of the unicycle in the form

$$\begin{aligned} \dot{y}_1 &= u_1 \\ \dot{y}_2 &= u_2 \\ \dot{\theta} &= \frac{u_2 \cos\theta - u_1 \sin\theta}{b}. \end{aligned} \tag{11.75}$$

An input/output linearization via feedback has therefore been obtained. At this point, a simple linear controller of the form

$$u_1 = \dot{y}_{1d} + k_1(y_{1d} - y_1) \tag{11.76}$$

$$u_2 = \dot{y}_{2d} + k_2(y_{2d} - y_2), \tag{11.77}$$

with $k_1 > 0$, $k_2 > 0$, guarantees exponential convergence to zero of the Cartesian tracking error, with decoupled dynamics on its two components. Note that the orientation, whose evolution is governed by the third equation in (11.75), is not controlled. In fact, this tracking scheme does not use the orientation error; hence, it is based on the output error rather than the state error.

It should be emphasized that the reference Cartesian trajectory for point B can be arbitrary, and in particular the associated path may exhibit isolated points with discontinuous geometric tangent (like in a broken line) without requiring the robot to stop and reorient itself at those points. This is true as long as $b \neq 0$, and therefore such a possibility does not apply to the contact point between the wheel and the ground, whose velocity, as already discussed, cannot have a component in the direction orthogonal to the sagittal axis of the vehicle.

Simulations

An example of application of the trajectory tracking controller (11.66), (11.67) based on linear approximation is shown in Fig. 11.14. The desired trajectory

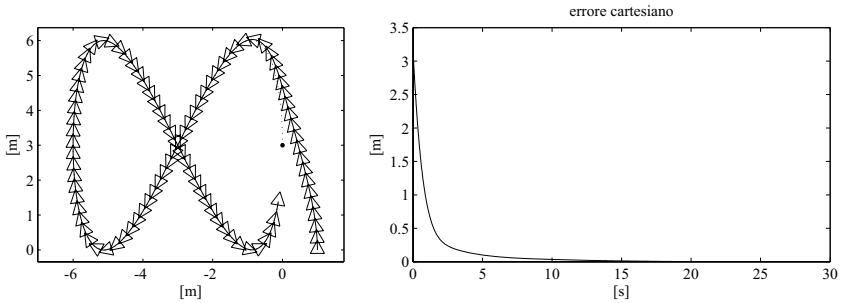

Fig. 11.15. Tracking a figure of eight-shaped reference trajectory (*dotted*) with the nonlinear controller; *left*: Cartesian motion of the unicycle, *right*: time evolution of the norm of the Cartesian error $\boldsymbol{e}_p$

in this case is the circle of centre (x_c, y_c) and radius R described by the parametric equations

$$x_d(t) = x_c + R\cos(\omega_d t)$$
$$y_d(t) = y_c + R\sin(\omega_d t),$$

with $R = 3$ m and $\omega_d = 1/3$ rad/s. Hence, the reference driving velocity on the circle is constant and equal to $v_d = R\,\omega_d = 1$ m/s. The controller gains have been chosen according to (11.69) with $\zeta = 0.7$ and $a = 1$. Note the exponential convergence to zero of the Cartesian error.

In the second simulation (Fig. 11.15) the reference trajectory is figure of eight-shaped, with centre in (x_c, y_c), and is described by the parametric equations

$$x_d(t) = x_c + R_1\sin(2\omega_d t)$$
$$y_d(t) = y_c + R_2\sin(\omega_d t),$$

with $R_1 = R_2 = 3$ m and $\omega_d = 1/15$ rad/s. The reference driving velocity $v_d(t)$ in this case varies over time and must be computed using (11.59). The results shown have been obtained with the nonlinear controller (11.71), (11.72), with k_1 and k_3 again given by (11.69) in which $\zeta = 0.7$ and $a = 1$, while k_2 has been set to 1. Also in this case, the error converges to zero very quickly.

The third Cartesian reference trajectory is a square with a side of 4 m, to be traced with constant velocity. This requirement means that the unicycle cannot stop at the vertices in order to reorient itself, and therefore the representative point (x, y) of the unicycle cannot follow the reference trajectory. As a consequence, the tracking scheme based on input/output linearization has been adopted. In particular, two simulations are presented, both obtained using the control law (11.76), (11.77) with $k_1 = k_2 = 2$. In the first, the point B that tracks the reference trajectory is located at a distance $b = 0.75$ m from the contact point between the wheel and the ground. As shown in Fig. 11.16,

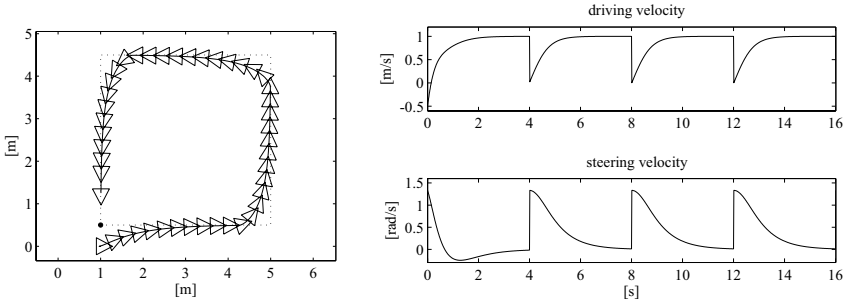

Fig. 11.16. Tracking a square reference trajectory (*dotted*) with the controller based on input/output linearization, with $b = 0.75$; *left*: Cartesian motion of the unicycle, *right*: time evolution of the velocity inputs v and ω

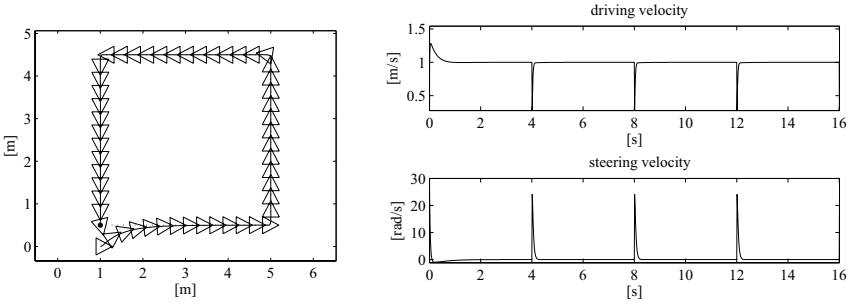

Fig. 11.17. Tracking a square reference trajectory (*dotted*) with the controller based on input/output linearization, with $b = 0.2$; *left*: Cartesian motion of the unicycle, *right*: time evolution of the velocity inputs v and ω

the unicycle actually moves on a 'smoothed' trajectory; therefore, an obstacle-free channel around the trajectory must be available to account for the area swept by the unicycle in correspondence of the square vertices.

The second simulation (Fig. 11.17) shows what can happen when b is reduced in order to achieve more accurate tracking for the unicycle representative point; in particular, $b = 0.2$ m was chosen in this case. While it is true that the unicycle tracks the square more closely with respect to the first simulation, the steering velocity is much higher in correspondence of the vertices, and this might be a problem in the presence of saturations on the velocity inputs. This situation is consistent with the fact that matrix $\boldsymbol{T}$ in (11.74) tends to become singular when b approaches zero.

11.6.2 Regulation

Consider now the problem of designing a feedback control law that drives the unicycle (11.13) to a desired configuration $\boldsymbol{q}_d$. A reasonable approach could

be to plan first a trajectory that stops in $\boldsymbol{q}_d$, and then track it via feedback. However, none of the tracking schemes so far described can be used to this end. In fact, the controller based on approximate linearization and the nonlinear controller both require a persistent state trajectory. The scheme based on input/output linearization can also track non-persistent trajectories, but will drive point B to the destination rather than the representative point of the unicycle. Besides, the final orientation at the destination will not be controlled.

Actually, the difficulty of identifying feedback control laws for tracking non-persistent trajectories is structural. It is in fact possible to prove that, due to the nonholonomy of the system, the unicycle does not admit any *universal* controller, i.e., a controller that can asymptotically stabilize arbitrary state trajectories, either persistent or not. This situation is completely different from the case of manipulators, for which the scheme based on inverse dynamics is an example of universal controller. As a consequence, the posture regulation problem in nonholonomic mobile robots must be addressed using purposely designed control laws.

Cartesian regulation

Consider first the problem of designing a feedback controller for a *partial* regulation task, in which the objective is to drive the unicycle to a given Cartesian position, without specifying the final orientation. This simplified version of the regulation problem is of practical interest. For example, a mobile robot exploring an unknown environment must visit a sequence of Cartesian positions (*view points*) from where it perceives the characteristics of the surrounding area using its on-board sensors. If these are isotropically distributed on the robot (as in the case of a ring of equispaced ultrasonic sensors, a rotating laser range finder, or a panoramic camera), the orientation of the robot at the view point is irrelevant.

Without loss of generality, assume that the desired Cartesian position is the origin; the Cartesian error $\boldsymbol{e}_p$ is then simply $[-x \quad -y]^T$. Consider the following control law

$$v = -k_1(x\cos\theta + y\sin\theta) \tag{11.78}$$

$$\omega = k_2(\text{Atan2}(y, x) - \theta + \pi), \tag{11.79}$$

where $k_1 > 0$, $k_2 > 0$. These two commands have an immediate geometric interpretation: the driving velocity v is proportional to the projection of the Cartesian error $\boldsymbol{e}_p$ on the sagittal axis of the unicycle, whereas the steering velocity ω is proportional to the difference between the orientation of the unicycle and that of vector $\boldsymbol{e}_p$ (*pointing error*).

Consider the following 'Lyapunov-like' function:

$$V = \frac{1}{2}(x^2 + y^2),$$

that is only positive semi-definite at the origin, because it is zero in all configurations such that $x = y = 0$, independently from the value of the orientation θ. Using the unicycle equations in (11.13) and the control inputs (11.78), (11.79) one obtains

$$\dot{V} = -k_1(x\cos\theta + y\sin\theta)^2,$$

that is negative semi-definite at the origin. This indicates that V, which is bounded below, tends to a limit value, and also that the position error $\boldsymbol{e}_p$ is bounded in norm. It is easy to verify that $\ddot{V}$ is also bounded, and thus $\dot{V}$ is uniformly continuous. Barbalat lemma[10] implies that $\dot{V}$ tends to zero. Hence

$$\lim_{t\to\infty}(x\cos\theta + y\sin\theta) = 0,$$

i.e., the projection of the Cartesian error vector $\boldsymbol{e}_p$ on the sagittal axis of the unicycle tends to vanish. This cannot happen in a point different from the origin, because the steering velocity (11.79) would then force the unicycle to rotate so as to align with $\boldsymbol{e}_p$. One may then conclude that the Cartesian error tends to zero for any initial configuration.

Posture regulation

To design a feedback controller that is able to regulate the whole configuration vector (Cartesian position and vehicle orientation) of the unicycle, it is convenient to formulate the problem in polar coordinates. It is again assumed, without loss of generality, that the desired configuration is the origin $\boldsymbol{q}_d = [\,0 \quad 0 \quad 0\,]^T$.

With reference to Fig. 11.18, let ρ be the distance between the representative point (x, y) of the unicycle and the origin of the Cartesian plane, γ the angle between vector $\boldsymbol{e}_p$ and the sagittal axis of the vehicle, and δ the angle between the same vector and the x axis. In formulae:

$$\begin{aligned}\rho &= \sqrt{x^2 + y^2}\\ \gamma &= \mathrm{Atan2}(y, x) - \theta + \pi\\ \delta &= \gamma + \theta.\end{aligned}$$

In these coordinates, the kinematic model of the unicycle is expressed as

$$\begin{aligned}\dot{\rho} &= -v\cos\gamma\\ \dot{\gamma} &= \frac{\sin\gamma}{\rho}\,v - \omega\\ \dot{\delta} &= \frac{\sin\gamma}{\rho}\,v.\end{aligned} \tag{11.80}$$

Note that the input vector field associated with v is singular for $\rho = 0$.

[10] La Salle theorem cannot be used because V is not positive definite at the origin.

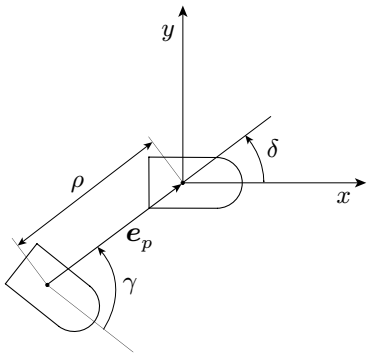

Fig. 11.18. Definition of polar coordinates for the unicycle

Define the feedback control as[11]

$$v = k_1 \rho \cos\gamma \tag{11.81}$$

$$\omega = k_2\,\gamma + k_1\,\frac{\sin\gamma\cos\gamma}{\gamma}\,(\gamma + k_3\,\delta), \tag{11.82}$$

with $k_1 > 0$, $k_2 > 0$. The kinematic model (11.80) under the action of the control law (11.81), (11.82) asymptotically converges to the desired configuration $[\,\rho \quad \gamma \quad \delta\,]^T = [\,0 \quad 0 \quad 0\,]^T$.

The proof of this result is based on the following Lyapunov candidate:

$$V = \frac{1}{2}\,\left(\rho^2 + \gamma^2 + k_3\,\delta^2\right),$$

whose time derivative along the closed-loop system trajectories

$$\dot{V} = -k_1 \cos^2\gamma\,\rho^2 - k_2\,\gamma^2$$

is negative semi-definite. As a consequence, V tends to a limit value and the system state is bounded. It can also be shown that $\ddot{V}$ is bounded, so that $\dot{V}$ is uniformly continuous. In view of Barbalat lemma, it can be inferred that $\dot{V}$ tends to zero, and likewise do ρ and γ. Further analysis of the closed-loop system leads to concluding that δ converges to zero as well.

Note that angles γ and δ are undefined for $x = y = 0$. They are, however, always well-defined during the motion of the unicycle, and asymptotically tend to the desired zero value.

[11] It is easy to verify that the expression (11.81) for v coincides with (11.78), the driving velocity prescribed by the Cartesian regulation scheme, except for the presence of ρ, whose effect is to modulate v according to the distance of the robot from the destination. As for the steering velocity, (11.82) differs from (11.79) for the presence of the second term, that contains the orientation error θ (through δ) in addition to the pointing error.

It should be noted that the control law (11.81), (11.82), once mapped back to the original coordinates, is discontinuous at the origin of the configuration space $\mathcal{C}$. As a matter of fact, it can be proven that any feedback law that can regulate the posture of the unicycle must be necessarily discontinuous with respect to the state and/or time-varying.[12]

Simulations

To illustrate the characteristics of the above regulation schemes, simulation results are now presented for two parking manoeuvres performed in feedback by a unicycle mobile robot.

Figure 11.19 shows the robot trajectories produced by the Cartesian regulator (11.78), (11.79), with $k = 1$ and $k_2 = 3$, for two different initial configurations. Note that the final orientation of the robot varies with the approach direction, and that the unicycle reaches the destination in forward motion, after inverting its motion at most once (like in the second manoeuvre). It is possible to prove that such behaviour is general with this controller.

The results of the application of the posture regulator (11.81), (11.82) starting from the same initial conditions are shown in Fig. 11.20. The gains have been set to $k_1 = 1$, $k_2 = 2.5$ and $k_3 = 3$. The trajectories obtained are quite similar to the previous ones, but as expected the orientation is driven to zero as well. As before, the final approach to the destination is always in forward motion, with at most one motion inversion in the transient phase.

11.7 Odometric Localization

The implementation of any feedback controller requires the availability of the robot configuration at each time instant. Unlike the case of manipulators, in which the joint encoders provide a direct measurement of the configuration, mobile robots are equipped with incremental encoders that measure the rotation of the wheels, but not directly the position and orientation of the vehicle with respect to a fixed world frame. It is therefore necessary to devise a *localization* procedure that estimates in real time the robot configuration.

Consider a unicycle moving under the action of velocity commands v and ω that are constant within each sampling interval. This assumption, which

[12] This result, which actually applies to all nonholonomic robots, derives from the application of a necessary condition (*Brockett theorem*) for the smooth stabilizability of control systems. In the particular case of a driftless system of the form (11.10), in which there are fewer inputs than states and the input vector fields are linearly independent, such a condition is violated and no control law that is continuous in the state $\boldsymbol{q}$ can asymptotically stabilize an equilibrium point. Brockett theorem does not apply to time-varying stabilizing controllers that may thus be continuous in $\boldsymbol{q}$.