10

Visual Servoing

Vision allows a robotic system to obtain geometrical and qualitative information on the surrounding environment to be used both for motion planning and control. In particular, control based on feedback of visual measurements is termed *visual servoing*. In the first part of this chapter, some basic algorithms for image processing, aimed at extracting numerical information referred to as *image feature parameters*, are presented. These parameters, relative to images of objects present in the scene observed by a camera, can be used to estimate the pose of the camera with respect to the objects and vice versa. To this end, analytic *pose estimation methods*, based on the measurement of a certain number of points or *correspondences* are presented. Also, numerical pose estimation methods, based on the integration of the linear mapping between the camera velocity in the operational space and the time derivative of the feature parameters in the *image plane*, are introduced. In cases in which multiple images of the same scene, taken from different viewpoints, are available, additional information can be obtained using *stereo vision* techniques and *epipolar geometry*. A fundamental operation is also *camera calibration*; to this end, a calibration method based on the measurement of a certain number of correspondences is presented. Then, the two main approaches to visual servoing are introduced, namely *position-based visual servoing* and *image-based visual servoing*, as well as a scheme, termed *hybrid visual servoing*, which combines the benefits of both approaches.

10.1 Vision for Control

Vision plays a key role in a robotic system, as it can be used to obtain geometrical and qualitative information on the environment where the robot operates, without physical interaction. Such information may be employed by the control system at different levels, for the sole task planning and also for feedback control.

As an example, consider the case of a robot manipulator, equipped with a camera, which has to grasp an object using a gripper. Through vision the robot may acquire information capable of identifying the relative pose of the object with respect to the gripper. This information allows the control system to plan a trajectory leading the manipulator in an appropriate grasping configuration, computed on the basis of the pose and of the shape of the object, from which the closure of the gripper can be commanded.

The planned trajectory can be executed using a simple motion controller. In this approach, termed *look-and-move*, visual measurements are used in open loop, making the system very sensitive to uncertainties due, for instance, to poor positioning accuracy of the manipulator or to the fact that the object may have moved while the gripper reaches the grasp position.

On the other hand, in *vision-based control* or *visual servoing*, the visual measurements are fed back to the control to compute an appropriate error vector defined between the current pose of the object and the pose of the manipulator's end-effector.

A key characteristic of visual servoing, compared to motion and force control, is the fact that the controlled variables are not directly measured by the sensor, but are obtained from the measured quantities through complex elaborations, based on algorithms of *image processing* and *computational vision*.

In Sect. 5.4.3 it was shown that a monochrome camera simply provides a two-dimensional matrix of values of light intensity. From this matrix, the so-called *image feature parameters* are to be extracted in real time. The geometric relationships between one or more two-dimensional views of a scene and the corresponding 3D space are the basis of techniques of *pose estimation* of objects in the manipulator workspace or of the end-effector with respect to the surrounding objects. In this regard, of fundamental importance is the operation of *camera calibration*, which is necessary for calculating the *intrinsic parameters*, relating the quantities measured in the image plane to those referred to the camera frame, and the *extrinsic parameters*, relating the latter to quantities defined in a frame attached to the manipulator.

The vision-based control schemes can be divided into two categories, namely, those that realize *visual servoing in operational space*, also termed *position-based visual servoing*, and those that realize *visual servoing in the image space*, also known as *image-based visual servoing*. The main difference lies in the fact that the schemes of the first category use visual measurements to reconstruct the relative pose of the object with respect to the robot, or vice versa, while the schemes of the second category are based on the comparison of the feature parameters of the image of the object between the current and the desired pose. There are also schemes combining characteristics common to both categories, that can be classified as *hybrid visual servoing*.

Another aspect to be considered for vision-based control is the type of camera (colour or monochrome, resolution, fixed or variable focal length, CCD or CMOS technology). In this chapter, only the case of monochrome cameras with fixed focal length will be considered.

Equally important is the choice of the number of cameras composing the visual system and their location; this issue is briefly discussed in the following.

10.1.1 Configuration of the Visual System

A visual system may consist of only one camera, or two or more cameras. If more cameras are used to observe the same object of a scene, it is possible to retrieve information about its depth by evaluating its distance with respect to the visual system. This situation is referred to as *3D vision* or *stereo vision*, where the term *stereo* derives from the Greek and means solid. The human capability of perceiving objects in three dimensions relies on the fact that the brain receives the same images from two eyes, observing the same scene from slightly different angles.

It is clear that 3D vision can be achieved even with one camera, provided that two images of the same object, taken from two different poses, are available. If only a single image is available, the depth can be estimated on the basis of certain geometrical characteristics of the object known in advance. This means that, in many applications, mono-camera systems are often preferred to multi-camera systems, because they are cheaper and easier to calibrate, although characterized by lower accuracy.

Another feature that distinguishes visual systems for robot manipulators is the placement of cameras. For mono-camera systems there are two options: the *fixed configuration*, often referred to as *eye-to-hand*, where the camera is mounted in a fixed location, and the *mobile configuration*, or *eye-in-hand*, with the camera attached to the robot. For multi-camera systems, in addition to the mentioned solutions, it is also possible to consider the *hybrid configuration*, consisting of one or more cameras in eye-to-hand configuration, and one or more cameras in eye-in-hand configuration.

In the eye-to-hand configuration, the visual system observes the objects to be manipulated by a fixed pose with respect to the base frame of the manipulator. The advantage is that the camera field of view does not change during the execution of the task, implying that the accuracy of such measurements is, in principle, constant. However, in certain applications, such as assembly, it is difficult to prevent that the manipulator, moving in the camera field of view, occludes, in part or in whole, the view of the objects.

In the eye-in-hand configuration, the camera is placed on the manipulator and can be mounted both before and after the wrist. In the first case, the camera can observe the end-effector by a favourable pose and without occlusions caused by the manipulator arm; in the latter case, the camera is attached to the end-effector and typically observes only the object. In both situations, the camera field of view changes significantly during the motion and this produces a high variability in the accuracy of measurements. However, when the end-effector is close to the object, the accuracy becomes almost constant and is usually higher than that achievable with eye-to-hand cameras, with the advantage that occlusions are virtually absent.

Finally, hybrid configuration combines the benefits of the other two configurations, namely, ensures a good accuracy throughout the workspace, while avoiding the problems of occlusions.

A separate category is represented by robotic heads, which are typically equipped with a stereo vision system consisting of two cameras mounted on motorized mechanisms that allow for yaw motion, or *pan*, and pitch motion, or *tilt*, hence the name of *pan-tilt* cameras.

In this chapter, only schemes based on a single eye-in-hand camera will be considered. The extension of the algorithms to the case of a eye-to-hand camera, or to the case of multiple cameras, requires only minor modifications.

10.2 Image Processing

Visual information, unlike the information provided by other types of sensors, is very rich and varied and thus requires complex and computational expensive transformations before it can be used for controlling a robotic system. The objective of these transformations is the extraction of numerical information from the image, which provides a synthetic and robust description of the objects of interest in the scene, through the so-called *image feature parameters*.

To this end, two basic operations are required. The first is so-called *segmentation*, which aims at obtaining a representation suitable for the identification of measurable features of the image. The subsequent operation, termed *interpretation* is concerned with the measurement of the feature parameters of the image.

The source information is contained in a framestore, namely the two-dimensional memory array representing the spatial sample of the image. On the set of pixels the so-called *image function* is defined which, in general, is a vector function whose components represent the values of one or more physical quantities related to the pixel, in a sampled and quantized form.

For example, in the case of color images, the image function defined on a pixel of coordinates (X_I, Y_I) has three components $I_r(X_I, Y_I)$, $I_g(X_I, Y_I)$ and $I_b(X_I, Y_I)$, corresponding to the light intensity in the wavelengths of red, green and blue. For a monocrome black-and-white image, the image function is scalar and coincides with the light intensity in shades of gray $I(X_I, Y_I)$, also referred to as *gray level*. In the following, for simplicity, only monochrome images will be considered.

The number of gray levels depends on the adopted grey-scale resolution. In all cases, the gray scale is bounded by two gray levels, black and white, corresponding to the minimum and maximum measurable light intensity respectively. Most current acquisition equipments adopt a scale consisting of 256 gray levels, that can be represented by a single byte of memory.

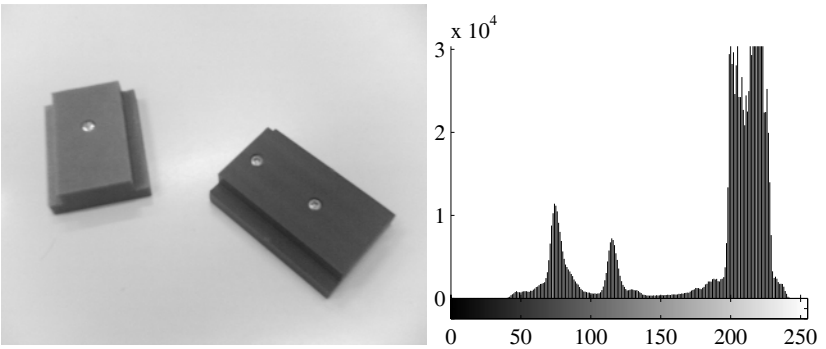

Fig. 10.1. Black-and-white image and corresponding gray-level histogram on the right

A representation of the framestore which is particularly useful for subsequent processing is the gray-level histogram, which provides the frequency of occurrence of each gray level in the image.

Where the gray levels are quantized from 0 to 255, the value $h(p)$ of the histogram at a particular gray level $p \in [0, 255]$ is the number of image pixels with gray level p. If this value is divided by the total number of pixels, the histogram is termed *normalized histogram*.

Figure 10.1 shows a black-and-white image and the corresponding gray-level histogram. Proceeding from left to right, three main peaks can be observed — from left to right — corresponding to the darkest object, the lightest object, and the background.

10.2.1 Image Segmentation

Segmentation consists of a grouping process, by which the image is divided into a certain number of groups, referred to as *segments*, so that the component of each group are similar with respect to one or more characteristics. Typically, distinct segments of the image correspond to distinct objects of the environment, or homogeneous object parts.

There are two complementary approaches to the problem of image segmentation: one is based on finding connected *regions* of the image, the other is concerned with detection of *boundaries*. The objective of region-based segmentation is that of grouping sets of pixels sharing common features into two-dimensional connected areas, with the implicit assumption that the resulting regions correspond to real-world surfaces or objects. On the other hand, boundary-based segmentation is aimed at identifying the pixels corresponding to object contours and isolating them from the rest of the image. The boundary of an object, once extracted, can be used to define the position and shape of the object itself.

The complementarity of the two approaches relies on the fact that a boundary can be achieved by isolating the contours of a region and, conversely, a region can be achieved simply by considering the set of pixels contained within a closed boundary.

The problem of segmentation is not trivial and there exist many solutions, some of which are sketched below. From the point of view of memory usage, boundary-based segmentation is more convenient, since boundaries contain a reduced number of pixels. However, from the computational load point of view, region-based segmentation is faster because it requires a reduced number of memory accesses.

Region-based segmentation

The central idea underlying region-based segmentation techniques is that of obtaining connected regions by continuous merging of initially small groups of adjacent pixels into larger ones.

Merging of two adjacent regions may happen only if the pixels belonging to these regions satisfy a common property, termed *uniformity predicate*. Often the uniformity predicate requires that the gray level of the pixels of the region belongs to a given interval.

In many applications of practical interest a thresholding approach is adopted and a light intensity scale composed of only two values (0 and 1) is considered. This operation is referred to as *binary segmentation* or image *binarization*, and corresponds to separating one or more objects present in the image from the background by comparing the gray level of each pixel with a threshold l. For light objects against a dark background, all the pixels whose gray level is greater than the threshold are considered to belong to a set S_o, corresponding to the objects, while all the other pixels are considered to belong to a set S_b corresponding to the background. It is obvious that this operation can be reversed for dark objects against a light background. When only an object is present in the image, the segmentation ends with the detection of sets S_o and S_b, representing two regions; in the presence of multiple objects, a further elaboration is required to separate the connected regions corresponding to the single objects. The image obtained assigning a light intensity equal to 0 to all the pixels of set S_o, and a light intensity equal to 1 to all the pixels of set S_b, or vice versa, is termed *binary image*.

A crucial factor for the success of binary segmentation is the choice of the threshold. A widely adopted method for selecting the threshold is based on the gray-level histogram, under the assumption that it contains clearly distinguishable minimum and maximum values, corresponding to the gray levels of the objects and of the background; the peaks of the histogram are also termed *modes*. For dark objects against a light background, the background corresponds to the mode which is located further to the right — as, for example, in the case of Fig. 10.1 — and the threshold can be chosen at the closest minimum to the left. For light objets against a dark background, the

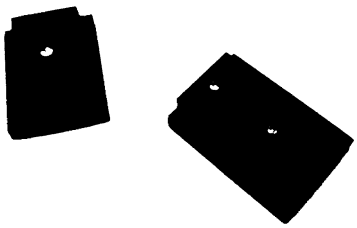

Fig. 10.2. Binary image corresponding to image of Fig. 10.1

background corresponds to the mode which is located further to the left and the threshold should be selected accordingly. With reference to Fig. 10.1, the threshold can be set to $l = 152$. The corresponding binary image is reported in Fig. 10.2.

In practice, the gray-scale histogram is noisy and the modes are difficult to identify. Often, there is no clear separation between the gray levels of the objects and those of the background. To this end, various techniques have been developed to increase the robustness of binary segmentation, which require appropriate filtering of the image before binarization and the adoption of algorithms for automatic selection of the threshold.

Boundary-based segmentation

Boundary-based segmentation techniques usually obtain a boundary by grouping many single local edges, corresponding to local discontinuities of image gray level. In other words, local edges are sets of pixels where the light intensity changes abruptly.

The algorithms for boundary detection first derive an intermediate image based on local edges from the original gray-scale image, then they construct short-curve segments by edge linking, and finally obtain the boundaries by joining these curve segments through geometric primitives often known in advance.

Boundary-based segmentation algorithms vary in the amount of a priori knowledge they incorporate in associating or linking the edges and their effectiveness clearly depends on the quality of the intermediate image based on local edges. The more reliable the local edges in terms of their position, orientation and 'authenticity', the easier the task of the boundary detection algorithm.

Notice that edge detection is essentially a filtering process and can often be implemented via hardware, whereas boundary detection is a higher level task usually requiring more sophisticated software. Therefore, the current trend

is that of using the most effective edge detector to simplify the boundary detection process. In the case of simple and well-defined shapes, boundary detection becomes straightforward and segmentation reduces to the sole edge detection.

Several edge detection techniques exist. Most of them require the calculation of the gradient or of the Laplacian of function $I(X_I, Y_I)$.

Since a local edge is defined as a transition between two regions of significantly different gray levels, it is obvious that the spatial gradient of function $I(X_I, Y_I)$, which measures the rate of change of the gray level, will have large magnitude close to these transitional boundary areas. Therefore, edge detection can be performed by grouping the pixels where the magnitude of the gradient is greater than a threshold. Moreover, the direction of the gradient vector will be the direction of maximum variation of the gray level.

Again, the choice of the value of the threshold is extremely important; in the presence of noise, the threshold is the result of a trade-off between the possibility of losing valid edges and that of detecting false edges.

For gradient computation, it suffices to evaluate the directional derivatives of function $I(X_I, Y_I)$ along two orthogonal directions. Since this function is defined on a discrete set of pixels, the derivatives are computed in an approximate way. The essential differences between gradient-based edge detection techniques are the directions used for the computation of the derivatives and the manner in which they approximate these derivatives and compute the gradient magnitude.

The most common operator for the computation of the gradient is that approximating the derivative along directions X_I and Y_I with the first differences:

$$\begin{aligned} \Delta_1 &= I(X_I + 1, Y_I) - I(X_I, Y_I) \\ \Delta_2 &= I(X_I, Y_I + 1) - I(X_I, Y_I). \end{aligned}$$

Other operators, less sensitive to noise effects are, for example, the *Roberts operator*, based on the first differences computed along the diagonals of a (2×2) square of pixels:

$$\begin{aligned} \Delta_1 &= I(X_I + 1, Y_I + 1) - I(X_I, Y_I) \\ \Delta_2 &= I(X_I, Y_I + 1) - I(X_I + 1, Y_I), \end{aligned}$$

and the *Sobel operator*, defined on a (3×3) square of pixels:

$$\begin{aligned} \Delta_1 = {} & (I(X_I + 1, Y_I - 1) + 2I(X_I + 1, Y_I) + I(X_I + 1, Y_I + 1)) \\ & -(I(X_I - 1, Y_I - 1) + 2I(X_I - 1, Y_I) + I(X_I - 1, Y_I + 1)) \\ \Delta_2 = {} & (I(X_I - 1, Y_I + 1) + 2I(X_I, Y_I + 1) + I(X_I + 1, Y_I + 1)) \\ & -(I(X_I - 1, Y_I - 1) + 2I(X_I, Y_I - 1) + I(X_I + 1, Y_I - 1)). \end{aligned}$$

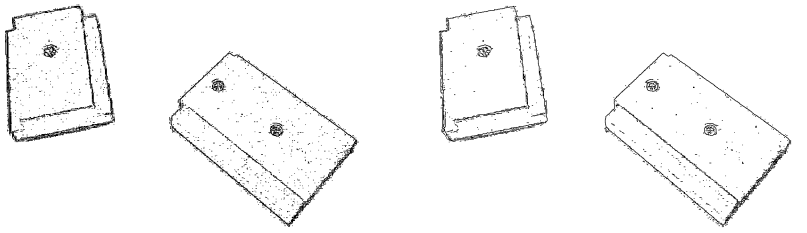

Fig. 10.3. Contours of image of Fig. 10.1 obtained using Roberts (*left*) and Sobel (*right*) operators

Then, the approximated magnitude, or norm, of gradient $G(X_I, Y_I)$ can be evaluated using one of the following two expressions:

$$G(X_I, Y_I) = \sqrt{\Delta_1^2 + \Delta_2^2}$$
$$G(X_I, Y_I) = |\Delta_1| + |\Delta_2|,$$

and direction $\theta(X_I, Y_I)$ with the relationship

$$\theta(X_I, Y_I) = \text{Atan2}\,(\Delta_2, \Delta_1)\,.$$

Figure 10.3 shows the images obtained from that of Fig. 10.1 by applying the gradient operators of Sobel and Roberts and binarization; the thresholds have been set to $l = 0.02$ and $l = 0.0146$, respectively.

An alternative edge detection method is based on the *Laplacian operator*, which requires the computation of the second derivatives of function $I(X_I, Y_I)$ along two orthogonal directions. Also in this case, suitable operators are used to discretize the computation of derivatives. One of the most common approximations is the following:

$$L(X_I, Y_I) = I(X_I, Y_I) - \frac{1}{4}\,(I(X_I, Y_I + 1) + I(X_I, Y_I - 1) + I(X_I + 1, Y_I) + I(X_I - 1, Y_I))\,.$$

In this case, the pixels of the contour are those where the Laplacian is lower than a threshold. The reason is that the Laplacian is null at the points of maximum magnitude of the gradient. The Laplacian, unlike the gradient, does not provide directional information; moreover, being based on the calculation of second derivatives, it is more sensitive to noise than the gradient.

10.2.2 Image Interpretation

Image interpretation is the process of calculating the image feature parameters from the segments, whether they are represented in terms of boundaries or in terms of regions.

The feature parameters used in visual servoing applications sometimes require the computation of the so-called *moments*. These parameters are defined on a region $\mathcal{R}$ of the image and can be used to characterize the position, orientation and shape of the two-dimensional object corresponding to the region itself.

The general definition of moment $m_{i,j}$ of a region $\mathcal{R}$ of a framestore, with $i, j = 0, 1, 2, \ldots$, is the following:

$$m_{i,j} = \sum_{X_I, Y_I \in \mathcal{R}} I(X_I, Y_I) X_I^i Y_I^j .$$

In the case of binary images, by assuming the light intensity equal to one for all the points of region $\mathcal{R}$, and equal to zero for all the points not belonging to $\mathcal{R}$, the following simplified definition of moment is obtained:

$$m_{i,j} = \sum_{X_I, Y_I \in \mathcal{R}} X_I^i Y_I^j . \tag{10.1}$$

In view of this definition, moment $m_{0,0}$ coincides with the area of the region, computed in terms of the total number of pixels of region $\mathcal{R}$.

The quantities

$$\bar{x} = \frac{m_{1,0}}{m_{0,0}} \qquad \bar{y} = \frac{m_{0,1}}{m_{0,0}}$$

define the coordinates of the so-called *centroid* of the region. These coordinates can be used to detect uniquely the position of region $\mathcal{R}$ on the image plane.

Using an analogy from mechanics, region $\mathcal{R}$ can be seen as a two-dimensional rigid body of density equal to light intensity. Hence, moment $m_{0,0}$ corresponds to the mass of the body and the centroid corresponds to the centre of mass.

The value of moment $m_{i,j}$ in (10.1) depends on the position of region $\mathcal{R}$ in the image plane. Therefore, the so-called *central moments* are often considered, defined as

$$\mu_{i,j} = \sum_{X_I, Y_I \in \mathcal{R}} (X_I - \bar{x})^i (Y_I - \bar{y})^j ,$$

which are invariant with respect to translation.

According to the mechanical analogy, it is easy to recognize that the central moments of second order $\mu_{2,0}$ and $\mu_{0,2}$ have the meaning of inertia moments with respect to axes X_I and Y_I respectively, while $\mu_{1,1}$ is an inertia product, and the matrix

$$\mathcal{I} = \begin{bmatrix} \mu_{2,0} & \mu_{1,1} \\ \mu_{1,1} & \mu_{0,2} \end{bmatrix} ,$$

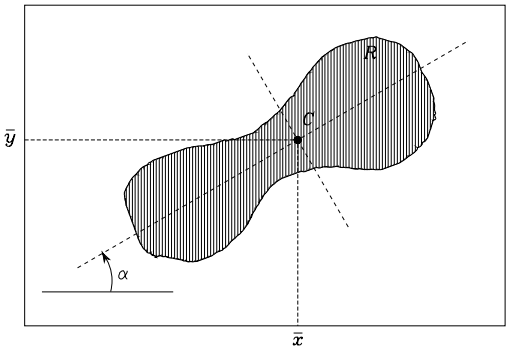

Fig. 10.4. Region of a binary image and some feature parameters

has the meaning of inertia tensor relative to the centre of mass. The eigenvalues of matrix $\mathcal{I}$ define the principal moments of inertia, termed *principal moments* of the region and the corresponding eigenvectors define the principal axes of inertia, termed *principal axes* of the region.

If region $\mathcal{R}$ is asymmetric, the principal moments of $\mathcal{I}$ are different and it is possible to characterize the orientation of $\mathcal{R}$ in terms of the angle α between the principal axis corresponding to the maximum moment and axis X. This angle can be computed with the equation (see Problem 10.1)

$$\alpha = \frac{1}{2}\tan^{-1}\left(\frac{2\mu_{1,1}}{\mu_{2,0} - \mu_{0,2}}\right). \tag{10.2}$$

As an example, in Fig. 10.4, the region of a binary image is shown; centroid C, the principal axes, and the angle α are evidenced.

Notice that the moments and the corresponding parameters can also be computed from the boundaries of the objects; moreover, these quantities are especially useful to characterize objects of generic form. Often, however, the objects present in the scene, especially those manufactured, have geometric characteristics which are useful to take into account for image interpretation.

For example, many objects have edges that, in the image plane, correspond to the intersection of linear parts of contour or to contour points of high curvature. The coordinates of these points on the image plane can be detected using algorithms robust against the noise, and therefore can be used as feature parameters of the image. They are usually termed *feature points*.

In other cases, it is possible to identify true geometric primitives such as lines or line segments, which are projections of linear edges or solids of revolution (cones, cylinders), or ellipses, obtained as projections of circles or spheres. These primitives can be characterized on the image plane in terms of a minimum set of parameters. For example, a line segment can be characterized by the coordinates of its endpoints, or alternatively, by the coordinates of its midpoint (centroid), its length (moment $m_{0,0}$) and its orientation (angle α); in both cases, the characterization of the line segment requires four parameters.

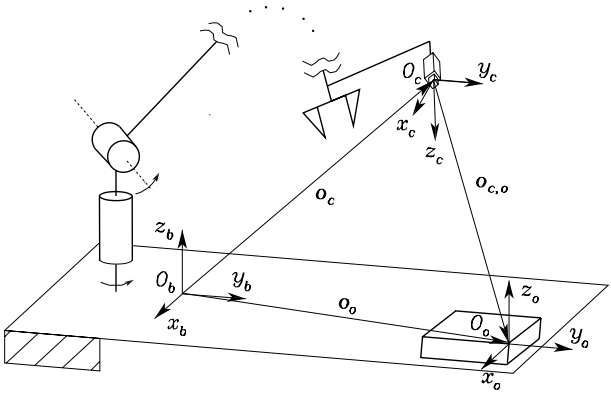

Fig. 10.5. Reference frames for an eye-in-hand camera

10.3 Pose Estimation

Visual servoing is based on the mapping between the feature parameters of an object measured in the image plane of the camera and the operational space variables defining the relative pose of the object with respect to the camera or, equivalently, of the camera with respect to the object. Often, it is sufficient to derive a differential mapping in terms of velocity. As for the computation of the inverse kinematics of a manipulator, the differential problem is easier to solve because the velocity mapping is linear; moreover, the solution to the differential problem can be used to compute the pose by using numerical integration algorithms.

The set of feature parameters of an image defines a $(k \times 1)$ vector $\boldsymbol{s}$, termed *feature vector*. In the following, to simplify notation, normalized coordinates (X, Y) defined in (5.44) will be used in place of pixel coordinates (X_I, Y_I) to define the feature vector. Since only pixel coordinates can be directly measured, the normalized coordinates should be computed from pixel coordinates using the inverse of mapping (5.45), provided that the intrinsic parameters of the camera are known.

The feature vector $\boldsymbol{s}$ of a point is defined as

$$\boldsymbol{s} = \begin{bmatrix} X \\ Y \end{bmatrix}, \tag{10.3}$$

while

$$\widetilde{\boldsymbol{s}} = \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

denotes its representation in homogeneous coordinates.

10.3.1 Analytic Solution

Consider a camera, for example an eye-in-hand camera, and a reference frame O_c–$x_c y_c z_c$ attached to the camera; consider also a reference frame O_o–$x_o y_o z_o$ attached to the object, supposed to be rigid, and let $\boldsymbol{T}_o^c$ be the homogeneous transformation matrix corresponding to the relative pose of the object with respect to the camera, defined as

$$\boldsymbol{T}_o^c = \begin{bmatrix} \boldsymbol{R}_o^c & \boldsymbol{o}_{c,o}^c \\ \boldsymbol{0}^T & 1 \end{bmatrix}, \tag{10.4}$$

with $\boldsymbol{o}_{c,o}^c = \boldsymbol{o}_o^c - \boldsymbol{o}_c^c$, where $\boldsymbol{o}_c^c$ is the position vector of the origin of the camera frame with respect to the base frame, expressed in camera frame, $\boldsymbol{o}_o^c$ is the position vector of the origin of the object frame with respect to the base frame, expressed in the camera frame, and $\boldsymbol{R}_o^c$ is the rotation matrix of the object frame with respect to the camera frame (Fig. 10.5).

The problem to solve is that of computing the elements of matrix $\boldsymbol{T}_o^c$ from the measurements of object feature parameters in the camera image plane. To this end, consider n points of the object and let $\boldsymbol{r}_{o,i}^o = \boldsymbol{p}_i^o - \boldsymbol{o}_o^o$, $i = 1, \ldots, n$, denote the corresponding position vectors with respect to the object frame. These quantities are assumed to be known, for example, from a CAD model of the object. The projections of these points on the image plane have coordinates

$$\boldsymbol{s}_i = \begin{bmatrix} X_i \\ Y_i \end{bmatrix},$$

and define the feature vector

$$\boldsymbol{s} = \begin{bmatrix} \boldsymbol{s}_1 \\ \vdots \\ \boldsymbol{s}_n \end{bmatrix}. \tag{10.5}$$

The homogeneous coordinates of the points of the object with respect to the camera frame can be expressed as

$$\widetilde{\boldsymbol{r}}_{o,i}^c = \boldsymbol{T}_o^c \widetilde{\boldsymbol{r}}_{o,i}^o.$$

Therefore, using (5.44), the homogeneous coordinates of the projections of these points on the image plane are given by

$$\lambda_i \widetilde{\boldsymbol{s}}_i = \boldsymbol{\Pi} \boldsymbol{T}_o^c \widetilde{\boldsymbol{r}}_{o,i}^o, \tag{10.6}$$

with $\lambda_i > 0$.

Assume that n *correspondences* are available, namely n equations of the form (10.6) for n points of the object, whose coordinates are known both in the object frame and in the image plane. These correspondences define a system of equations to be solved for the unknown elements of matrix $\boldsymbol{T}_o^c$.

Computing the solution is a difficult task because, depending on the type and on the number of correspondences, multiple solutions may exist. This problem, in photogrammetry, is known as PnP (Perspective-n-Point) problem. In particular, it can be shown that:

- P3P problem has four solutions, in the case of three non-collinear points.
- P4P and P5P problems each have at least two solutions, in the case of non-coplanar points, while the solution is unique in the case of at least four coplanar points and no triplets of collinear points.
- PnP problem, with $n \geq 6$ non-coplanar points, has only one solution.

The analytic solution to PnP problem is rather laborious. However, the derivation becomes simpler in some particular cases as, for example, in the case of coplanar points.

Without loss of generality, assume that the plane containing the points of the object coincides with one of the three coordinate planes of the object frame, for instance, with the plane of equation $z_o = 0$; this implies that all the points of the plane have the third coordinate equal to zero. Multiplying both sides of (10.6) by the skew-symmetric matrix $\boldsymbol{S}(\widetilde{\boldsymbol{s}}_i)$, the product on the left-hand side is zero, leading to the homogeneous equation

$$\boldsymbol{S}(\widetilde{\boldsymbol{s}}_i)\boldsymbol{H}\,[\,r_{x,i} \quad r_{y,i} \quad 1\,]^T = \boldsymbol{0}, \tag{10.7}$$

where $r_{x,i}$ and $r_{y,i}$ are the two non-null components of vector $\boldsymbol{r}_{o,i}^o$ and $\boldsymbol{H}$ is the (3×3) matrix

$$\boldsymbol{H} = [\,\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{o}_{c,o}^c\,], \tag{10.8}$$

$\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ being the first and the second column of rotation matrix $\boldsymbol{R}_o^c$, respectively.

Vector equation (10.7), defined on homogeneous coordinates of points belonging to two planes, is know as *planar homography*; this denomination is used also for matrix $\boldsymbol{H}$.

Notice that Eq. (10.7) is linear with respect to $\boldsymbol{H}$ and, therefore, can be rewritten in the form

$$\boldsymbol{A}_i(\boldsymbol{s}_i)\boldsymbol{h} = \boldsymbol{0},$$

where $\boldsymbol{h}$ is the (9×1) column vector obtained by staking the columns of matrix $\boldsymbol{H}$, while $\boldsymbol{A}_i$ is the (3×9) matrix

$$\boldsymbol{A}_i(\boldsymbol{s}_i) = [\,r_{x,i}\boldsymbol{S}(\widetilde{\boldsymbol{s}}_i) \quad r_{y,i}\boldsymbol{S}(\widetilde{\boldsymbol{s}}_i) \quad \boldsymbol{S}(\widetilde{\boldsymbol{s}}_i)\,]\,. \tag{10.9}$$

Since the rank of $\boldsymbol{S}(\cdot)$ is at most 2, then the rank of matrix $\boldsymbol{A}_i$ is also at most 2; therefore, to compute $\boldsymbol{h}$ (up to a scale factor), it is necessary to consider at least 4 equations of the form (10.9) written for 4 points of the plane, leading to the system of 12 equations with 9 unknowns

$$\begin{bmatrix} \boldsymbol{A}_1(\boldsymbol{s}_1) \\ \boldsymbol{A}_2(\boldsymbol{s}_2) \\ \boldsymbol{A}_3(\boldsymbol{s}_3) \\ \boldsymbol{A}_4(\boldsymbol{s}_4) \end{bmatrix} \boldsymbol{h} = \boldsymbol{A}(\boldsymbol{s})\boldsymbol{h} = \boldsymbol{0}, \tag{10.10}$$

with $\boldsymbol{s}$ defined in (10.5).

It can be shown that, considering a set of four points with no triplets of collinear points, matrix $\boldsymbol{A}$ has rank 8 and the system of equations in (10.10) admits a non-null solution $\zeta\boldsymbol{h}$, defined up to a scaling factor ζ (see Problem 10.2). As a result, matrix $\zeta\boldsymbol{H}$ can be computed up to a scaling factor ζ. The presented derivation is general and can be applied to any kind of planar homography defined by an equation of the form (10.7).

In view of (10.8), it is

$$\begin{aligned}\boldsymbol{r}_1 &= \zeta\boldsymbol{h}_1\\ \boldsymbol{r}_2 &= \zeta\boldsymbol{h}_2\\ \boldsymbol{o}_{c,o}^c &= \zeta\boldsymbol{h}_3\end{aligned}$$

where $\boldsymbol{h}_i$ denotes the i-th column of matrix $\boldsymbol{H}$. The absolute value of constant ζ can be computed by imposing the unit norm constraint to vectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$:

$$|\zeta| = \frac{1}{\|\boldsymbol{h}_1\|} = \frac{1}{\|\boldsymbol{h}_2\|},$$

while the sign of ζ can be determined by choosing the solution corresponding to the object in front of the camera. Finally, the third column $\boldsymbol{r}_3$ of matrix $\boldsymbol{R}_o^c$ can be computed as

$$\boldsymbol{r}_3 = \boldsymbol{r}_1 \times \boldsymbol{r}_2.$$

Notice that, because of the noise affecting the measurements of the coordinates in the image plane, the results of this derivation are affected by errors that can be reduced by considering a number $n > 4$ of correspondences and computing the solution $\zeta\boldsymbol{h}$ to the $3n$ equations in (10.10), up to a scaling factor ζ, using least-squares techniques. This, however, does not guarantee that the resulting matrix $\boldsymbol{Q} = [\,\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{r}_3\,]$ is a rotation matrix.

A possible solution overcoming this problem consists of computing the rotation matrix 'closest' to $\boldsymbol{Q}$ with respect to a given norm such as the matrix which minimizes the Frobenius norm[1]

$$\|\boldsymbol{R}_o^c - \boldsymbol{Q}\|_F = \left(\mathrm{Tr}\left((\boldsymbol{R}_o^c - \boldsymbol{Q})^T(\boldsymbol{R}_o^c - \boldsymbol{Q})\right)\right)^{1/2}, \tag{10.11}$$

with the constraint that $\boldsymbol{R}_o^c$ is a rotation matrix. The problem of minimizing norm (10.11) is equivalent to that of maximizing the trace of matrix $\boldsymbol{R}_o^{cT}\boldsymbol{Q}$. It can be shown that the solution to this problem is

$$\boldsymbol{R}_o^c = \boldsymbol{U}\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & \sigma\end{bmatrix}\boldsymbol{V}^T \tag{10.12}$$

where $\boldsymbol{U}$ and $\boldsymbol{V}^T$ are, respectively, the left and right orthogonal matrices of the singular value decomposition of $\boldsymbol{Q} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$. The choice $\sigma = \det(\boldsymbol{U}\boldsymbol{V}^T)$ ensures that the determinant of $\boldsymbol{R}_o^c$ is equal to one (see Problem 10.3).

[1] The Frobenius norm is defined in Sect. A.4.

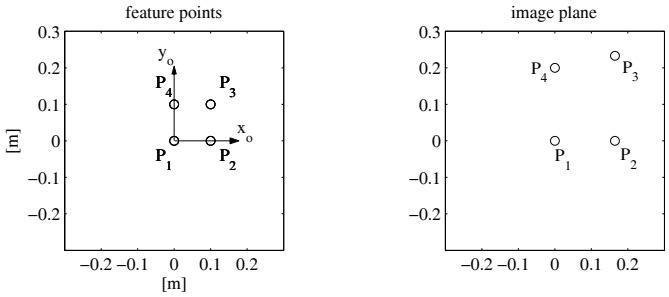

Fig. 10.6. Planar object; *left*: object frame and feature points; *right*: feature points projections on the normalized image plane of a camera in the pose of Example 10.1

Example 10.1

Consider a planar object characterized by four feature points shown in Fig. 10.6, where the object frame is represented. The feature points P_1, P_2, P_3, P_4 are the vertices of a square with side length $l = 0.1$ m. In Fig. 10.6, the images of the projections of the four points of the object on the normalized image plane of the camera are shown as well, under the assumption that the relative pose of the object frame with respect to the camera frame is characterized by rotation matrix

$$\boldsymbol{R}_o^c = \boldsymbol{R}_z(0)\boldsymbol{R}_y(\pi/4)\boldsymbol{R}_x(0) = \begin{bmatrix} 0.7071 & 0 & 0.7071 \\ 0 & 1 & 0 \\ -0.7071 & 0 & 0.7071 \end{bmatrix}$$

and position vector $\boldsymbol{o}_{c,o}^c = [\,0 \quad 0 \quad 0.5\,]^T$ m. The normalized coordinates of the four points of the object can be computed from the position vectors in the object frame $\boldsymbol{r}_{o,1}^o = [\,0 \quad 0 \quad 0\,]^T$ m, $\boldsymbol{r}_{o,2}^o = [\,0.1 \quad 0 \quad 0\,]^T$ m, $\boldsymbol{r}_{o,3}^o = [\,0.1 \quad 0.1 \quad 0\,]^T$ m, $\boldsymbol{r}_{o,4}^o = [\,0 \quad 0.1 \quad 0\,]^T$ m, using (10.6), which gives

$$\boldsymbol{s}_1 = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \quad \boldsymbol{s}_2 = \begin{bmatrix} 0.1647 \\ 0 \end{bmatrix} \quad \boldsymbol{s}_3 = \begin{bmatrix} 0.1647 \\ 0.2329 \end{bmatrix} \quad \boldsymbol{s}_4 = \begin{bmatrix} 0 \\ 0.2 \end{bmatrix}.$$

To solve the inverse problem, namely that of computing matrix $\boldsymbol{T}_o^c$ from the coordinates of the four points both in the image plane and in the object frame, it is necessary to build matrix $\boldsymbol{A}(\boldsymbol{s})$ from four matrices $\boldsymbol{A}_i(\boldsymbol{s}_i)$ defined in (10.9). It is easy to verify that matrix $\boldsymbol{A}(\boldsymbol{s})$ has rank 8; moreover, a non-null solution to the system of equations in (10.10) can be computed using the singular value decomposition

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$$

and coincides with the last column of matrix $\boldsymbol{V}$, namely, with the right eigenvector corresponding to the null singular value of $\boldsymbol{A}$. From this computation, matrix

$$\zeta\boldsymbol{H} = \begin{bmatrix} -0.4714 & 0 & 0 \\ 0 & -0.6667 & 0 \\ 0.4714 & 0 & -0.3333 \end{bmatrix}$$

can be obtained. Normalization of the first column yields $|\zeta| = 1.5$. It is possible to verify that, with the choice $\zeta = -1.5$, the exact solution for $\boldsymbol{o}_{c,o}^{c}$ is obtained; moreover, matrix $\boldsymbol{Q}$ coincides numerically with the sought rotation matrix $\boldsymbol{R}_{o}^{c}$, without using any kind of approximate solution. This result was expected, due to the absence of measurement noise affecting image plane coordinates $\boldsymbol{s}_i$.

The above derivation is a particular case of a method, termed *direct linear transformation*, which is aimed at computing the elements of matrix $\boldsymbol{T}_{o}^{c}$ by solving a system of linear equations obtained using n correspondences relative to points in generic configuration. In detail, from equalities

$$\boldsymbol{S}(\widetilde{\boldsymbol{s}}_i) \, [\, \boldsymbol{R}_{o}^{c} \quad \boldsymbol{o}_{c,o}^{c} \,] \, \widetilde{\boldsymbol{r}}_{o,i}^{o} = \boldsymbol{0}, \tag{10.13}$$

which coincide with (10.7) in the case that points $\boldsymbol{r}_{o,i}^{o}$ belong to plane $z_0 = 0$, two independent linear equations with 12 unknowns are obtained, taking into account that matrix $\boldsymbol{S}(\cdot)$ is, at most, of rank 2. Therefore, n correspondences produce $2n$ equations.

It can be shown that, considering a set of 6 points not all coplanar, the matrix of coefficients of the corresponding system of 12 equations with 12 unknowns has rank 11; therefore, the solution is defined up to a scaling factor. Once this solution has been computed, the elements of rotation matrix $\boldsymbol{R}_{o}^{c}$ and of vector $\boldsymbol{o}_{c,o}^{c}$ can be obtained using a derivation similar to that presented above. Notice that, in practical applications, due to the presence of noise, the system of equations has rank 12 and admits only the null solution. In this case, it is necessary to consider a number $n > 6$ of correspondences and to compute the solution of the resulting system of equations, defined up to a scaling factor, using least-squares techniques.

In conclusion, the presented method permits the computation of matrix $\boldsymbol{T}_{o}^{c}$, characterizing the relative pose of the object frame with respect to the camera frame, from the projections of n points of the object on the camera image plane. To this end, it is necessary to know the position of these points with respect to the object frame, and thus the object geometry, besides the camera *intrinsic parameters*. The latter are required for computing normalized coordinates $\boldsymbol{s}_i$ from pixel coordinates.

Notice that, if it is required to compute the object pose with respect to the base frame (as usually happens in the case of eye-to-hand cameras) or to the end-effector frame (as usually happens in the case of eye-in-hand cameras), then it is also necessary to know the camera *extrinsic parameters*. In fact, in the first case, it is

$$\boldsymbol{T}_{o}^{b} = \boldsymbol{T}_{c}^{b} \boldsymbol{T}_{o}^{c}, \tag{10.14}$$

where the elements of matrix $\boldsymbol{T}_{c}^{b}$ represent the extrinsic parameters of an eye-to-hand camera; on the other hand, in the case of eye-in-hand camera, it is

$$\boldsymbol{T}_{o}^{e} = \boldsymbol{T}_{c}^{e} \boldsymbol{T}_{o}^{c}, \tag{10.15}$$

where the extrinsic parameters matrix $\boldsymbol{T}_c^e$ characterize the pose of the camera with respect to the end-effector frame.

10.3.2 Interaction Matrix

If the object is in motion with respect to the camera, the feature vector $\boldsymbol{s}$ is, in general, time-varying. Therefore, it is possible to define a $(k \times 1)$ velocity vector in the image plane $\dot{\boldsymbol{s}}$.

The motion of the object with respect to the camera is characterized by the relative velocity

$$\boldsymbol{v}_{c,o}^c = \begin{bmatrix} \dot{\boldsymbol{o}}_{c,o}^c \\ \boldsymbol{R}_c^T(\boldsymbol{\omega}_o - \boldsymbol{\omega}_c) \end{bmatrix}, \tag{10.16}$$

where $\dot{\boldsymbol{o}}_{c,o}^c$ is the time derivative of vector $\boldsymbol{o}_{c,o}^c = \boldsymbol{R}_c^T(\boldsymbol{o}_o - \boldsymbol{o}_c)$, representing the relative position of the origin of the object frame with respect to the origin of the camera frame, while $\boldsymbol{\omega}_o$ and $\boldsymbol{\omega}_c$ are the angular velocities of the object frame and camera frame, respectively.

The equation relating $\dot{\boldsymbol{s}}$ to $\boldsymbol{v}_{c,o}^c$ is

$$\dot{\boldsymbol{s}} = \boldsymbol{J}_s(\boldsymbol{s}, \boldsymbol{T}_o^c)\boldsymbol{v}_{c,o}^c, \tag{10.17}$$

where $\boldsymbol{J}_s$ is a $(k \times 6)$ matrix termed *image Jacobian*. This equation is linear but $\boldsymbol{J}_s$ depends, in general, on the current value of the feature vector $\boldsymbol{s}$ and on the relative pose of the object with respect to the camera $\boldsymbol{T}_o^c$.

It is useful to consider also the mapping between the image plane velocity $\dot{\boldsymbol{s}}$, the absolute velocity of the camera frame

$$\boldsymbol{v}_c^c = \begin{bmatrix} \boldsymbol{R}_c^T \dot{\boldsymbol{o}}_c \\ \boldsymbol{R}_c^T \boldsymbol{\omega}_c, \end{bmatrix}$$

and the absolute velocity of the object frame

$$\boldsymbol{v}_o^c = \begin{bmatrix} \boldsymbol{R}_c^T \dot{\boldsymbol{o}}_o \\ \boldsymbol{R}_c^T \boldsymbol{\omega}_o \end{bmatrix}.$$

To this end, vector $\dot{\boldsymbol{o}}_{c,o}^c$ can be expressed as

$$\dot{\boldsymbol{o}}_{c,o}^c = \boldsymbol{R}_c^T(\dot{\boldsymbol{o}}_o - \dot{\boldsymbol{o}}_c) + \boldsymbol{S}(\boldsymbol{o}_{c,o}^c)\boldsymbol{R}_c^T\boldsymbol{\omega}_c,$$

which allows equality (10.16) to be rewritten in the compact form

$$\boldsymbol{v}_{c,o}^c = \boldsymbol{v}_o^c + \boldsymbol{\Gamma}(\boldsymbol{o}_{c,o}^c)\boldsymbol{v}_c^c, \tag{10.18}$$

with

$$\boldsymbol{\Gamma}(\cdot) = \begin{bmatrix} -\boldsymbol{I} & \boldsymbol{S}(\cdot) \\ \boldsymbol{O} & -\boldsymbol{I} \end{bmatrix}.$$

Therefore, Eq. (10.17) can be rewritten in the form

$$\dot{\boldsymbol{s}} = \boldsymbol{J}_s \boldsymbol{v}_o^c + \boldsymbol{L}_s \boldsymbol{v}_c^c, \tag{10.19}$$

where the $(k \times 6)$ matrix

$$\boldsymbol{L}_s = \boldsymbol{J}_s(\boldsymbol{s}, \boldsymbol{T}_o^c)\boldsymbol{\Gamma}(\boldsymbol{o}_{c,o}^c) \tag{10.20}$$

is termed *interaction matrix*. This matrix, in view of (10.19), defines the linear mapping between the absolute velocity of the camera $\boldsymbol{v}_c^c$ and the corresponding image plane velocity $\dot{\boldsymbol{s}}$, in the case that the object is fixed with respect to the base frame ($\boldsymbol{v}_o^c = \boldsymbol{0}$).

The analytic expression of the interaction matrix is, in general, simpler than that of the image Jacobian. The latter can be computed from the interaction matrix using the equation

$$\boldsymbol{J}_s(\boldsymbol{s}, \boldsymbol{T}_o^c) = \boldsymbol{L}_s \boldsymbol{\Gamma}(-\boldsymbol{o}_{c,o}^c), \tag{10.21}$$

obtained from (10.20), with $\boldsymbol{\Gamma}^{-1}(\boldsymbol{o}_{c,o}^c) = \boldsymbol{\Gamma}(-\boldsymbol{o}_{c,o}^c)$. In the following, examples of computation of interaction matrix and image Jacobian for some of the most common cases in applications are provided.

Interaction matrix of a point

Consider a point P of the object characterized, with respect to the camera frame, by the vector of coordinates

$$\boldsymbol{r}_c^c = \boldsymbol{R}_c^T(\boldsymbol{p} - \boldsymbol{o}_c), \tag{10.22}$$

where $\boldsymbol{p}$ is the position of point P with respect to the base frame. Choose vector $\boldsymbol{s}$ of normalized coordinates (10.3) as the feature vector of the point. In view of (5.44), the following expression holds:

$$\boldsymbol{s} = \boldsymbol{s}(\boldsymbol{r}_c^c), \tag{10.23}$$

with

$$\boldsymbol{s}(\boldsymbol{r}_c^c) = \frac{1}{z_c}\begin{bmatrix} x_c \\ y_c \end{bmatrix} = \begin{bmatrix} X \\ Y \end{bmatrix}, \tag{10.24}$$

and $\boldsymbol{r}_c^c = [\, x_c \quad y_c \quad z_c \,]^T$. Computing the time derivative of (10.23) and using (10.24) yields

$$\dot{\boldsymbol{s}} = \frac{\partial \boldsymbol{s}(\boldsymbol{r}_c^c)}{\partial \boldsymbol{r}_c^c}\dot{\boldsymbol{r}}_c^c, \tag{10.25}$$

with

$$\frac{\partial \boldsymbol{s}(\boldsymbol{r}_c^c)}{\partial \boldsymbol{r}_c^c} = \frac{1}{z_c}\begin{bmatrix} 1 & 0 & -x_c/z_c \\ 0 & 1 & -y_c/z_c \end{bmatrix} = \frac{1}{z_c}\begin{bmatrix} 1 & 0 & -X \\ 0 & 1 & -Y \end{bmatrix}.$$

To compute the interaction matrix, vector $\dot{\boldsymbol{r}}_c^c$ can be obtained from the time derivative of (10.22) under the assumption that $\boldsymbol{p}$ is constant:

$$\dot{\boldsymbol{r}}_c^c = -\boldsymbol{R}_c^T \dot{\boldsymbol{o}}_c + \boldsymbol{S}(\boldsymbol{r}_c^c)\boldsymbol{R}_c^T \boldsymbol{\omega}_c = [\, -\boldsymbol{I} \quad \boldsymbol{S}(\boldsymbol{r}_c^c)\,]\, \boldsymbol{v}_c^c. \tag{10.26}$$

Combining Eqs. (10.25), (10.26), the following expression of interaction matrix of a point can be found:

$$\boldsymbol{L}_s(\boldsymbol{s}, z_c) = \begin{bmatrix} -\dfrac{1}{z_c} & 0 & \dfrac{X}{z_c} & XY & -(1+X^2) & Y \\ 0 & -\dfrac{1}{z_c} & \dfrac{Y}{z_c} & 1+Y^2 & -XY & -X \end{bmatrix}, \tag{10.27}$$

revealing that this matrix depends on the components of vector $\boldsymbol{s}$ and the sole component z_c of vector $\boldsymbol{r}_c^c$.

The image Jacobian of a point can be computed using (10.21), (10.27) and has the expression

$$\boldsymbol{J}_s(\boldsymbol{s}, \boldsymbol{T}_o^c) = \frac{1}{z_c} \begin{bmatrix} 1 & 0 & -X & -r_{o,y}^c X & r_{o,z}^c + r_{o,x}^c X & -r_{o,y}^c \\ 0 & 1 & -Y & -(r_{o,z}^c + r_{o,y}^c Y) & r_{o,x}^c Y & r_{o,x}^c \end{bmatrix},$$

where $r_{o,x}^c$, $r_{o,y}^c$, $r_{o,z}^c$ are the components of vector $\boldsymbol{r}_o^c = \boldsymbol{r}_c^c - \boldsymbol{o}_{c,o}^c = \boldsymbol{R}_o^c \boldsymbol{r}_o^o$, $\boldsymbol{r}_o^o$ being the constant vector expressing the position of point P with respect to the object frame.

Interaction matrix of a set of points

The interaction matrix of a set of n points of the object $P_1, \dots P_n$ can be built by considering the $(2n \times 1)$ feature vector (10.5). If $\boldsymbol{L}_{s_i}(\boldsymbol{s}_i, z_{c,i})$ denotes the interaction matrix corresponding to point P_i, then the interaction matrix of the set of points will be the $(2n \times 6)$ matrix

$$\boldsymbol{L}_s(\boldsymbol{s}, \boldsymbol{z}_c) = \begin{bmatrix} \boldsymbol{L}_{s_1}(\boldsymbol{s}_1, z_{c,1}) \\ \vdots \\ \boldsymbol{L}_{s_n}(\boldsymbol{s}_n, z_{c,n}) \end{bmatrix},$$

with $\boldsymbol{z}_c = [\, z_{\,c,1} \dots z_{c,n}]^T$.

The image Jacobian of a set of points can be easily computed from the interaction matrix, using (10.21).

Interaction matrix of a line segment

A line segment is the part of the line connecting two points P_1 and P_2. The projection on the image plane is still a line segment that can be characterized in terms of the middle point coordinates $\bar{x}$, $\bar{y}$, the length L, and the angle α

formed by the line with respect to axis X. Therefore, the feature vector can be defined as

$$\boldsymbol{s} = \begin{bmatrix} \bar{x} \\ \bar{y} \\ L \\ \alpha \end{bmatrix} = \begin{bmatrix} (X_1 + X_2)/2 \\ (Y_1 + Y_2)/2 \\ \sqrt{\Delta X^2 + \Delta Y^2} \\ \tan^{-1}(\Delta Y / \Delta X) \end{bmatrix} = \boldsymbol{s}(\boldsymbol{s}_1, \boldsymbol{s}_2) \tag{10.28}$$

with $\Delta X = X_2 - X_1$, $\Delta Y = Y_2 - Y_1$ and $\boldsymbol{s}_i = [\, X_i \quad Y_i \,]^T$, $i = 1, 2$. Computing the time derivative of this equation yields

$$\begin{aligned} \dot{\boldsymbol{s}} &= \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{s}_1} \dot{\boldsymbol{s}}_1 + \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{s}_2} \dot{\boldsymbol{s}}_2 \\ &= \left(\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{s}_1} \boldsymbol{L}_{s_1}(\boldsymbol{s}_1, z_{c,1}) + \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{s}_2} \boldsymbol{L}_{s_2}(\boldsymbol{s}_2, z_{c,2}) \right) \boldsymbol{v}_c^c, \end{aligned}$$

where $\boldsymbol{L}_{s_i}$ is the interaction matrix of point P_i, under the assumption that the line segment is fixed with respect to the base frame. Therefore, the interaction matrix of a line segment is

$$\boldsymbol{L}_s(\boldsymbol{s}, \boldsymbol{z}_c) = \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{s}_1} \boldsymbol{L}_{s_1}(\boldsymbol{s}_1, z_{c,1}) + \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{s}_2} \boldsymbol{L}_{s_2}(\boldsymbol{s}_2, z_{c,2}),$$

with

$$\frac{\partial \boldsymbol{s}}{\partial \boldsymbol{s}_1} = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \\ -\Delta X/L & -\Delta Y/L \\ \Delta Y/L^2 & -\Delta X/L^2 \end{bmatrix} \quad \frac{\partial \boldsymbol{s}}{\partial \boldsymbol{s}_2} = \begin{bmatrix} 1/2 & 0 \\ 0 & 1/2 \\ \Delta X/L & \Delta Y/L \\ -\Delta Y/L^2 & \Delta X/L^2 \end{bmatrix}.$$

Notice that vectors $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ can be computed as functions of parameters $\bar{x}$, $\bar{y}$, L, α, using (10.28). Therefore, the interaction matrix can be expressed as a function of the feature vector $\boldsymbol{s} = [\, \bar{x} \quad \bar{y} \quad L \quad \alpha \,]^T$, besides the components $z_{c,1}$ and $z_{c,2}$ of the endpoints P_1 and P_2 of the line segment.

The image Jacobian of a line segment can be easily computed from the interaction matrix using (10.21).

10.3.3 Algorithmic Solution

The interaction matrix $\boldsymbol{L}_s$ is, in general, a matrix of dimension $(k \times m)$, where k is equal to the number of feature parameters of the image and m is the dimension of velocity vector $\boldsymbol{v}_c^c$. Usually $m = 6$, but it may happen that $m < 6$, when the relative motion of the object with respect to the camera is constrained.

The image Jacobian $\boldsymbol{J}_s$ is also of dimension $(k \times m)$, being related to $\boldsymbol{L}_s$ by mapping (10.21). Since this mapping is invertible, the rank of $\boldsymbol{J}_s$ coincides with that of $\boldsymbol{L}_s$.

In the case that $\boldsymbol{L}_s$ is full rank, by using (10.17), it is possible to compute $\boldsymbol{v}_{c,o}^c$ from $\dot{\boldsymbol{s}}$.

In particular, if $k = m$, the velocity $\boldsymbol{v}_{c,o}^c$ can be obtained using the expression

$$\boldsymbol{v}_{c,o}^c = \boldsymbol{\Gamma}(\boldsymbol{o}_{c,o}^c)\boldsymbol{L}_s^{-1}\dot{\boldsymbol{s}}, \tag{10.29}$$

which requires the computation of the inverse of the interaction matrix.

In the case $k > m$, the interaction matrix has more rows than columns and Eq. (10.17) can be solved using a least-squares technique, whose solution can be written in the form

$$\boldsymbol{v}_{c,o}^c = \boldsymbol{\Gamma}(\boldsymbol{o}_{c,o}^c)(\boldsymbol{L}_s^T\boldsymbol{L}_s)^{-1}\boldsymbol{L}_s^T\dot{\boldsymbol{s}}, \tag{10.30}$$

where $(\boldsymbol{L}_s^T\boldsymbol{L}_s)^{-1}\boldsymbol{L}_s^T$ is the left pseudo-inverse of $\boldsymbol{L}_s$. This situation is rather frequent in applications, because it permits using interaction matrices with good condition numbers.

Finally, in the case $k < m$, the interaction matrix has more columns than rows and Eq. (10.17) admits infinite solutions. This implies that the number of parameters of the observed image is not sufficient to determine uniquely the relative motion of the object with respect to the camera. Hence, there exist relative motions of the object with respect to the camera (or vice versa) that do not produce variations of the image feature parameters. The velocities associated with these relative motions belong to the null subspace of $\boldsymbol{J}_s$, which has the same dimension of the null subspace of $\boldsymbol{L}_s$. If the problem is that of computing uniquely the relative pose of the object with respect to the camera from feature parameters in the image plane, this case has no interest.

Example 10.2

The interaction matrix of a point P is a matrix with more columns than rows of dimension (2×6) and rank 2; therefore, the null subspace has dimension 4. It can be seen immediately that this subspace contains the velocity of the camera translational motion along the visual ray projecting point P on the image plane, proportional to vector

$$\boldsymbol{v}_1 = [\,X \quad Y \quad 1 \quad 0 \quad 0 \quad 0\,]^T,$$

as well as the velocity of the camera rotational motion about this visual ray, proportional to vector

$$\boldsymbol{v}_2 = [\,0 \quad 0 \quad 0 \quad X \quad Y \quad 1\,]^T.$$

Vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are independent and belong to a base of the null subspace of $\boldsymbol{L}_s$. The remaining base vectors are not easy to find geometrically, but can be easily computed analytically.

The pose estimation problem may be cast in a form analogous to that of inverse kinematics algorithms for robot manipulators. To this end, it is

necessary to represent the relative pose of the object with respect to the camera using a minimum number of coordinates, in terms of the $(m \times 1)$ vector

$$\boldsymbol{x}_{c,o} = \begin{bmatrix} \boldsymbol{o}_{c,o}^{c} \\ \boldsymbol{\phi}_{c,o} \end{bmatrix}, \tag{10.31}$$

where $\boldsymbol{o}_{c,o}^{c}$ characterizes the position of the origin of the object frame with respect to the camera frame and $\boldsymbol{\phi}_{c,o}$ characterizes the relative orientation. If Euler angles are used to represent orientation, then $\boldsymbol{\phi}_{c,o}$ is the vector of the angles extracted from rotation matrix $\boldsymbol{R}_{o}^{c}$ and the mapping between $\boldsymbol{v}_{c,o}^{c}$ and $\dot{\boldsymbol{x}}_{c,o}$ is expressed by

$$\boldsymbol{v}_{c,o}^{c} = \begin{bmatrix} \boldsymbol{I} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{T}(\boldsymbol{\phi}_{c,o}) \end{bmatrix} \dot{\boldsymbol{x}}_{c,o} = \boldsymbol{T}_{A}(\boldsymbol{\phi}_{c,o}) \dot{\boldsymbol{x}}_{c,o}. \tag{10.32}$$

Example 10.3

Consider a camera mounted on the end-effector of the SCARA manipulator of Fig. 2.36. Choose the camera frame parallel to the end-effector frame, with axis z_c pointing downward. Assume that the camera observes a fixed planar object, parallel to the image plane, and that axis z_o of the object frame is parallel to axis z_c and points downward.

The geometry of the problem suggests that the relative position of the object with respect to the camera can be represented by a vector $\boldsymbol{o}_{c,o}^{c}$, whereas the relative orientation can be defined by the angle α between the object frame and the camera frame about axis z_c. Therefore, $m = 4$ and

$$\boldsymbol{x}_{c,o} = \begin{bmatrix} \boldsymbol{o}_{c,o}^{c} \\ \alpha \end{bmatrix}. \tag{10.33}$$

Moreover, the time derivative $\dot{\alpha}$ coincides with the component of $\boldsymbol{\omega}_{c,o}^{c}$ along z_c, and this is the sole non-null component of the angular velocity of the relative motion of the object frame with respect to the camera frame. Hence, in (10.32), $\boldsymbol{T}_{A}(\boldsymbol{\phi}_{c,o})$ is the (4×4) identity matrix.

Equation (10.17) can be rewritten in the form

$$\dot{\boldsymbol{s}} = \boldsymbol{J}_{A_s}(\boldsymbol{s}, \boldsymbol{x}_{c,o}) \dot{\boldsymbol{x}}_{c,o}, \tag{10.34}$$

where the matrix

$$\boldsymbol{J}_{A_s}(\boldsymbol{s}, \boldsymbol{x}_{c,o}) = \boldsymbol{L}_{s} \boldsymbol{\Gamma}(-\boldsymbol{o}_{c,o}^{c}) \boldsymbol{T}_{A}(\boldsymbol{\phi}_{c,o}) \tag{10.35}$$

has a meaning analogous to that of the analytic Jacobian of a manipulator.

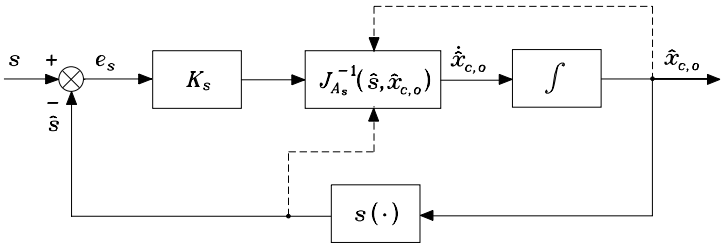

Fig. 10.7. Pose estimation algorithm based on the inverse of image Jacobian

Equation (10.34) is the starting point of a numeric integration algorithm for the computation of $\boldsymbol{x}_{c,o}$, similar to inverse kinematics algorithms. Let $\widehat{\boldsymbol{x}}_{c,o}$ denote the current estimate of vector $\boldsymbol{x}_{c,o}$ and let

$$\widehat{\boldsymbol{s}} = \boldsymbol{s}(\widehat{\boldsymbol{x}}_{c,o})$$

be the corresponding vector of image feature parameters computed from the pose specified by $\widehat{\boldsymbol{x}}_{c,o}$; the objective of this algorithm is the minimization of the error

$$\boldsymbol{e}_s = \boldsymbol{s} - \widehat{\boldsymbol{s}}. \tag{10.36}$$

Notice that, for the purpose of numerical integration, vector $\boldsymbol{s}$ is constant while the current estimate $\widehat{\boldsymbol{s}}$ depends on the current integration time. Therefore, computing the time derivative of (10.36) yields

$$\dot{\boldsymbol{e}}_s = -\dot{\widehat{\boldsymbol{s}}} = -\boldsymbol{J}_{A_s}(\widehat{\boldsymbol{s}}, \widehat{\boldsymbol{x}}_{c,o})\dot{\widehat{\boldsymbol{x}}}_{c,o}. \tag{10.37}$$

Assumed that matrix $\boldsymbol{J}_{A_s}$ is square and nonsingular, the choice

$$\dot{\widehat{\boldsymbol{x}}}_{c,o} = \boldsymbol{J}_{A_s}^{-1}(\widehat{\boldsymbol{s}}, \widehat{\boldsymbol{x}}_{c,o})\boldsymbol{K}_s\boldsymbol{e}_s \tag{10.38}$$

leads to the equivalent linear system

$$\dot{\boldsymbol{e}}_s + \boldsymbol{K}_s\boldsymbol{e}_s = \boldsymbol{0}. \tag{10.39}$$

Therefore, if $\boldsymbol{K}_s$ is a positive definite matrix (usually diagonal), system (10.39) is asymptotically stable and the error tends to zero with a convergence speed that depends on the eigenvalues of matrix $\boldsymbol{K}_s$. The convergence to zero of error $\boldsymbol{e}_s$ ensures the asymptotic convergence of the estimate $\widehat{\boldsymbol{x}}_{c,o}$ to the true value $\boldsymbol{x}_{c,o}$.

The block scheme of the pose estimation algorithm is shown in Fig. 10.7, where $\boldsymbol{s}(\cdot)$ denotes the function computing the feature vector of the 'virtual' image corresponding to the current estimate $\widehat{\boldsymbol{x}}_{c,o}$ of the object pose with respect to the camera. This algorithm can be used as an alternative to the analytic methods for pose estimation illustrated in Sect. 10.3.1. Obviously, the convergence properties depend on the choice of the image feature parameters

and on the initial value of estimate $\widehat{\boldsymbol{x}}_{c,o}(0)$, which may produce instability problems related to the singularities of matrix $\boldsymbol{J}_{A_s}$.

Notice that, in view of (10.35), the singularities of matrix $\boldsymbol{J}_{A_s}$ are both the representation singularities of the orientation and those of the interaction matrix. The most critical singularities are those of the interaction matrix, since they depend on the choice of the image feature parameters.

To separate the effects of the two types of singularities, it is convenient to compute (10.38) in two steps, evaluating first

$$\widehat{\boldsymbol{v}}_{c,o}^{c} = \boldsymbol{\Gamma}(\boldsymbol{o}_{c,o}^{c})\boldsymbol{L}_s^{-1}\boldsymbol{K}_s\boldsymbol{e}_s, \tag{10.40}$$

and then

$$\dot{\widehat{\boldsymbol{x}}}_{c,o} = \boldsymbol{T}_A^{-1}(\boldsymbol{\phi}_{c,o})\widehat{\boldsymbol{v}}_{c,o}^{c}. \tag{10.41}$$

Assumed to work far from representation singularities, the problem of singularities of $\boldsymbol{L}_s$ can be overcome by using a number k of feature parameters greater than the minimum required m. This choice also allows a reduction of the effects of measurement noise. The resulting estimation algorithm requires the use of the left pseudo-inverse of $\boldsymbol{L}_s$ in place of the inverse, namely

$$\widehat{\boldsymbol{v}}_{c,o}^{c} = \boldsymbol{\Gamma}(\boldsymbol{o}_{c,o}^{c})(\boldsymbol{L}_s^T\boldsymbol{L}_s)^{-1}\boldsymbol{L}_s^T\boldsymbol{K}_s\boldsymbol{e}_s \tag{10.42}$$

in place of (10.40). The convergence of error (10.36) can be shown using the direct Lyapunov method based on the positive definite function [2]

$$V(\boldsymbol{e}_s) = \frac{1}{2}\boldsymbol{e}_s^T\boldsymbol{K}_s\boldsymbol{e}_s > 0 \qquad \forall \boldsymbol{e}_s \neq \boldsymbol{0}.$$

Computing the time derivative of this function, and using (10.37), (10.35), (10.41), (10.42), yields

$$\dot{V} = -\boldsymbol{e}_s^T\boldsymbol{K}_s\boldsymbol{L}_s(\boldsymbol{L}_s^T\boldsymbol{L}_s)^{-1}\boldsymbol{L}_s^T\boldsymbol{K}_s\boldsymbol{e}_s,$$

which is negative semi-definite because $\mathcal{N}(\boldsymbol{L}_s^T) \neq \emptyset$, $\boldsymbol{L}_s^T$ being a matrix with more columns than rows. Therefore, the system is stable but not asymptotically stable. This implies that the error is bounded, but in some cases the algorithm can get stuck with $\boldsymbol{e}_s \neq \boldsymbol{0}$ and $\boldsymbol{K}_s\boldsymbol{e}_s \in \mathcal{N}(\boldsymbol{L}_s^T)$.

Notice that the pose estimation methods based on inverse Jacobian are as efficient in terms of accuracy, speed of convergence and computational load, as the initial estimate $\widehat{\boldsymbol{x}}_{c,o}(0)$ is close to the true value $\boldsymbol{x}_{c,o}$. Therefore, these methods are mainly adopted for real-time 'visual tracking' applications, where the estimate on an image taken at time $\bar{t}$ is computed assuming as initial value the estimate computed on the image taken at time $\bar{t}-T$, T being the sampling time of the image (multiple of the sampling time of the numerical integration algorithm).

[2] See Sect. C.3 for the illustration of the direct Lyapunov method.

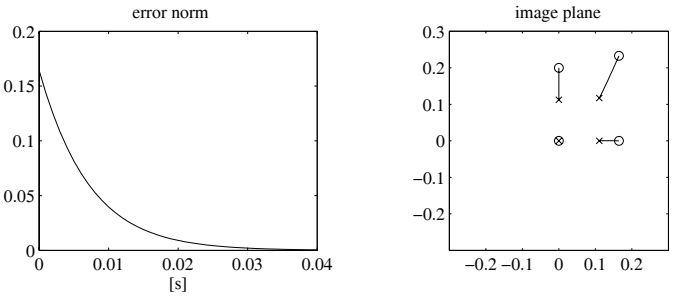

Fig. 10.8. Time history of the norm of the estimation error and corresponding paths of the feature points projections on image plane

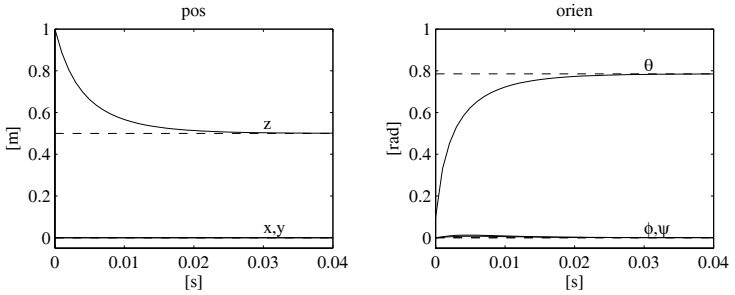

Fig. 10.9. Time history of camera pose estimate

Example 10.4

Consider the object of Example 10.1. Using the algorithmic solution, it is desired to compute the relative pose of the object frame, with respect to the camera frame, from the image plane coordinates of the projections of the four feature points of the object. The same numerical values of Example 10.1 are used.

Since the image Jacobian has dimension (6×8), it is necessary to use the algorithm based on the pseudo-inverse of the interaction matrix. This algorithm was simulated on a computer by adopting the Euler numerical integration scheme with an integration time $\Delta t = 1\,\text{ms}$, matrix gain $\boldsymbol{K}_s = 160\boldsymbol{I}_8$, and initial estimate $\widehat{\boldsymbol{x}}_{c,o} = [\,0 \quad 0 \quad 1 \quad 0 \quad \pi/32 \quad 0\,]^T$.

The results in Fig. 10.8 show that the norm of the estimation error of the feature parameters $\boldsymbol{e}_s$ tends to zero asymptotically with convergence of exponential type; moreover, due to the fact that matrix gain $\boldsymbol{K}_s$ was chosen diagonal with equal elements, the paths of the projections of the feature points on the image plane (between the initial positions marked with crosses and the final positions marked with circles) are line segments.

The corresponding time histories of the components of vector $\widehat{\boldsymbol{x}}_{c,o}$ for position and orientation are reported in Fig. 10.9, together with the time histories of the components of the true value $\boldsymbol{x}_{c,o} = [\,0 \quad 0 \quad 0.5 \quad 0 \quad \pi/4 \quad 0\,]^T$ (represented with

dashed lines). It can be verified that, with the chosen value of $\boldsymbol{K}_s$, the algorithm converges to the true value in about 0.03 s, corresponding to 30 iterations.

The time histories of Fig. 10.9 can be interpreted as the position and orientation trajectories of a 'virtual' camera in motion between the initial pose $\widehat{\boldsymbol{x}}_{c,o}(0)$ and the final pose $\boldsymbol{x}_{c,o}$.

Notice that, for the purpose of pose estimation, the illustrated algorithm may converge also in the case that only three feature points are used. In fact, in this case, $\boldsymbol{J}_{As}$ is a square (6×6) matrix and the convergence is ensured provided that this matrix is nonsingular. However, since P3P problem has four solutions, the algorithm may converge to a solution different from that sought, unless, as for visual tracking applications, the initial estimate $\widehat{\boldsymbol{x}}_{c,o}(0)$ is close enough to the true value $\boldsymbol{x}_{c,o}$.

10.4 Stereo Vision

The bidimensional image provided by a camera does not give any explicit information on depth, namely, the distance of the observed object from the camera. This information can be recovered in an indirect way from the geometric model of the object, assumed to be known.

On the other hand, the depth of a point can be directly computed in the case that two images of the same scene are available, taken from two different points of view. The two images can be obtained using two cameras, or sequentially, using a moving camera. These cases are referred to as *stereo vision*.

In the framework of stereo vision, two fundamental problems can be devised. The first is the *correspondence problem*, which consists of the identification of the points of the two images that are projections of the same point of the scene. These points are termed *conjugate* or *corresponding*. This problem is not easy to solve, and the solution is based on the existence of geometric constraints between two images of the same point, besides the fact that some details of the scene appear to be similar in the two images.

The second problem, illustrated below in some fundamental aspects, is that of *3D reconstruction* which, in general, consists of the computation of the relative pose of the cameras (calibrated and not) and thus, starting from this pose, the position in the 3D space of the points of the observed object.

10.4.1 Epipolar Geometry

Assume that two cameras are available, with respective reference frames, denoted as 1 and 2. Moreover, let $\boldsymbol{o}_{1,2}^1$ denote the position vector and $\boldsymbol{R}_2^1$ the rotation matrix of Frame 2 with respect to Frame 1 and let $\boldsymbol{T}_2^1$ be the corresponding homogeneous transformation matrix. The coordinates of a point P expressed in the two frames are related by equation

$$\boldsymbol{p}^1 = \boldsymbol{o}_{1,2}^1 + \boldsymbol{R}_2^1 \boldsymbol{p}^2. \tag{10.43}$$

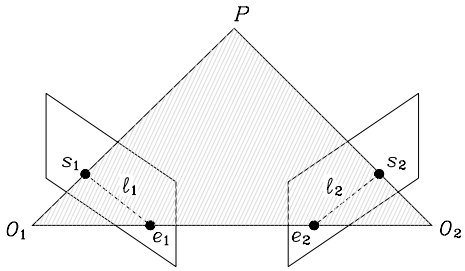

Fig. 10.10. Epipolar geometry

Let $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ be the coordinates of the projections of P on the image planes of the cameras; in view of (5.44) it is

$$\lambda_i \widetilde{\boldsymbol{s}}_i = \boldsymbol{\Pi} \widetilde{\boldsymbol{p}}^i = \boldsymbol{p}^i, \qquad i = 1, 2. \tag{10.44}$$

Substituting (10.44) into (10.43) yields

$$\lambda_1 \widetilde{\boldsymbol{s}}_1 = \boldsymbol{o}^1_{1,2} + \lambda_2 \boldsymbol{R}^1_2 \widetilde{\boldsymbol{s}}_2. \tag{10.45}$$

Premultiplying both sides of (10.45) by $\boldsymbol{S}(\boldsymbol{o}^1_{1,2})$ gives

$$\lambda_1 \boldsymbol{S}(\boldsymbol{o}^1_{1,2}) \widetilde{\boldsymbol{s}}_1 = \lambda_2 \boldsymbol{S}(\boldsymbol{o}^1_{1,2}) \boldsymbol{R}^1_2 \widetilde{\boldsymbol{s}}_2.$$

Hence, premultiplying both sides of the above equation by $\widetilde{\boldsymbol{s}}_1^T$, the following equality is obtained

$$\lambda_2 \widetilde{\boldsymbol{s}}_1^T \boldsymbol{S}(\boldsymbol{o}^1_{1,2}) \boldsymbol{R}^1_2 \widetilde{\boldsymbol{s}}_2 = 0,$$

which has to be satisfied for any value of scalar λ_2. Therefore, this equality is equivalent to the so-called *epipolar constraint* equation

$$\widetilde{\boldsymbol{s}}_1^T \boldsymbol{E} \widetilde{\boldsymbol{s}}_2 = 0, \tag{10.46}$$

where $\boldsymbol{E} = \boldsymbol{S}(\boldsymbol{o}^1_{1,2}) \boldsymbol{R}^1_2$ is a (3×3) matrix known as *essential matrix*. Equation (10.46) expresses in analytic form the geometric constraint existing between the projections of the same point on the image planes of the two cameras.

The geometric interpretation of the epipolar constraint can be derived from Fig. 10.10, where the projections of a point P on the image planes of the two cameras are reported as well as the respective optical centers O_1 and O_2. Notice that points O_1, O_2 and P are the vertices of a triangle whose sides O_1P and O_2P belong to the visual rays projecting point P into the points of coordinates $\boldsymbol{s}_1$ and $\boldsymbol{s}_2$ of the image plane, respectively. These rays lay along the directions of vectors $\widetilde{\boldsymbol{s}}_1$ and $\boldsymbol{R}^1_2 \widetilde{\boldsymbol{s}}_2$ respectively, expressed with respect to Frame 1. Line segment O_1O_2, termed *base line*, is represented by vector $\boldsymbol{o}^1_{1,2}$. The epipolar constraint (10.46) corresponds to imposing that vectors $\widetilde{\boldsymbol{s}}_1$, $\boldsymbol{R}^1_2 \widetilde{\boldsymbol{s}}_2$

and $\boldsymbol{o}_{1,2}^1$ are coplanar. The plane containing these vectors is termed *epipolar plane*.

Notice that line segment O_1O_2 belongs to the visual ray projecting point O_2 on the image plane of Camera 1 and, at the same time, to the ray projecting point O_1 on the image plane of Camera 2. These projections, of coordinates $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$, respectively, are termed *epipoles*. Line ℓ_1, passing through the points of coordinates $\boldsymbol{s}_1$ and $\boldsymbol{e}_1$, and line ℓ_2, passing through the points of coordinates $\boldsymbol{s}_2$ and $\boldsymbol{e}_2$, are termed *epipolar lines*. The epipolar lines can also be obtained as the intersection of the epipolar plane with the image planes of the two cameras. Notice that, by varying point P, the epipolar plane describes a set of planes about the base line and the epipoles do not change.

For the purpose of computing the correspondences, the epipolar constraint can be exploited to reduce the complexity of the problem to find conjugate points. In fact, if $\boldsymbol{s}_1$ is the image of a point of the visual ray passing through O_1 and the point of coordinates $\boldsymbol{s}_1$, the corresponding conjugate point of the image plane of Camera 2 must necessarily belong to the epipolar line ℓ_2, which is known because the epipolar plane is uniquely defined by O_1, O_2 and by the point of coordinates $\boldsymbol{s}_1$. Finding correspondences, therefore, reduces to searching for a point along the epipolar line and not on the whole image plane.

In the framework of 3D reconstruction, different scenarios may arise, depending on the type of information which is available a priori.

10.4.2 Triangulation

In the case that both intrinsic and extrinsic parameters of the two cameras are known, the reconstruction problem consists of computing the position in the scene of the points projected on the two image planes using a geometric method known as *triangulation*. This method allows the computation of coordinates $\boldsymbol{p} = [\, p_x \quad p_y \quad p_z \,]^T$ of a point P with respect to the base frame, starting from normalized coordinates $\boldsymbol{s}_1 = [\, X_1 \quad Y_1 \,]^T$ and $\boldsymbol{s}_2 = [\, X_2 \quad Y_2 \,]^T$ of the projections of P on the image planes of the two cameras. Assume that, for simplicity, the base frame coincides with Frame 1, then $\boldsymbol{p}^1 = \boldsymbol{p}$, $\boldsymbol{o}_{1,2}^1 = \boldsymbol{o}$ and $\boldsymbol{R}_2^1 = \boldsymbol{R}$.

From (10.44), (10.45) the following equalities can be derived:

$$\boldsymbol{p} = \lambda_1 \widetilde{\boldsymbol{s}}_1 \tag{10.47}$$

$$\boldsymbol{p} = \boldsymbol{o} + \lambda_2 \boldsymbol{R} \widetilde{\boldsymbol{s}}_2, \tag{10.48}$$

where the first equality is the parametric equation of the visual ray passing through O_1 and the point of coordinates $\boldsymbol{s}_1$, while the second represents the parametric equation of the visual ray passing through O_2 and the point of coordinates $\boldsymbol{s}_2$; both equations are expressed in the base frame.

Therefore, the coordinates of point P at the intersection of the two visual rays can be computed by solving the system of two Eqs. (10.47), (10.48) with

respect to $\boldsymbol{p}$. To this end, from (10.47), by computing λ_1 in the third equation and replacing its value into the other two equations, the following system is obtained:

$$\begin{bmatrix} 1 & 0 & -X_1 \\ 0 & 1 & -Y_1 \end{bmatrix} \boldsymbol{p} = \boldsymbol{0}. \tag{10.49}$$

Using a similar derivation for the equation obtained by premultiplying both sides of (10.48) by $\boldsymbol{R}^T$, the following system is obtained

$$\begin{bmatrix} \boldsymbol{r}_1^T - X_2 \boldsymbol{r}_3^T \\ \boldsymbol{r}_2^T - Y_2 \boldsymbol{r}_3^T \end{bmatrix} \boldsymbol{p} = \begin{bmatrix} o_x - o_z X_2 \\ o_y - o_z Y_2 \end{bmatrix}, \tag{10.50}$$

with $\boldsymbol{R} = [\, \boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{r}_3 \,]$ and $\boldsymbol{R}^T \boldsymbol{o} = [\, o_x \quad o_y \quad o_z \,]^T$. Equations (10.49), (10.50) define a system of four equations and three unknowns, which is linear with respect to $\boldsymbol{p}$. Of these equations, only three are independent in the ideal case that the two visual rays intersect at point P. In practical applications, because of noise, these equations are all independent and the system has no solution; hence, suitable algorithms based on least-squares techniques have to be adopted to compute an approximate solution.

Computation of $\boldsymbol{p}$ can be greatly simplified in the case, rather frequent in applications, that the two cameras have parallel and aligned image planes, with $\boldsymbol{R} = \boldsymbol{I}$ and $\boldsymbol{R}^T \boldsymbol{o} = [\, b \quad 0 \quad 0 \,]^T$, $b > 0$ being the distance between the origins of the two camera frames. This implies that the solution to the system of Eqs. (10.49), (10.50) is

$$p_x = \frac{X_1 b}{X_1 - X_2} \tag{10.51}$$

$$p_y = \frac{Y_1 b}{X_1 - X_2} = \frac{Y_2 b}{X_1 - X_2} \tag{10.52}$$

$$p_z = \frac{b}{X_1 - X_2}. \tag{10.53}$$

10.4.3 Absolute Orientation

In the case of a calibrated system of two cameras observing a rigid object of unknown shape, triangulation can be used to compute the variation of pose of the object or of the system of cameras, due to the relative motion of the system with respect to the cameras. This problem, known as *absolute orientation*, requires the measurement of the positions of the projections of a certain number of feature points of the object.

If the stereo camera system is moving and the object is fixed, let $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_n$ denote the position vectors of n points of the rigid object measured at time t and $\boldsymbol{p}'_1, \ldots, \boldsymbol{p}'_n$ the position vectors of the same points measured at time t' using triangulation. These vectors are all referred to Frame 1 and, under the assumption of rigid motion, satisfy the equations

$$\boldsymbol{p}_i = \boldsymbol{o} + \boldsymbol{R} \boldsymbol{p}'_i \qquad i = 1, \ldots, n \tag{10.54}$$

where vector $\boldsymbol{o}$ and rotation matrix $\boldsymbol{R}$ define the position and orientation displacement of Frame 1 between time t and time t'. The absolute orientation problem consists of the computation of $\boldsymbol{R}$ and $\boldsymbol{o}$ from $\boldsymbol{p}_i$ and $\boldsymbol{p}'_i$.

From rigid body mechanics it is known that this problem has a unique solution in the case of three non-collinear points. In this case, nine nonlinear equations can be derived from (10.54), in terms of the nine independent parameters which characterize $\boldsymbol{o}$ and $\boldsymbol{R}$. However, since the points are obtained using triangulation, the measurements are affected by error and the system may have no solution. In this case, it is convenient to consider a number $n > 3$ of points and compute $\boldsymbol{o}$ and $\boldsymbol{R}$ as the quantities which minimize the linear quadratic function

$$\sum_{i=1}^{n} \|\boldsymbol{p}_i - \boldsymbol{o} - \boldsymbol{R}\boldsymbol{p}'_i\|^2, \tag{10.55}$$

with the constraint that $\boldsymbol{R}$ is a rotation matrix. The problem of computing $\boldsymbol{o}$ can be separated from the problem of computing $\boldsymbol{R}$ observing that the value of $\boldsymbol{o}$ which minimizes function (10.55) is (see Problem 10.6)

$$\boldsymbol{o} = \bar{\boldsymbol{p}} - \boldsymbol{R}\bar{\boldsymbol{p}}' \tag{10.56}$$

where $\bar{\boldsymbol{p}}$ and $\bar{\boldsymbol{p}}'$ are the centroids of the set of points $\{\boldsymbol{p}_i\}$ and $\{\boldsymbol{p}'_i\}$, defined as

$$\bar{\boldsymbol{p}} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{p}_i, \qquad \bar{\boldsymbol{p}}' = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{p}'_i.$$

Hence the problem becomes that of computing the rotation matrix $\boldsymbol{R}$ which minimizes the linear quadratic function

$$\sum_{i=1}^{n} \|\bar{\boldsymbol{p}}_i - \boldsymbol{R}\bar{\boldsymbol{p}}'_i\|^2, \tag{10.57}$$

where $\bar{\boldsymbol{p}}_i = \boldsymbol{p}_i - \bar{\boldsymbol{p}}$ and $\bar{\boldsymbol{p}}'_i = \boldsymbol{p}'_i - \bar{\boldsymbol{p}}'$ are the deviations with respect to the centroids.

It can be proven that the matrix $\boldsymbol{R}$ which minimizes function (10.57) is the matrix which maximizes the trace of $\boldsymbol{R}^T\boldsymbol{K}$, with

$$\boldsymbol{K} = \sum_{i=1}^{n} \bar{\boldsymbol{p}}_i \bar{\boldsymbol{p}}_i'^T;$$

see Problem 10.7. Therefore, the solution has the form (10.12) where, for the purpose of this problem, $\boldsymbol{U}$ and $\boldsymbol{V}$ are respectively the left and right orthogonal matrices of the singular value decomposition of $\boldsymbol{K}$. Once that rotation matrix $\boldsymbol{R}$ is known, vector $\boldsymbol{o}$ can be computed using (10.56).

10.4.4 3D Reconstruction from Planar Homography

Another interesting case of application of 3D reconstruction occurs when the feature points of the observed object lie on the same plane. This geometric property represents an additional constraint between the projections of each point on the image plane of the two cameras, besides the epipolar constraint. This constraint is a *planar homography*.

Let $\boldsymbol{p}^2$ be the position vector of a point P of the object, expressed with respect to Frame 2. Moreover, let $\boldsymbol{n}^2$ denote the unit vector orthogonal to the plane containing the feature points and $d_2 > 0$ the distance of the plane from the origin of Frame 2. By virtue of simple geometric considerations, the following equation can be derived:

$$\frac{1}{d_2}\boldsymbol{n}^{2T}\boldsymbol{p}^2 = 1$$

which defines the set of points $\boldsymbol{p}^2$ belonging to the plane. In view of the above equality, Eq. (10.43) can be rewritten in the form

$$\boldsymbol{p}^1 = \boldsymbol{H}\boldsymbol{p}^2, \tag{10.58}$$

with

$$\boldsymbol{H} = \boldsymbol{R}_2^1 + \frac{1}{d_2}\boldsymbol{o}_{1,2}^1\boldsymbol{n}^{2T}. \tag{10.59}$$

Replacing (10.44) into (10.58) yields

$$\widetilde{\boldsymbol{s}}_1 = \lambda\boldsymbol{H}\widetilde{\boldsymbol{s}}_2, \tag{10.60}$$

where $\lambda = \lambda_2/\lambda_1 > 0$ is an arbitrary constant. Premultiplication of both sides of (10.60) by $\boldsymbol{S}(\widetilde{\boldsymbol{s}}_1)$ yields the equality

$$\boldsymbol{S}(\widetilde{\boldsymbol{s}}_1)\boldsymbol{H}\widetilde{\boldsymbol{s}}_2 = \boldsymbol{0}, \tag{10.61}$$

representing a planar homography defined by matrix $\boldsymbol{H}$.

Using a derivation similar to that presented in Sect. 10.3.1, it is possible to compute numerically matrix $\zeta\boldsymbol{H}$, up to a scaling factor ζ, starting from the coordinates of n points of the plane, with $n \geq 4$.

The value of the scaling factor ζ can be computed using a numerical algorithm based on expression (10.59) of matrix $\boldsymbol{H}$; once $\boldsymbol{H}$ is known, it is possible to compute quantities $\boldsymbol{R}_2^1$, $\boldsymbol{o}_{1,2}^1/d_2$ and $\boldsymbol{n}^2$ in (10.59) — actually, it can be shown that two admissible solutions exist.

This result is of a certain relevance for visual servoing applications. For example, in the case of a camera in motion with respect to the object, if Frames 1 and 2 represent the poses of the camera in two different time instants, the computation of $\boldsymbol{H}$ with decomposition (10.59) can be used to evaluate the orientation displacement of the camera frame and the position displacement of the origin, the latter defined up to a scaling factor d_2. This information can be achieved without knowing the object geometry, as long as the feature points all belong to the same plane.

Example 10.5

Consider the SCARA manipulator of Example 10.3 and the planar object of Example 10.1. Assume that the feature vector of the four points of the object, measured by the camera at time t', is

$$\boldsymbol{s}' = [\,0 \quad 0 \quad 0.2 \quad 0 \quad 0.2 \quad 0.2 \quad 0 \quad 0.2\,]^T,$$

while the feature vector, measured by the camera at time t'', is

$$\boldsymbol{s}'' = [\,-0.1667 \quad 0.1667 \quad -0.0833 \quad 0.0223 \quad 0.0610 \quad 0.1057 \quad -0.0223 \quad 0.2500\,]^T.$$

It is desired to compute the quantities $\boldsymbol{R}_2^1$, $(1/d_2)\boldsymbol{o}_{1,2}^1$ and $\boldsymbol{n}^2$ of the planar homography (10.59).

For simplicity, assume that the orientation of the planar object is known, namely

$$\boldsymbol{n}^2 = [\,0 \quad 0 \quad 1\,]^T.$$

A further simplification to the problem derives from the fact that, in this case, matrix $\boldsymbol{R}_2^1$ corresponds to a rotation about axis z of an angle β, namely $\boldsymbol{R}_2^1 = \boldsymbol{R}_z(\beta)$. Therefore, in view of (10.59), planar homography $\boldsymbol{H}$ has the symbolic expression

$$\boldsymbol{H} = \begin{bmatrix} c_\beta & -s_\beta & o_x/d_2 \\ s_\beta & c_\beta & o_y/d_2 \\ 0 & 0 & 1 + o_z/d_2 \end{bmatrix},$$

with $\boldsymbol{o}_{1,2}^1 = [\,o_x \quad o_y \quad o_z\,]^T$.

On the other hand, starting from numerical values of $\boldsymbol{s}'$ and $\boldsymbol{s}''$ and using a derivation similar to that used in Example 10.1, the following matrix can be obtained:

$$\zeta\boldsymbol{H} = \begin{bmatrix} 0.3015 & -0.5222 & 0.1373 \\ 0.5222 & 0.3015 & 0.0368 \\ 0 & 0 & 0.5025 \end{bmatrix},$$

which is the numerical value of the planar homography, up to a scaling factor ζ.

The symbolic expression of $\boldsymbol{H}$ reveals that the first and second column of this matrix have unit norm. This property can be used to compute $|\zeta| = 1.6583$. The sign of ζ can be evaluated by imposing, for any of the feature points of the object, the constraint

$$\boldsymbol{p}^{1T}\boldsymbol{p}^1 = \boldsymbol{p}^{1T}\boldsymbol{H}\boldsymbol{p}^2 = \lambda_1\lambda_2\widetilde{\boldsymbol{s}}_1^T\boldsymbol{H}\widetilde{\boldsymbol{s}}_2 > 0.$$

Since scalars λ_i are positive, this inequality is equivalent to

$$\widetilde{\boldsymbol{s}}_1^T\boldsymbol{H}\widetilde{\boldsymbol{s}}_2 > 0,$$

hence, in this case, it is $\zeta > 0$. Therefore

$$\boldsymbol{H} = \begin{bmatrix} 0.5 & -0.8660 & 0.2277 \\ 0.8660 & 0.5 & 0.0610 \\ 0 & 0 & 0.8333 \end{bmatrix}.$$

At this point, the value of angle β can be computed from the elements h_{11} and h_{21} of matrix $\boldsymbol{H}$ using equation

$$\beta = \mathrm{Atan2}(h_{21}, h_{11}) = \frac{\pi}{3}.$$

Finally, vector $(1/d_2)\boldsymbol{o}_{1,2}^1$ can be computed from the elements of the last row of matrix $\boldsymbol{H}$ using the equality

$$\frac{1}{d_2}\boldsymbol{o}_{1,2}^1 = \begin{bmatrix} h_{13} \\ h_{23} \\ h_{33}-1 \end{bmatrix} = \begin{bmatrix} 0.2277 \\ 0.0610 \\ -0.0667 \end{bmatrix}.$$

Notice that the derivation illustrated in Example 10.5, in the case that $\boldsymbol{n}^2$ is unknown and $\boldsymbol{R}_2^1$ is a generic rotation matrix, becomes much more complex.

10.5 Camera Calibration

An important problem for visual servoing applications is the calibration of the camera, which is the sensor providing the information fed back to the controller. Calibration consists of the estimation of the *intrinsic parameters*, characterizing matrix $\boldsymbol{\Omega}$ defined in (5.41), and of the *extrinsic parameters*, characterizing the pose of the camera frame with respect to the base frame (for eye-to-end cameras) or to the end-effector frame (for eye-in-hand cameras). Various calibration techniques exist, which are based on algorithms similar to those used for the pose estimation of an object with respect to a camera.

In particular, the solution method of a PnP problem with n coplanar points illustrated in Sect. 10.3.1 can be directly used for the computation of the extrinsic parameters of the camera, if the intrinsic parameters are known.

In fact, in view of (10.14), extrinsic parameters of an eye-to-hand camera can be computed as

$$\boldsymbol{T}_c^b = \boldsymbol{T}_o^b(\boldsymbol{T}_o^c)^{-1}, \tag{10.62}$$

where matrix $\boldsymbol{T}_o^c$ is the output of the algorithm solving a PnP planar problem, provided that matrix $\boldsymbol{T}_o^b$, expressing the position and the orientation of the object frame with respect to the base frame, is known. Similarly, in view of (10.15), the extrinsic parameters of an eye-in-hand camera can be computed as

$$\boldsymbol{T}_c^e = \boldsymbol{T}_o^e(\boldsymbol{T}_o^c)^{-1}, \tag{10.63}$$

provided that matrix $\boldsymbol{T}_o^e$, expressing the pose of the object frame with respect to the end-effector frame, is known.

If the intrinsic parameters are not known, the derivation of Sect. 10.3.1 has to be suitably extended and can be broken down in the three phases described below.

Phase 1

A planar homography can be computed starting from pixel coordinates

$$\boldsymbol{c}_i = \begin{bmatrix} X_{Ii} \\ Y_{Ii} \end{bmatrix}$$

in place of normalized coordinates $\boldsymbol{s}_i$. In detail, an equation formally identical to (10.7) can be obtained:

$$\boldsymbol{S}(\tilde{\boldsymbol{c}}_i)\boldsymbol{H}'\,[\,r_{x,i} \quad r_{y,i} \quad 1\,]^T = \boldsymbol{0}, \tag{10.64}$$

where $\boldsymbol{H}'$ is the (3×3) matrix

$$\boldsymbol{H}' = \boldsymbol{\Omega}\boldsymbol{H}, \tag{10.65}$$

$\boldsymbol{\Omega}$ being the matrix of intrinsic parameters (5.41) and $\boldsymbol{H}$ the matrix defined in (10.8). Using an algorithm similar to that presented in Sect. 10.3.1, planar homography $\zeta\boldsymbol{H}'$ can be computed, up to a scaling factor ζ, from the coordinates of n points of the plane, with $n \geq 4$.

Phase 2

Matrix $\boldsymbol{\Omega}$ can be computed from the elements of matrix $\zeta\boldsymbol{H}'$. In fact, taking into account (10.65), and the definition of $\boldsymbol{H}$ in (10.8), yields

$$\zeta\,[\,\boldsymbol{h}'_1 \quad \boldsymbol{h}'_2 \quad \boldsymbol{h}'_3\,] = \boldsymbol{\Omega}\,[\,\boldsymbol{r}_1 \quad \boldsymbol{r}_2 \quad \boldsymbol{o}^c_{c,o}\,],$$

where $\boldsymbol{h}'_i$ denotes the i-th column of matrix $\boldsymbol{H}'$. Computing $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ from this equation, and imposing the orthogonality and unit norm constraints on these vectors, the following two scalar equations are obtained:

$$\begin{aligned} &\boldsymbol{h}'^{\,T}_1 \boldsymbol{\Omega}^{-T}\boldsymbol{\Omega}^{-1}\boldsymbol{h}'_2 = 0 \\ &\boldsymbol{h}'^{\,T}_1 \boldsymbol{\Omega}^{-T}\boldsymbol{\Omega}^{-1}\boldsymbol{h}'_1 = \boldsymbol{h}'^{\,T}_2 \boldsymbol{\Omega}^{-T}\boldsymbol{\Omega}^{-1}\boldsymbol{h}'_2 \end{aligned}$$

which, being linear, can be rewritten in the form

$$\boldsymbol{A}'\boldsymbol{b} = \boldsymbol{0}. \tag{10.66}$$

In the above equation, $\boldsymbol{A}'$ is a (2×6) matrix of coefficients depending on $\boldsymbol{h}'_1$, $\boldsymbol{h}'_2$, while $\boldsymbol{b} = [\,b_{11} \quad b_{12} \quad b_{22} \quad b_{13} \quad b_{23} \quad b_{33}\,]^T$, where b_{ij} is the generic element of the symmetric matrix

$$\boldsymbol{B} = \boldsymbol{\Omega}^{-T}\boldsymbol{\Omega}^{-1} = \begin{bmatrix} 1/\alpha_x^2 & 0 & -X_0/\alpha_x^2 \\ * & 1/\alpha_y^2 & -Y_0/\alpha_y^2 \\ * & * & 1 + X_0^2/\alpha_x^2 + Y_0^2/\alpha_y^2 \end{bmatrix}. \tag{10.67}$$

By repeating Phase 1 k times, with the same plane placed each time in a different pose, $2k$ equations of the form (10.66) are obtained. These equations, in the case $k \geq 3$, have a unique solution $\gamma\boldsymbol{b}$ defined up to a scaling factor γ. From matrix $\gamma\boldsymbol{B}$, in view of (10.67), the following expressions for the intrinsic parameters can be found:

$$X_0 = -b'_{13}/b'_{11}$$

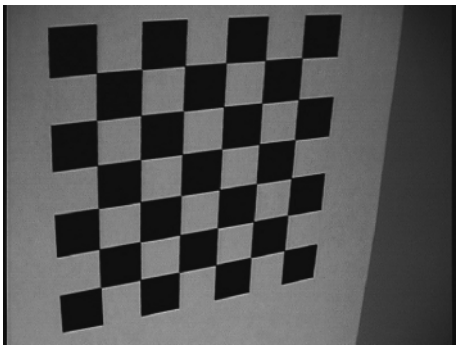

Fig. 10.11. Example of calibration plane

$$
\begin{aligned}
Y_0 &= -b'_{23}/b'_{22} \\
\alpha_x &= \sqrt{\gamma/b'_{11}} \\
\alpha_y &= \sqrt{\gamma/b'_{22}},
\end{aligned}
$$

where $b'_{i,j} = \gamma b_{i,j}$ and γ can be computed as

$$\gamma = b'_{13} + b'_{23} + b'_{33}.$$

Phase 3

Once the matrix $\boldsymbol{\Omega}$ of intrinsic parameters has been evaluated, it is possible to compute $\boldsymbol{H}$ from $\boldsymbol{H}'$, up to a scaling factor ζ, using (10.65) for one of the k poses of the plane. Hence, matrix $\boldsymbol{T}_o^c$ can be computed from $\boldsymbol{H}$ as for the case of the solution of a PnP problem shown in Sect. 10.3.1. Finally, using equations (10.62), (10.63), the extrinsic parameters of the camera can be evaluated.

The method illustrated above is merely conceptual, because it does not provide satisfactory solutions in the presence of measurement noise or lens distortion — especially for the intrinsic parameters; however, the accuracy of the result can be improved using models that take into account the distortion phenomena of the lenses, together with nonlinear optimization techniques.

From the experimental point of view, the calibration method described above requires the use of calibration planes where a certain number of points can be easily detected; also, the position of these points with respect to a suitable reference frame must be known with high accuracy. An example of calibration plane with a chessboard pattern is shown in Fig. 10.11.

Finally, notice that a calibration method can also be set up starting from the solution of a nonplanar PnP problem, using the direct linear transformation method.

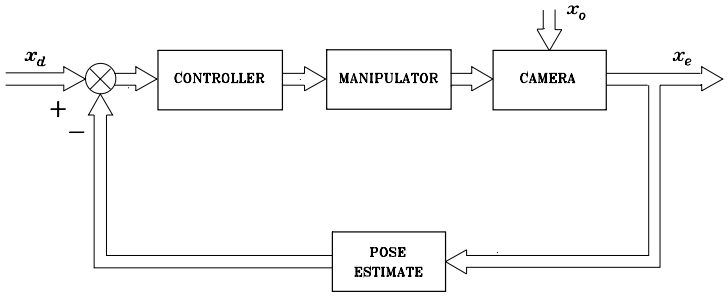

Fig. 10.12. General block scheme of position-based visual servoing

10.6 The Visual Servoing Problem

Visual measurements allow a robot to collect information on the surrounding environment. In the case of robot manipulators, such information is typically used to compute the end-effector pose with respect to an object observed by the camera. The objective of visual servoing is to ensure that the end-effector, on the basis of visual measurements elaborated in real time, reaches and keeps a (constant or time-varying) desired pose with respect to the observed object.

It is worth remarking that the direct measurements provided by the visual system are concerned with feature parameters in the image plane, while the robotic task is defined in the operational space, in terms of the relative pose of the end-effector with respect to the object. This fact naturally leads to considering two kinds of control approaches, illustrated by the block schemes of Figs. 10.12 and 10.13, namely *position-based visual servoing*, also termed *visual servoing in the operational space*, and *image-based visual servoing*, also termed *visual servoing in the image space*. In these schemes, the case of eye-in-hand camera is considered; for eye-to-hand cameras, similar schemes can be adopted.

The *position-based visual servoing* approach is conceptually similar to the operational space control illustrated in Fig. 8.2. The main difference is that feedback is based on the real-time estimation of the pose of the observed object with respect to the camera using visual measurements. Estimation can be performed analytically or using iterative numerical algorithms. Its conceptual advantage regards the possibility of acting directly on operational space variables. Therefore, the control parameters can be selected on the basis of suitable specifications imposed to the time response of the end-effector motion variables, both at steady state and during the transient. The drawback of this approach is that, due to the absence of direct control of the image features, the object may exit from the camera field of view during the transient or as a consequence of planning errors; hence, the feedback loop turns out to be open due to lack of visual measurements and instability may occur.

In the *image-space visual servoing* approach, the control action is computed on the basis of the error defined as the difference between the value of

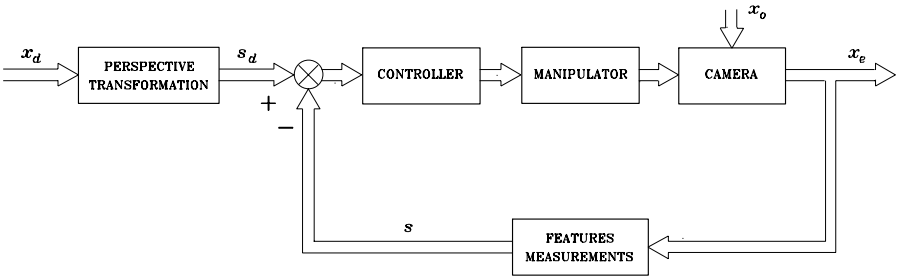

Fig. 10.13. General block scheme of image-based visual servoing

the image feature parameters in the desired configuration — computed using perspective transformation or directly measured with the camera in the desired pose — and the value of the parameters measured with the camera in the current pose. The conceptual advantage of this solution regards the fact that the real-time estimate of the pose of the object with respect to the camera is not required. Moreover, since the control acts directly in the image feature parameters, it is possible to keep the object within the camera field of view during the motion. However, due to the nonlinearity of the mapping between the image feature parameters and the operational space variables, singular configurations may occur, which cause instability or saturation of the control action. Also, the end-effector trajectories cannot be easily predicted in advance and may produce collisions with obstacles or joint limits violation.

To compare the two control strategies it is also worth considering the operating conditions. Of particular importance is the issue of camera calibration. It is easy to understand that position-based visual servoing is more sensitive to camera calibration errors compared to image-based visual servoing. In fact, for the first approach, the presence of uncertainties on calibration parameters, both intrinsic and extrinsic, produces errors on the estimate of operational space variables that may be seen as an external disturbance acting on the feedback path of the control loop, where disturbance rejection capability is low. On the other hand, in the image-based visual servoing approach, the quantities used for the computation of the control action are directly defined in the image plane and measured in pixel units; moreover, the desired value of the feature parameters is measured using the camera. This implies that the uncertainty affecting calibration parameters can be seen as a disturbance acting on the forward path of the control loop, where disturbance rejection capability is high.

A further aspect to analyze concerns knowledge of the geometric model of the object. It is evident that, for position-based visual servoing, the object geometry must be known if only one camera is used, because it is necessary for pose estimation, while it may be unknown when a stereo camera system is

employed. On the other hand, image-based visual servoing does not require, in principle, knowledge of the object geometry, even for mono-camera systems.

On the above premises, in the following, the main position-based and image-based visual servoing schemes are illustrated. For both approaches, the problem of regulation to a constant set-point is presented and the object is assumed to be fixed with respect to the base frame. Without loss of generality, the case of a single calibrated camera, mounted on the manipulator's end-effector, is considered (see Fig. 10.5); moreover, the end-effector frame is chosen so as to coincide with the camera frame.

10.7 Position-based Visual Servoing

In position-based visual servoing schemes, visual measurements are used to estimate in real time the homogeneous transformation matrix $\boldsymbol{T}_o^c$, representing the relative pose of the object frame with respect to the camera frame. From matrix $\boldsymbol{T}_o^c$, the $(m \times 1)$ vector of independent coordinates $\boldsymbol{x}_{c,o}$, defined in (10.31), can be extracted.

Assumed that the object is fixed with respect to the base frame, the position-based visual servoing problem can be formulated by imposing a desired value to the relative pose of the object frame with respect to the camera frame. This quantity can be specified in terms of homogeneous transformation matrix $\boldsymbol{T}_o^d$, where superscript d denotes the desired pose of the camera frame. From this matrix, the $(m \times 1)$ operational space vector $\boldsymbol{x}_{d,o}$ can be extracted.

Matrices $\boldsymbol{T}_o^c$ and $\boldsymbol{T}_o^d$ can be used to obtain the homogeneous transformation matrix

$$\boldsymbol{T}_c^d = \boldsymbol{T}_o^d (\boldsymbol{T}_o^c)^{-1} = \begin{bmatrix} \boldsymbol{R}_c^d & \boldsymbol{o}_{d,c}^d \\ \boldsymbol{0}^T & 1 \end{bmatrix}, \tag{10.68}$$

expressing the position and orientation displacement of the camera frame in the current pose with respect to the desired pose. From this matrix, a suitable error vector in the operational space can be computed, defined as

$$\widetilde{\boldsymbol{x}} = - \begin{bmatrix} \boldsymbol{o}_{d,c}^d \\ \boldsymbol{\phi}_{d,c} \end{bmatrix}, \tag{10.69}$$

where $\boldsymbol{\phi}_{d,c}$ is the vector of the Euler angles extracted from rotation matrix $\boldsymbol{R}_c^d$. Vector $\widetilde{\boldsymbol{x}}$ does not depend on the object pose and represents the error between the desired pose and the current pose of the camera frame. It is worth observing that this vector does not coincide with the difference between $\boldsymbol{x}_{d,o}$ and $\boldsymbol{x}_{c,o}$, but it can be computed from the corresponding homogeneous transformation matrices, using (10.68), (10.69).

The control has to be designed so that the operational space error $\widetilde{\boldsymbol{x}}$ tends to zero asymptotically.

Notice that, for the choice of the set point $\boldsymbol{x}_{d,o}$, the knowledge of the object pose is not required. However, the control objective can be satisfied provided

that the desired pose of the camera frame with respect to the base frame, corresponding to the homogeneous transformation matrix

$$\boldsymbol{T}_d = \boldsymbol{T}_c(\boldsymbol{T}_c^d)^{-1} = \begin{bmatrix} \boldsymbol{R}_d & \boldsymbol{o}_d \\ \boldsymbol{0}^T & 1 \end{bmatrix}, \tag{10.70}$$

belongs to the dexterous workspace of the manipulator. If the object is fixed with respect to the base frame, this matrix is constant.

10.7.1 PD Control with Gravity Compensation

Position-based visual servoing can be implemented using PD control with gravity compensation, suitably modified with respect to that used for motion control.

Computing the time derivative of (10.69), for the position part, gives

$$\dot{\boldsymbol{o}}_{d,c}^d = \dot{\boldsymbol{o}}_c^d - \dot{\boldsymbol{o}}_d^d = \boldsymbol{R}_d^T \dot{\boldsymbol{o}}_c,$$

while, for the orientation part, it gives

$$\dot{\boldsymbol{\phi}}_{d,c} = \boldsymbol{T}^{-1}(\boldsymbol{\phi}_{d,c})\boldsymbol{\omega}_{d,c}^d = \boldsymbol{T}^{-1}(\boldsymbol{\phi}_{d,c})\boldsymbol{R}_d^T \boldsymbol{\omega}_c.$$

To compute the above expressions, equalities $\dot{\boldsymbol{o}}_d^d = \boldsymbol{0}$ and $\boldsymbol{\omega}_d^d = \boldsymbol{0}$ have been taken into account, observing that $\boldsymbol{o}_d$ and $\boldsymbol{R}_d$ are constant. Therefore, $\dot{\tilde{\boldsymbol{x}}}$ has the expression

$$\dot{\tilde{\boldsymbol{x}}} = -\boldsymbol{T}_A^{-1}(\boldsymbol{\phi}_{d,c}) \begin{bmatrix} \boldsymbol{R}_d^T & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}_d^T \end{bmatrix} \boldsymbol{v}_c. \tag{10.71}$$

Since the end-effector frame and the camera frame coincide, the following equality holds:

$$\dot{\tilde{\boldsymbol{x}}} = -\boldsymbol{J}_{A_d}(\boldsymbol{q}, \tilde{\boldsymbol{x}})\dot{\boldsymbol{q}}, \tag{10.72}$$

where the matrix

$$\boldsymbol{J}_{A_d}(\boldsymbol{q}, \tilde{\boldsymbol{x}}) = \boldsymbol{T}_A^{-1}(\boldsymbol{\phi}_{d,c}) \begin{bmatrix} \boldsymbol{R}_d^T & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}_d^T \end{bmatrix} \boldsymbol{J}(\boldsymbol{q}) \tag{10.73}$$

has the meaning of analytic Jacobian of the manipulator in the operational space, as for the Jacobian in (9.14).

Position-based visual servoing of PD type with gravity compensation has the expression

$$\boldsymbol{u} = \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{J}_{A_d}^T(\boldsymbol{q}, \tilde{\boldsymbol{x}})(\boldsymbol{K}_P \tilde{\boldsymbol{x}} - \boldsymbol{K}_D \boldsymbol{J}_{A_d}(\boldsymbol{q}, \tilde{\boldsymbol{x}})\dot{\boldsymbol{q}}), \tag{10.74}$$

analogous to motion control law (8.110), but using a different definition of operational space error. The asymptotic stability of the equilibrium pose corresponding to $\tilde{\boldsymbol{x}} = \boldsymbol{0}$, under the assumption of symmetric and positive definite matrices $\boldsymbol{K}_P$, $\boldsymbol{K}_D$, can be proven using the Lyapunov function

$$\boldsymbol{V}(\dot{\boldsymbol{q}}, \tilde{\boldsymbol{x}}) = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}} + \frac{1}{2}\tilde{\boldsymbol{x}}^T \boldsymbol{K}_P \tilde{\boldsymbol{x}} > 0 \qquad \forall \dot{\boldsymbol{q}}, \tilde{\boldsymbol{x}} \neq \boldsymbol{0},$$

similarly to the case of control law (8.110).

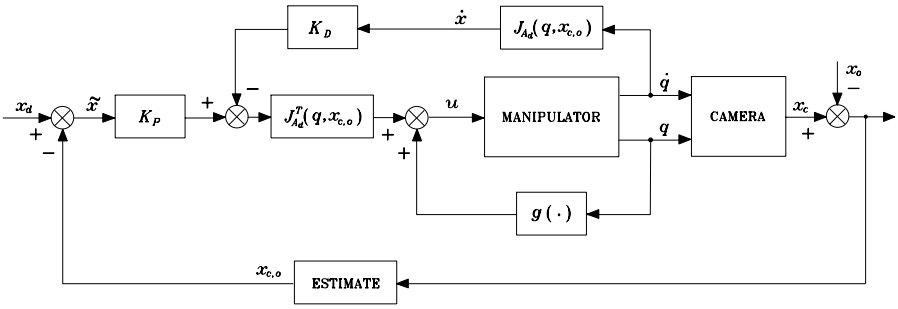

Fig. 10.14. Block scheme of position-based visual servoing of PD type with gravity compensation

Notice that, for the computation of control law (10.74), the estimation of $\boldsymbol{x}_{c,o}$ and the measurements of $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ are required. Moreover, the derivative term can also be chosen as $-\boldsymbol{K}_D\dot{\boldsymbol{q}}$.

The block scheme of position-based visual servoing of PD type with gravity compensation is shown in Fig. 10.14. Notice that the sum block computing error $\tilde{\boldsymbol{x}}$ and that computing the output of the controlled system have a purely conceptual meaning and do not correspond to algebraic sums.

10.7.2 Resolved-velocity Control

The information deriving from visual measurements is computed at a frequency lower or equal to the camera frame rate. This quantity, especially for CCD cameras, is at least of one order of magnitude lower than the typical frequencies used for motion control of robot manipulators. As a consequence, in the digital implementation of control law (10.74), to preserve stability of the closed-loop system, the control gains must be set to values much lower than those typically used for motion control; therefore, the performance of the closed-loop system in terms of speed of convergence and disturbance rejection capability turns out to be poor.

This problem can be avoided assuming that the manipulator is equipped with a high-gain motion controller in the joint space or in the operational space. Neglecting the effects on the tracking errors deriving from manipulator dynamics and disturbances, the controlled manipulator can be considered as an ideal positioning device. This implies that, in the case of joint space motion control, the following equality holds:

$$\boldsymbol{q}(t) \approx \boldsymbol{q}_r(t), \tag{10.75}$$

$\boldsymbol{q}_r(t)$ being the imposed reference trajectory for the joint variables.

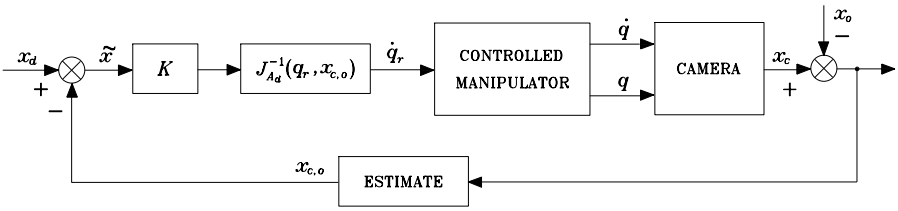

Fig. 10.15. Block scheme of resolved-velocity position-based visual servoing

Therefore, visual servoing can be achieved by computing the trajectory $\boldsymbol{q}_r(t)$ on the basis of visual measurements, so that the operational space tracking error (10.69) goes asymptotically to zero.

To this end, Eq. (10.72) suggests the following choice for the joint space reference velocity:

$$\dot{\boldsymbol{q}}_r = \boldsymbol{J}_{A_d}^{-1}(\boldsymbol{q}_r, \widetilde{\boldsymbol{x}})\boldsymbol{K}\widetilde{\boldsymbol{x}} \tag{10.76}$$

which, replaced in (10.72), by virtue of equality (10.75), yields the linear equation

$$\dot{\widetilde{\boldsymbol{x}}} + \boldsymbol{K}\widetilde{\boldsymbol{x}} = \boldsymbol{0}. \tag{10.77}$$

This equality, for a positive definite matrix $\boldsymbol{K}$, implies that the operational space error tends to zero asymptotically with a convergence of exponential type and speed depending on the eigenvalues of matrix $\boldsymbol{K}$; the larger the eigenvalues, the faster the convergence.

The above scheme is termed *resolved-velocity control* in the operational space, because it is based on the computation of velocity $\dot{\boldsymbol{q}}_r$ from the operational space error. Trajectory $\boldsymbol{q}_r(t)$ is computed from (10.76) via a simple integration.

The block scheme of resolved-velocity position-based visual servoing is reported in Fig. 10.15. Also in this case, the sum block computing the error $\widetilde{\boldsymbol{x}}$ and that computing the output of the scheme have a purely conceptual meaning and do not correspond to algebraic sums.

Notice that the choice of $\boldsymbol{K}$ influences the transient behaviour of the trajectory of the camera frame, which is the solution to the differential equation (10.77). If $\boldsymbol{K}$ is a diagonal matrix with the same gains for the positional part, the origin of the camera frame follows the line segment connecting the initial position to the desired position. On the other hand, the orientation trajectory depends on the particular choice of Euler angles and, more in general, of the orientation error. The possible choices of the orientation error are those presented in Sect. 3.7.3, with the appropriate definition of Jacobian (10.73). The possibility of knowing in advance the trajectory of the camera is important because, during the motion, the object may exit from the camera field of view, making visual measurements unavailable.

10.8 Image-based Visual Servoing

If the object is fixed with respect to the base frame, image-based visual servoing can be formulated by stipulating that the vector of the object feature parameters has a desired constant value $\boldsymbol{s}_d$ corresponding to the desired pose of the camera. Therefore, it is implicitly assumed that a desired pose $\boldsymbol{x}_{d,o}$ exists so that the camera pose belongs to the dexterous workspace of the manipulator and

$$\boldsymbol{s}_d = \boldsymbol{s}(\boldsymbol{x}_{d,o}). \tag{10.78}$$

Moreover, $\boldsymbol{x}_{d,o}$ is supposed to be unique. To this end, the feature parameters can be chosen as the coordinates of n points of the object, with $n \geq 4$ for coplanar points (and no triplets of collinear points) or $n \geq 6$ in case of non-coplanar points. Notice that, if the operational space dimension is $m < 6$, as for the case of SCARA manipulator, a reduced number of points can be used.

The interaction matrix $\boldsymbol{L}_s(\boldsymbol{s}, \boldsymbol{z}_c)$ depends on variables $\boldsymbol{s}$ and $\boldsymbol{z}_c$ with $\boldsymbol{z}_c = [\, z_{\,c,1} \ldots z_{c,n}]^T$, $z_{c,i}$ being the third coordinate of the generic feature point of the object.

It is worth noticing that the task is assigned directly in terms of feature vector $\boldsymbol{s}_d$, while pose $\boldsymbol{x}_{d,o}$ does not need to be known. In fact, $\boldsymbol{s}_d$ can be computed by measuring the feature parameters when the object is in the desired pose with respect to the camera.

The control law must be designed so as to guarantee that the image space error

$$\boldsymbol{e}_s = \boldsymbol{s}_d - \boldsymbol{s} \tag{10.79}$$

tends asymptotically to zero.

10.8.1 PD Control with Gravity Compensation

Image-based visual servoing can be implemented using a PD control with gravity compensation defined on the basis of the image space error.

To this end, consider the following positive definite quadratic form as Lyapunov function candidate:

$$\boldsymbol{V}(\dot{\boldsymbol{q}}, \boldsymbol{e}_s) = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{B}(\boldsymbol{q})\boldsymbol{q} + \frac{1}{2}\boldsymbol{e}_s^T \boldsymbol{K}_{Ps}\boldsymbol{e}_s > 0 \qquad \forall \dot{\boldsymbol{q}}, \boldsymbol{e}_s \neq \boldsymbol{0}, \tag{10.80}$$

with $\boldsymbol{K}_{Ps}$ symmetric and positive definite $(k \times k)$ matrix.

Computing the time derivative of (10.80) and taking into account the expression (8.7) of the joint space dynamic model of the manipulator and property (7.49) yields

$$\dot{\boldsymbol{V}} = -\dot{\boldsymbol{q}}^T \boldsymbol{F}\dot{\boldsymbol{q}} + \dot{\boldsymbol{q}}^T(\boldsymbol{u} - \boldsymbol{g}(\boldsymbol{q})) + \dot{\boldsymbol{e}}_s^T \boldsymbol{K}_{Ps}\boldsymbol{e}_s. \tag{10.81}$$

Since $\dot{\boldsymbol{s}}_d = \boldsymbol{0}$ and the object is fixed with respect to the base frame, the following equality holds:

$$\dot{\boldsymbol{e}}_s = -\dot{\boldsymbol{s}} = -\boldsymbol{J}_L(\boldsymbol{s}, \boldsymbol{z}_c, \boldsymbol{q})\dot{\boldsymbol{q}}, \tag{10.82}$$

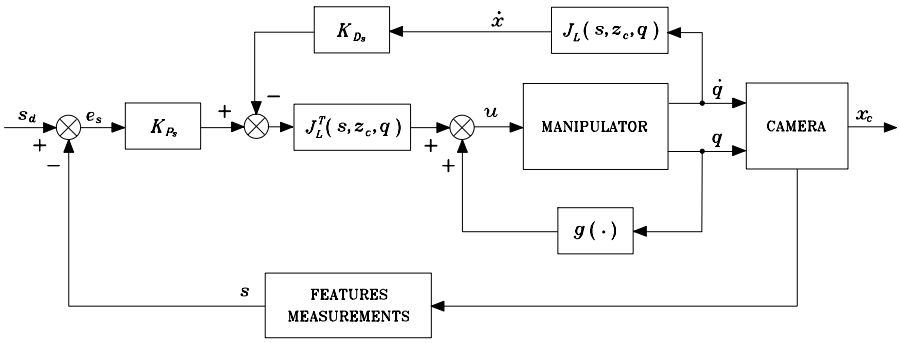

Fig. 10.16. Block scheme of image-based visual servoing of PD type with gravity compensation

where

$$\boldsymbol{J}_L(\boldsymbol{s}, \boldsymbol{z}_c, \boldsymbol{q}) = \boldsymbol{L}_s(\boldsymbol{s}, \boldsymbol{z}_c) \begin{bmatrix} \boldsymbol{R}_c^T & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}_c^T \end{bmatrix} \boldsymbol{J}(\boldsymbol{q}), \tag{10.83}$$

the camera frame and the end-effector frame being coincident.

Therefore, with the choice

$$\boldsymbol{u} = \boldsymbol{g}(\boldsymbol{q}) + \boldsymbol{J}_L^T(\boldsymbol{s}, \boldsymbol{z}_c, \boldsymbol{q}) \left(\boldsymbol{K}_{Ps} \boldsymbol{e}_s - \boldsymbol{K}_{Ds} \boldsymbol{J}_L(\boldsymbol{s}, \boldsymbol{z}_c, \boldsymbol{q}) \dot{\boldsymbol{q}} \right), \tag{10.84}$$

where $\boldsymbol{K}_{Ds}$ is a symmetric and positive definite $(k \times k)$ matrix, Eq. (10.81) becomes

$$\dot{V} = -\dot{\boldsymbol{q}}^T \boldsymbol{F} \dot{\boldsymbol{q}} - \dot{\boldsymbol{q}}^T \boldsymbol{J}_L^T \boldsymbol{K}_{Ds} \boldsymbol{J}_L \dot{\boldsymbol{q}}. \tag{10.85}$$

Control law (10.84) includes a nonlinear compensation action of gravitational forces in the joint space and a linear PD action in the image space. The last term, in view of (10.82), corresponds to a derivative action in the image space and has been added to increase damping. The resulting block scheme is reported in Fig. 10.16.

The direct measurement of $\dot{\boldsymbol{s}}$ would permit the computation of the derivative term as $-\boldsymbol{K}_{Ds} \dot{\boldsymbol{s}}$; this measurement, however, is not available. As an alternative, the derivative term can simply be set as $-\boldsymbol{K}_D \dot{\boldsymbol{q}}$, with $\boldsymbol{K}_D$ symmetric and positive definite $(n \times n)$ matrix.

Equation (10.85) reveals that, for all trajectories of the system, the Lyapunov function decreases until $\dot{\boldsymbol{q}} \neq \boldsymbol{0}$. Therefore the system reaches an equilibrium state, characterized by

$$\boldsymbol{J}_L^T(\boldsymbol{s}, \boldsymbol{z}_c, \boldsymbol{q}) \boldsymbol{K}_{Ps} \boldsymbol{e}_s = \boldsymbol{0}. \tag{10.86}$$

Equations (10.86), (10.83) show that, if the interaction matrix and the geometric Jacobian of the manipulator are full rank, then $\boldsymbol{e}_s = \boldsymbol{0}$, which is the sought result.

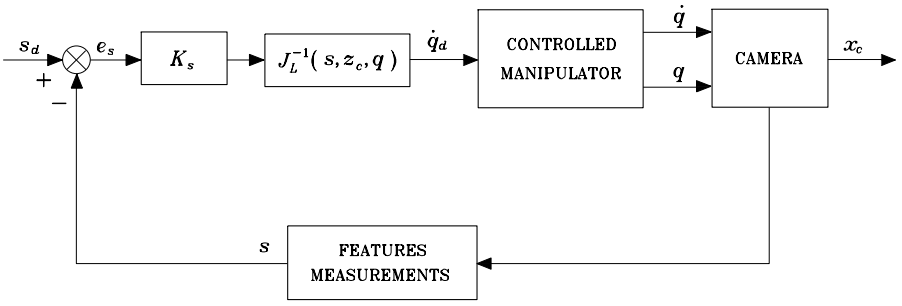

Fig. 10.17. Block scheme of resolved-velocity image-based visual servoing

Notice that control law (10.84) requires not only the measurement of $\boldsymbol{s}$ but also the computation of vector $\boldsymbol{z}_c$ which, in the image-based visual servoing philosophy, should be avoided. In some applications $\boldsymbol{z}_c$ is known with good approximation, as in the case that the relative motion of the camera with respect to the object belongs to a plane. Alternatively, estimated or constant values can be used for $\boldsymbol{z}_c$, as the value in the initial configuration or that in the desired configuration. This is equivalent to using an estimate $\widehat{\boldsymbol{L}}_s$ of the interaction matrix. In such cases, however, the stability proof becomes much more complex.

10.8.2 Resolved-velocity Control

The concept of resolved-velocity control can easily be extended to the image space. In such a case, Eq. (10.82) suggests the following choice of the reference velocity in joint space

$$\dot{\boldsymbol{q}}_r = \boldsymbol{J}_L^{-1}(\boldsymbol{s}, \boldsymbol{z}_c, \boldsymbol{q}_r)\boldsymbol{K}_s\boldsymbol{e}_s, \tag{10.87}$$

under the assumption of invertible matrix $\boldsymbol{J}_L$. This control law, replaced in (10.82), yields the linear equation

$$\dot{\boldsymbol{e}}_s + \boldsymbol{K}_s\boldsymbol{e}_s = \boldsymbol{0}. \tag{10.88}$$

Therefore, if $\boldsymbol{K}_s$ is a positive definite matrix, Eq. (10.88) is asymptotically stable and error $\boldsymbol{e}_s$ tends asymptotically to zero with convergence of exponential type and speed depending on the eigenvalues of matrix $\boldsymbol{K}_s$. The convergence to zero of the image space error $\boldsymbol{e}_s$ ensures the asymptotic convergence of $\boldsymbol{x}_{c,o}$ to the desired pose $\boldsymbol{x}_{d,o}$.

The block scheme of resolved-velocity image-based visual servoing is shown in Fig. 10.17.

Notice that this control scheme requires the computation of the inverse of matrix $\boldsymbol{J}_L$; therefore it is affected by problems related to the singularities of this matrix which, in view of (10.83), are both those of the geometric Jacobian and those of the interaction matrix. The most critical singularities are those

of the interaction matrix, since they depend on the choice of the image feature parameters.

Therefore, it is convenient to compute control law (10.87) in two steps. The first step is the computation of vector

$$\boldsymbol{v}_r^c = \boldsymbol{L}_s^{-1}(\boldsymbol{s}, \boldsymbol{z}_c)\boldsymbol{K}_s\boldsymbol{e}_s. \tag{10.89}$$

The second step is the computation of the joint space reference velocity using the relationship

$$\dot{\boldsymbol{q}}_r = \boldsymbol{J}^{-1}(\boldsymbol{q}) \begin{bmatrix} \boldsymbol{R}_c & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{R}_c \end{bmatrix} \boldsymbol{v}_r^c. \tag{10.90}$$

Far from the kinematic singularities of the manipulator, the problem of the singularities of the interaction matrix can be overcome by using a number k of feature parameters greater than the minimum required m, similarly to the case considered in Sect. 10.3.3. The control law can be modified by using the left pseudo-inverse of interaction matrix $\boldsymbol{L}_s$ in place of the inverse, namely

$$\boldsymbol{v}_r^c = (\boldsymbol{L}_s^T\boldsymbol{L}_s)^{-1}\boldsymbol{L}_s^T\boldsymbol{K}_s\boldsymbol{e}_s \tag{10.91}$$

in place of (10.89). Stability of the closed-loop system with control law (10.90), (10.91) can be shown using the direct Lyapunov method based on the positive definite function

$$V(\boldsymbol{e}_s) = \frac{1}{2}\boldsymbol{e}_s^T\boldsymbol{K}_s\boldsymbol{e}_s > 0 \qquad \forall \boldsymbol{e}_s \neq \boldsymbol{0}.$$

Computing the time derivative of this function and taking into account (10.82), (10.83), (10.90), (10.91), yields

$$\dot{V} = -\boldsymbol{e}_s^T\boldsymbol{K}_s\boldsymbol{L}_s(\boldsymbol{L}_s^T\boldsymbol{L}_s)^{-1}\boldsymbol{L}_s^T\boldsymbol{K}_s\boldsymbol{e}_s$$

which is negative semi-definite because $\mathcal{N}(\boldsymbol{L}_s^T) \neq \emptyset$, $\boldsymbol{L}_s^T$ being a matrix with more columns than rows. Therefore, the closed-loop system is stable but not asymptotically stable. This implies that the error is bounded, but in some cases the system may reach an equilibrium with $\boldsymbol{e}_s \neq \boldsymbol{0}$ and $\boldsymbol{K}_s\boldsymbol{e}_s \in \mathcal{N}(\boldsymbol{L}_s^T)$.

Another problem connected with the implementation of control law (10.89) or (10.91) and (10.90) depends on the fact that the computation of interaction matrix $\boldsymbol{L}_s$ requires knowledge of $\boldsymbol{z}_c$. Similar to Sect. 10.8.1, this problem can be solved by using an estimate of matrix $\widehat{\boldsymbol{L}}_s^{-1}$ (or of its pseudo-inverse). In this case, the Lypapunov method can be used to prove that the control scheme remains stable provided that matrix $\boldsymbol{L}_s\widehat{\boldsymbol{L}}_s^{-1}$ is positive definite. Notice that $\boldsymbol{z}_c$ is the only information depending on object geometry. Therefore, it can also be seen that image-based visual servoing, in the case that only one camera is used, does not require exact knowledge of object geometry.

The choice of the elements of matrix $\boldsymbol{K}_s$ influences the trajectories of the feature parameters, which are solution to differential equation (10.88). In the

case of feature points, if a diagonal matrix $\boldsymbol{K}_s$ with equal elements is set, the projections of these points on the image plane will follow line segments. The corresponding camera motion, however, cannot be easily predicted, because of the nonlinearity of the mapping between image plane variables and operational space variables.

10.9 Comparison Among Various Control Schemes

In order to make a comparison between the various control schemes presented, consider the SCARA manipulator of Example 10.3 and the planar object of Example 10.1. The base frame of the SCARA manipulator of Fig. 2.36 is set with the origin at the intersection of the axis of joint 1 with the horizontal plane containing the origin of the end-effector frame when $d_3 = 0$, d_3 being the prismatic joint variable; the axis z of the base frame points downward. The operational space, of dimension $m = 4$, is characterized by vector $\boldsymbol{x}_{c,o}$ in (10.33).

With reference to the dynamic model of Problem 7.3, the same data of Example 7.2 are considered; in addition, $m_{\ell_3} = 2\,\mathrm{kg}$ and $I_{\ell_4} = 1\,\mathrm{kg{\cdot}m^2}$, while the contribution of the motors of the last two links are neglected.

For position-based visual servoing schemes, the real-time estimation of vector $\boldsymbol{x}_{c,o}$ from a suitable feature vector is required. To this end, the algorithm of Sect. 10.3.3 can be used, based on the inverse of the image Jacobian. This is a classical visual tracking application that can be accomplished using only two feature points, because the corresponding Jacobian $\boldsymbol{J}_{A_s}$ is a (4×4) matrix. The selected points are P_1 and P_2 of the object of Fig. 10.6.

The same points can also be used for image-based visual servoing, because the corresponding Jacobian $\boldsymbol{J}_L$ is a (4×4) matrix.

It is assumed that at time $t = 0$ the pose of the camera frame, with respect to the base frame, is defined by the operational space vector

$$\boldsymbol{x}_c(0) = [\,1 \quad 1 \quad 1 \quad \pi/4\,]^T\,\mathrm{m}$$

and the pose of the object frame, with respect to the camera frame, is defined by the operational space vector

$$\boldsymbol{x}_{c,o}(0) = [\,0 \quad 0 \quad 0.5 \quad 0\,]^T\,\mathrm{m}.$$

The desired pose of the object frame with respect to the camera frame is defined by vector

$$\boldsymbol{x}_{d,o} = [\,-0.1 \quad 0.1 \quad 0.6 \quad -\pi/3\,]^T\,\mathrm{m}.$$

This quantity is assumed as the initial value of the pose estimation algorithm used by position-based visual servoing schemes.

For image-based visual servoing schemes, the desired value of the feature parameters of points P_1 and P_2 of the object, in the desired pose $\boldsymbol{x}_{d,o}$, is

$$\boldsymbol{s}_d = [\, -0.1667 \quad 0.1667 \quad -0.0833 \quad 0.0223 \,]^T .$$

For all the schemes, a discrete-time implementation of the controller with sampling time of 0.04 s has been adopted, corresponding to a 25 Hz frequency. This value coincides with the minimum frame rate of analog cameras and allows the use of visual measurements also in the worst case.

In the numerical simulations, the following control schemes have been utilized:

A. Position-based visual servoing of PD type with gravity compensation with the following data:

$$\boldsymbol{K}_P = \text{diag}\{500, 500, 10, 10\}$$
$$\boldsymbol{K}_D = \text{diag}\{500, 500, 10, 10\}.$$

B. Resolved-velocity position-based visual servoing with the following data:

$$\boldsymbol{K} = \text{diag}\{1, 1, 1, 2\},$$

corresponding to a time constant of 1 s for the three position variables and of 0.5 s for the orientation variable.

C. Image-based visual servoing of PD type with gravity compensation with the following data:

$$\boldsymbol{K}_{Ps} = 300\boldsymbol{I}_4 \qquad \boldsymbol{K}_{Ds} = 330\boldsymbol{I}_4.$$

D. Resolved-velocity image-based visual servoing with the following data:

$$\boldsymbol{K}_s = \boldsymbol{I}_4,$$

corresponding to a time constant of 1 s for the feature parameters.

For the simulation of resolved-velocity control schemes, the dynamics of the velocity controlled manipulator has been neglected. Therefore, a pure kinematic model has been considered, based on the analytic Jacobian of the manipulator.

For position-based control schemes, the pose estimation algorithm based on the inverse of the image Jacobian has been adopted, with integration step $\Delta t = 1$ ms and gain $\boldsymbol{K}_s = 160\boldsymbol{I}_4$. As shown in Example 10.4, this implies that the algorithm converges in a time of about 0.03 s, which is lower than the sampling time of the control, as required for a correct operation of position-based control.

For image-based control schemes, matrix $\boldsymbol{L}_s(\boldsymbol{s}, \boldsymbol{z}_c)$ has been approximated with matrix $\widehat{\boldsymbol{L}}_s = \boldsymbol{L}_s(\boldsymbol{s}, z_d)$, where z_d is the third component of vector $\boldsymbol{x}_{d,o}$.

The parameters of the various controllers have been chosen in such a way as to show the particular features of the different control laws and, at the same time, to allow a significant comparison of the performance of each scheme in response to congruent control actions. In particular, it can be observed what follows:

- The gains in schemes **A** and **C** have been tuned in simulation so as to obtain transient behaviors similar to those of schemes **B** and **D**.
- In control scheme **B** the gains of the position variables have been intentionally chosen equal to one another but different from the gain of the orientation variable to show that a desired dynamics can be imposed to each operational space variable.
- In control scheme **D** the gains have all been chosen equal to one another, since imposing different dynamics to different coordinates of the projections of the feature points on the image plane is not significant.

The results obtained with the various control schemes are illustrated in Figs. 10.18–10.25 in terms of:

- The time history of position and orientation of the camera frame with respect to the base frame and corresponding desired values (represented with dashed lines).
- The time history of feature parameters and corresponding desired values (represented with dashed lines).
- The path of feature points projections on the camera image plane, from initial positions (marked with crosses) to final positions (marked with circles).

Regarding performance of the various control schemes, the following considerations can be drawn from the obtained results.

In principle, if position-based visual servoing is adopted, a desired transient behaviour can be assigned to the operational space variables. This is only partially true for control scheme **A**, because the dynamics of the closed-loop system, for PD control with gravity compensation, is nonlinear and coupled. Therefore, the transient behaviour shown in Fig. 10.18 may be different if the manipulator starts from a different initial pose or has to reach a different desired pose. Vice versa, for control scheme **B**, the time history of operational space variables reported in Fig. 10.20 shows transient behaviours of exponential type, whose characteristics depend only on the choice of matrix $\boldsymbol{K}$.

For both schemes **A** and **B**, the trajectories of the projections of the feature points and the corresponding paths in the image plane (Figs. 10.19 and 10.21 respectively) have evolutions that cannot be predicted in advance. This implies that, although the feature points projections are inside the image plane both in the initial and in the desired configuration, they may exit from the image plane during the transient, thus causing problems of convergence to the controlled system.

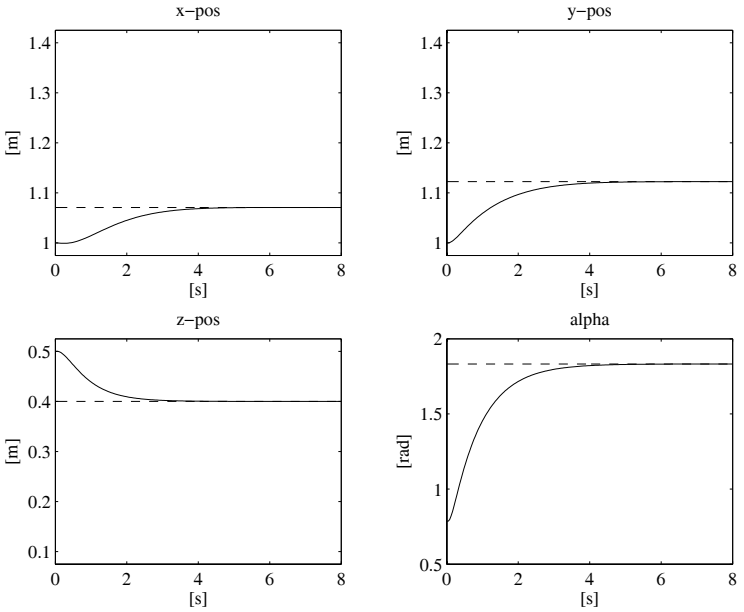

Fig. 10.18. Time history of camera frame position and orientation with control **A**

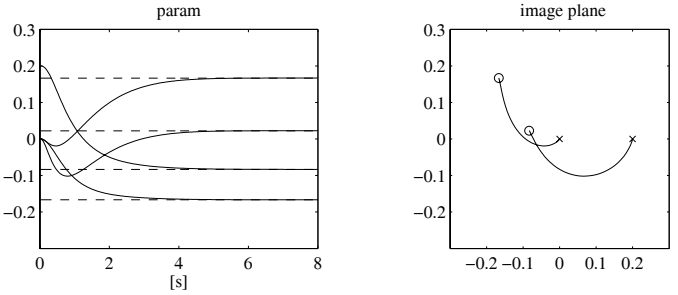

Fig. 10.19. Time history of feature parameters and corresponding path of feature points projections on image plane with control **A**

If image-based control is adopted, a desired transient behaviour can be assigned to the time histories of feature parameters and not to the operational space variables, in a dual fashion with respect to position-based control. This is confirmed by the results shown in Figs. 10.22–10.25, relative to control schemes **C** and **D**, respectively. In detail, especially in the case of control **C**, the time histories of the operational space variables are quite different from those reported in Figs. 10.18 and 10.20, despite the same initial and final configuration and a similar transient duration. This implies that the camera path, being unpredictable, may lead to joint limits violation or to collision of

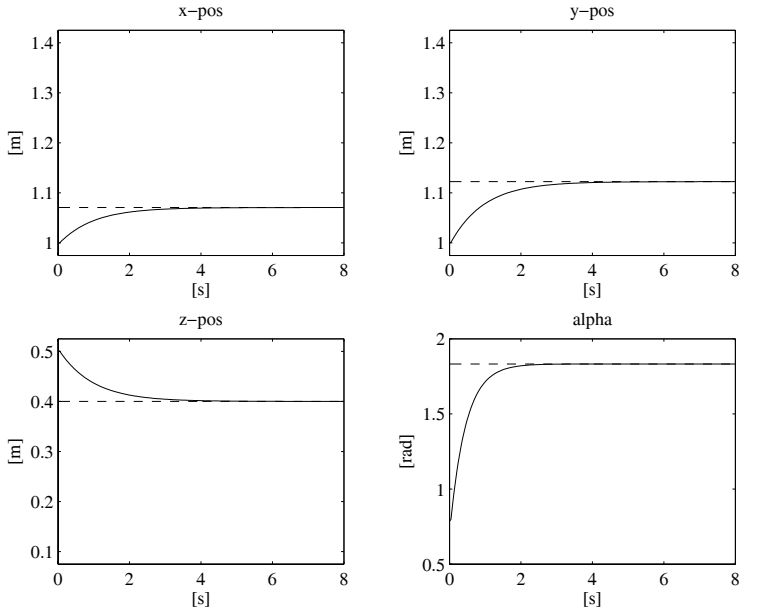

Fig. 10.20. Time history of camera frame position and orientation with control **B**

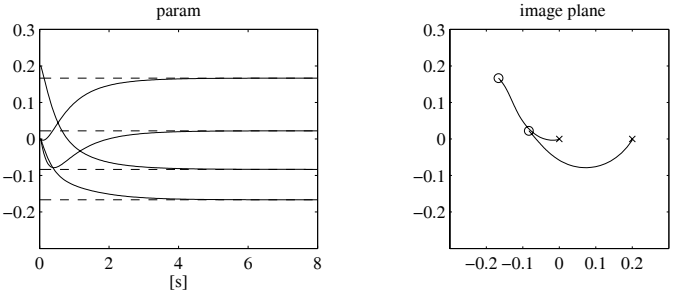

Fig. 10.21. Time history of feature parameters and corresponding path of feature points projections on image plane with control **B**

the manipulator with an obstacle. In the specific case of control scheme **C**, a 300% overshoot is present on the z component of the camera trajectory (see Fig. 10.22), which corresponds to a camera retreat movement with respect to the object of amplitude much higher than the distance reached at the end of the transient. The overshoot on the z component is present also for control scheme **D**, but is 'only' of 50% (see Fig. 10.24).

Notice that, for control scheme **C**, the presence of large displacements on some operational space variables does not correspond to significant deviations of the feature parameters with respect to their final values during the transient

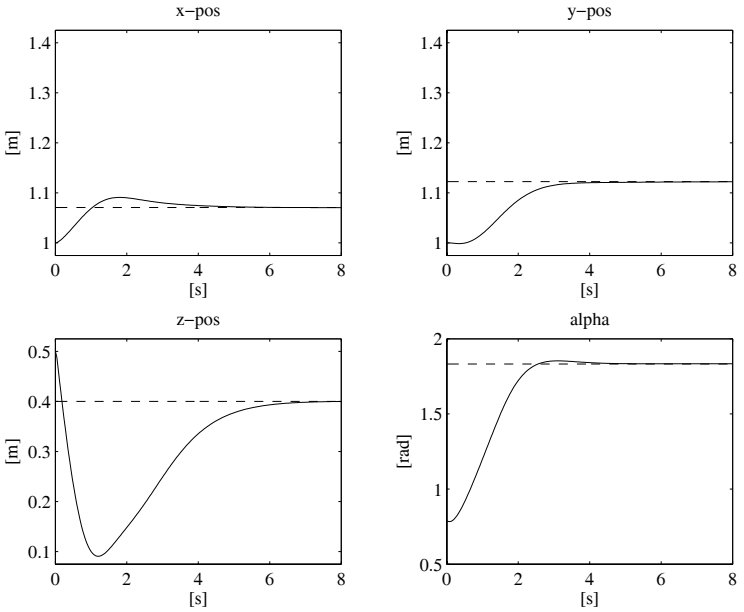

Fig. 10.22. Time history of camera frame position and orientation with control **C**

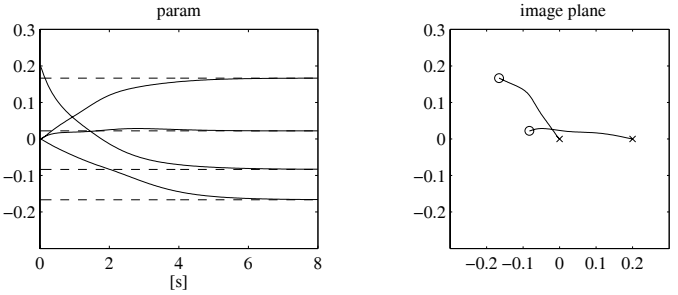

Fig. 10.23. Time history of feature parameters and corresponding path of feature points projections on image plane with control **C**

(see Fig. 10.23). Indeed, the paths of the feature points projections do not deviate much from the line segments connecting these points.

Figure 10.25, relative to control scheme **D**, reveals that the trajectories of the feature parameters are of exponential type. In this case, the transient behaviour depends only on matrix $\boldsymbol{K}_s$; choosing a diagonal matrix with equal elements implies that the paths of the feature points projections are linear. In the case at hand, in view of the approximation $\boldsymbol{L}_s(\boldsymbol{s}, \boldsymbol{z}_c) \approx \boldsymbol{L}_s(\boldsymbol{s}, z_d)$, the paths of the feature points projections shown in Fig. 10.25 are not perfectly linear.

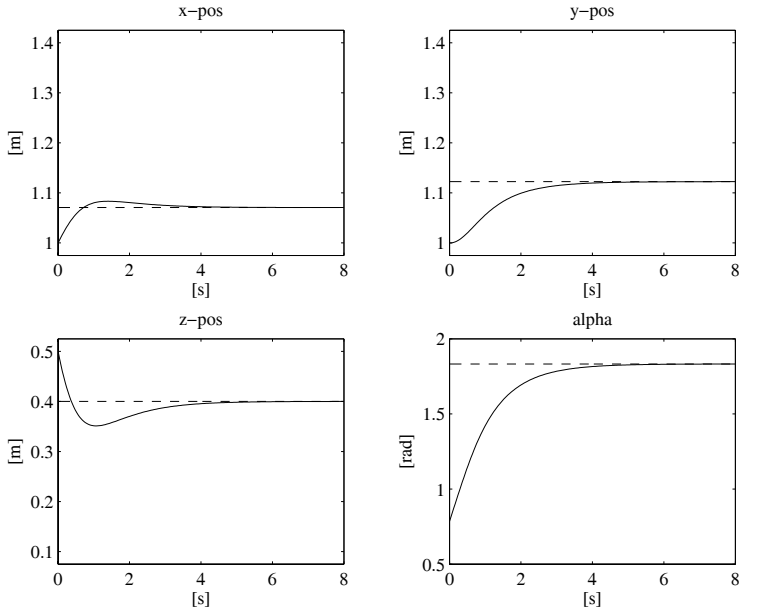

Fig. 10.24. Time history of camera frame position and orientation with control **D**

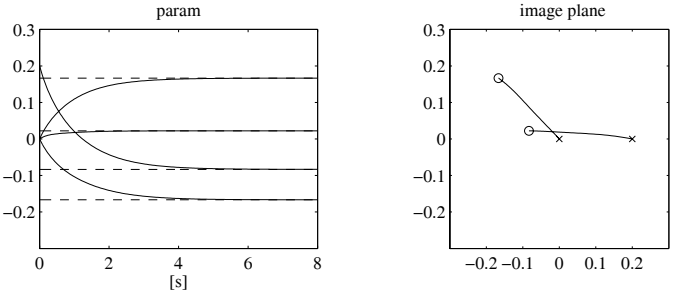

Fig. 10.25. Time history of feature parameters and corresponding path of feature points projections on image plane with control **D**

To conclude, Fig. 10.26 shows the paths of the origin of the camera frame obtained using the four control schemes. It can be observed that, with control scheme **B**, a perfectly linear path is obtained, thanks to the choice of a diagonal gain matrix $\boldsymbol{K}_s$ with equal weights for the positional part. Using control scheme **A**, the path is almost linear, because, unlike case **B**, this type of control does not guarantee a decoupled dynamics for each operational space variable. Vice versa, using control schemes **C** and **D**, the path of the origin of the camera frame is far from being linear. In both cases, the phenomenon of camera retreat with respect to the object can be observed. To this end, notice

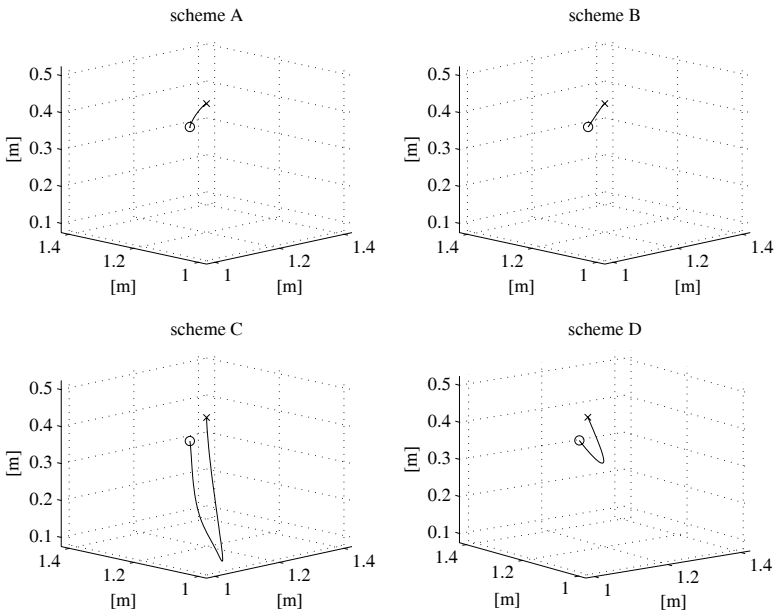

Fig. 10.26. Paths of camera frame origin in Cartesian space with the four control schemes

the axis z of the base frame of the SCARA manipulator and the axis z_c of the camera frame are aligned to the vertical direction and point downward; therefore, with respect to the reference frames of Fig. 10.26, the object is on the top and the camera points upward.

The phenomenon of camera retreat appears whenever the camera is required to perform large rotations about the optical axis; this phenomenon can be intuitively explained through a simple example. Assume that a pure rotation of the camera about the axis z_c is required and control scheme **D** is adopted. Therefore, visual servoing imposes that the feature points projections on the image plane follow rectilinear paths from the initial to the desired positions, whereas simple camera rotation would have required circular paths. The constraint on the path in the image plane implies that, during rotation, the origin of the camera frame must move, with respect to the object, first backward and then forward to reach again, asymptotically, the initial position. It can be shown that, if the desired rotation tends to π, then the distance of the camera from the object tends to ∞ and the system becomes unstable.

10.10 Hybrid Visual Servoing

An approach which combines the benefits of position-based and image-based visual servoing is *hybrid visual servoing*. The name stems from the fact that

the control error is defined in the operational space for some components and in the image space for the others. This implies that a desired motion can be specified, at least partially, in the operational space so that the camera trajectory during visual servoing can be predicted in advance for some components. On the other hand, the presence of error components in the image space helps keep the image features in the camera field of view, which is a difficult task in position-based approaches.

Hybrid visual servoing requires the estimation of some operational space variables. Assume that the object has a planar surface where at least four feature points, and no triplets of collinear points, can be selected. Using the coordinates of these points in the camera image plane, both in the current and in the desired pose of the camera frame, it is possible to compute the planar homography $\boldsymbol{H}$ as described in Sect. 10.4.4. Notice that, for this computation, knowledge of the current and the desired camera pose is not required, provided that the feature vectors $\boldsymbol{s}$ and $\boldsymbol{s}_d$ are known.

In view of (10.59), assuming that Frame 1 coincides with the camera frame in the current pose and Frame 2 coincides with the camera frame in the desired pose, the following equality holds:

$$\boldsymbol{H} = \boldsymbol{R}_d^c + \frac{1}{d_d}\boldsymbol{o}_{c,d}^c \boldsymbol{n}^{dT},$$

where $\boldsymbol{R}_d^c$ is the rotation matrix between the desired orientation and the current orientation of the camera frame, $\boldsymbol{o}_{c,d}^c$ is the position vector of the origin of the camera frame in the desired pose with respect to the current pose, $\boldsymbol{n}^d$ is the unit vector normal to the plane containing the feature points, and d_d is the distance between this plane and the origin of the camera frame in the desired pose. The quantities $\boldsymbol{R}_d^c$, $\boldsymbol{n}^d$, $(1/d_d)\boldsymbol{o}_{c,d}^c$, in the current camera pose, can be computed at each sampling time from matrix $\boldsymbol{H}$.

Adopting a resolved-velocity approach, the control objective consists of computing the reference absolute velocity of the camera frame

$$\boldsymbol{v}_r^c = \begin{bmatrix} \boldsymbol{\nu}_r^c \\ \boldsymbol{\omega}_r^c \end{bmatrix}$$

from a suitably defined error vector.

To this end, the orientation error between the desired and the current camera pose can be computed from matrix $\boldsymbol{R}_d^c$, as for position-based visual servoing. If $\boldsymbol{\phi}_{c,d}$ denotes the vector of the Euler angles extracted from $\boldsymbol{R}_d^c$, the control vector $\boldsymbol{\omega}_r^c$ can be chosen as

$$\boldsymbol{\omega}_r^c = -\boldsymbol{T}(\boldsymbol{\phi}_{c,d})\boldsymbol{K}_o\boldsymbol{\phi}_{c,d}, \tag{10.92}$$

where $\boldsymbol{K}_o$ is a (3×3) matrix. With this choice, the equation of the orientation error has the form

$$\dot{\boldsymbol{\phi}}_{c,d} + \boldsymbol{K}_o\boldsymbol{\phi}_{c,d} = \boldsymbol{0}. \tag{10.93}$$

Equation (10.93), if $\boldsymbol{K}_o$ is a symmetric and positive definite matrix, implies that the orientation error tends to zero asymptotically with convergence of exponential type and speed depending on the eigenvalues of matrix $\boldsymbol{K}_o$.

The control vector $\boldsymbol{\nu}_r^c$ should be selected so that the positional part of the error between the desired and the current camera pose converges to zero. The position error could be defined as the difference of the coordinates of a point of the object in the desired camera frame $\boldsymbol{r}_d^c = [\, x_d \quad y_d \quad z_d \,]^T$ and those in the current camera frame $\boldsymbol{r}_c^c = [\, x_c \quad y_c \quad z_c \,]^T$, namely $\boldsymbol{r}_d^c - \boldsymbol{r}_c^c$. These coordinates, however, cannot be directly measured, unlike the corresponding coordinates in the image plane, defining the feature vectors $\boldsymbol{s}_{p,d} = [\, X_d \quad Y_d \,]^T = [\, x_d/z_d \quad y_d/z_d \,]^T$ and $\boldsymbol{s}_p = [\, X \quad Y \,]^T = [\, x_c/z_c \quad y_c/z_c \,]^T$.

The information deriving from the computation of homography $\boldsymbol{H}$ can be used to rewrite the ratio

$$\rho_z = z_c/z_d$$

in terms of known or measurable quantities in the form

$$\rho_z = \frac{d_c}{d_d} \frac{\boldsymbol{n}^{dT} \widetilde{\boldsymbol{s}}_{p,d}}{\boldsymbol{n}^{cT} \widetilde{\boldsymbol{s}}_p} \tag{10.94}$$

with

$$\frac{d_c}{d_d} = 1 + \boldsymbol{n}^{cT} \frac{\boldsymbol{o}_{c,d}^c}{d_d} = \det(\boldsymbol{H}), \tag{10.95}$$

and $\boldsymbol{n}^c = \boldsymbol{R}_d^c \boldsymbol{n}^d$, where vectors $\widetilde{\boldsymbol{s}}_p$ and $\widetilde{\boldsymbol{s}}_{p,d}$ denote the representations in homogeneous coordinates of $\boldsymbol{s}_p$ and $\boldsymbol{s}_{p,d}$, respectively (see Problem 10.12).

The position error, expressed in terms of known or measurable quantities, can be defined as

$$\boldsymbol{e}_p(\boldsymbol{r}_d^c, \boldsymbol{r}_c^c) = \begin{bmatrix} X_d - X \\ Y_d - Y \\ \ln \rho_z \end{bmatrix}.$$

Notice that, in view of (5.44), convergence to zero of $\boldsymbol{e}_p$ implies convergence to zero of $\boldsymbol{r}_d^c - \boldsymbol{r}_c^c$ and vice versa.

Computing the time derivative of $\boldsymbol{e}_p$ yields

$$\dot{\boldsymbol{e}}_p = \frac{\partial \boldsymbol{e}_p(\boldsymbol{r}_c^c)}{\partial \boldsymbol{r}_c^c} \dot{\boldsymbol{r}}_c^c,$$

$\boldsymbol{r}_d^c$ being constant. By taking into account (10.26) and the decomposition

$$\boldsymbol{v}_c^c = \begin{bmatrix} \boldsymbol{\nu}_c^c \\ \boldsymbol{\omega}_c^c \end{bmatrix}$$

with $\boldsymbol{\nu}_c^c = \boldsymbol{R}_c^T \dot{\boldsymbol{o}}_c$, the above expression can be rewritten in the form

$$\dot{\boldsymbol{e}}_p = -\boldsymbol{J}_p \boldsymbol{\nu}_c^c - \boldsymbol{J}_o \boldsymbol{\omega}_c^c, \tag{10.96}$$

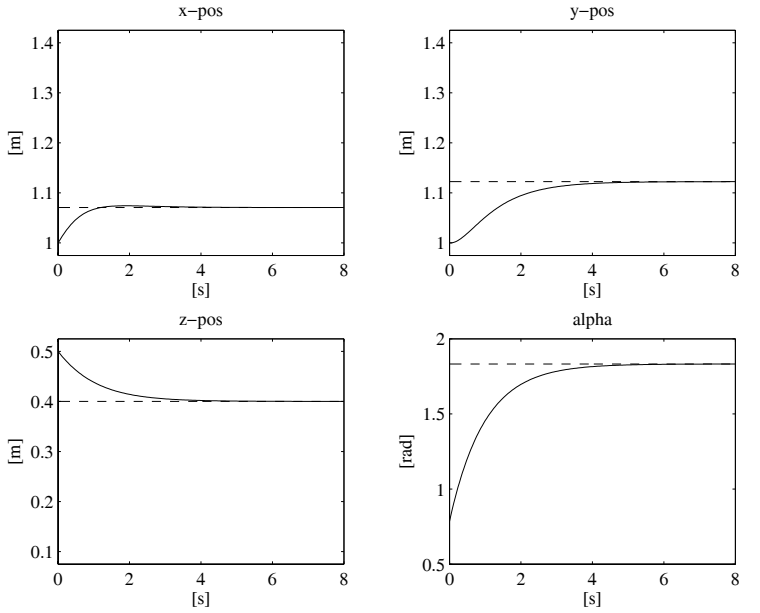

Fig. 10.27. Time history of position and orientation of camera frame with hybrid visual servoing

with (see Problem 10.13)

$$\boldsymbol{J}_p = \frac{1}{z_d \rho_z} \begin{bmatrix} -1 & 0 & X \\ 0 & -1 & Y \\ 0 & 0 & -1 \end{bmatrix}$$

and

$$\boldsymbol{J}_o = \begin{bmatrix} XY & -1 - X^2 & Y \\ 1 + Y^2 & -XY & -X \\ -Y & X & 0 \end{bmatrix}.$$

Equation (10.96) suggests the following choice of control vector $\boldsymbol{\nu}_r^c$

$$\boldsymbol{\nu}_r^c = \boldsymbol{J}_p^{-1}(\boldsymbol{K}_p \boldsymbol{e}_p - \boldsymbol{J}_o \boldsymbol{\omega}_r^c), \tag{10.97}$$

$\boldsymbol{J}_p$ being a nonsingular matrix.

Notice that, for the computation of $\boldsymbol{J}_p^{-1}$, knowledge of the constant quantity z_d is required.

If z_d is known, control law (10.97), in view of assumptions $\dot{\boldsymbol{o}}_c^c \approx \boldsymbol{\nu}_r^c$ and $\boldsymbol{\omega}_c^c \approx \boldsymbol{\omega}_r^c$, yields the following error equation:

$$\dot{\boldsymbol{e}}_p + \boldsymbol{K}_p \boldsymbol{e}_p = \boldsymbol{0},$$

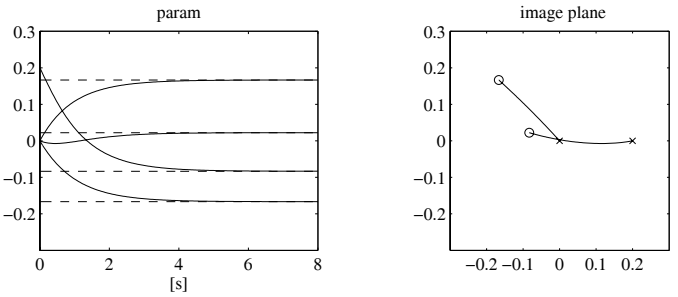

Fig. 10.28. Time history of feature parameters and corresponding path of feature points projections on image plane with hybrid visual servoing

which implies the exponential convergence of $\boldsymbol{e}_p$ to zero, provided that $\boldsymbol{K}_p$ is a positive definite matrix.

If z_d is not known, an estimate $\widehat{z}_d$ can be adopted. Therefore, in control law (10.97), matrix $\boldsymbol{J}_p^{-1}$ can be replaced by an estimate $\widehat{\boldsymbol{J}}_p^{-1}$. In view of the equality

$$\widehat{\boldsymbol{J}}_p^{-1} = \frac{\widehat{z}_d}{z_d}\boldsymbol{J}_p^{-1},$$

the following error equation is obtained:

$$\dot{\boldsymbol{e}}_p + \frac{\widehat{z}_d}{z_d}\boldsymbol{K}_p\boldsymbol{e}_p = \left(1 - \frac{\widehat{z}_d}{z_d}\right)\boldsymbol{J}_o\boldsymbol{\omega}_r^c.$$

This equation shows that the use of an estimate $\widehat{z}_d$ in place of the true value z_d implies a simple gain scaling in the error equation, and asymptotic stability is preserved. Moreover, due to the presence in the right-hand side of the above equation of a term depending on $\boldsymbol{\omega}_r^c$, the time history of $\boldsymbol{e}_p$ is influenced by the orientation error, which evolves according to (10.93).

Example 10.6

For the SCARA manipulator and the task of Sect. 10.9, consider the hybrid visual servoing law with gains

$$\boldsymbol{K}_p = \boldsymbol{I}_3 \quad k_o = 2,$$

and compute the positional part of the error with respect to point P_1. The planar homography and the corresponding parameters are estimated as in Example 10.5, using four points. The results are reported in Figs. 10.27 and 10.28, in terms of the same variables shown in Sect. 10.9. Notice that the time histories of the variables in the operational space of Fig. 10.27 are quite similar to that obtained with resolved-velocity position-based visual servoing (Fig. 10.20). On the other hand, the time histories of the feature parameters of Fig. 10.28 are substantially similar to those obtained with resolved-velocity image-based visual servoing (Fig. 10.25) except for

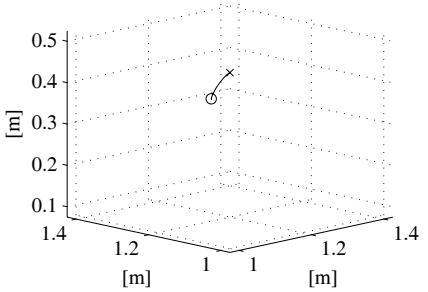

Fig. 10.29. Path of camera frame origin in Cartesian space with hybrid visual servoing

the fact that the path in the image plane of the projection of point P_1 is perfectly linear, as imposed by the control. The corresponding path of the camera frame origin, reported in Fig. 10.29, shows a substantial improvement with respect to those obtained with the image-based visual servoing schemes of Fig. 10.26.

The method illustrated above is only one of the possible visual servoing approaches based on the computation of the planar homography and on its decomposition. It is worth pointing out that knowledge of $(1/d_d)\boldsymbol{o}_{c,d}^c$ and $\boldsymbol{R}_d^c$ allows the computation of the operational space error (10.69), up to a scaling factor for the positional part; therefore, it is also possible to use the position-based visual servoing schemes presented in Sect. 10.7. On the other hand, in hybrid visual servoing approaches, different choices are possible for the error components depending on feature parameters as well as for those depending on the operational space variables.

Bibliography

The literature dealing with computational vision is quite extensive and various. Image processing is treated, e.g., in [93], while geometrical issues are considered in [75] and [146]. The concept of interaction matrix was originally proposed in [239] under a different name, while the actual name was introduced in [71]. The pose estimation algorithm based on the inverse of the image Jacobian is also termed virtual visual servoing [47]. Position-based visual servoing was proposed in [239] and, more recently, in [244] and [134]. Several papers dealing with image-based visual servoing can be found, starting from early works of [239] and [79]. A rigorous stability analysis is reported in [108] for PD control with gravity compensation and in [71] for resolved-velocity control. Hybrid visual servoing, presented in [148], is only one of advanced control schemes based on decoupling of the camera DOFs, e.g., the partitioned approach [49] and the control based on image moments [35]. Finally,

visual servoing based on stereo vision is considered in [92]. An interesting review of the state of the art of visual servoing until mid 1990s is presented in [103], while a more recent review can be found in [36].

Problems

10.1. Derive Eq. (10.2).

10.2. Show that a non-null solution to (10.10) is the right eigenvector corresponding to the null singular value of matrix $\boldsymbol{A}$.

10.3. Show that (10.12) is the matrix which minimizes Frobenius norm (10.11) with the constraint that $\boldsymbol{R}_o^c$ is a rotation matrix, in the case $\sigma > 0$. [*Hint*: consider that $\mathrm{Tr}(\boldsymbol{R}_o^{cT}\boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T) = \mathrm{Tr}(\boldsymbol{V}^T\boldsymbol{R}_o^{cT}\boldsymbol{U}\boldsymbol{\Sigma})$; moreover, the absolute values of the diagonal elements of matrix $\boldsymbol{V}^T\boldsymbol{R}_o^{cT}\boldsymbol{U}$ are lower or equal to 1.]

10.4. Consider the SCARA manipulator of Example 10.3. Perform a computer implementation of the pose estimation algorithm based on the inverse of the image Jacobian considering the points P_1 and P_2 of the object of Example 10.1. Compute the homogeneous transformation matrix corresponding to the feature vector $\boldsymbol{s} = [\,-0.1667 \quad 0.1667 \quad -0.0833 \quad 0.0223\,]^T$. Assume that, in the initial pose, the axes of the camera frame are parallel to those of the object frame and the origin is at a distance of 0.5 m along the vertical axis.

10.5. Solve the previous problem using the feature parameters of the line segment P_1P_2.

10.6. Show that the value of $\boldsymbol{o}$ which minimizes function (10.55) has expression (10.56). [*Hint*: Use the equalities $\boldsymbol{p}_i = \bar{\boldsymbol{p}}_i + \bar{\boldsymbol{p}}$ and $\boldsymbol{p}'_i = \bar{\boldsymbol{p}}'_i + \bar{\boldsymbol{p}}'$ in (10.55), and the properties $\|\boldsymbol{a}+\boldsymbol{b}\|^2 = \|\boldsymbol{a}\|^2+2\boldsymbol{a}^T\boldsymbol{b}+\|\boldsymbol{b}\|^2$ and $\sum_{i=1}^n \bar{\boldsymbol{p}}_i = \sum_{i=1}^n \bar{\boldsymbol{p}}'_i = \boldsymbol{0}$.]

10.7. Show that the matrix $\boldsymbol{R}$ which minimizes Frobenius norm (10.57) is the matrix which maximizes the trace of $\boldsymbol{R}\boldsymbol{K}$.

10.8. For the SCARA manipulator of Example 10.3, design a position-based visual servoing scheme of PD type with gravity compensation using the measurement of the feature parameters of line segment P_1P_2 of Example 10.1. Perform a computer simulation in the same operating conditions of Sect. 10.9 and compare the results.

10.9. For the SCARA manipulator of Example 10.3, design a resolved-velocity position-based visual servoing scheme using the measurement of the feature parameters of line segment P_1P_2 of Example 10.1. Perform a computer simulation in the same operating conditions of Sect. 10.9 and compare the results.

10.10. For the SCARA manipulator of Example 10.3, design an image-based visual servoing scheme of PD type with gravity compensation using the measurement of the feature parameters of the line segment P_1P_2 of Example 10.1. Perform a computer simulation in the same operating conditions of Sect. 10.9 and compare the results.

10.11. For the SCARA manipulator of Example 10.3, design a resolved-velocity image-based visual servoing scheme using the measurement of the feature parameters of line segment P_1P_2 of Example 10.1. Perform a computer simulation in the same operating conditions of Sect. 10.9 and compare the results.

10.12. Derive the expressions (10.94), (10.95).

10.13. Derive the expressions of $\boldsymbol{J}_p$ and $\boldsymbol{J}_o$ in (10.96).

11

Mobile Robots

The previous chapters deal mainly with articulated manipulators that represent the large majority of robots used in industrial settings. However, *mobile robots* are becoming increasingly important in advanced applications, in view of their potential for autonomous intervention. This chapter presents techniques for modelling, planning and control of *wheeled* mobile robots. The structure of the kinematic constraints arising from the pure rolling of the wheels is first analyzed; it is shown that such constraints are in general *nonholonomic* and consequently reduce the local mobility of the robot. The *kinematic model* associated with the constraints is introduced to describe the instantaneous admissible motions, and conditions are given under which it can be put in *chained form*. The *dynamic model*, that relates the admissible motions to the generalized forces acting on the robot DOFs, is then derived. The peculiar nature of the kinematic model, and in particular the existence of *flat outputs*, is exploited to devise *trajectory planning* methods that guarantee that the nonholonomic constraints are satisfied. The structure of *minimum-time trajectories* is also analyzed. The *motion control* problem for mobile robots is then discussed, with reference to two basic motion tasks, i.e., *trajectory tracking* and *posture regulation*. The chapter concludes by surveying some techniques for *odometric localization* that is necessary to implement feedback control schemes.

11.1 Nonholonomic Constraints

Wheels are by far the most common mechanism to achieve locomotion in mobile robots. Any wheeled vehicle is subject to kinematic constraints that reduce in general its local mobility, while leaving intact the possibility of reaching arbitrary configurations by appropriate manoeuvres. For example, any driver knows by experience that, while it is impossible to move instantaneously a car in the direction orthogonal to its heading, it is still possible to

park it arbitrarily, at least in the absence of obstacles. It is therefore important to analyze in detail the structure of these constraints.

In accordance with the terminology introduced in Sect. B.4, consider a mechanical system whose *configuration* $\boldsymbol{q} \in \mathcal{C}$ 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[1] with $\mathbb{R}^n$. The motion of the system that is represented by the evolution of $\boldsymbol{q}$ over time may be subject to constraints that can be classified under various criteria. For example, they may be expressed as equalities or 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

$$h_i(\boldsymbol{q}) = 0 \qquad i = 1, \ldots, k < n \tag{11.1}$$

are called *holonomic* (or *integrable*). In the following, it is assumed that the functions $h_i : \mathcal{C} \mapsto \mathbb{R}$ 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 (11.1) is called *holonomic*.

In the presence of holonomic constraints, the implicit function theorem can be used in principle to solve the equations in (11.1) 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 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 mechanical interconnections between the various bodies of the system. For example, prismatic and revolute joints used in robot manipulators are a typical source of such constraints, and joint variables are an example of reduced sets of coordinates in the above sense. Constraints of the form (11.1) may also arise in particular operating conditions; for example, one may mention the case of a kinematically redundant manipulator that moves while keeping the end-effector fixed at a certain pose (*self-motion*).

Constraints that involve generalized coordinates and velocities

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

[1] This assumption is taken for simplicity. In the general case, the configuration space $\mathcal{C}$ may be identified with a Euclidean space only on a local basis, because its global geometric structure is more complex; this will be further discussed in Chap. 12. The material presented in this chapter is, however, still valid.

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:

$$\boldsymbol{a}_i^T(\boldsymbol{q})\dot{\boldsymbol{q}} = 0 \qquad i = 1, \ldots, k < n, \tag{11.2}$$

or, in matrix form

$$\boldsymbol{A}^T(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{0}. \tag{11.3}$$

Vectors $\boldsymbol{a}_i : \mathcal{C} \mapsto \mathbb{R}^n$ are assumed to be smooth as well as linearly independent.

Clearly, the existence of k holonomic constraints (11.1) implies that of an equal number of kinematic constraints:

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

However, the converse is not true in general. A system of kinematic constraints in the form (11.3) may or may not be integrable to the form (11.1). 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 respect to holonomic constraints. To appreciate this fact, consider a single Pfaffian constraint

$$\boldsymbol{a}^T(\boldsymbol{q})\dot{\boldsymbol{q}} = 0. \tag{11.4}$$

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

$$h(\boldsymbol{q}) = c, \tag{11.5}$$

where $\partial h/\partial \boldsymbol{q} = \gamma(\boldsymbol{q})\,\boldsymbol{a}^T(\boldsymbol{q})$, with $\gamma(\boldsymbol{q}) \neq 0$ 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 confined to a particular *level surface* of the scalar function h. This surface, which depends on the initial configuration $\boldsymbol{q}_0$ through the value of $h(\boldsymbol{q}_0) = c$, has dimension $n - 1$.

Assume instead that the constraint (11.4) 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 $\boldsymbol{a}^T(\boldsymbol{q})$. 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 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

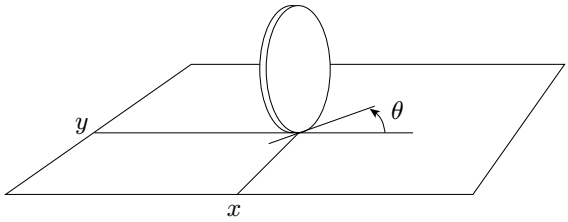

Fig. 11.1. Generalized coordinates for a disk rolling on a plane

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 is particularly relevant in the study of mobile robots.

Example 11.1

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 (Fig. 11.1). Its configuration is described by three[2] 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 respect to the x axis. The configuration vector is therefore $\boldsymbol{q} = [\, x \quad y \quad \theta \,]^T$.

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

$$\dot{x} \sin\theta - \dot{y} \cos\theta = [\, \sin\theta \quad -\cos\theta \quad 0 \,]\, \dot{\boldsymbol{q}} = 0, \tag{11.6}$$

and entails that, in the 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 unconstrained.

Constraint (11.6) 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 $\boldsymbol{q}_i = [\, x_i \quad y_i \quad \theta_i \,]^T$ to any final configuration $\boldsymbol{q}_f = [\, x_f \quad y_f \quad \theta_f \,]^T$ through the following sequence of movements that do not violate constraint (11.6):

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 its final position (x_f, y_f);
3. rotate again the disk around its vertical axis to change the orientation from θ_v to θ_f.

[2] One could add to this description an angle ϕ measuring the rotation of the disk around the horizontal axis passing through its centre. Such a coordinate is however irrelevant for the analysis presented in this chapter, and is therefore ignored in the following.

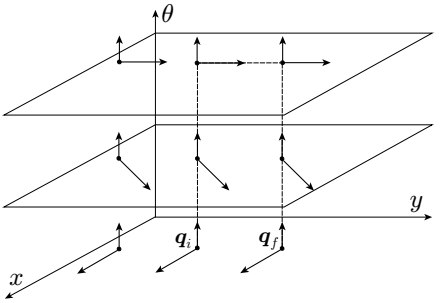

Fig. 11.2. A local representation of the configuration space for the rolling disk with an example manoeuvre that transfers the configuration from $\boldsymbol{q}_i$ to $\boldsymbol{q}_f$ (*dashed line*)

An example of this manoeuvre is shown in Fig. 11.2. Two possible directions of instantaneous motion are shown at each configuration: the first, that is aligned with the sagittal axis, moves the contact point while keeping the orientation constant (rolling); the second varies the orientation while keeping the contact point fixed (rotation around the vertical axis).

It is interesting to note that, in addition to wheeled vehicles, there exist other robotic systems that are nonholonomic in nature. For example, the pure rolling constraint also arises in manipulation problems with round-fingered robot hands. Another kind of nonholonomic behaviour is found in multibody systems that 'float' freely (i.e., without a fixed base), such as manipulators used in space operations. In fact, in the absence of external generalized forces, the conservation of the angular momentum represents a non-integrable Pfaffian constraint for the system.

11.1.1 Integrability Conditions

In the presence of Pfaffian kinematic constraints, integrability conditions can be used to decide whether the system is holonomic or nonholonomic.

Consider first the case of a *single* Pfaffian constraint:

$$\boldsymbol{a}^T(\boldsymbol{q})\dot{\boldsymbol{q}} = \sum_{j=1}^{n} a_j(\boldsymbol{q})\dot{q}_j = 0. \tag{11.7}$$

For this constraint to be integrable, there must exist a scalar function $h(\boldsymbol{q})$ and an integrating factor $\gamma(\boldsymbol{q}) \neq 0$ such that the following condition holds:

$$\gamma(\boldsymbol{q})a_j(\boldsymbol{q}) = \frac{\partial h(\boldsymbol{q})}{\partial q_j} \qquad j = 1, \ldots, n. \tag{11.8}$$

The converse is also true: if there exists an integrating factor $\gamma(\boldsymbol{q}) \neq 0$ such that $\gamma(\boldsymbol{q})\boldsymbol{a}(\boldsymbol{q})$ is the gradient of a scalar function $h(\boldsymbol{q})$, constraint (11.7) is integrable. By using Schwarz theorem on the symmetry of second derivatives, the integrability condition (11.8) may be replaced by the following system of partial differential equations:

$$\frac{\partial(\gamma a_k)}{\partial q_j} = \frac{\partial(\gamma a_j)}{\partial q_k} \qquad j,k = 1,\ldots,n, \quad j \neq k, \tag{11.9}$$

that does not contain the unknown function $h(\boldsymbol{q})$. Note that condition (11.9) implies that a Pfaffian constraint with constant coefficients a_j is always holonomic.

Example 11.2

Consider the following kinematic constraint in $\mathcal{C} = \mathbb{R}^3$:

$$\dot{q}_1 + q_1\dot{q}_2 + \dot{q}_3 = 0.$$

The holonomy condition (11.9) gives

$$\begin{aligned} \frac{\partial\gamma}{\partial q_2} &= \gamma + q_1\,\frac{\partial\gamma}{\partial q_1} \\ \frac{\partial\gamma}{\partial q_3} &= \frac{\partial\gamma}{\partial q_1} \\ q_1\,\frac{\partial\gamma}{\partial q_3} &= \frac{\partial\gamma}{\partial q_2}. \end{aligned}$$

By substituting the second and third equations into the first, it is easy to conclude that the only solution is $\gamma = 0$. Therefore, the constraint is nonholonomic.

Example 11.3

Consider the pure rolling constraint (11.6). In this case, the holonomy condition (11.9) gives

$$\begin{aligned} \sin\theta\,\frac{\partial\gamma}{\partial y} &= -\cos\theta\,\frac{\partial\gamma}{\partial x} \\ \cos\theta\,\frac{\partial\gamma}{\partial\theta} &= \gamma\sin\theta \\ \sin\theta\,\frac{\partial\gamma}{\partial\theta} &= -\gamma\cos\theta. \end{aligned}$$

Squaring and adding the last two equations gives $\partial\gamma/\partial\theta = \pm\gamma$. Assume for example $\partial\gamma/\partial\theta = \gamma$. Using this in the above equations leads to

$$\begin{aligned} \gamma\cos\theta &= \gamma\sin\theta \\ \gamma\sin\theta &= -\gamma\cos\theta \end{aligned}$$

whose only solution is $\gamma = 0$. The same conclusion is reached by letting $\partial\gamma/\partial\theta = -\gamma$. This confirms that constraint (11.6) is nonholonomic.

The situation becomes more complicated when dealing with a system of $k > 1$ kinematic constraints of the form (11.3). In fact, in this case it may happen that the single constraints are not integrable if taken separately, but the whole system is integrable. In particular, if $p \leq k$ independent linear combinations of the constraints

$$\sum_{i=1}^{k} \gamma_{ji}(\boldsymbol{q}) \boldsymbol{a}_i^T(\boldsymbol{q}) \dot{\boldsymbol{q}} \qquad j = 1, \ldots, p$$

are integrable, there exist p independent scalar functions $h_j(\boldsymbol{q})$ such that

$$\operatorname{span} \left\{ \frac{\partial h_1(\boldsymbol{q})}{\partial \boldsymbol{q}}, \ldots, \frac{\partial h_p(\boldsymbol{q})}{\partial \boldsymbol{q}} \right\} \subset \operatorname{span} \left\{ \boldsymbol{a}_1^T(\boldsymbol{q}), \ldots, \boldsymbol{a}_k^T(\boldsymbol{q}) \right\} \quad \forall \boldsymbol{q} \in \mathcal{C}.$$

Therefore, the configurations that are accessible for the mechanical system belong to the $(n-p)$-dimensional subspace consisting of the particular level surfaces of the functions h_j:

$$\{\boldsymbol{q} \in \mathcal{C} \colon h_1(\boldsymbol{q}) = c_1, \ldots, h_p(\boldsymbol{q}) = c_p\}$$

on which the motion is started (see Problem 11.2). In the case $p = k$, the system of kinematic constraints (11.3) is completely integrable, and hence holonomic.

Example 11.4

Consider the system of Pfaffian constraints

$$\begin{aligned} \dot{q}_1 + q_1 \dot{q}_2 + \dot{q}_3 &= 0 \\ \dot{q}_1 + \dot{q}_2 + q_1 \dot{q}_3 &= 0. \end{aligned}$$

Taken separately, these constraints are found to be non-integrable (in particular, the first is the nonholonomic constraint of Example 11.2). However, subtracting the second from the first gives

$$(q_1 - 1)(\dot{q}_2 - \dot{q}_3) = 0$$

so that $\dot{q}_2 = \dot{q}_3$, because the constraints must be satisfied for any value of $\boldsymbol{q}$. The assigned system of constraints is then equivalent to

$$\begin{aligned} \dot{q}_2 &= \dot{q}_3 \\ \dot{q}_1 + (1 + q_1)\dot{q}_2 &= 0, \end{aligned}$$

which can be integrated as

$$\begin{aligned} q_2 - q_3 &= c_1 \\ \log(q_1 + 1) + q_2 &= c_2 \end{aligned}$$

with integration constants c_1, c_2.

The integrability conditions of a system of Pfaffian kinematic constraints are quite complex and derive from a fundamental result of differential geometry known as *Frobenius theorem*. However, as shown later in the chapter, it is possible to derive such conditions more directly from a different perspective.

11.2 Kinematic Model

The system of k Pfaffian constraints (11.3) entails that the admissible generalized velocities at each configuration $\boldsymbol{q}$ belong to the $(n-k)$-dimensional null space of matrix $\boldsymbol{A}^T(\boldsymbol{q})$. Denoting by $\{\boldsymbol{g}_1(\boldsymbol{q}), \ldots, \boldsymbol{g}_{n-k}(\boldsymbol{q})\}$ a basis of $\mathcal{N}(\boldsymbol{A}^T(\boldsymbol{q}))$, the admissible trajectories for the mechanical system can then be characterized as the solutions of the nonlinear dynamic system

$$\dot{\boldsymbol{q}} = \sum_{j=1}^{m} \boldsymbol{g}_j(\boldsymbol{q}) u_j = \boldsymbol{G}(\boldsymbol{q})\boldsymbol{u} \qquad m = n-k, \tag{11.10}$$

where $\boldsymbol{q} \in \mathbb{R}^n$ is the state vector and $\boldsymbol{u} = [\, u_1 \quad \ldots \quad u_m \,]^T \in \mathbb{R}^m$ is the input vector. System (11.10) is said to be *driftless* because one has $\dot{\boldsymbol{q}} = \boldsymbol{0}$ if the input is zero.

The choice of the *input vector fields* $\boldsymbol{g}_1(\boldsymbol{q}), \ldots, \boldsymbol{g}_m(\boldsymbol{q})$ (and thus of matrix $\boldsymbol{G}(\boldsymbol{q})$) in (11.10) is not unique. Correspondingly, the components of $\boldsymbol{u}$ may have different meanings. In general, it is possible to choose the basis of $\mathcal{N}(\boldsymbol{A}^T(\boldsymbol{q}))$ in such a way that the u_js have a physical interpretation, as will be shown later for some examples of mobile robots. In any case, vector $\boldsymbol{u}$ may not be directly related to the actual control inputs, that are in general forces and/or torques. For this reason, Eq. (11.10) is referred to as the *kinematic model* of the constrained mechanical system.

The holonomy or nonholonomy of constraints (11.3) can be established by analyzing the controllability[3] properties of the associated kinematic model (11.10). In fact, two cases are possible:

1. If system (11.10) is controllable, given two arbitrary configurations $\boldsymbol{q}_i$ and $\boldsymbol{q}_f$ in $\mathcal{C}$, there exists a choice of $\boldsymbol{u}(t)$ that steers the system from $\boldsymbol{q}_i$ to $\boldsymbol{q}_f$, i.e., there exists a trajectory that joins the two configurations and satisfies the kinematic constraints (11.3). Therefore, these do not affect in any way the accessibility of $\mathcal{C}$, and they are (completely) nonholonomic.
2. If system (11.10) is not controllable, the kinematic constraints (11.3) reduce the set of accessible configurations in $\mathcal{C}$. Hence, the constraints are partially or completely integrable depending on the dimension $\nu < n$ of the accessible configuration space. In particular:

[3] Refer to Appendix D for a short survey of nonlinear controllability theory, including the necessary tools from differential geometry.

2a. If $m < \nu < n$, the loss of accessibility is not maximal, and thus constraints (11.3) are only partially integrable. The mechanical system is still nonholonomic.
2b. If $\nu = m$, the loss of accessibility is maximal, and constraints (11.3) are completely integrable. Therefore, the mechanical system is holonomic.

Note how this particular viewpoint, i.e., the equivalence between controllability and nonholonomy, was already implicitly adopted in Example 11.1, where the controllability of the kinematic system was proven *constructively*, i.e., by exhibiting a reconfiguration manoeuvre. A more systematic approach is to take advantage of the controllability conditions for nonlinear driftless systems. In particular, controllability may be verified using the *accessibility rank condition*

$$\dim \Delta_{\mathcal{A}}(\boldsymbol{q}) = n, \tag{11.11}$$

where $\Delta_{\mathcal{A}}$ is the *accessibility distribution* associated with system (11.10), i.e., the involutive closure of distribution $\Delta = \text{span}\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m\}$. The following cases may occur:

1. If (11.11) holds, system (11.10) is controllable and the kinematic constraints (11.3) are (completely) nonholonomic.
2. If (11.11) does not hold, system (11.10) is not controllable and the kinematic constraints (11.3) are at least partially integrable. In particular, let

$$\dim \Delta_{\mathcal{A}}(\boldsymbol{q}) = \nu < n.$$

Then

2a. If $m < \nu < n$, constraints (11.3) are only partially integrable.
2b. If $\nu = m$, constraints (11.3) are completely integrable, and hence holonomic. This happens when $\Delta_{\mathcal{A}}$ coincides with $\Delta = \text{span}\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m\}$, i.e., when the latter distribution is involutive.

It is easy to verify that, in the case of a single kinematic constraint (11.7), the integrability condition given by (11.9) is equivalent to the involutivity of $\Delta = \text{span}\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{n-1}\}$. Another remarkable situation is met when the number of Pfaffian constraints is $k = n - 1$; in this case, the associated kinematic model (11.10) consists of a single vector field $\boldsymbol{g}$ ($m = 1$). Hence, $n - 1$ Pfaffian constraints are always integrable, because the distribution associated with a single vector field is always involutive. For example, a mechanical system with two generalized coordinates that is subject to a scalar Pfaffian constraint is always holonomic.

In the following, the kinematic models of two wheeled vehicles of particular interest will be analyzed in detail. A large part of the existing mobile robots have a kinematic model that is equivalent to one of these two.

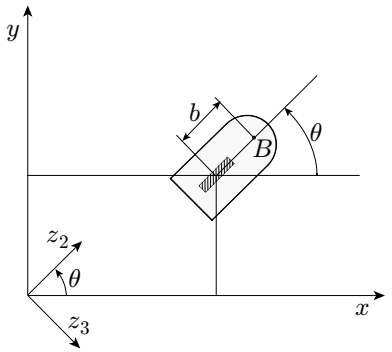

Fig. 11.3. Generalized coordinates for a unicycle

11.2.1 Unicycle

A *unicycle* is a vehicle with a single orientable wheel. Its configuration is completely described by $\boldsymbol{q} = [\, x \quad y \quad \theta \,]^T$, where (x, y) are the Cartesian coordinates of the contact point of the wheel with the ground (or equivalently, of the wheel centre) and θ is the orientation of the wheel with respect to the x axis (see Fig. 11.3).

As already seen in Example 11.1, the pure rolling constraint for the wheel is expressed as

$$\dot{x} \sin\theta - \dot{y} \cos\theta = [\, \sin\theta \quad -\cos\theta \quad 0 \,]\, \dot{\boldsymbol{q}} = 0, \tag{11.12}$$

entailing that the velocity of the contact point is zero in the direction orthogonal to the sagittal axis of the vehicle. The line passing through the contact point and having such direction is therefore called *zero motion line*. Consider the matrix

$$\boldsymbol{G}(\boldsymbol{q}) = [\, \boldsymbol{g}_1(\boldsymbol{q}) \quad \boldsymbol{g}_2(\boldsymbol{q}) \,] = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix},$$

whose columns $\boldsymbol{g}_1(\boldsymbol{q})$ and $\boldsymbol{g}_2(\boldsymbol{q})$ are, for each $\boldsymbol{q}$, a basis of the null space of the matrix associated with the Pfaffian constraint. All the admissible generalized velocities at $\boldsymbol{q}$ are therefore obtained as a linear combination of $\boldsymbol{g}_1(\boldsymbol{q})$ and $\boldsymbol{g}_2(\boldsymbol{q})$. The kinematic model of the unicycle is then

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega, \tag{11.13}$$

where the inputs v and ω have a clear physical interpretation. In particular, v is the *driving velocity*, i.e., the modulus[4] (with sign) of the contact point

[4] Note that v is given by the angular speed of the wheel around its horizontal axis multiplied by the wheel radius.

velocity vector, whereas the *steering velocity* ω is the wheel angular speed around the vertical axis.

The Lie bracket of the two input vector fields is

$$[\boldsymbol{g}_1, \boldsymbol{g}_2](\boldsymbol{q}) = \begin{bmatrix} \sin\theta \\ -\cos\theta \\ 0 \end{bmatrix},$$

that is always linearly independent from $\boldsymbol{g}_1(\boldsymbol{q}), \boldsymbol{g}_2(\boldsymbol{q})$. Therefore, the iterative procedure (see Sect. D.2) for building the accessibility distribution $\Delta_{\mathcal{A}}$ ends with

$$\dim \Delta_{\mathcal{A}} = \dim \Delta_2 = \dim \operatorname{span}\{\boldsymbol{g}_1, \boldsymbol{g}_2, [\boldsymbol{g}_1, \boldsymbol{g}_2]\} = 3.$$

This indicates that the unicycle is controllable with degree of nonholonomy $\kappa = 2$, and that constraint (11.12) is nonholonomic — the same conclusion reached in Example 11.3 by applying the integrability condition.

A unicycle in the strict sense (i.e., a vehicle equipped with a single wheel) is a robot with a serious problem of balance in static conditions. However, there exist vehicles that are kinematically equivalent to a unicycle but more stable from a mechanical viewpoint. Among these, the most important are the *differential drive* and the *synchro drive* vehicles, already introduced in Sect. 1.2.2.

For the differential drive mobile robot of Fig. 1.13, denote by (x, y) the Cartesian coordinates of the midpoint of the segment joining the two wheel centres, and by θ the common orientation of the fixed wheels (hence, of the vehicle body). Then, the kinematic model (11.13) of the unicycle also applies to the differential drive vehicle, provided that the driving and steering velocities v and ω are expressed as a function of the actual velocity inputs, i.e., the angular speeds ω_R and ω_L of the right and left wheel, respectively. Simple arguments (see Problem 11.6) can be used to show that there is a one-to-one correspondence between the two sets of inputs:

$$v = \frac{r\,(\omega_R + \omega_L)}{2} \qquad \omega = \frac{r\,(\omega_R - \omega_L)}{d}, \tag{11.14}$$

where r is the radius of the wheels and d is the distance between their centres.

The equivalence with the kinematic model (11.13) is even more straightforward for the *synchro drive* mobile robot of Fig. 1.14, whose control inputs are indeed the driving velocity v and the steering velocity ω, that are common to the three orientable wheels. The Cartesian coordinates (x, y) may represent in this case any point of the robot (for example, its centroid), while θ is the common orientation of the wheels. Note that, unlike a differential drive vehicle, the orientation of the body of a synchro drive vehicle never changes, unless a third actuator is added for this specific purpose.

11.2.2 Bicycle

Consider now a *bicycle*, i.e., a vehicle having an orientable wheel and a fixed wheel arranged as in Fig. 11.4. A possible choice for the generalized coordi-

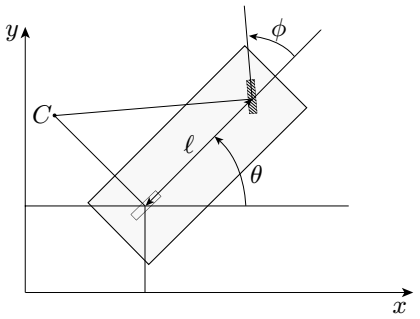

Fig. 11.4. Generalized coordinates and instantaneous centre of rotation for a bicycle

nates is $\boldsymbol{q} = [\, x \quad y \quad \theta \quad \phi \,]^T$, where (x, y) are the Cartesian coordinates of the contact point between the rear wheel and the ground (i.e., of the rear wheel centre), θ is the orientation of the vehicle with respect to the x axis, and ϕ is the steering angle of the front wheel with respect to the vehicle.

The motion of the vehicle is subject to two pure rolling constraints, one for each wheel:

$$\dot{x}_f \sin(\theta + \phi) - \dot{y}_f \cos(\theta + \phi) = 0 \tag{11.15}$$

$$\dot{x} \sin\theta - \dot{y} \cos\theta = 0, \tag{11.16}$$

where (x_f, y_f) is the Cartesian position of the centre of the front wheel. The geometric meaning of these constraints is obvious: the velocity of the centre of the front wheel is zero in the direction orthogonal to the wheel itself, while the velocity of the centre of the rear wheel is zero in the direction orthogonal to the sagittal axis of the vehicle. The zero motion lines of the two wheels meet at a point C called *instantaneous centre of rotation* (Fig. 11.4), whose position depends only on (and changes with) the configuration $\boldsymbol{q}$ of the bicycle. Each point of the vehicle body then moves instantaneously along an arc of circle with centre in C (see also Problem 11.7).

Using the rigid body constraint

$$\begin{aligned} x_f &= x + \ell \cos\theta \\ y_f &= y + \ell \sin\theta, \end{aligned}$$

where ℓ is the distance between the wheels, constraint (11.15) can be rewritten as

$$\dot{x} \sin(\theta + \phi) - \dot{y} \cos(\theta + \phi) - \ell\, \dot{\theta} \cos\phi = 0. \tag{11.17}$$

The matrix associated with the Pfaffian constraints (11.16), (11.17) is then

$$\boldsymbol{A}^T(\boldsymbol{q}) = \begin{bmatrix} \sin\theta & -\cos\theta & 0 & 0 \\ \sin(\theta + \phi) & -\cos(\theta + \phi) & -\ell \cos\phi & 0 \end{bmatrix},$$

with constant rank $k = 2$. The dimension of its null space is $n - k = 2$, and all the admissible velocities at $\boldsymbol{q}$ may be written as a linear combination of a basis of $\mathcal{N}(\boldsymbol{A}^T(\boldsymbol{q}))$, for example

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\phi \\ \sin\theta\cos\phi \\ \sin\phi/\ell \\ 0 \end{bmatrix} u_1 + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} u_2.$$

Since the front wheel is orientable, it is immediate to set $u_2 = \omega$, where ω is the *steering* velocity. The expression of u_1 depends instead on how the vehicle is driven.

If the bicycle has *front-wheel drive*, one has directly $u_1 = v$, where v is the *driving* velocity of the front wheel. The corresponding kinematic model is

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta\cos\phi \\ \sin\theta\cos\phi \\ \sin\phi/\ell \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \omega. \tag{11.18}$$

Denoting by $\boldsymbol{g}_1(\boldsymbol{q})$ and $\boldsymbol{g}_2(\boldsymbol{q})$ the two input vector fields, simple computations give

$$\boldsymbol{g}_3(\boldsymbol{q}) = [\boldsymbol{g}_1, \boldsymbol{g}_2](\boldsymbol{q}) = \begin{bmatrix} \cos\theta\sin\phi \\ \sin\theta\sin\phi \\ -\cos\phi/\ell \\ 0 \end{bmatrix} \quad \boldsymbol{g}_4(\boldsymbol{q}) = [\boldsymbol{g}_1, \boldsymbol{g}_3](\boldsymbol{q}) = \begin{bmatrix} -\sin\theta/\ell \\ \cos\theta/\ell \\ 0 \\ 0 \end{bmatrix},$$

both linearly independent from $\boldsymbol{g}_1(\boldsymbol{q})$ and $\boldsymbol{g}_2(\boldsymbol{q})$. Hence, the iterative procedure for building the accessibility distribution $\Delta_{\mathcal{A}}$ ends with

$$\dim \Delta_{\mathcal{A}} = \dim \Delta_3 = \dim \operatorname{span}\{\boldsymbol{g}_1, \boldsymbol{g}_2, \boldsymbol{g}_3, \boldsymbol{g}_4\} = 4.$$

This means that the front-wheel drive bicycle is controllable with degree of nonholonomy $\kappa = 3$, and constraints (11.15), (11.16) are (completely) nonholonomic.

The kinematic model of a bicycle with *rear-wheel drive* can be derived by noting that in this case the first two equations must coincide with those of the unicycle model (11.13). It is then sufficient to set $u_1 = v/\cos\phi$ to obtain

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ \dot{\phi} \end{bmatrix} = \begin{bmatrix} \cos\theta \\ \sin\theta \\ \tan\phi/\ell \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \omega, \tag{11.19}$$

where v is the *driving* velocity of the rear wheel.[5] In this case, one has

$$\boldsymbol{g}_3(\boldsymbol{q}) = [\boldsymbol{g}_1, \boldsymbol{g}_2](\boldsymbol{q}) = \begin{bmatrix} 0 \\ 0 \\ -\dfrac{1}{\ell \cos^2 \phi} \\ 0 \end{bmatrix} \qquad \boldsymbol{g}_4(\boldsymbol{q}) = [\boldsymbol{g}_1, \boldsymbol{g}_3](\boldsymbol{q}) = \begin{bmatrix} -\dfrac{\sin\theta}{\ell \cos^2 \phi} \\ \dfrac{\cos\theta}{\ell \cos^2 \phi} \\ 0 \\ 0 \end{bmatrix},$$

again linearly independent from $\boldsymbol{g}_1(\boldsymbol{q})$ and $\boldsymbol{g}_2(\boldsymbol{q})$. Hence, the rear-wheel drive bicycle is also controllable with degree of nonholonomy $\kappa = 3$.

Like the unicycle, the bicycle is also unstable in static conditions. Kinematically equivalent vehicles that are mechanically balanced are the *tricycle* and the *car-like* robot, introduced in Sect. 1.2.2 and shown respectively in Fig. 1.15 and 1.16. In both cases, the kinematic model is given by (11.18) or by (11.19) depending on the wheel drive being on the front or the rear wheels. In particular, (x, y) are the Cartesian coordinates of the midpoint of the rear wheel axle, θ is the orientation of the vehicle, and ϕ is the steering angle.

11.3 Chained Form

The possibility of transforming the kinematic model (11.10) of a mobile robot in a canonical form is of great interest for solving planning and control problems with efficient, systematic procedures. Here, the analysis is limited to systems with two inputs, like the unicycle and bicycle models.

A $(2, n)$ *chained form* is a two-input driftless system

$$\dot{\boldsymbol{z}} = \boldsymbol{\gamma}_1(\boldsymbol{z}) v_1 + \boldsymbol{\gamma}_2(\boldsymbol{z}) v_2,$$

whose equations are expressed as

$$\begin{aligned} \dot{z}_1 &= v_1 \\ \dot{z}_2 &= v_2 \\ \dot{z}_3 &= z_2 v_1 \\ &\vdots \\ \dot{z}_n &= z_{n-1} v_1. \end{aligned} \tag{11.20}$$

Using the following notation for a 'repeated' Lie bracket:

$$\mathrm{ad}_{\boldsymbol{\gamma}_1} \boldsymbol{\gamma}_2 = [\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2] \qquad \mathrm{ad}^k_{\boldsymbol{\gamma}_1} \boldsymbol{\gamma}_2 = [\boldsymbol{\gamma}_1, \mathrm{ad}^{k-1}_{\boldsymbol{\gamma}_1} \boldsymbol{\gamma}_2],$$

[5] Note that the kinematic model (11.19) is no longer valid for $\phi = \pm\pi/2$, where the first vector field is not defined. This corresponds to the mechanical jam in which the front wheel is orthogonal to the sagittal axis of the vehicle. This singularity does not arise in the front-wheel drive bicycle (11.18), that in principle can still pivot around the rear wheel contact point in such a situation.

one has for system (11.20)

$$\boldsymbol{\gamma}_1 = \begin{bmatrix} 1 \\ 0 \\ z_2 \\ z_3 \\ \vdots \\ z_{n-1} \end{bmatrix} \quad \boldsymbol{\gamma}_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \quad \Rightarrow \quad \mathrm{ad}^k_{\boldsymbol{\gamma}_1} \boldsymbol{\gamma}_2 = \begin{bmatrix} 0 \\ \vdots \\ (-1)^k \\ \vdots \\ 0 \end{bmatrix},$$

where $(-1)^k$ is the $(k+2)$-th component. This implies that the system is controllable, because the accessibility distribution

$$\Delta_{\mathcal{A}} = \mathrm{span}\,\{\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_2, \mathrm{ad}_{\boldsymbol{\gamma}_1}\boldsymbol{\gamma}_2, \ldots, \mathrm{ad}^{n-2}_{\boldsymbol{\gamma}_1}\boldsymbol{\gamma}_2\}$$

has dimension n. In particular, the degree of nonholonomy is $\kappa = n - 1$.

There exist necessary and sufficient conditions for transforming a generic two-input driftless system

$$\dot{\boldsymbol{q}} = \boldsymbol{g}_1(\boldsymbol{q})u_1 + \boldsymbol{g}_2(\boldsymbol{q})u_2 \tag{11.21}$$

in the chained form (11.20) via coordinate and input transformations

$$\boldsymbol{z} = \boldsymbol{T}(\boldsymbol{q}) \qquad \boldsymbol{v} = \boldsymbol{\beta}(\boldsymbol{q})\boldsymbol{u}. \tag{11.22}$$

In particular, it can be shown that systems like (11.21) with dimension n not larger than 4 can *always* be put in chained form. This applies, for example, to the kinematic models of the unicycle and the bicycle.

There also exist sufficient conditions for transformability in chained form that are relevant because they are constructive. Define the distributions

$$\begin{aligned} \Delta_0 &= \mathrm{span}\,\{\boldsymbol{g}_1, \boldsymbol{g}_2, \mathrm{ad}_{\boldsymbol{g}_1}\boldsymbol{g}_2, \ldots, \mathrm{ad}^{n-2}_{\boldsymbol{g}_1}\boldsymbol{g}_2\} \\ \Delta_1 &= \mathrm{span}\,\{\boldsymbol{g}_2, \mathrm{ad}_{\boldsymbol{g}_1}\boldsymbol{g}_2, \ldots, \mathrm{ad}^{n-2}_{\boldsymbol{g}_1}\boldsymbol{g}_2\} \\ \Delta_2 &= \mathrm{span}\,\{\boldsymbol{g}_2, \mathrm{ad}_{\boldsymbol{g}_1}\boldsymbol{g}_2, \ldots, \mathrm{ad}^{n-3}_{\boldsymbol{g}_1}\boldsymbol{g}_2\}. \end{aligned}$$

Assume that, in a certain set, it is $\dim \Delta_0 = n$, Δ_1 and Δ_2 are involutive, and there exists a scalar function $h_1(\boldsymbol{q})$ whose differential $d\boldsymbol{h}_1$ satisfies

$$d\boldsymbol{h}_1 \cdot \Delta_1 = 0 \qquad d\boldsymbol{h}_1 \cdot \boldsymbol{g}_1 = 1,$$

where the symbol $\cdot$ denotes the inner product between a row vector and a column vector — in particular, $\cdot\Delta_1$ is the inner product with any vector generated by distribution Δ_1. In this case, system (11.21) can be put in the form (11.20) through the coordinate transformation[6]

$$z_1 = h_1$$

[6] This transformation makes use of the Lie derivative (see Appendix D).

$$
\begin{aligned}
z_2 &= L_{\boldsymbol{g}_1}^{n-2} h_2 \\
&\vdots \\
z_{n-1} &= L_{\boldsymbol{g}_1} h_2 \\
z_n &= h_2,
\end{aligned}
$$

where h_2 must be chosen independent of h_1 and such that $dh_2 \cdot \Delta_2 = 0$. The input transformation is given by

$$
\begin{aligned}
v_1 &= u_1 \\
v_2 &= \left(L_{\boldsymbol{g}_1}^{n-1} h_2\right) u_1 + \left(L_{\boldsymbol{g}_2} L_{\boldsymbol{g}_1}^{n-2} h_2\right) u_2.
\end{aligned}
$$

In general, the coordinate and input transformations are not unique.

Consider the kinematic model (11.13) of the unicycle. With the change of coordinates

$$
\begin{aligned}
z_1 &= \theta \\
z_2 &= x \cos\theta + y \sin\theta \\
z_3 &= x \sin\theta - y \cos\theta
\end{aligned}
\tag{11.23}
$$

and the input transformation

$$
\begin{aligned}
v &= v_2 + z_3 v_1 \\
\omega &= v_1,
\end{aligned}
\tag{11.24}
$$

one obtains the (2,3) chained form

$$
\begin{aligned}
\dot{z}_1 &= v_1 \\
\dot{z}_2 &= v_2 \\
\dot{z}_3 &= z_2 v_1.
\end{aligned}
\tag{11.25}
$$

Note that, while z_1 is simply the orientation θ, coordinates z_2 and z_3 represent the position of the unicycle in a moving reference frame whose z_2 axis is aligned with the sagittal axis of the vehicle (see Fig. 11.3).

As for mobile robots with bicycle-like kinematics, consider for example the model (11.19) corresponding to the rear-wheel drive case. Using the change of coordinates

$$
\begin{aligned}
z_1 &= x \\
z_2 &= \frac{1}{\ell} \sec^3\theta \tan\phi \\
z_3 &= \tan\theta \\
z_4 &= y
\end{aligned}
$$

and the input transformation

$$v = \frac{v_1}{\cos\theta}$$
$$\omega = -\frac{3}{\ell} v_1 \sec\theta \sin^2\phi + \frac{1}{\ell} v_2 \cos^3\theta \cos^2\phi,$$

the (2,4) chained form is obtained:

$$\dot{z}_1 = v_1$$
$$\dot{z}_2 = v_2$$
$$\dot{z}_3 = z_2 v_1$$
$$\dot{z}_4 = z_3 v_1.$$

This transformation is defined everywhere in the configuration space, with the exception of points where $\cos\theta = 0$. The equivalence between the two models is then subject to the condition $\theta \neq \pm k\pi/2$, with $k = 1, 2, \ldots$.

11.4 Dynamic Model

The derivation of the dynamic model of a mobile robot is similar to the manipulator case, the main difference being the presence of nonholonomic constraints on the generalized coordinates. An important consequence of nonholonomy is that exact linearization of the dynamic model via feedback is no longer possible. In the following, the Lagrange formulation is used to obtain the dynamic model of an n-dimensional mechanical system subject to $k < n$ kinematic constraints in the form (11.3), and it is shown how this model can be partially linearized via feedback.

As usual, define the Lagrangian $\mathcal{L}$ of the mechanical system as the difference between its kinetic and potential energy:

$$\mathcal{L}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \mathcal{T}(\boldsymbol{q}, \dot{\boldsymbol{q}}) - \mathcal{U}(\boldsymbol{q}) = \frac{1}{2}\dot{\boldsymbol{q}}^T \boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}} - \mathcal{U}(\boldsymbol{q}), \tag{11.26}$$

where $\boldsymbol{B}(\boldsymbol{q})$ is the (symmetric and positive definite) inertia matrix of the mechanical system. The Lagrange equations are in this case

$$\frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}}\right)^T - \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{q}}\right)^T = \boldsymbol{S}(\boldsymbol{q})\boldsymbol{\tau} + \boldsymbol{A}(\boldsymbol{q})\boldsymbol{\lambda}, \tag{11.27}$$

where $\boldsymbol{S}(\boldsymbol{q})$ is an $(n \times m)$ matrix mapping the $m = n - k$ external inputs $\boldsymbol{\tau}$ to generalized forces performing work on $\boldsymbol{q}$, $\boldsymbol{A}(\boldsymbol{q})$ is the transpose of the $(k \times n)$ matrix characterizing the kinematic constraints (11.3), and $\boldsymbol{\lambda} \in \mathbb{R}^m$ is the vector of *Lagrange multipliers*. The term $\boldsymbol{A}(\boldsymbol{q})\boldsymbol{\lambda}$ represents the vector of reaction forces at the generalized coordinate level. It has been assumed that the number of available inputs matches the number of DOFs (*full actuation*),

that is, in turn, equal to the number n of generalized coordinates minus the number k of constraints.

Using (11.26), (11.27), the *dynamic model* of the constrained mechanical system is expressed as

$$\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \boldsymbol{S}(\boldsymbol{q})\boldsymbol{\tau} + \boldsymbol{A}(\boldsymbol{q})\boldsymbol{\lambda} \tag{11.28}$$

$$\boldsymbol{A}^T(\boldsymbol{q})\dot{\boldsymbol{q}} = \boldsymbol{0}, \tag{11.29}$$

where

$$\boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) = \dot{\boldsymbol{B}}(\boldsymbol{q})\dot{\boldsymbol{q}} - \frac{1}{2}\left(\frac{\partial}{\partial \boldsymbol{q}}\left(\dot{\boldsymbol{q}}^T \boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{q}}\right)\right)^T + \left(\frac{\partial \mathcal{U}(\boldsymbol{q})}{\partial \boldsymbol{q}}\right)^T.$$

Consider now a matrix $\boldsymbol{G}(\boldsymbol{q})$ whose columns are a basis for the null space of $\boldsymbol{A}^T(\boldsymbol{q})$, so that $\boldsymbol{A}^T(\boldsymbol{q})\boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{0}$. One can replace the constraint given by (11.29) with the kinematic model

$$\dot{\boldsymbol{q}} = \boldsymbol{G}(\boldsymbol{q})\boldsymbol{v} = \sum_{i=1}^{m} \boldsymbol{g}_i(\boldsymbol{q})\, v_i, \tag{11.30}$$

where $\boldsymbol{v} \in \mathbb{R}^m$ is the vector of *pseudo-velocities*;[7] for example, in the case of a unicycle the components of this vector are the driving velocity v and the steering velocity ω. Moreover, the Lagrange multipliers in (11.28) can be eliminated premultiplying both sides of the equation by $\boldsymbol{G}^T(\boldsymbol{q})$. This leads to the *reduced dynamic model*

$$\boldsymbol{G}^T(\boldsymbol{q})\left(\boldsymbol{B}(\boldsymbol{q})\ddot{\boldsymbol{q}} + \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}})\right) = \boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{S}(\boldsymbol{q})\boldsymbol{\tau}, \tag{11.31}$$

a system of m differential equations.

Differentiation of (11.30) with respect to time gives

$$\ddot{\boldsymbol{q}} = \dot{\boldsymbol{G}}(\boldsymbol{q})\boldsymbol{v} + \boldsymbol{G}(\boldsymbol{q})\dot{\boldsymbol{v}}.$$

Premultiplying this by $\boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{B}(\boldsymbol{q})$ and using the reduced dynamic model (11.31), one obtains

$$\boldsymbol{M}(\boldsymbol{q})\dot{\boldsymbol{v}} + \boldsymbol{m}(\boldsymbol{q}, \boldsymbol{v}) = \boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{S}(\boldsymbol{q})\boldsymbol{\tau}, \tag{11.32}$$

where

$$\begin{aligned} \boldsymbol{M}(\boldsymbol{q}) &= \boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{B}(\boldsymbol{q})\boldsymbol{G}(\boldsymbol{q}) \\ \boldsymbol{m}(\boldsymbol{q}, \boldsymbol{v}) &= \boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{B}(\boldsymbol{q})\dot{\boldsymbol{G}}(\boldsymbol{q})\boldsymbol{v} + \boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{n}(\boldsymbol{q}, \boldsymbol{G}(\boldsymbol{q})\boldsymbol{v}), \end{aligned}$$

[7] In the dynamic modeling context, the use of this term emphasizes the difference between $\boldsymbol{v}$ and $\dot{\boldsymbol{q}}$, that are the actual (generalized) velocities of the mechanical system.

with $\boldsymbol{M}(\boldsymbol{q})$ positive definite and

$$\dot{\boldsymbol{G}}(\boldsymbol{q})\boldsymbol{v} = \sum_{i=1}^{m} \left(v_i \frac{\partial \boldsymbol{g}_i}{\partial \boldsymbol{q}}(\boldsymbol{q}) \right) \boldsymbol{G}(\boldsymbol{q})\boldsymbol{v}.$$

This finally leads to the *state-space reduced model*

$$\dot{\boldsymbol{q}} = \boldsymbol{G}(\boldsymbol{q})\boldsymbol{v} \tag{11.33}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{m}(\boldsymbol{q},\boldsymbol{v}) + \boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{S}(\boldsymbol{q})\boldsymbol{\tau}, \tag{11.34}$$

that represents in a compact form the kinematic and dynamic models of the constrained system as a set of $n+m$ differential equations.

Suppose now that

$$\det\left(\boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{S}(\boldsymbol{q})\right) \neq 0,$$

an assumption on the 'control availability' that is satisfied in many cases of interest. It is then possible to perform a *partial linearization via feedback* of (11.33), (11.34) by letting

$$\boldsymbol{\tau} = \left(\boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{S}(\boldsymbol{q})\right)^{-1} \left(\boldsymbol{M}(\boldsymbol{q})\boldsymbol{a} + \boldsymbol{m}(\boldsymbol{q},\boldsymbol{v})\right), \tag{11.35}$$

where $\boldsymbol{a} \in \mathbb{R}^m$ is the *pseudo-acceleration* vector. The resulting system is

$$\dot{\boldsymbol{q}} = \boldsymbol{G}(\boldsymbol{q})\boldsymbol{v} \tag{11.36}$$

$$\dot{\boldsymbol{v}} = \boldsymbol{a}. \tag{11.37}$$

Note the structure of this system: the first n equations are the kinematic model, of which the last m — that represent the inclusion of m integrators on the input channels — are a *dynamic extension*. If the system is unconstrained and fully actuated, it is $\boldsymbol{G}(\boldsymbol{q}) = \boldsymbol{S}(\boldsymbol{q}) = \boldsymbol{I}_n$; then, the feedback law (11.35) simply reduces to an inverse dynamics control analogous to (8.57), and correspondingly the closed-loop system is equivalent to n decoupled double integrators.

The implementation of the feedback control (11.35) in principle requires the measurement of $\boldsymbol{v}$, and this may not be available. However, pseudo-velocities can be computed via the kinematic model as

$$\boldsymbol{v} = \boldsymbol{G}^{\dagger}(\boldsymbol{q})\dot{\boldsymbol{q}} = \left(\boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{G}(\boldsymbol{q})\right)^{-1} \boldsymbol{G}^T(\boldsymbol{q})\dot{\boldsymbol{q}}, \tag{11.38}$$

provided that $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$ are measured. Note that the left pseudo-inverse of $\boldsymbol{G}(\boldsymbol{q})$ has been used here.

By defining the state $\boldsymbol{x} = (\boldsymbol{q},\boldsymbol{v}) \in \mathbb{R}^{n+m}$ and the input $\boldsymbol{u} = \boldsymbol{a} \in \mathbb{R}^m$, system (11.36), (11.37) can be expressed as

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{G}(\boldsymbol{x})\boldsymbol{u} = \begin{bmatrix} \boldsymbol{G}(\boldsymbol{q})\boldsymbol{v} \\ \boldsymbol{0} \end{bmatrix} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{I}_m \end{bmatrix} \boldsymbol{u}, \tag{11.39}$$

i.e., a nonlinear system with drift also known as the *second-order kinematic model* of the constrained mechanical system. Accordingly, Eq. (11.36) is sometimes called *first-order kinematic model*. In view of the results recalled in Appendix D, the controllability of the latter guarantees the controllability of system (11.39).

Summarizing, in nonholonomic mechanical systems — such as wheeled mobile robots — it is possible to 'cancel' the dynamic effects via nonlinear state feedback, provided that the dynamic parameters are exactly known and the complete state of the system (generalized coordinates and velocities $\boldsymbol{q}$ and $\dot{\boldsymbol{q}}$) is measured.

Under these assumptions, the control problem can be directly addressed at the (pseudo-)velocity level, i.e., by choosing $\boldsymbol{v}$ in such a way that the kinematic model

$$\dot{\boldsymbol{q}} = \boldsymbol{G}(\boldsymbol{q})\boldsymbol{v}$$

behaves as desired. From $\boldsymbol{v}$, it is possible to derive the actual control inputs at the generalized force level through (11.35). Since $\boldsymbol{a} = \dot{\boldsymbol{v}}$ appears in this equation, the pseudo-velocities $\boldsymbol{v}$ must be differentiable with respect to time.

Example 11.5

For illustration, the above procedure for deriving, reducing and partially linearizing the dynamic model is now applied to the unicycle. Let m be the mass of the unicycle, I its moment of inertia around the vertical axis through its centre, $\boldsymbol{\tau}_1$ the driving force and $\boldsymbol{\tau}_2$ the steering torque. With the kinematic constraint expressed as (11.12), the dynamic model (11.28), (11.29) takes on the form

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I \end{bmatrix} \begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{\theta} \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 \\ \sin\theta & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} + \begin{bmatrix} \sin\theta \\ -\cos\theta \\ 0 \end{bmatrix} \lambda$$

$$\dot{x}\sin\theta - \dot{y}\cos\theta = 0.$$

In this case one has

$$\begin{aligned} \boldsymbol{n}(\boldsymbol{q}, \dot{\boldsymbol{q}}) &= \boldsymbol{0} \\ \boldsymbol{G}(\boldsymbol{q}) &= \boldsymbol{S}(\boldsymbol{q}) \\ \boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{S}(\boldsymbol{q}) &= \boldsymbol{I} \\ \boldsymbol{G}^T(\boldsymbol{q})\boldsymbol{B}\dot{\boldsymbol{G}}(\boldsymbol{q}) &= \boldsymbol{0}, \end{aligned}$$

and thus the reduced model in state-space is obtained as

$$\begin{aligned} \dot{\boldsymbol{q}} &= \boldsymbol{G}(\boldsymbol{q})\boldsymbol{v} \\ \dot{\boldsymbol{v}} &= \boldsymbol{M}^{-1}(\boldsymbol{q})\boldsymbol{\tau} \end{aligned}$$

where

$$\boldsymbol{M}^{-1}(\boldsymbol{q}) = \begin{bmatrix} 1/m & 0 \\ 0 & 1/I \end{bmatrix}.$$

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

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

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

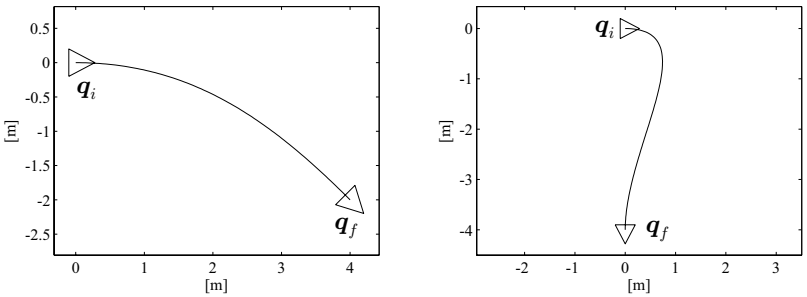

Fig. 11.5. Two parking manoeuvres planned via cubic Cartesian polynomials; in both cases $k = 5$ has been used

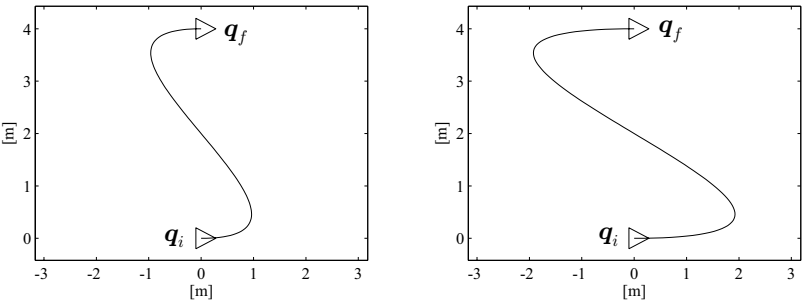

Fig. 11.6. Planning a parallel parking manoeuvre via cubic Cartesian polynomials; *left*: with $k = 10$, *right*: with $k = 20$

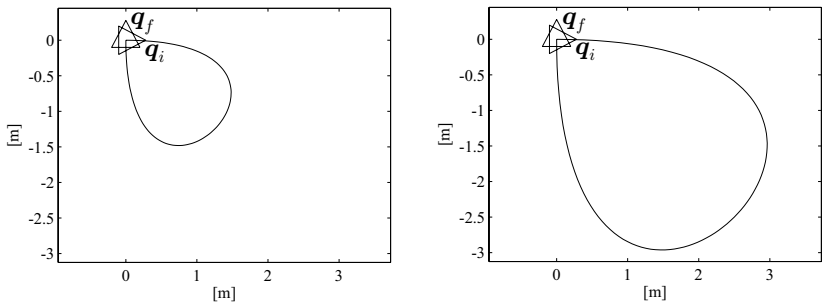

Fig. 11.7. Planning a pure reorientation manoeuvre via cubic Cartesian polynomials; *left*: with $k = 10$, *right*: with $k = 20$

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-

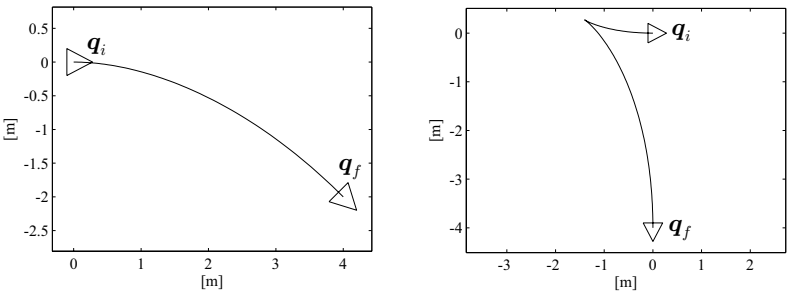

Fig. 11.8. Two parking manoeuvres planned via the chained form

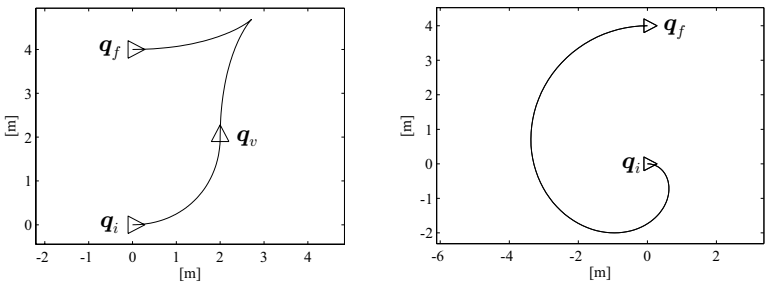

Fig. 11.9. Planning a parallel parking manoeuvre via the chained form; *left*: adding a via point $\boldsymbol{q}_v$, *right*: letting $\theta_f = \theta_i + 2\pi$

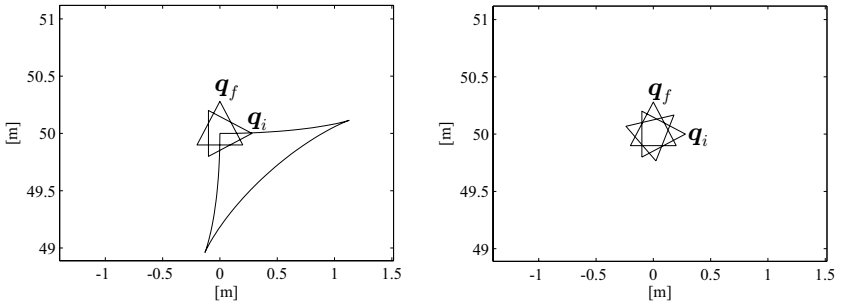

Fig. 11.10. Planning a pure reorientation manoeuvre via the chained form; *left*: with the coordinate transformation (11.23), *right*: with the coordinate transformation (11.52)

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.

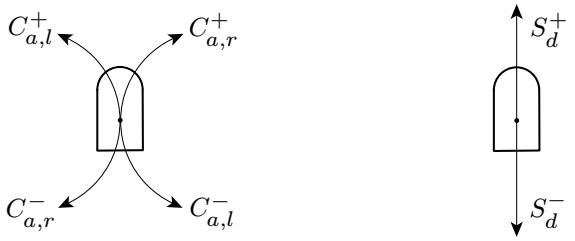

Fig. 11.11. The elementary arcs that constitute the trajectories of the sufficient family for the minimum-time planning problem for the unicycle

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

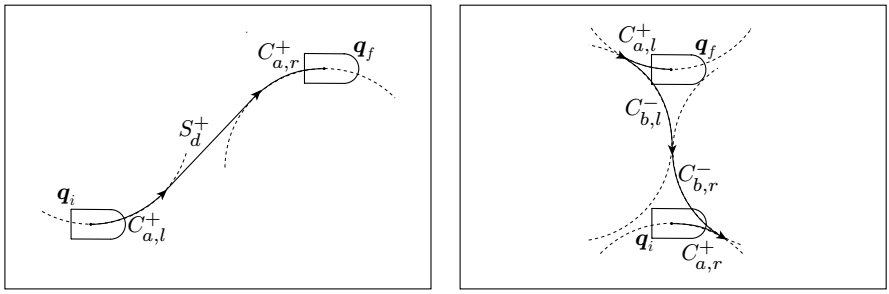

Fig. 11.12. Two examples of minimum-time trajectories for the unicycle

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):

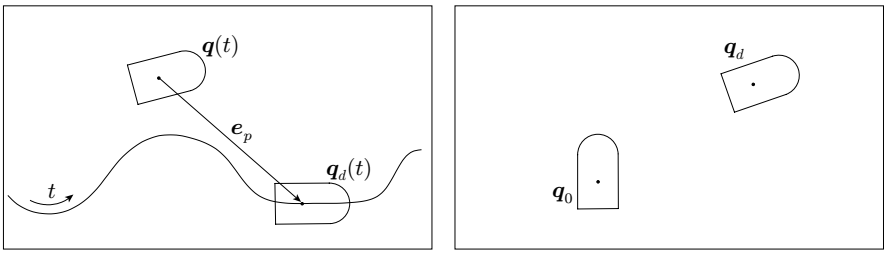

Fig. 11.13. Control problems for a unicycle; *left*: trajectory tracking, *right*: posture regulation

- *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 $\boldsymbol{e} = \mathbf{0}$. 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

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

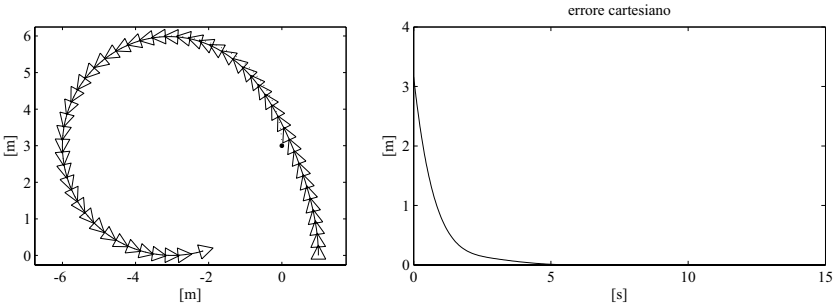

Fig. 11.14. Tracking a circular reference trajectory (*dotted*) with the controller based on approximate linearization; *left*: Cartesian motion of the unicycle, *right*: time evolution of the norm of the Cartesian error $\boldsymbol{e}_p$

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

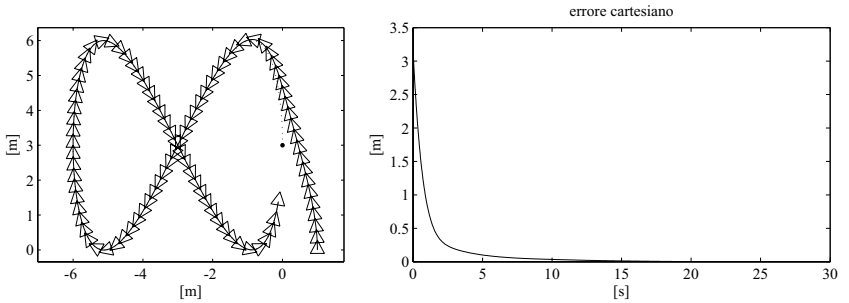

Fig. 11.15. Tracking a figure of eight-shaped reference trajectory (*dotted*) with the nonlinear controller; *left*: Cartesian motion of the unicycle, *right*: time evolution of the norm of the Cartesian error $\boldsymbol{e}_p$

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,

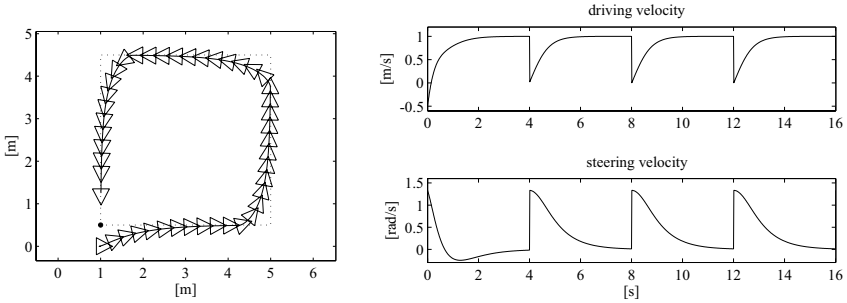

Fig. 11.16. Tracking a square reference trajectory (*dotted*) with the controller based on input/output linearization, with $b = 0.75$; *left*: Cartesian motion of the unicycle, *right*: time evolution of the velocity inputs v and ω

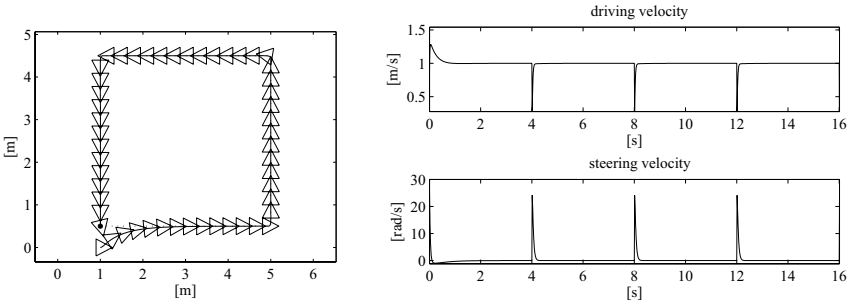

Fig. 11.17. Tracking a square reference trajectory (*dotted*) with the controller based on input/output linearization, with $b = 0.2$; *left*: Cartesian motion of the unicycle, *right*: time evolution of the velocity inputs v and ω

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

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

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

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.

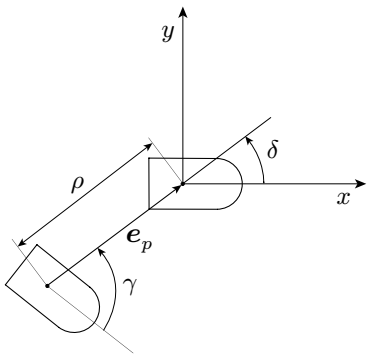

Fig. 11.18. Definition of polar coordinates for the unicycle

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 $\boldsymbol{q}$.

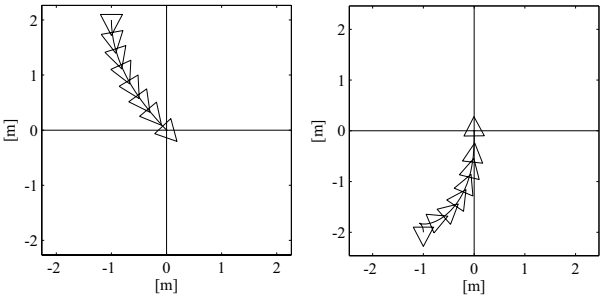

Fig. 11.19. Regulation to the origin of the Cartesian position of the unicycle with the controller (11.78), (11.79), for two different initial configurations

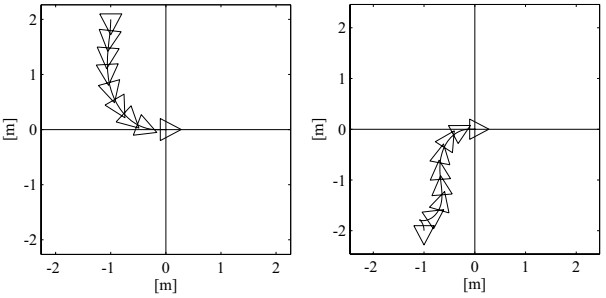

Fig. 11.20. Regulation to the origin of the posture of the unicycle with the controller (11.81), (11.82), for two different initial configurations

is generally satisfied[13] in digital control implementations, implies that during the interval the robot moves along an arc of circle of radius $R = v_k/\omega_k$, which degenerates to a line segment if $\omega_k = 0$. Assume that the robot configuration $\boldsymbol{q}(t_k) = \boldsymbol{q}_k$ at the sampling time t_k is known, together with the value of the velocity inputs v_k and ω_k applied in the interval $[t_k, t_{k+1})$. The value of the configuration variables $\boldsymbol{q}_{k+1}$ at the sampling time t_{k+1} can then be reconstructed by forward integration of the kinematic model (11.13).

A first possibility is to use the approximate formulae

$$\begin{aligned} x_{k+1} &= x_k + v_k\, T_s \cos\theta_k \\ y_{k+1} &= y_k + v_k\, T_s \sin\theta_k \\ \theta_{k+1} &= \theta_k + \omega_k\, T_s, \end{aligned} \tag{11.83}$$

where $T_s = t_{k+1} - t_k$ is the duration of the sampling interval. These equations, which correspond to the use of the Euler method for numerical integration

[13] In particular, this is certainly true if the velocity commands computed by the digital controller are converted to control inputs for the robot through a zero-order hold (ZOH).

of (11.13), introduce an error in the computation of x_{k+1} and y_{k+1}, that is performed as if the orientation θ_k remained constant throughout the interval. This error becomes smaller as T_s is decreased. The third formula instead is exact.

If the accuracy of the Euler method proves to be inadequate, one may use the following estimate with the same T_s:

$$\begin{aligned} x_{k+1} &= x_k + v_k\, T_s \cos\left(\theta_k + \frac{\omega_k\, T_s}{2}\right) \\ y_{k+1} &= y_k + v_k\, T_s \sin\left(\theta_k + \frac{\omega_k\, T_s}{2}\right) \\ \theta_{k+1} &= \theta_k + \omega_k\, T_s, \end{aligned} \tag{11.84}$$

corresponding to the adoption of the second-order Runge–Kutta integration method. Note how the first two formulae use the average value of the unicycle orientation in the sampling interval.

To obtain an exact reconstruction of $\boldsymbol{q}_{k+1}$ under the assumption of constant velocity inputs within the sampling interval one may use simple geometric arguments or exploit the transformability of the unicycle kinematic model in the chained form (11.25). As already seen, this form is easily integrable in closed form, leading to an exact expression for $\boldsymbol{z}_{k+1}$. The configuration $\boldsymbol{q}_{k+1}$ can then be computed by inverting the coordinate and input transformations (11.23) and (11.24). This procedure, which can be generalized to any mobile robot that can be put in chained form, provides the formulae:

$$\begin{aligned} x_{k+1} &= x_k + \frac{v_k}{\omega_k}(\sin\theta_{k+1} - \sin\theta_k) \\ y_{k+1} &= y_k - \frac{v_k}{\omega_k}(\cos\theta_{k+1} - \cos\theta_k) \\ \theta_{k+1} &= \theta_k + \omega_k\, T_s. \end{aligned} \tag{11.85}$$

Note that the first two are still defined for $\omega_k = 0$; in this case, they coincide with the corresponding formulae of Euler and Runge–Kutta methods (which are exact over line segments). In the implementation, however, it is necessary to handle this situation with a conditional instruction.

Figure 11.21 allows a comparison among the configurations $\boldsymbol{q}_{k+1}$ reconstructed via the three aforementioned integration methods. In practice, the difference is obviously much less dramatic, and tends to disappear as the duration T_s of the sampling interval is decreased.

In the previous formulae it has been assumed that the velocity inputs v_k and ω_k applied in the sampling interval are available. In view of the non-ideality of any actuation system, rather than relying on the 'commanded' values, it is convenient to reconstruct v_k and ω_k using the robot proprioceptive sensors. First of all, note that

$$v_k T_s = \Delta s \qquad \omega_k T_s = \Delta\theta \qquad \frac{v_k}{\omega_k} = \frac{\Delta s}{\Delta\theta}, \tag{11.86}$$

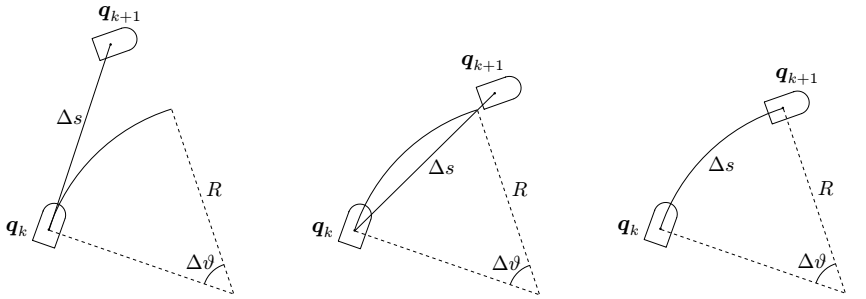

Fig. 11.21. Odometric localization for a unicycle moving along an elementary tract corresponding to an arc of circle; *left*: integration via Euler method, *centre*: integration via Runge–Kutta method, *right*: exact integration

where Δs is the length of the elementary tract travelled by the robot and $\Delta\theta$ is the total variation of orientation along the tract. For example, in the case of a differential drive unicycle, denote by $\Delta\phi_R$ and $\Delta\phi_L$ the rotation of the right and left wheel, respectively, as measured by the incremental encoders during the sampling interval. From (11.14) one easily finds

$$\Delta s = \frac{r}{2}\left(\Delta\phi_R + \Delta\phi_L\right) \qquad \Delta\theta = \frac{r}{d}\left(\Delta\phi_R - \Delta\phi_L\right)$$

that, used in (11.86), allow the implementation of all the previous formulae for the reconstruction of $\boldsymbol{q}_{k+1}$.

The forward integration of the kinematic model using the velocity commands reconstructed via the proprioceptive sensors — the encoders of the wheel actuators — is referred to as *odometric localization* or *passive localization* or *dead reckoning* — the latter is a term of uncertain etymology used in marine navigation. This method, relying on the iterated use of the previous formulae starting from an estimate of the initial configuration, provides an estimate whose accuracy cannot be better than that of $\boldsymbol{q}_0$. In any case, odometric localization — independently from the adopted integration method — is subject in practice to an error that grows over time (*drift*) and quickly becomes significant over sufficiently long paths. This error is the result of several causes, that include wheel slippage, inaccuracy in the calibration of kinematic parameters (e.g., the radius of the wheels), as well as the numerical error introduced by the integration method, if Euler or Runge–Kutta methods are used. It should also be noted that, once an odometric localization technique has been chosen, its performance also depends on the specific kinematic arrangement of the robot; for example, differential drive is usually better than synchro drive in this respect.

A more robust solution is represented by *active localization* methods. For example, this kind of approach can be adopted when the robot is equipped with proximity exteroceptive sensors (like a laser range finder) and knows a

map of the workspace, either given in advance or built by the robot itself during the motion. It is then possible to correct the estimate provided by the passive localization methods by comparing the expected measures of the exteroceptive sensors with the actual readings. These techniques, which make use of tools from *Bayesian estimation theory* such as the *Extended Kalman Filter* or the *Particle Filter*, provide greater accuracy than pure odometric localization and are therefore essential in navigation tasks over long paths.

Bibliography

The literature on mobile robots is very rich and includes some comprehensive treatises, among which one of the most recent is [210].

Many books deal with the realization of mobile robots, with particular emphasis on electro-mechanical design and sensor equipment, e.g., [106, 72, 21]. For the modelling of systems subject to nonholonomic constraints, see [164]. A general classification of wheeled mobile robots based on the number, placement and type of wheels is proposed in [18], where the derivation of the kinematic and dynamic models is also presented.

Conditions for transforming driftless systems in chained form are discussed in detail in [159], while a general reference on flat outputs is [80]. The presented schemes for Cartesian trajectory tracking based on linear and nonlinear control are taken from [34], while the method for posture regulation based on polar coordinates was proposed in [2]. Complete (input/state) linearization of the unicycle kinematic model can be obtained using a dynamic feedback law, as shown for example in [174]. The non-existence of a universal controller for nonholonomic robots is proven in [135].

A detailed extension of some of the planning and control techniques described in this chapter to the case of bicycle-like kinematics is given in [58]. The design of time-varying and/or discontinuous feedback control laws for posture regulation in mobile robots was addressed in many works, including [193] and [153].

The reader interested in sensor-based robot localization and map building can refer, e.g., to [231].

Problems

11.1. Consider the mobile robot obtained by connecting N trailers to a rear-wheel drive tricycle. Each trailer is a rigid body with an axle carrying two fixed wheels, that can be assimilated to a single wheel located at the midpoint of the axle, and is hinged to the midpoint of the preceding axle through a revolute joint. Find a set of generalized coordinates for the robot.

11.2. Consider an omnidirectional mobile robot having three Mecanum wheels placed at the vertices of an equilateral triangle, each oriented in the direction orthogonal to the bisectrix of its angle. Let q_1 and q_2 be the Cartesian coordinates of the centre of the robot, q_3 the vehicle orientation, while q_4, q_5, and q_6 represent the angle of rotation of each wheel around its axis. Also, denote by r the radius of the wheels and by ℓ the distance between the centre of the robot and the centre of each wheel. This mechanical system is subject to the following Pfaffian constraints:

$$\boldsymbol{A}_1^T(\boldsymbol{q}) \begin{bmatrix} \dot{q}_1 \\ \dot{q}_2 \\ \dot{q}_3 \end{bmatrix} + \boldsymbol{A}_2^T \begin{bmatrix} \dot{q}_4 \\ \dot{q}_5 \\ \dot{q}_6 \end{bmatrix} = 0,$$

where

$$\boldsymbol{A}_1(q) = \begin{bmatrix} \dfrac{\sqrt{3}}{2}\cos q_3 - \dfrac{1}{2}\sin q_3 & \sin q_3 & -\dfrac{1}{2}\sin q_3 - \dfrac{\sqrt{3}}{2}\cos q_3 \\ \dfrac{1}{2}\cos q_3 + \dfrac{\sqrt{3}}{2}\sin q_3 & -\cos q_3 & \dfrac{1}{2}\cos q_3 - \dfrac{\sqrt{3}}{2}\sin q_3 \\ \ell & \ell & \ell \end{bmatrix}$$

and $\boldsymbol{A}_2 = r\,\boldsymbol{I}_3$. Show that this set of constraints is partially integrable, and that in particular the orientation of the vehicle is a linear function of the wheel rotation angles. [*Hint*: add the kinematic constraints side-by-side.]

11.3. Show that, for a single kinematic constraint in the form (11.7), the integrability condition expressed by (11.9) coincides with the involutivity of the distribution $\Delta = \text{span}\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_{n-1}\}$ associated with the input vector fields of the corresponding kinematic model.

11.4. Using the controllability condition (11.11), show that a set of Pfaffian constraints that does not depend on the generalized coordinates is always integrable.

11.5. With reference to the kinematic model (11.13) of the unicycle, consider the following sequence of velocity inputs:

$$\begin{array}{llll} v(t) = 1 & \omega(t) = 0, & t \in [0, \varepsilon) \\ v(t) = 0 & \omega(t) = 1 & t \in [\varepsilon, 2\varepsilon) \\ v(t) = -1 & \omega(t) = 0, & t \in [2\varepsilon, 3\varepsilon) \\ v(t) = 0 & \omega = -1 & t \in [3\varepsilon, 4\varepsilon). \end{array}$$

By forward integration of the model equations, show that when ε is infinitesimal the displacement of the unicycle in configuration space is aligned with the Lie bracket of the input vector fields.

11.6. Prove the relationships (11.14) that allow the transformation of the velocity inputs of a differential drive vehicle to those of the equivalent unicycle.

[*Hint*: for the velocity of the midpoint of the segment joining the two wheels, it is sufficient to differentiate with respect to time the Cartesian coordinates of such a point expressed as a function of the coordinates of the wheel centres. As for ω, use the formula (B.4) for the composition of the velocities of a rigid body.]

11.7. For a generic configuration of a bicycle mobile robot, determine the Cartesian position of the instantaneous centre of rotation, and derive the expression of the angular velocity of the body as a function of the robot configuration $\boldsymbol{q}$ and of the modulus of the velocity of rear wheel centre. In the particular case of the rear-wheel drive bicycle, show that such expression is consistent with the evolution of θ predicted by the kinematic model (11.19). Moreover, compute the velocity v_P of a generic point P on the robot chassis.

11.8. Derive the kinematic model of the tricycle robot towing N trailers considered in Problem 11.1. Denote by ℓ the distance between the front wheel and rear wheel axle of the tricycle, and by ℓ_i the joint-to-joint length of the i-th trailer.

11.9. Consider a differential drive robot whose angular speed inputs — one for the right wheel and one for the left wheel — are subject to the following bounds:

$$|\omega_R(t)| \leq \omega_{RL} \qquad |\omega_L(t)| \leq \omega_{RL} \qquad \forall t,$$

that correspond to a square admissible region in the ω_R, ω_L plane. Derive the expression of the resulting constraints for the driving and steering velocity v and ω of the equivalent unicycle model. In particular, show that the admissible region in the v, ω plane is a rhombus.

11.10. Compute the expression of matrix $\boldsymbol{D}(\Delta)$ and vector $\boldsymbol{d}(\boldsymbol{z}_i, \boldsymbol{z}_f, \Delta)$ in (11.50) for the case of the (2,4) chained form.

11.11. Modify the path planning scheme based on parameterized inputs so as to obtain $s_f = 1$.

11.12. Show that the path planning schemes based on polynomials of different degree and on parameterized inputs give exactly the same result in the case of the (2,3) chained form.

11.13. Implement in a computer program the path planning method for a unicycle based on cubic Cartesian polynomials. Use the program to plan a path leading the robot from the configuration $\boldsymbol{q}_i = [\,0 \quad 0 \quad 0\,]^T$ [m,m,rad] to the configuration $\boldsymbol{q}_f = [\,2 \quad 1 \quad \pi/2\,]^T$ [m,m,rad]. Then, determine a timing law over the path so as to satisfy the following velocity bounds:

$$|v(t)| \leq 1\,\text{m/s} \qquad |\omega(t)| \leq 1\,\text{rad/s} \qquad \forall t.$$

11.14. Formulate the trajectory tracking problem for a (2,3) chained form and derive the corresponding error dynamics. Derive a feedback controller using the linear approximation around the reference trajectory.

11.15. Consider the kinematic model (11.19) of the rear-wheel drive bicycle. In analogy to the case of the unicycle, identify two outputs y_1, y_2 for which it is possible to perform a static input/output linearization. [*Hint*: consider a point P on the line passing through the centre of the front wheel and oriented as the wheel itself.]

11.16. Implement in a computer program the Cartesian regulator (11.78), (11.79), including a modification to allow the unicycle to reach the origin either in forward or in backward motion. [*Hint*: modify (11.79) so as to force the robot to orient itself as vector $\boldsymbol{e}_p$ or vector $-\boldsymbol{e}_p$, depending on which choice is the most convenient.]

11.17. Prove formulae (11.85) for the exact odometric localization of a unicycle under velocity inputs that are constant in the sampling interval.

11.18. Implement in a computer program the unicycle posture regulator based on polar coordinates, with the state feedback computed through the Runge–Kutta odometric localization method.

12

Motion Planning

The trajectory planning methods presented in Chaps. 4 and 11, respectively for manipulators and mobile robots, operate under the simplifying assumption that the workspace is empty. In the presence of obstacles, it is necessary to plan motions that enable the robot to execute the assigned task without colliding with them. This problem, referred to as *motion planning*, is the subject of this chapter. After defining a canonical version of the problem, the concept of *configuration space* is introduced in order to achieve an efficient formulation. A selection of representative planning techniques is then presented. The method based on the notion of *retraction* characterizes the connectivity of the free configuration space using a *roadmap*, i.e., a set of collision-free paths, while the *cell decomposition* method identifies a network of *channels* with the same property. The *PRM* and *bidirectional RRT* techniques are *probabilistic* in nature and rely on the randomized sampling of the configuration space and the memorization of those samples that do not cause a collision between the robot and the obstacles. The *artificial potential* method is also described as a heuristic approach particularly suited to *on-line* planning problems, where the geometry of the workspace obstacles is unknown in advance. The chapter ends with a discussion of the application of the presented planning methods to the *robot manipulator* case.

12.1 The Canonical Problem

Robotic systems are expected to perform tasks in a workspace that is often populated by physical objects, which represent an obstacle to their motion. For example, a manipulator working in a robotized cell must avoid collision with its static structures, as well as with other moving objects that may access it, such as other manipulators. Similarly, a mobile robot carrying baggage in an airport has to navigate among obstacles that may be fixed (fittings, conveyor belts, construction elements) or mobile (passengers, workers). Planning a motion amounts then to deciding which path the robot must follow in order

to execute a transfer task from an initial to a final posture without colliding with the obstacles. Clearly, one would like to endow the robot with the capability of *autonomously* planning its motion, starting from a high-level description of the task provided by the user and a geometric characterization of the workspace, either made available entirely in advance (*off-line* planning) or gathered by the robot itself during the motion by means of on-board sensors (*on-line* planning).

However, developing automatic methods for motion planning is a very difficult endeavour. In fact, the spatial reasoning that humans instinctively use to move safely among obstacles has proven hard to replicate and codify in an algorithm that can be executed by a robot. To this date, motion planning is still an active topic of research, with contributions coming from different areas such as algorithm theory, computational geometry and automatic control.

To address the study of methods for motion planning, it is convenient to introduce a version of the problem that highlights its fundamental issues. The *canonical problem* of motion planning is then formulated as follows.

Consider a robot $\mathcal{B}$, which may consist of a single rigid body (mobile robot) or of a kinematic chain whose base is either fixed (standard manipulator) or mobile (mobile robot with trailers or mobile manipulator). The robot moves in a Euclidean space $\mathcal{W} = \mathbb{R}^N$, with $N = 2$ or 3, called *workspace*. Let $\mathcal{O}_1, \ldots, \mathcal{O}_p$ be the *obstacles*, i.e., fixed rigid objects in $\mathcal{W}$. It is assumed that both the geometry of $\mathcal{B}$, $\mathcal{O}_1, \ldots, \mathcal{O}_p$ and the pose of $\mathcal{O}_1, \ldots, \mathcal{O}_p$ in $\mathcal{W}$ are known. Moreover, it is supposed that $\mathcal{B}$ is *free-flying*, that is, the robot is not subject to any kinematic constraint. The motion planning problem is the following: given an initial and a final posture of $\mathcal{B}$ in $\mathcal{W}$, find if exists a *path*, i.e., a continuous sequence of postures, that drives the robot between the two postures while avoiding collisions (including contacts) between $\mathcal{B}$ and the obstacles $\mathcal{O}_1, \ldots, \mathcal{O}_p$; report a failure if such a path does not exist.

In the particular case in which the robot is a single body moving in $\mathbb{R}^2$, the canonical motion planning problem is also known as the *piano movers' problem*, as it captures the difficulties faced by movers when manoeuvring a piano (without lifting it) among obstacles. The *generalized movers' problem* is the canonical problem for a single-body robot moving in $\mathbb{R}^3$.

Clearly, some of the hypotheses of the canonical problem may not be satisfied in applications. For example, the assumption that the robot is the only object in motion in the workspace rules out the relevant case of moving obstacles (e.g., other robots). Advance knowledge of obstacle geometry and placement is another strong assumption: especially in *unstructured* environments, which are not purposely designed to accommodate robots, the robot itself is typically in charge of detecting obstacles by means of its sensors, and the planning problem must therefore be solved on-line during the motion. Moreover, as shown in Chap. 11, the free-flying robot hypothesis does not hold in nonholonomic mechanical systems, which cannot move along arbitrary paths in the workspace. Finally, manipulation and assembly problems are excluded from the canonical formulation since they invariably involve contacts between rigid

bodies. As a matter of fact, all the above assumptions are introduced in order to reduce motion planning to the purely geometrical — but still quite difficult — problem of generating a collision-free path. However, many methods that successfully solve this simplified version of the problem lend themselves to an extension to more general versions.

The notion of configuration space is essential to obtain a more convenient formulation of the canonical motion planning problem, as well as to envisage approaches to its solution.

12.2 Configuration Space

A very effective scheme for motion planning is obtained by representing the robot as a mobile point in an appropriate space, where the images of the workspace obstacles are also reported. To this end, it is natural to refer to the generalized coordinates of the mechanical system, whose value identifies the *configuration* of the robot (see Sect. B.4). This associates to each posture of the latter a point in the *configuration space* $\mathcal{C}$, i.e., the set of all the configurations that the robot can assume.

Generalized coordinates of robots are essentially of two types. Cartesian coordinates are used to describe the position of selected points on the links of the kinematic chain and take value in Euclidean spaces. Angular coordinates are used to represent the orientations of the bodies; independently from the adopted representation (rotation matrices, Euler angles, quaternions), they take values in $SO(m)$ ($m = 2, 3$), the *special orthonormal group* of real $(m \times m)$ matrices with orthonormal columns and determinant equal to 1 (see Sect. 2.2). It is well known that a minimal parameterization of $SO(m)$ requires $m(m-1)/2$ parameters. The configuration space of a robot is then obtained in general as a Cartesian product of these spaces.

Some examples of configuration spaces are presented below:

- The configuration of a polygonal mobile robot in $\mathcal{W} = \mathbb{R}^2$ is described by the position of a representative point on the body (e.g., a vertex) and by the orientation of the polygon, both expressed with respect to a fixed reference frame. The configuration space $\mathcal{C}$ is then $\mathbb{R}^2 \times SO(2)$, whose dimension is $n = 3$.
- For a polyhedral mobile robot in $\mathcal{W} = \mathbb{R}^3$, the configuration space $\mathcal{C}$ is $\mathbb{R}^3 \times SO(3)$, whose dimension is $n = 6$.
- For a fixed-base planar manipulator with n revolute joints, the configuration space is a subset of $(\mathbb{R}^2 \times SO(2))^n$. The dimension of $\mathcal{C}$ equals the dimension of $(\mathbb{R}^2 \times SO(2))^n$ minus the number of constraints due to the presence of the joints, i.e., $3n-2n = n$. In fact, in a planar kinematic chain, each joint imposes two holonomic constraints on the following body.
- For a fixed-base spatial manipulator with n revolute joints, the configuration space is a subset of $(\mathbb{R}^3 \times SO(3))^n$. Since in this case each joint imposes five constraints on the following body, the dimension of $\mathcal{C}$ is $6n - 5n = n$.

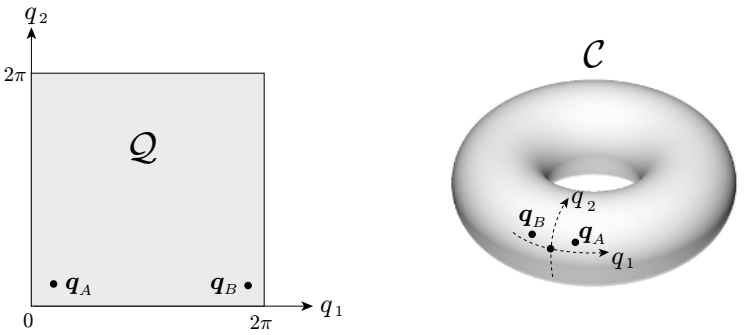

Fig. 12.1. The configuration space of a 2R manipulator; *left*: a locally valid representation as a subset of $\mathbb{R}^2$, *right*: a topologically correct representation as a two-dimensional torus

- For a unicycle-like vehicle with a trailer in $\mathbb{R}^2$, the configuration space is a subset of $(\mathbb{R}^2 \times SO(2)) \times (\mathbb{R}^2 \times SO(2))$. If the trailer is connected to the unicycle by a revolute joint, the configuration of the robot can be described by the position and orientation of the unicycle and the orientation of the trailer. The dimension of $\mathcal{C}$ is therefore $n = 4$.

If n is the dimension of $\mathcal{C}$, a configuration in $\mathcal{C}$ can be described by a vector $\boldsymbol{q} \in \mathbb{R}^n$. However, this description is only valid locally: the geometric structure of the configuration space $\mathcal{C}$ is in general more complex than that of a Euclidean space, as shown in the following example.

Example 12.1

Consider the planar manipulator with two revolute joints (2R manipulator) of Fig. 2.14. The configuration space has dimension 2, and may be locally represented by $\mathbb{R}^2$, or more precisely by its subset

$$\mathcal{Q} = \{\boldsymbol{q} = (q_1, q_2) : q_1 \in [0, 2\pi), q_2 \in [0, 2\pi)\}.$$

This guarantees that the representation is injective, i.e., that a single value of $\boldsymbol{q}$ exists for each manipulator posture. However, this representation is not topologically correct: for example, the configurations denoted as $\boldsymbol{q}_A$ and $\boldsymbol{q}_B$ in Fig. 12.1, left, which correspond to manipulator postures that are 'close' in the workspace $\mathcal{W}$, appear to be 'far' in $\mathcal{Q}$. To take this into account, one should 'fold' the square $\mathcal{Q}$ onto itself (so as to make opposite sides meet) in sequence along its two axes. This procedure generates a ring, properly called *torus*, which can be visualized as a two-dimensional surface immersed in $\mathbb{R}^3$ (Fig. 12.1, right). The correct expression of this space is $SO(2) \times SO(2)$.

When the configuration of a robot (either articulated or mobile) includes angular generalized coordinates, its configuration space is properly described as an n-dimensional *manifold*, i.e., a space in which a neighbourhood of a point can be put in correspondence with $\mathbb{R}^n$ through a continuous bijective function whose inverse is also continuous (a *homeomorphism*).

12.2.1 Distance

Having discussed the nature of the robot configuration space, it is useful to define a distance function in $\mathcal{C}$. In fact, the planning methods that will be discussed in the following make use of this notion.

Given a configuration $\boldsymbol{q}$, let $\mathcal{B}(\boldsymbol{q})$ be the subset of the workspace $\mathcal{W}$ occupied by the robot $\mathcal{B}$, and $p(\boldsymbol{q})$ be the position in $\mathcal{W}$ of a point p on $\mathcal{B}$. Intuition suggests that the distance between two configurations $\boldsymbol{q}_1$ and $\boldsymbol{q}_2$ should go to zero when the two regions $\mathcal{B}(\boldsymbol{q}_1)$ and $\mathcal{B}(\boldsymbol{q}_2)$ tend to coincide. A definition that satisfies this property is

$$d_1(\boldsymbol{q}_1, \boldsymbol{q}_2) = \max_{p \in \mathcal{B}} \|p(\boldsymbol{q}_1) - p(\boldsymbol{q}_2)\|, \tag{12.1}$$

where $\|\cdot\|$ denotes the Euclidean distance in $\mathcal{W} = \mathbb{R}^N$. In other words, the distance between two configurations in $\mathcal{C}$ is the maximum displacement in $\mathcal{W}$ they induce on a point, as the point moves all over the robot.

However, the use of function d_1 is cumbersome, because it requires characterizing the volume occupied by the robot in the two configurations and the computation of the maximum distance in $\mathcal{W}$ between corresponding points. For algorithmic purposes, the simple Euclidean norm is often chosen as a configuration space distance:

$$d_2(\boldsymbol{q}_1, \boldsymbol{q}_2) = \|\boldsymbol{q}_1 - \boldsymbol{q}_2\|. \tag{12.2}$$

Nevertheless, one must keep in mind that this definition is appropriate only when $\mathcal{C}$ is a Euclidean space. Going back to Example 12.1, it is easy to realize that, unlike $d_1(\boldsymbol{q}_A, \boldsymbol{q}_B)$, the Euclidean norm $d_2(\boldsymbol{q}_A, \boldsymbol{q}_B)$ does not represent correctly the distance on the torus. A possible solution is to modify the definition of d_2 by suitably computing differences of angular generalized coordinates (see Problem 12.2).

12.2.2 Obstacles

In order to characterize paths that represent a solution to the canonical motion planning problem — those that avoid collisions between the robot and the workspace obstacles — it is necessary to build the 'images' of the obstacles in the configuration space of the robot.

In the following, it is assumed that the obstacles are closed (i.e., they contain their boundaries) but not necessarily limited subsets of $\mathcal{W}$. Given an

obstacle $\mathcal{O}_i$ $(i = 1, \ldots, p)$ in $\mathcal{W}$, its image in configuration space $\mathcal{C}$ is called $\mathcal{C}$-*obstacle* and is defined as

$$\mathcal{CO}_i = \{\boldsymbol{q} \in \mathcal{C} : \mathcal{B}(\boldsymbol{q}) \cap \mathcal{O}_i \neq \emptyset\}. \tag{12.3}$$

In other words, $\mathcal{CO}_i$ is the subset of configurations that cause a collision (including simple contacts) between the robot $\mathcal{B}$ and the obstacle $\mathcal{O}_i$ in the workspace. The union of all $\mathcal{C}$-obstacles

$$\mathcal{CO} = \bigcup_{i=1}^{p} \mathcal{CO}_i \tag{12.4}$$

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

$$\mathcal{C}_{\text{free}} = \mathcal{C} - \mathcal{CO} = \{\boldsymbol{q} \in \mathcal{C} : \mathcal{B}(\boldsymbol{q}) \cap \left(\bigcup_{i=1}^{p} \mathcal{O}_i\right) = \emptyset\} \tag{12.5}$$

is the *free configuration space*, that is, the subset of robot configurations that do not cause collision with the obstacles. A path in configuration space is called *free* if it is entirely contained in $\mathcal{C}_{\text{free}}$.

Although $\mathcal{C}$ in itself is a connected space — given two arbitrary configuration there exists a path that joins them — the free configuration space $\mathcal{C}_{\text{free}}$ may not be connected as a consequence of occlusions due to $\mathcal{C}$-obstacles. Note also that the assumption of free-flying robot in the canonical problem means that the robot can follow any path in the free configuration space $\mathcal{C}_{\text{free}}$.

It is now possible to give a more compact formulation of the canonical motion planning problem. Assume that the initial and final posture of the robot $\mathcal{B}$ in $\mathcal{W}$ are mapped to the corresponding configurations in $\mathcal{C}$, respectively called *start* configuration $\boldsymbol{q}_s$ and *goal* configuration $\boldsymbol{q}_g$. Planning a collision-free motion for the robot means then generating a safe path between $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$ if they belong to the same connected component of $\mathcal{C}_{\text{free}}$, and reporting a failure otherwise.

12.2.3 Examples of Obstacles

In the following, the $\mathcal{C}$-obstacle generation procedure is presented in some representative cases. For the sake of simplicity, it is assumed that obstacles in $\mathcal{W}$ are either polygonal or polyhedral.

Example 12.2

Consider the case of a point robot $\mathcal{B}$. In this case, the configuration of the robot is described by the coordinates of point $\mathcal{B}$ in the workspace $\mathcal{W} = \mathbb{R}^N$ and the configuration space $\mathcal{C}$ is a copy of $\mathcal{W}$. Similarly, the $\mathcal{C}$-obstacles are copies of the obstacles in $\mathcal{W}$.

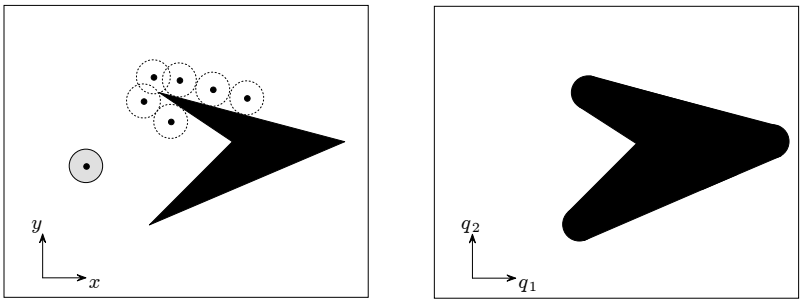

Fig. 12.2. $\mathcal{C}$-obstacles for a circular robot in $\mathbb{R}^2$; *left*: the robot $\mathcal{B}$, an obstacle $\mathcal{O}_i$ and the growing procedure for building $\mathcal{C}$-obstacles, *right*: the configuration space $\mathcal{C}$ and the $\mathcal{C}$-obstacle $\mathcal{CO}_i$

Example 12.3

If the robot is a sphere[1] in $\mathcal{W} = \mathbb{R}^N$, its configuration can be described by the Cartesian coordinates of a representative point, e.g., its centre — note that the orientation of the sphere is irrelevant for collision checking. Therefore, as in the previous example, the configuration space $\mathcal{C}$ is a copy of the workspace $\mathcal{W}$. However, the $\mathcal{C}$-obstacles are no longer simple copies of the obstacles in $\mathcal{W}$, and they must be built through a *growing* procedure. In particular, the boundary of the $\mathcal{C}$-obstacle $\mathcal{CO}_i$ is the locus of configurations that put the robot is in contact with the obstacle $\mathcal{O}_i$, and it can be obtained as the surface described by the representative point as the robot slides on the boundary of $\mathcal{O}_i$. As a consequence, to build $\mathcal{CO}_i$ it is sufficient to grow $\mathcal{O}_i$ isotropically by the radius of the robot. This procedure is shown in Fig. 12.2 for the case $N = 2$ (circular robot in $\mathbb{R}^2$); in this case, each $\mathcal{C}$-obstacle is a *generalized polygon*, i.e., a planar region whose boundary consists of line segments and/or circular arcs. If the representative point of the robot is different from the centre of the sphere, the growing procedure is not isotropic.

Example 12.4

Consider now the case of a polyhedral robot that is free to translate (with a fixed orientation) in $\mathbb{R}^N$. Its configuration can be described by the Cartesian coordinates of a representative point, for example a vertex of the polyhedron. Therefore, the configuration space $\mathcal{C}$ is again a copy of $\mathbb{R}^N$. Again, a growing procedure must be applied to the workspace obstacles to obtain their image in the configuration space. In particular, the boundary of the $\mathcal{C}$-obstacle $\mathcal{CO}_i$ is the surface described by the representative point of the robot when the robot $\mathcal{B}$ slides at a fixed orientation on the boundary of $\mathcal{O}_i$. Figure 12.3 shows this procedure for the case $N = 2$ (polygonal robot in $\mathbb{R}^2$). The resulting shape of $\mathcal{CO}_i$ depends on the position of the representative point on the robot, but in any case the $\mathcal{C}$-obstacle is itself a polyhedron. $\mathcal{CO}_i$ has in general a larger number of vertices than $\mathcal{O}_i$, and is a convex polyhedron provided that $\mathcal{B}$ and $\mathcal{O}_i$ are convex. Note also that, although the result of the growing

[1] For simplicity, the term *sphere* will be used in Euclidean spaces of arbitrary dimension n in place of *n–sphere*.

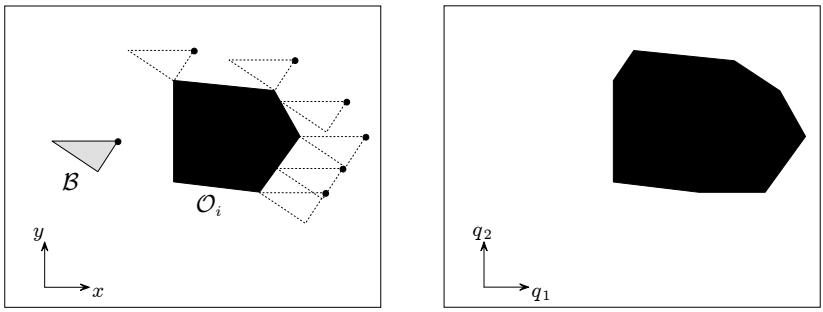

Fig. 12.3. $\mathcal{C}$-obstacles for a polygonal robot translating in $\mathbb{R}^2$; *left*: the robot $\mathcal{B}$, an obstacle $\mathcal{O}_i$ and the growing procedure for building $\mathcal{C}$-obstacles, *right*: the configuration space $\mathcal{C}$ and the $\mathcal{C}$-obstacle $\mathcal{CO}_i$

procedure — and thus the shape of the $\mathcal{C}$-obstacles — depends on the choice of the representative point on the robot, all the obtained planning problems in configuration space are equivalent. In particular, the existence of a solution for any of them implied the existence of a solution for all the others. Moreover, to each path in configuration space that solves one of these problems corresponds a (different) free path in any of the others, to which the same motion of the robot in the workspace is associated.

Example 12.5

For a polyhedral robot that can translate and rotate in $\mathbb{R}^N$, the dimension of the configuration space is increased with respect to the previous example, because it is also necessary to describe the orientation DOFs. For example, consider the case of a polygon that can translate and rotate in $\mathbb{R}^2$. The configuration of the robot can be characterized by the Cartesian coordinates of a representative point (for example, a vertex of the polygon) and an angular coordinate θ representing the orientation of the polygon with respect to a fixed reference frame. The configuration space $\mathcal{C}$ is then $\mathbb{R}^2 \times SO(2)$, which can be locally represented by $\mathbb{R}^3$. To build the image in $\mathcal{C}$ of an obstacle $\mathcal{O}_i$, one should in principle repeat the procedure illustrated in Fig. 12.3 for each possible value of the robot orientation θ. The $\mathcal{C}$-obstacle $\mathcal{CO}_i$ is the volume generated by 'stacking' (in the direction of the θ axis) all the constant-orientation slices thus obtained.

Example 12.6

For a robot manipulator $\mathcal{B}$ made by rigid links $\mathcal{B}_1, \ldots, \mathcal{B}_n$ connected by joints, there exist in principle two kinds of $\mathcal{C}$-obstacles: those that represent the collision between a body $\mathcal{B}_i$ and an obstacle $\mathcal{O}_j$, and those accounting for *self-collisions*, i.e., interference between two links $\mathcal{B}_i$ and $\mathcal{B}_j$ of the kinematic chain. Even considering for simplicity only the first type, the procedure for building $\mathcal{C}$-obstacles is much more complicated than in the previous examples. In fact, to obtain the boundary of the $\mathcal{C}$-obstacle $\mathcal{CO}_i$, it is necessary to identify through appropriate inverse kinematics computations all the configurations that bring one or more links of the manipulator $\mathcal{B}$ in contact with $\mathcal{O}_i$. Figure 12.4 shows the result of the $\mathcal{C}$-obstacle building

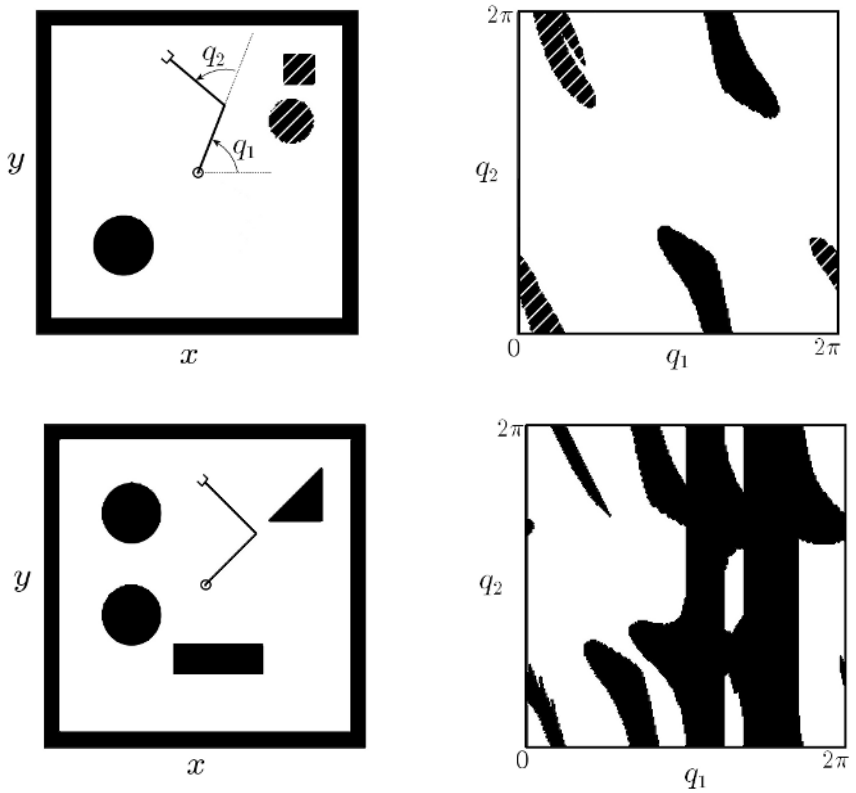

Fig. 12.4. $\mathcal{C}$-obstacles for a wire-frame 2R manipulator in two different cases; *left*: the robot and the obstacles in $\mathcal{W} = \mathbb{R}^2$, *right*: the configuration space $\mathcal{C}$ and the $\mathcal{C}$-obstacle region $\mathcal{CO}$

procedure for a wire-frame 2R manipulator in two different cases (self-collisions are not considered); note how, in spite of the simple shape of the obstacles in $\mathcal{W}$, the profile of the $\mathcal{C}$-obstacles is quite complicated. For simplicity, the configuration space has been represented as a subset (a square) of $\mathbb{R}^2$; however, to correctly visualize the $\mathcal{C}$-obstacles, one should keep in mind that the correct representation of $\mathcal{C}$ is a two-dimensional torus, so that the upper/lower and left/right sides of the square are actually coincident. Note also that in the first case (top of Fig. 12.4) the images of the two obstacles in the upper right corner of $\mathcal{W}$ merge in a single $\mathcal{C}$-obstacle, and the free configuration space $\mathcal{C}_{\text{free}}$ consists of a single connected component. In the second case (bottom of Fig. 12.4) $\mathcal{C}_{\text{free}}$ is instead partitioned into three disjoint connected components.

Whatever the nature of the robot, an *algebraic* model of the workspace obstacles is needed (e.g., derived from a CAD model of the workspace) to compute exactly the $\mathcal{C}$-obstacle region $\mathcal{CO}$. However, except for the most elementary cases, the procedures for generating $\mathcal{CO}$ are extremely complex. As a consequence, it is often convenient to resort to approximate representations of $\mathcal{CO}$. A simple (although computationally intensive) way to build such a representation is to extract configuration samples from $\mathcal{C}$ using a regular grid, compute the corresponding volume occupied by the robot via direct kinematics, and finally identify through *collision checking*[2] those samples that bring the robot to collide with obstacles. These samples can be considered as a discrete representation of $\mathcal{CO}$, whose accuracy can be improved at will by increasing the resolution of the grid in $\mathcal{C}$.

Some of the motion planning methods that will be presented in this chapter do not require an explicit computation of the $\mathcal{C}$-obstacle region. In particular, this is true for the probabilistic planners described in Sect. 12.5, and for the technique based on artificial potential fields and control points discussed in Sect. 12.7.

12.3 Planning via Retraction

The basic idea of motion planning via retraction is to represent the free configuration space by means of a *roadmap* $\mathcal{R} \subset \mathcal{C}_{\text{free}}$, i.e., a network of paths that describe adequately the connectivity of $\mathcal{C}_{\text{free}}$. The solution of a particular instance of a motion planning problem is then obtained by connecting (*retracting*) to the roadmap the start configuration $\boldsymbol{q}_s$ and the goal configuration $\boldsymbol{q}_g$, and finding a path on $\mathcal{R}$ between the two connection points. Depending on the type of roadmap, and on the retraction procedure, this general approach leads to different planning methods. One of these is described in the following, under the simplifying assumption that $\mathcal{C}_{\text{free}}$ is a limited subset of $\mathcal{C} = \mathbb{R}^2$ and is *polygonal*, i.e., its boundary is entirely made of line segments.[3] As the boundary of $\mathcal{C}_{\text{free}}$ coincides with the boundary of $\mathcal{CO}$, this assumption implies that the $\mathcal{C}$-obstacle region is itself a polygonal subset of $\mathcal{C}$.

For each configuration $\boldsymbol{q}$ in $\mathcal{C}_{\text{free}}$, let its *clearance* be defined as

$$\gamma(\boldsymbol{q}) = \min_{\boldsymbol{s} \in \partial\mathcal{C}_{\text{free}}} \|\boldsymbol{q} - \boldsymbol{s}\|, \tag{12.6}$$

where $\partial\mathcal{C}_{\text{free}}$ is the boundary of $\mathcal{C}_{\text{free}}$. The clearance $\gamma(\boldsymbol{q})$ represents the minimum Euclidean distance between the configuration $\boldsymbol{q}$ and the $\mathcal{C}$-obstacle re-

[2] Many algorithms based on computational geometry techniques are available (in the literature, but also implemented in software packages) to test collision in $\mathbb{R}^2$ o $\mathbb{R}^3$. The most efficient, such as *I-Collide* and *V-Collide*, use a hierarchical representation of geometric models of bodies, and can speed up collision checking by re-using the results of previous checks in spatially similar situations.

[3] According to the definition, a polygonal subset is not necessarily connected, and may contain 'holes'.

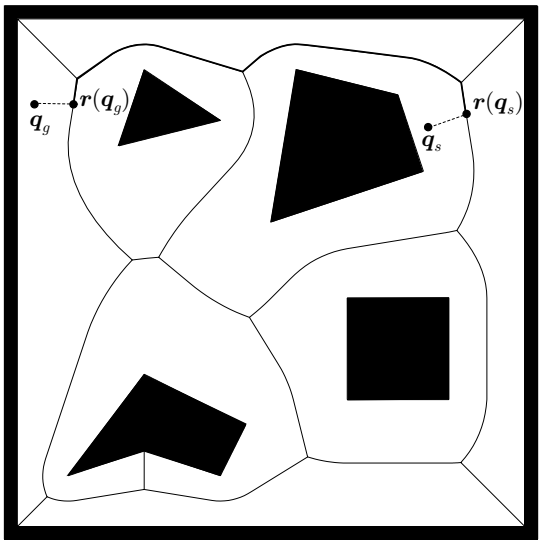

Fig. 12.5. An example of generalized Voronoi diagram and the solution of a particular instance of the planning problem, obtained by retracting $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$ on the diagram. The solution path from $\boldsymbol{q}_s$ to $\boldsymbol{q}_g$ consists of the two *dashed* segments and the *thick* portion of the diagram joining them

gion. Moreover, consider the set of points on the boundary of $\mathcal{C}_{\text{free}}$ that are *neighbours* of $\boldsymbol{q}$:

$$N(\boldsymbol{q}) = \{\boldsymbol{s} \in \partial\mathcal{C}_{\text{free}} : \|\boldsymbol{q} - \boldsymbol{s}\| = \gamma(\boldsymbol{q})\}, \tag{12.7}$$

i.e., the points on $\partial\mathcal{C}_{\text{free}}$ that determine the value of the clearance for $\boldsymbol{q}$. With these definitions, the *generalized*[4] *Voronoi diagram* of $\mathcal{C}_{\text{free}}$ is the locus of its configurations having more than one neighbour:

$$\mathcal{V}(\mathcal{C}_{\text{free}}) = \{\boldsymbol{q} \in \mathcal{C}_{\text{free}} : \text{card}(N(\boldsymbol{q})) > 1\}, \tag{12.8}$$

in which $\text{card}(\cdot)$ denotes the cardinality of a set. Figure 12.5 shows an example of generalized Voronoi diagram for a polygonal free configuration space; note how the connectivity of $\mathcal{C}_{\text{free}}$ is well captured by the diagram.

It is easy to show that $\mathcal{V}(\mathcal{C}_{\text{free}})$ is made of elementary arcs that are either rectilinear — each made of contiguous configurations whose clearance is due to the same pair of edges or vertices — or parabolic — each made of contiguous configurations whose clearance is determined by the same pair edge-vertex. Therefore, one can build an analytical expression of $\mathcal{V}(\mathcal{C}_{\text{free}})$ starting from the pair of features (side/side, side/vertex, vertex/vertex) that determine the

[4] A proper *Voronoi diagram* is obtained in the particular case in which the $\mathcal{C}$-obstacles are isolated points.

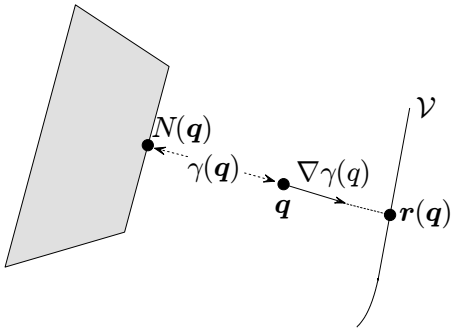

Fig. 12.6. The retraction procedure for connecting a generic configuration $\boldsymbol{q}$ in $\mathcal{C}_{\text{free}}$ to $\mathcal{V}(\mathcal{C}_{\text{free}})$

appearance of each arc. From an abstract point of view, $\mathcal{V}(\mathcal{C}_{\text{free}})$ can be considered as a *graph* having elementary arcs of the diagram as *arcs* and endpoints of arcs as *nodes*.

By construction, the generalized Voronoi diagram has the property of locally maximizing clearance, and is therefore a natural choice as a roadmap of $\mathcal{C}_{\text{free}}$ for planning motions characterized by a healthy safety margin with respect to the possibility of collisions. To use $\mathcal{V}(\mathcal{C}_{\text{free}})$ as a roadmap, a retraction procedure must be defined for connecting a generic configuration in $\mathcal{C}_{\text{free}}$ to the diagram. To this end, consider the geometric construction shown in Fig. 12.6. Since $\boldsymbol{q} \notin \mathcal{V}(\mathcal{C}_{\text{free}})$, it is $\text{card}\big(N(\boldsymbol{q})\big) = 1$, i.e., there exists a single point on the polygonal boundary of $\mathcal{CO}$ (either a vertex or a point on a side) that determines the value of the clearance $\gamma(\boldsymbol{q})$. The gradient $\nabla\gamma(\boldsymbol{q})$, which identifies the direction of steepest ascent for the clearance at the configuration $\boldsymbol{q}$, is directed as the half-line originating in $N(\boldsymbol{q})$ and passing through $\boldsymbol{q}$. The first intersection of this half-line with $\mathcal{V}(\mathcal{C}_{\text{free}})$ defines $\boldsymbol{r}(\boldsymbol{q})$, i.e., the connection point of $\boldsymbol{q}$ to the generalized Voronoi diagram. To guarantee that $\boldsymbol{r}(\cdot)$ is continuous, it is convenient to extend its domain of definition to all $\mathcal{C}_{\text{free}}$ by letting $\boldsymbol{r}(\boldsymbol{q}) = \boldsymbol{q}$ if $\boldsymbol{q} \in \mathcal{V}(\mathcal{C}_{\text{free}})$.

From a topological viewpoint, the function $\boldsymbol{r}(\cdot)$ defined above is actually an example of *retraction* of $\mathcal{C}_{\text{free}}$ on $\mathcal{V}(\mathcal{C}_{\text{free}})$, i.e., a continuous surjective map from $\mathcal{C}_{\text{free}}$ to $\mathcal{V}(\mathcal{C}_{\text{free}})$ such that its restriction to $\mathcal{V}(\mathcal{C}_{\text{free}})$ is the identity map. In view of its definition, $\boldsymbol{r}(\cdot)$ preserves the connectivity of $\mathcal{C}_{\text{free}}$, in the sense that $\boldsymbol{q}$ and $\boldsymbol{r}(\boldsymbol{q})$ — as well as the segment joining them — always lie in the same connected component of $\mathcal{C}_{\text{free}}$. This property is particularly important, because it is possible to show that, given a generic retraction ρ of $\mathcal{C}_{\text{free}}$ on a connectivity-preserving roadmap $\mathcal{R}$, there exists a free path between two configurations $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$ if and only if there exists a path on $\mathcal{R}$ between $\rho(\boldsymbol{q}_s)$ and $\rho(\boldsymbol{q}_g)$. As a consequence, the problem of planning a path in $\mathcal{C}_{\text{free}}$ reduces to the problem of planning a path on its retraction $\mathcal{R} = \rho(\mathcal{C}_{\text{free}})$.

Given the start and goal configurations $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$, the motion planning method via retraction goes through the following steps (see Fig. 12.5):

1. Build the generalized Voronoi diagram $\mathcal{V}(\mathcal{C}_{\text{free}})$.
2. Compute the retractions $\boldsymbol{r}(\boldsymbol{q}_s)$ and $\boldsymbol{r}(\boldsymbol{q}_g)$ on $\mathcal{V}(\mathcal{C}_{\text{free}})$.
3. Search $\mathcal{V}(\mathcal{C}_{\text{free}})$ for a sequence of consecutive arcs such that $\boldsymbol{r}(\boldsymbol{q}_s)$ belongs to the first and $\boldsymbol{r}(\boldsymbol{q}_g)$ to the last.
4. If the search is successful, replace the first arc of the sequence with its subarc originating in $\boldsymbol{r}(\boldsymbol{q}_s)$ and the last arc of the sequence with its subarc terminating in $\boldsymbol{r}(\boldsymbol{q}_g)$, and provide as output the path consisting of the line segment joining $\boldsymbol{q}_s$ to $\boldsymbol{r}(\boldsymbol{q}_s)$, the modified arc sequence, and the line segment joining $\boldsymbol{q}_g$ to $\boldsymbol{r}(\boldsymbol{q}_g)$; otherwise, report a failure.

If simplicity of implementation is desired, the graph search[5] required at Step 3 can be performed using basic strategies such as breadth-first or depth-first search. On the other hand, if one wishes to identify the minimum-length path (among those which can be produced by the method) between $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$, each arc must be labelled with a cost equal to its actual length. The minimum-cost path can then be computed with an *informed* algorithm — i.e., an algorithm using a heuristic estimate of the minimum cost path from a generic node to the goal, in this case the Euclidean distance — such as $A^\star$.

Whatever the adopted search strategy, the above motion planning method via retraction of $\mathcal{C}_{\text{free}}$ on $\mathcal{V}(\mathcal{C}_{\text{free}})$ is *complete*, i.e., it is guaranteed to find a solution if one exists, and to report a failure otherwise. Its time complexity is a function of the number v of vertices of the polygonal region $\mathcal{C}_{\text{free}}$, and depends essentially on the construction of the generalized Voronoi diagram (Step 1), on the retraction of $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$ (Step 2), and on the search on the diagram (Step 3). As for Step 1, the most efficient algorithms can build $\mathcal{V}(\mathcal{C}_{\text{free}})$ in time $O(v \log v)$. The retraction procedure requires $O(v)$, mainly to compute $N(\boldsymbol{q}_s)$ and $N(\boldsymbol{q}_g)$. Finally, since it is possible to prove that $\mathcal{V}(\mathcal{C}_{\text{free}})$ has $O(v)$ arcs, the complexity of breadth-first or depth-first search would be $O(v)$, whereas $A^\star$ would require $O(v \log v)$. Altogether, the time complexity of the motion planning method via retraction is $O(v \log v)$.

It should be noted that, once the generalized Voronoi diagram of $\mathcal{C}_{\text{free}}$ has been computed, it can be used again to solve quickly other instances (*queries*) of the same motion planning problem, i.e., to generate collision-free paths between different start and goal configurations in the same $\mathcal{C}_{\text{free}}$. For example, this is useful when a robot must repeatedly move between different postures in the same static workspace. The motion planning method based on retraction can then be considered *multiple-query*. It is also possible to extend the method to the case in which the $\mathcal{C}$-obstacles are generalized polygons.

[5] See Appendix E for a quick survey on graph search strategies and algorithm complexity.

12.4 Planning via Cell Decomposition

Assume that the free configuration space $\mathcal{C}_{\text{free}}$ can be decomposed in simply-shaped regions, called *cells*, with the following basic characteristics:

- Given two configurations belonging to the same cell, it is 'easy' to compute a collision-free path that joins them.
- Given two adjacent cells — i.e., two cells having in common a portion of their boundaries of non-zero measure — it is 'easy' to generate a collision-free path going from one cell to the other.

Starting from one such cell decomposition of $\mathcal{C}_{\text{free}}$, it is easy to build the associated *connectivity graph*. The nodes of this graph represent cells, while an arc between two nodes indicates that the two corresponding cells are adjacent. By searching the connectivity graph for a path from the cell containing the start configuration $\boldsymbol{q}_s$ to the cell containing the goal configuration $\boldsymbol{q}_g$, one obtains (if it exists) a sequence of adjacent cells, called *channel*, from which it is possible — in view of the above mentioned characteristics of cells — to extract a path that joins $\boldsymbol{q}_s$ to $\boldsymbol{q}_g$ and is entirely contained in $\mathcal{C}_{\text{free}}$.

The general approach so far outlined generates different motion planning methods, essentially depending on the type of cells used for the decomposition. In the following, two algorithms are described, respectively based on exact and approximate cell decomposition of $\mathcal{C}_{\text{free}}$. As before, it is assumed that $\mathcal{C}_{\text{free}}$ is a limited polygonal subset of $\mathcal{C} = \mathbb{R}^2$.

12.4.1 Exact Decomposition

When an exact decomposition is used, the free configuration space is partitioned in a collection of cells whose union gives exactly $\mathcal{C}_{\text{free}}$. A typical choice for cells are convex polygons. In fact, convexity guarantees that the line segments joining two configurations belonging to the same cell lies entirely in the cell itself, and therefore in $\mathcal{C}_{\text{free}}$. Moreover, it is easy to travel safely between two adjacent cells by passing through the midpoint of the segment that constitutes their common boundary. A simple way to decompose $\mathcal{C}_{\text{free}}$ in a collection of convex polygons is to use the *sweep line* algorithm, which turns out to be useful in a number of computational geometry problems. In the present setting, the application of this algorithm proceeds as follows.

Choose a line that is not parallel to any side of the boundary of $\mathcal{C}_{\text{free}}$, and let it translate ('sweep') all over $\mathcal{C}_{\text{free}}$. Whenever the line passes through one of the vertices of $\mathcal{C}_{\text{free}}$, consider the two segments (*extensions*) that originate from the vertex, lie on the line and point in opposite directions, terminating at the first intersection with $\partial\mathcal{C}_{\text{free}}$. Every extension that lies in $\mathcal{C}_{\text{free}}$ (except for its endpoints) is part of the boundary of a cell; the rest of the boundary is made of (part of) sides of $\partial\mathcal{C}_{\text{free}}$ and possibly other extensions. This procedure is illustrated in Fig. 12.7, where a vertical sweep line has been used; note that some of the vertices of $\mathcal{C}_{\text{free}}$ contribute to the decomposition with a single or no

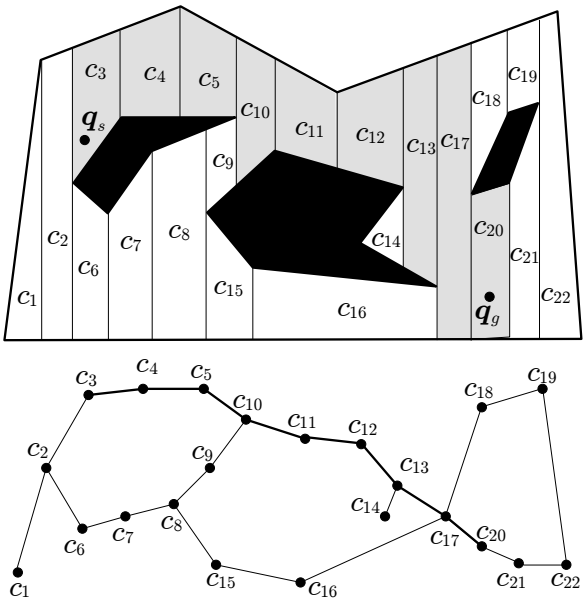

Fig. 12.7. An example of trapezoidal decomposition via the sweep line algorithm (*above*) and the associated connectivity graph (*below*). Also shown is the solution of a particular planning problem ($c_s = c_3$ and $c_g = c_{20}$), both as a channel in $\mathcal{C}_{\text{free}}$ and as a path on the graph

extension at all. The result is a special case of *convex polygonal decomposition*, that is called *trapezoidal* because its cells are trapezoids — triangular cells, if present, are regarded as degenerate trapezoids having one side of zero length.

After the decomposition of $\mathcal{C}_{\text{free}}$ has been computed, it is possible to build the associated *connectivity graph* C. This is the graph whose nodes are the cells of the decomposition, while an arc exists between two nodes if the corresponding cells are adjacent, i.e., the intersection of their boundary is a line segment of non-zero length; therefore, cells with side-vertex or vertex-vertex contacts are not adjacent. At this point, it is necessary to identify the cells c_s and c_g in the decomposition that respectively contain $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$, the start and goal configurations for the considered planning problem. A graph search algorithm can then be used to find a *channel* from c_s to c_g, i.e., a path on C that joins the two corresponding nodes (see Fig. 12.7). From the channel, which is a sequence of adjacent cells, one must extract a path in $\mathcal{C}_{\text{free}}$ going from $\boldsymbol{q}_s$ to $\boldsymbol{q}_g$. Since the interior of the channel is contained in $\mathcal{C}_{\text{free}}$ and the cells are convex polygons, such extraction is straightforward. For example, one may identify the midpoints of the segments that represent the common boundaries between consecutive cells of the channel, and connect them through a broken line starting in $\boldsymbol{q}_s$ and ending in $\boldsymbol{q}_g$.

Wrapping up, given the two configurations $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$, the motion planning algorithm via exact cell decomposition is based on the following steps:

1. Compute a convex polygonal (e.g., trapezoidal) decomposition of $\mathcal{C}_{\text{free}}$.
2. Build the associated connectivity graph C.
3. Search C for a channel, i.e., a sequence of adjacent cells from c_s to c_g.
4. If a channel has been found, extract and provide as output a collision-free path from $\boldsymbol{q}_s$ to $\boldsymbol{q}_g$; otherwise, report a failure.

As in motion planning via retraction, using a non-informed graph search algorithm in Step 3 will result in a channel that is not optimal, in the sense that all paths from $\boldsymbol{q}_s$ to $\boldsymbol{q}_g$ that can be extracted from the channel may be longer than necessary. To compute efficient paths, the use of $A^\star$ is advisable as a search algorithm. To this end, one should build a modified connectivity graph C' having as nodes $\boldsymbol{q}_s$, $\boldsymbol{q}_g$ and all the midpoints of adjacency segments between cells, and arcs joining nodes belonging to the same cell (note that nodes on adjacency segments belong to two cells). Each arc is then labelled with a cost equal to the distance between the nodes connected by the arc. If the heuristic function is chosen as the distance between the current node and $\boldsymbol{q}_g$, the use of $A^\star$ will produce the shortest path in C', if a solution exists.

The motion planning method based on exact cell decomposition is complete. As for its time complexity, it depends mainly on the cell decomposition and on the connectivity graph search. Using the sweep line algorithm, the decomposition procedure (including the generation of the connectivity graph) has complexity $O(v \log v)$, where v is the number of vertices of $\mathcal{C}_{\text{free}}$. Moreover, it can be shown that the connectivity graph C has $O(v)$ arcs. Hence, regardless of the adopted search strategy, the motion planning method based on exact cell decomposition has complexity $O(v \log v)$.

Note the following facts:

- Any planner based on exact cell decomposition can be considered multiple-query. In fact, once computed, the connectivity graph associated with the decomposition can be used to solve different instances of the same motion planning problem.
- The connectivity graph represents a network of channels, each implicitly containing an *infinity* of paths that traverse the channel and are topologically equivalent, i.e., differ only for a continuous deformation. Therefore, cell decomposition provides as output a structure that is more flexible than the roadmap used by retraction-based methods. This may be useful to plan paths in the channel that are also admissible with respect to possible kinematic constraints, as well as to avoid unexpected obstacles during the actual execution of the motion.
- The solution paths produced by the planning method based on exact cell decomposition are broken lines. It is however possible to smooth the path using *curve fitting* techniques. In practice, one selects a sufficient number

of intermediate points (*via points*) on the path, among which it is necessary to include $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$, and then interpolates them using functions with an appropriate level of differentiability (e.g., polynomial functions of sufficiently high degree).

- The above method can be extended to the case in which $\mathcal{C}_{\text{free}}$ is a limited polyhedral subset of $\mathcal{C} = \mathbb{R}^3$. In particular, the decomposition of $\mathcal{C}_{\text{free}}$ can obtained through the *sweep plane* algorithm, which produces polyhedral cells. The common boundary between adjacent cells consists of trapezoid of non-zero area. This boundary can be safely crossed, e.g., at the barycentre of the trapezoid.

Finally, it should be mentioned that there exist in the literature methods based on exact cell decomposition that can solve essentially any motion planning problem, regardless of the dimension of $\mathcal{C}$ and of the geometry of $\mathcal{C}_{\text{free}}$, which can also be non-polyhedral. However, the complexity of these planners is prohibitive, being exponential in the dimension of $\mathcal{C}$, and their importance is therefore mainly theoretical.

12.4.2 Approximate Decomposition

In approximate decompositions of $\mathcal{C}_{\text{free}}$, disjoint cells of predefined shape are used; for example, when $\mathcal{C} = \mathbb{R}^2$ one may choose square or rectangular cells. In general, the union of all cells will represent an approximation by defect of $\mathcal{C}_{\text{free}}$. To achieve a reasonable trade-off between the accuracy of the approximation and the efficiency of the decomposition procedure, a recursive algorithm is typically used, which starts with a coarse grid whose resolution is locally increased to adapt better to the geometry of $\mathcal{C}_{\text{free}}$. As in motion planning methods based on exact decomposition, the connectivity graph associated with the obtained approximate decomposition is searched for a channel, from which a solution path can be extracted.

In the following, a motion planning method based on approximate decomposition is described for the case in which $\mathcal{C}_{\text{free}}$ is a limited polygonal subset of $\mathcal{C} = \mathbb{R}^2$. Without loss of generality, it will be assumed that the 'external' boundary of $\mathcal{C}_{\text{free}}$ is a square, and that square cells are therefore used. The decomposition algorithm (Fig. 12.8) starts by dividing initially $\mathcal{C}$ into four cells, that are classified according to the categories below:

- *free* cells, whose interior has no intersection with the $\mathcal{C}$-obstacle region;
- *occupied* cells, entirely contained in the $\mathcal{C}$-obstacle region;
- *mixed* cells, that are neither free nor occupied.

At this point, one builds the connectivity graph C associated with the current level of decomposition: this is the graph having free and mixed cells as nodes, and arcs that join nodes representing adjacent cells. Once the nodes corresponding to the cells that contain $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$ have been identified, C is searched for a path between them, e.g., using the $A^\star$ algorithm. If such a path

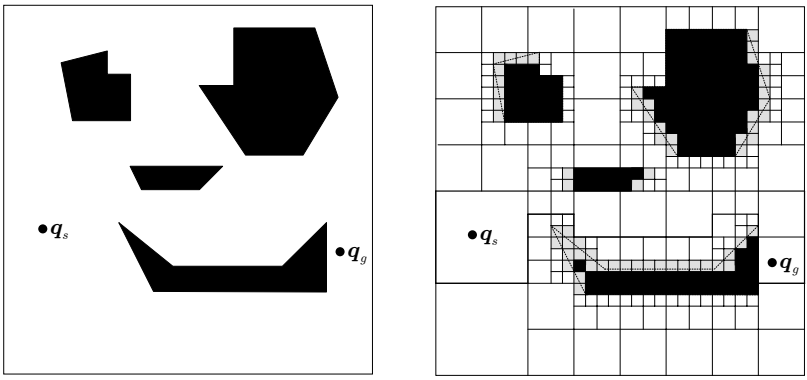

Fig. 12.8. An example of motion planning via approximate cell decomposition; *left*: the assigned problem, *right*: the solution as a free channel (*thick line*)

does not exist, a failure is reported. If the path exists, it consists of a sequence of cells that may be either all free (*free channel*) or not (*mixed channel*). In the first case, a solution to the motion planning problem has been found; in particular, a configuration space path can be easily extracted from the free channel as in the method based on exact cell decomposition. Instead, if the channel contains mixed cells, each of them is further divided into fourcells, which are then classified as free, occupied or mixed. The algorithm proceeds by iterating these steps, until a free channel going from $\boldsymbol{q}_s$ to $\boldsymbol{q}_g$ has been found or a minimum admissible size has been reached for the cells. Figure 12.8 shows an example of application of this technique. Note that, at the resolution level where the solution has been found, free and occupied cells represent an approximation by defect of $\mathcal{C}_{\text{free}}$ and $\mathcal{CO}$, respectively. The missing part of $\mathcal{C}$ is occupied by mixed cells (in gray).

At each iteration, the search algorithm on the connectivity graph can find a free channel going from $\boldsymbol{q}_s$ to $\boldsymbol{q}_g$ only if such a channel exists on the approximation of $\mathcal{C}_{\text{free}}$ (i.e., on the free cells) at the current level of decomposition. This motion planning method is therefore *resolution complete*, in the sense that a sufficient reduction of the minimum admissible size for cells guarantees that a solution is found whenever one exists.

A comparison between Figs. 12.7 and 12.8 clearly shows that, unlike what happens in exact decomposition, the boundary of a cell in an approximate decomposition does not correspond in general to a change in the spatial constraints imposed by obstacles. One of the consequences of this fact is that the implementation of a motion planning method based on approximate decomposition is remarkably simpler, as it only requires a recursive division of cells followed by a collision check between the cells and the $\mathcal{C}$-obstacle region. In particular, the first can be realized using a data structure called *quadtree*. This is a tree in which any internal node (i.e., a node that is not a leaf) has exactly four child nodes. In the cell decomposition case, the tree is rooted at

$\mathcal{C}$, i.e., the whole configuration space. Lower level nodes represent cells that are free, occupied or mixed; only the latter have children.

Another difference with respect to the method based on exact decomposition is that an approximate decomposition is intrinsically associated with a particular instance of a planning problem, as the decomposition procedure itself is guided by the search for a free channel between start and goal.

The planning method based on approximate decomposition is conceptually applicable in configuration spaces of arbitrary dimension. For example, in $\mathbb{R}^3$ it is possible to use an *octree*, a tree in which any internal node has eight children. In $\mathbb{R}^n$, the corresponding data structure is a 2^n*-tree*. Since the maximum number of leaves of a 2^n-tree is 2^{np}, where p is the depth (number of levels) of the tree, the complexity of approximate cell decomposition — and thus of the associated motion planning method — is exponential in the dimension of $\mathcal{C}$ and in the maximal resolution of the decomposition. As a consequence, this technique is effective in practice only in configuration spaces of low dimension (typically, not larger than 4).

12.5 Probabilistic Planning

Probabilistic planners represent a class of methods of remarkable efficiency, especially in problems involving high-dimensional configuration spaces. They belong to the general family of *sampling-based* methods, whose basic idea consists of determining a finite set of collision-free configurations that adequately represent the connectivity of $\mathcal{C}_{\text{free}}$, and using these configurations to build a roadmap that can be employed for solving motion planning problems. This is realized by choosing at each iteration a sample configuration and checking if it entails a collision between the robot and the workspace obstacles. If the answer is affirmative, the sample is discarded. A configuration that does not cause a collision is instead added to the current roadmap and connected if possible to other already stored configurations.

The above strategy is quite general and may lead to different planning methods depending on the specific design choices, and mainly on the criterion for selecting the samples in $\mathcal{C}$ to be checked for collision. One may proceed in a *deterministic* fashion, choosing the samples by means of a regular grid that is applied to $\mathcal{C}$. However, it is preferable to use a *randomized* approach, in which the sample configurations are chosen according to some probability distribution. In the following, two planners of this type are described.

12.5.1 PRM Method

The basic iteration of the PRM (*Probabilistic Roadmap*) method begins by generating a random sample $\boldsymbol{q}_{\text{rand}}$ of the configuration space using a uniform probability distribution in $\mathcal{C}$. Then, $\boldsymbol{q}_{\text{rand}}$ is tested for collision, by using kinematic and geometric relationships to compute the corresponding posture of

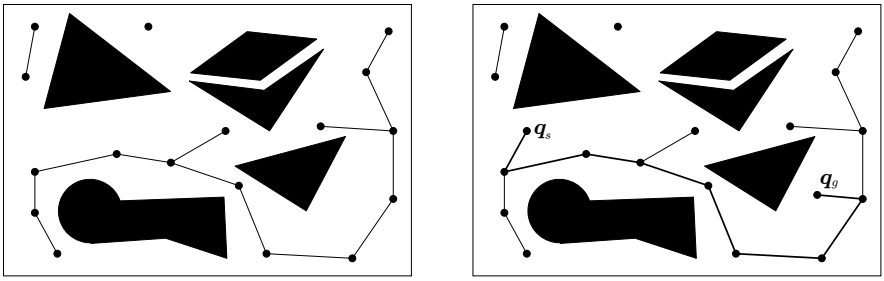

Fig. 12.9. A PRM in a two-dimensional configuration space (*left*) and its use for solving a particular planning problem (*right*)

the robot and invoking an algorithm that can detect collisions (including contacts) between the latter and the obstacles. If $\boldsymbol{q}_{\text{rand}}$ does not cause collisions, it is added to the roadmap and connected (if possible) through free *local paths* to sufficiently 'near' configurations already in the roadmap. Usually, 'nearness' is defined on the basis of the Euclidean distance in $\mathcal{C}$, but it is possible to use different distance notions; for example, as mentioned in Sect. 12.2.1, one may use a configuration space distance notion induced by a distance in the workspace. The generation of a free local path between $\boldsymbol{q}_{\text{rand}}$ and a near configuration $\boldsymbol{q}_{\text{near}}$ is delegated to a procedure known as *local planner*. A common choice is to throw a rectilinear path in $\mathcal{C}$ between $\boldsymbol{q}_{\text{rand}}$ and $\boldsymbol{q}_{\text{near}}$ and test it for collision, for example by sampling the segment with sufficient resolution and checking the single samples for collision. If the local path causes a collision, it is discarded and no direct connection between $\boldsymbol{q}_{\text{rand}}$ and $\boldsymbol{q}_{\text{near}}$ appears in the roadmap.

The PRM incremental generation procedure stops when either a maximum number of iterations has been reached, or the number of connected components in the roadmap becomes smaller than a given threshold. At this point, one verifies whether it is possible to solve the assigned motion planning problem by connecting $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$ to the *same* connected component of the PRM by free local paths. Figure 12.9 shows an example of PRM and the solution to a particular problem. Note the presence of multiple connected components of the PRM, one of which consists of a single configuration.

If a solution cannot be found, the PRM can be improved by performing more iterations of the basic procedure, or using special strategies aimed at reducing the number of its connected components. For example, a possible technique consists of trying to connect configurations that are close but belong to different components via more general (e.g., not rectilinear) local paths.

The main advantage of the PRM method is its remarkable speed in finding a solution to motion planning problems, provided that the roadmap has been sufficiently developed. In this respect, it should be noted that new instances of the same problem induce a potential enhancement of the PRM, which improves with usage both in terms of connectivity and of time efficiency. The

PRM method is therefore intrinsically multiple-query. In high-dimensional configuration spaces, the time needed by this method to compute a solution can be several orders of magnitude smaller than with the previously presented techniques. Another aspect of the method that is worth mentioning is the simplicity of implementation. In particular, note that the generation of $\mathcal{C}$-obstacles is completely eliminated.

The downside of the PRM method is that it is only *probabilistically complete*, i.e., the probability of finding a solution to the planning problem when one exists tends to 1 as the execution time tends to infinity. This means that, if no solution exists, the algorithm will run indefinitely. In practice, a maximum number of iterations is enforced so as to guarantee its termination.

A situation that is critical for the PRM method is the presence of *narrow passages* in $\mathcal{C}_{\text{free}}$, such as the one shown in the upper-right quadrant of the scene in Fig. 12.9. In fact, using a uniform distribution for generating $\boldsymbol{q}_{\text{rand}}$, the probability of placing a sample in a certain region of $\mathcal{C}_{\text{free}}$ is proportional to its volume. As a consequence, depending on its size, it may be very unlikely that a path crossing a narrow passage appears in the PRM within a reasonable time. To alleviate this problem, the method can be modified by using non-uniform probability distributions. For example, there exist strategies for generating $\boldsymbol{q}_{\text{rand}}$ that are biased towards those regions of $\mathcal{C}$ that contain fewer samples, and therefore are more likely to contain close obstacles and the associated narrow passages.

12.5.2 Bidirectional RRT Method

Single-query probabilistic methods are aimed at quickly solving a particular instance of a motion planning problem. Unlike multiple-query planners such as PRM, these techniques do not rely on the generation of a roadmap that represents exhaustively the connectivity of the free configuration space; in fact, they tend to explore only a subset of $\mathcal{C}_{\text{free}}$ that is relevant for solving the problem at hand. This results in a further reduction of the time needed to compute a solution.

An example of single-query probabilistic planner is the *bidirectional RRT* method, which makes use of a data structure called RRT (*Rapidly-exploring Random Tree*). The incremental expansion of an RRT, denoted by T in the following, relies on a simple randomized procedure to be repeated at each iteration (see Fig. 12.10). The first step is the generation of a random configuration $\boldsymbol{q}_{\text{rand}}$ according to a uniform probability distribution in $\mathcal{C}$ (as in the PRM method). Then, the configuration $\boldsymbol{q}_{\text{near}}$ in T that is closer to $\boldsymbol{q}_{\text{rand}}$ is found, and a new candidate configuration $\boldsymbol{q}_{\text{new}}$ is produced on the segment joining $\boldsymbol{q}_{\text{near}}$ to $\boldsymbol{q}_{\text{rand}}$ at a predefined distance δ from $\boldsymbol{q}_{\text{near}}$. A collision check is then run to verify that both $\boldsymbol{q}_{\text{new}}$ and the segment going from $\boldsymbol{q}_{\text{near}}$ to $\boldsymbol{q}_{\text{new}}$ belong to $\mathcal{C}_{\text{free}}$. If this is the case, T is *expanded* by incorporating $\boldsymbol{q}_{\text{new}}$ and the segment joining it to $\boldsymbol{q}_{\text{near}}$. Note that $\boldsymbol{q}_{\text{rand}}$ is not added to the tree, so

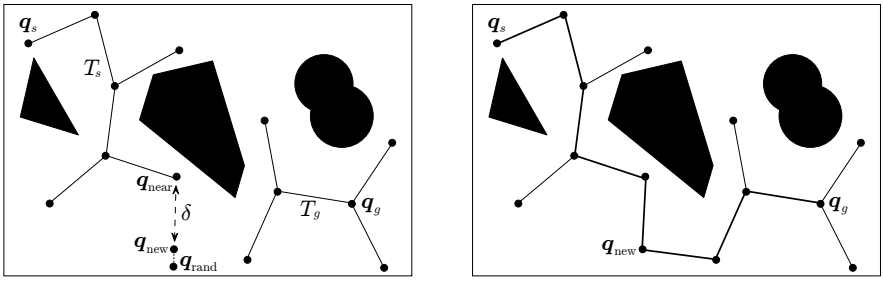

Fig. 12.10. The bidirectional RRT method in a two-dimensional configuration space *left*: the randomized mechanism for expanding a tree, *right*: the extension procedure for connecting the two trees

that it is not necessary to check whether it belongs to $\mathcal{C}_{\text{free}}$; its only function is to indicate a direction of expansion for T.

It is worth pointing out that the RRT expansion procedure, although quite simple, results in a very efficient 'exploration' of $\mathcal{C}$. In fact, it may be shown that the procedure for generating new candidate configuration is intrinsically biased towards those regions of $\mathcal{C}_{\text{free}}$ that have not been visited yet. Moreover, the probability that a generic configuration of $\mathcal{C}_{\text{free}}$ is added to the RRT tends to 1 as the execution time tends to infinity, provided that the configuration lies in the same connected component of $\mathcal{C}_{\text{free}}$ where the RRT is rooted.

To speed up the search for a free path going from $\boldsymbol{q}_s$ to $\boldsymbol{q}_g$, the bidirectional RRT method uses two trees T_s and T_g, respectively rooted at $\boldsymbol{q}_s$ and $\boldsymbol{q}_g$. At each iteration, both trees are expanded with the previously described randomized mechanism. After a certain number of expansion steps, the algorithm enters a phase where it tries to connect the two trees by *extending* each one of them towards the other. This is realized by generating a $\boldsymbol{q}_{\text{new}}$ as an expansion of T_s, and trying to connect T_g to $\boldsymbol{q}_{\text{new}}$. To this end, one may modify the above expansion procedure. In particular, once $\boldsymbol{q}_{\text{new}}$ has been generated from T_s, it acts as a $\boldsymbol{q}_{\text{rand}}$ for T_g: one finds the closest configuration $\boldsymbol{q}_{\text{near}}$ in T_g, and moves from $\boldsymbol{q}_{\text{near}}$ trying to actually *reach* $\boldsymbol{q}_{\text{rand}} = \boldsymbol{q}_{\text{new}}$, hence with a variable stepsize as opposed to a constant δ.

If the segment joining $\boldsymbol{q}_{\text{near}}$ to $\boldsymbol{q}_{\text{new}}$ is collision-free, the extension is complete and the two trees have been connected; otherwise the free portion of the segment is extracted and added to T_g together with its endpoint. At this point, T_g and T_s exchange their roles and the connection attempt is repeated. If this is not successful within a certain number of iterations, one may conclude that the two trees are still far apart and resume the expansion phase.

Like the PRM method, bidirectional RRT is probabilistically complete. A number of variations can be made on the basic scheme. For example, rather than using a constant δ for generating $\boldsymbol{q}_{\text{new}}$, one may define the stepsize as a function of the available free space, possibly going as far as $\boldsymbol{q}_{\text{rand}}$ (as in the extension procedure). This *greedy* version of the method can be much more

efficient if $\mathcal{C}$ contains extensive free regions. Moreover, RRT-based methods can be adapted to robots that do not satisfy the free-flying assumption, such as robots that are subject to nonholonomic constraints.

Extension to nonholonomic robots

Consider now the motion planning problem for a nonholonomic mobile robot whose kinematic model is expressed as (11.10). As seen in the previous chapter, admissible paths in configuration space must satisfy constraint (11.40). For example, in the case of a robot with unicycle kinematics, rectilinear paths in configuration space — such as those used in the RRT expansion to move from $\boldsymbol{q}_{\text{near}}$ to $\boldsymbol{q}_{\text{new}}$ — are not admissible in general.

A simple yet general approach to the design of nonholonomic motion planning methods is to use *motion primitives*, i.e., a finite set of admissible local paths in configuration space, each produced by a specific choice of the velocity inputs in the kinematic model. Admissible paths are generated as a concatenation of motion primitives. In the case of a unicycle robot, for example, the following set of velocity inputs

$$v = \bar{v} \qquad \omega = \{-\bar{\omega}, 0, \bar{\omega}\} \qquad t \in [0, \Delta] \tag{12.9}$$

results in three[6] admissible local paths: the first and the third are respectively a left turn and a right turn along arcs of circle, while the second is a rectilinear path (Fig. 12.11, left).

The expansion of an RRT for a nonholonomic mobile robot equipped with a set of motion primitives is quite similar to the previously described procedure. The difference is that, once identified the configuration $\boldsymbol{q}_{\text{near}}$ on T that is closest to $\boldsymbol{q}_{\text{rand}}$, the new configuration $\boldsymbol{q}_{\text{new}}$ is generated by applying the motion primitives starting from $\boldsymbol{q}_{\text{near}}$ and choosing one of the produced configurations, either randomly or as the closest to $\boldsymbol{q}_{\text{rand}}$. Clearly, $\boldsymbol{q}_{\text{new}}$ and the admissible local path joining $\boldsymbol{q}_{\text{near}}$ to $\boldsymbol{q}_{\text{new}}$ are subject to a collision test. Figure 12.11, right, shows an example of RRT — more precisely, its projection on the workspace — for a unicycle equipped with the motion primitives (12.9).

Under suitable assumptions, it is possible to show that if the goal configuration $\boldsymbol{q}_g$ can be reached from the start configuration $\boldsymbol{q}_s$ through a collision-free concatenation of motion primitives,[7] the probability that $\boldsymbol{q}_g$ is added to

[6] Note that these particular motion primitives include neither backward motion nor rotation on the spot. A unicycle with constant positive driving velocity v and bounded steering velocity is called *Dubins car* in the literature.

[7] This hypothesis, obviously necessary, implies that the choice of motion primitives must be sufficiently 'rich' to guarantee that the set of configuration reachable through concatenation is 'dense' enough with respect to $\mathcal{C}_{\text{free}}$. For example, the unicycle with the motion primitives (12.9) with a variable time interval Δ can reach any configuration in $\mathcal{C}_{\text{free}}$, although not necessarily with a path that is entirely contained in $\mathcal{C}_{\text{free}}$. This property is instead guaranteed if the *Reeds–Shepp curves* given by (11.56) are used as motion primitives.

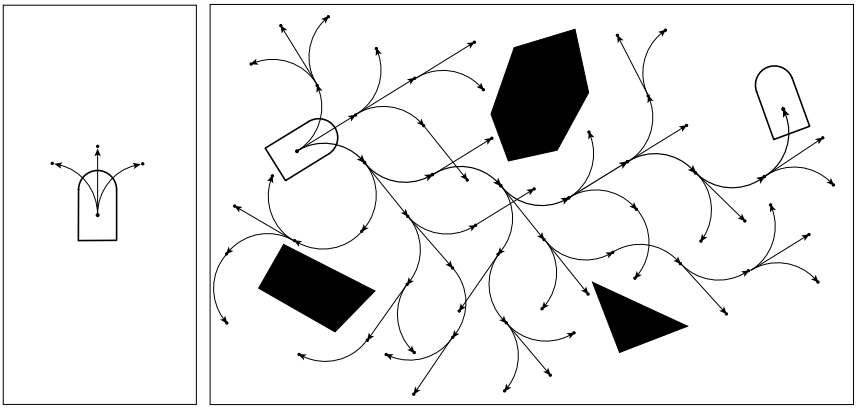

Fig. 12.11. RRT-based motion planning for a unicycle; *left*: a set of motion primitives, *right*: an example of RRT

the tree T tends to 1 as the execution time tends to infinity. To increase the efficiency of the search, also in this case it is possible to devise a bidirectional version of the method.

12.6 Planning via Artificial Potentials

All the methods so far presented are suitable for off-line motion planning, because they require a priori knowledge of the geometry and the pose of the obstacles in the robot workspace. This assumption is reasonable in many cases, e.g., when an industrial manipulator is moving in a robotized cell. However, in service robotics applications the robot must be able to plan its motion on-line, i.e., using partial information on the workspace gathered during the motion on the basis of sensor measurements.

An effective paradigm for on-line planning relies on the use of *artificial potential fields*. Essentially, the point that represents the robot in configuration space moves under the influence of a potential field U obtained as the superposition of an *attractive* potential to the goal and a *repulsive* potential from the $\mathcal{C}$-obstacle region. Planning takes place in an incremental fashion: at each robot configuration $\boldsymbol{q}$, the artificial force generated by the potential is defined as the negative gradient $-\nabla U(\boldsymbol{q})$ of the potential, which indicates the most promising direction of local motion.

12.6.1 Attractive Potential

The attractive potential is designed so as to guide the robot to the goal configuration $\boldsymbol{q}_g$. To this end, one may use a *paraboloid* with vertex in $\boldsymbol{q}_g$:

$$U_{a1}(\boldsymbol{q}) = \frac{1}{2}\, k_a \boldsymbol{e}^T(\boldsymbol{q})\boldsymbol{e}(\boldsymbol{q}) = \frac{1}{2}\, k_a \|\boldsymbol{e}(\boldsymbol{q})\|^2, \tag{12.10}$$

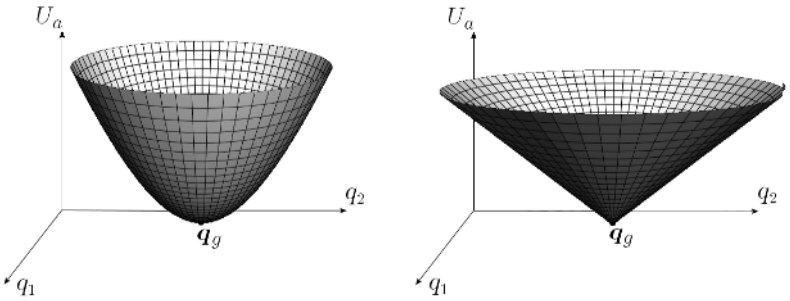

Fig. 12.12. The shape of the paraboloidic attractive potential U_{a1} (*left*) and of the conical attractive potential U_{a2} (*right*) in the case $\mathcal{C} = \mathbb{R}^2$, for $k_a = 1$

where $k_a > 0$ and $\boldsymbol{e} = \boldsymbol{q}_g - \boldsymbol{q}$ is the 'error' vector with respect to the goal configuration $\boldsymbol{q}_g$. This function is always positive and has a global minimum in $\boldsymbol{q}_g$, where it is zero. The resulting attractive force is defined as

$$\boldsymbol{f}_{a1}(\boldsymbol{q}) = -\nabla U_{a1}(\boldsymbol{q}) = k_a \boldsymbol{e}(\boldsymbol{q}). \tag{12.11}$$

Hence, $\boldsymbol{f}_{a1}$ converges linearly to zero when the robot configuration $\boldsymbol{q}$ tends to the goal configuration $\boldsymbol{q}_g$.

Alternatively, it is possible to define a *conical* attractive potential as

$$U_{a2}(\boldsymbol{q}) = k_a \|\boldsymbol{e}(\boldsymbol{q})\|. \tag{12.12}$$

Also U_{a2} is always positive, and zero in $\boldsymbol{q}_g$. The corresponding attractive force is

$$\boldsymbol{f}_{a2}(\boldsymbol{q}) = -\nabla U_{a2}(\boldsymbol{q}) = k_a \frac{\boldsymbol{e}(\boldsymbol{q})}{\|\boldsymbol{e}(\boldsymbol{q})\|}, \tag{12.13}$$

that is constant in modulus. This represents an advantage with respect to the force $\boldsymbol{f}_{a1}$ generated by the paraboloidic attractive potential, which tends to grow indefinitely as the error vector increases in norm. On the other hand, $\boldsymbol{f}_{a2}$ is indefinite in $\boldsymbol{q}_g$. Figure 12.12 shows the shape of U_{a1} and U_{a2} in the case $\mathcal{C} = \mathbb{R}^2$, with $k_a = 1$.

A choice that combines the advantages of the above two potentials is to define the attractive potential as a conical surface away from the goal and as a paraboloid in the vicinity of $\boldsymbol{q}_g$. In particular, by placing the transition between the two potentials where $\|\boldsymbol{e}(\boldsymbol{q})\| = 1$ (i.e., on the surface of the sphere of unit radius centred in $\boldsymbol{q}_g$) one obtains an attractive force that is continuous for any $\boldsymbol{q}$ (see also Problem 12.9).

12.6.2 Repulsive Potential

The repulsive potential U_r is added to the attractive potential U_a to prevent the robot from colliding with obstacles as it moves under the influence of the

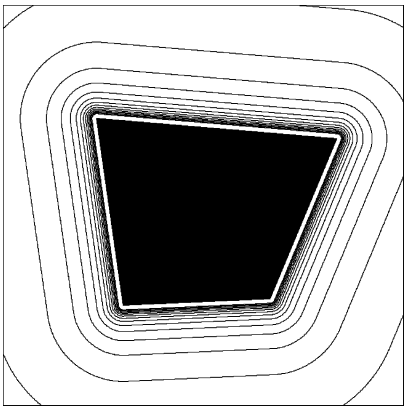

Fig. 12.13. The equipotential contours of the repulsive potential U_r in the range of influence of a polygonal $\mathcal{C}$-obstacle in $\mathcal{C} = \mathbb{R}^2$, for $k_r = 1$ and $\gamma = 2$

attractive force $\boldsymbol{f}_a$. In particular, the idea is to build a barrier potential in the vicinity of the $\mathcal{C}$-obstacle region, so as to repel the point that represents the robot in $\mathcal{C}$.

In the following, it will be assumed that the $\mathcal{C}$-obstacle region has been partitioned in convex components $\mathcal{CO}_i$, $i = 1, \ldots, p$. These components may coincide with the $\mathcal{C}$-obstacles themselves; this happens, for example, when the robot $\mathcal{B}$ is a convex polygon (polyhedron) translating with a fixed orientation in $\mathbb{R}^2$ ($\mathbb{R}^3$) among convex polygonal (polyhedral) obstacles (see Sect. 12.2.2). In the presence of non-convex $\mathcal{C}$-obstacles, however, it is necessary to perform the decomposition in convex components before building the repulsive potential.

For each convex component $\mathcal{CO}_i$, define an associated repulsive potential as

$$U_{r,i}(\boldsymbol{q}) = \begin{cases} \dfrac{k_{r,i}}{\gamma} \left(\dfrac{1}{\eta_i(\boldsymbol{q})} - \dfrac{1}{\eta_{0,i}} \right)^{\gamma} & \text{if } \eta_i(\boldsymbol{q}) \leq \eta_{0,i} \\ 0 & \text{if } \eta_i(\boldsymbol{q}) > \eta_{0,i}, \end{cases} \tag{12.14}$$

where $k_{r,i} > 0$, $\eta_i(\boldsymbol{q}) = \min_{\boldsymbol{q}' \in \mathcal{CO}_i} \|\boldsymbol{q} - \boldsymbol{q}'\|$ is the distance of $\boldsymbol{q}$ from $\mathcal{CO}_i$, $\eta_{0,i}$ is the *range of influence* of $\mathcal{CO}_i$ and $\gamma = 2, 3, \ldots$. The potential $U_{r,i}$ is zero outside and positive inside the range of influence $\eta_{0,i}$ and tends to infinity as the boundary of $\mathcal{CO}_i$ is approached, more abruptly as γ is increased (a typical choice is $\gamma = 2$).

When $\mathcal{C} = \mathbb{R}^2$ and the convex component $\mathcal{CO}_i$ is polygonal, an *equipotential contour* of $U_{r,i}$ (i.e., the locus of configurations $\boldsymbol{q}$ such that $U_{r,i}$ has a certain constant value) consists of rectilinear tracts that are parallel to the sides of the polygon, connected by arcs of circle in correspondence of the vertices, as shown in Fig. 12.13. Note how the contours get closer to each other in the proximity of the $\mathcal{C}$-obstacle boundary, due to the hyperboloidic profile

of the potential. When $\mathcal{C} = \mathbb{R}^3$ and the convex component $\mathcal{CO}_i$ is polyhedral, the *equipotential surfaces* of $U_{r,i}$ are copies of the faces of $\mathcal{CO}_i$, connected by patches of cylindrical surfaces in correspondence of the edges and spherical surfaces in correspondence of the vertices of $\mathcal{CO}_i$.

The repulsive force resulting from $U_{r,i}$ is

$$\boldsymbol{f}_{r,i}(\boldsymbol{q}) = -\nabla U_{r,i}(\boldsymbol{q}) = \begin{cases} \dfrac{k_{r,i}}{\eta_i^2(\boldsymbol{q})} \left(\dfrac{1}{\eta_i(\boldsymbol{q})} - \dfrac{1}{\eta_{0,i}} \right)^{\gamma-1} \nabla \eta_i(\boldsymbol{q}) & \text{if } \eta_i(\boldsymbol{q}) \leq \eta_{0,i} \\ \\ 0 & \text{if } \eta_i(\boldsymbol{q}) > \eta_{0,i}. \end{cases} \tag{12.15}$$

Denote by $\boldsymbol{q}_m$ the configuration of $\mathcal{CO}_i$ that is closer to $\boldsymbol{q}$ ($\boldsymbol{q}_m$ is uniquely determined in view of the convexity of $\mathcal{CO}_i$). The gradient vector $\nabla \eta_i(\boldsymbol{q})$, which is orthogonal to the equipotential contour (or surface) passing through $\boldsymbol{q}$, is directed as the half-line originating from $\boldsymbol{q}_m$ and passing through $\boldsymbol{q}$. If the boundary of $\mathcal{CO}_i$ is piecewise differentiable, function η_i is differentiable everywhere in $\mathcal{C}_{\text{free}}$ and $\boldsymbol{f}_{r,i}$ is continuous in the same space.[8]

The aggregate repulsive potential is obtained by adding up the individual potentials associated with the convex components of $\mathcal{CO}$:

$$U_r(\boldsymbol{q}) = \sum_{i=1}^{p} U_{r,i}(\boldsymbol{q}). \tag{12.16}$$

If $\eta_i(\boldsymbol{q}_g) \geq \eta_{0,i}$ for $i = 1, \ldots, p$ (i.e., if the goal is placed outside the range of influence of each obstacle component $\mathcal{CO}_i$), the value of the aggregate repulsive field U_r is zero in $\boldsymbol{q}_g$. In the following, it will be assumed that this is the case.

12.6.3 Total Potential

The total potential U_t is obtained by superposition of the attractive and the aggregate repulsive potentials:

$$U_t(\boldsymbol{q}) = U_a(\boldsymbol{q}) + U_r(\boldsymbol{q}). \tag{12.17}$$

This results in the force field

$$\boldsymbol{f}_t(\boldsymbol{q}) = -\nabla U_t(\boldsymbol{q}) = \boldsymbol{f}_a(\boldsymbol{q}) + \sum_{i=1}^{p} \boldsymbol{f}_{r,i}(\boldsymbol{q}). \tag{12.18}$$

[8] Note the relevance in this sense of the assumption that the component $\mathcal{CO}_i$ is convex. If it were otherwise, there would exist configurations in $\mathcal{C}_{\text{free}}$ for which $\boldsymbol{q}_m$ would not be uniquely defined. In these configurations, belonging by definition to the generalized Voronoi diagram $\mathcal{V}(\mathcal{C}_{\text{free}})$, function η_i would not be differentiable, resulting in a discontinuous repulsive force. This might induce undesired effects on the planned path (for example, oscillations).

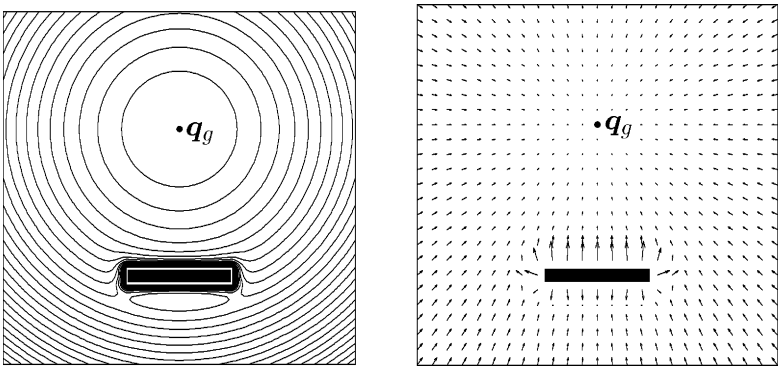

Fig. 12.14. The total potential in $\mathcal{C} = \mathbb{R}^2$ obtained by superposition of a hyperboloidic attractive potential and a repulsive potential for a rectangular $\mathcal{C}$-obstacle: *left*: the equipotential contours, *right*: the resulting force field

U_t clearly has a global minimum in $\boldsymbol{q}_g$, but there may also exist some *local minima* where the force field is zero. Considering for simplicity the case $\mathcal{C} = \mathbb{R}^2$, this happens in the 'shadow zone' of a $\mathcal{C}$-obstacle when the repulsive potential $U_{r,i}$ has equipotential contours with lower curvature (e.g., segments) than the attractive potential in the same area. See for example Fig. 12.14, where a local minimum is clearly present 'below' the $\mathcal{C}$-obstacle. A remarkable exception is the case (*sphere world*) in which all the convex components $\mathcal{CO}_i$ of the $\mathcal{C}$-obstacle region are spheres. In this situation, the total potential exhibits isolated saddle points (where the force field is still zero) but no local minima.

12.6.4 Planning Techniques

There are three different approaches for planning collision-free motions on the basis of a total artificial potential U_t and the associated force field $\boldsymbol{f}_t = -\nabla U_t$. They are briefly discussed below:

1. The first possibility is to let
$$\boldsymbol{\tau} = \boldsymbol{f}_t(\boldsymbol{q}), \tag{12.19}$$
hence considering $\boldsymbol{f}_t(\boldsymbol{q})$ as a vector of generalized forces that induce a motion of the robot in accordance with its dynamic model.
2. The second method regards the robot as a unit point mass moving under the influence of $\boldsymbol{f}_t(\boldsymbol{q})$, as in
$$\ddot{\boldsymbol{q}} = \boldsymbol{f}_t(\boldsymbol{q}). \tag{12.20}$$
3. The third possibility is to interpret the force field $\boldsymbol{f}_t(\boldsymbol{q})$ as a desired velocity for the robot, by letting
$$\dot{\boldsymbol{q}} = \boldsymbol{f}_t(\boldsymbol{q}). \tag{12.21}$$

In principle, one could use these three approaches for on-line as well as off-line motion planning. In the first case, (12.19) directly represents control inputs for the robot, whereas the implementation of (12.20) requires the solution of the inverse dynamics problem, i.e., the substitution of $\ddot{\boldsymbol{q}}$ in the robot dynamic model to compute the generalized forces $\boldsymbol{\tau}$ that realize such accelerations. Equation (12.21) can instead be used on-line in a kinematic control scheme, in particular to provide the reference inputs for the low-level controllers that are in charge of reproducing such generalized velocities as accurately as possible. In any case, the artificial force field $\boldsymbol{f}_t$ represents, either directly or indirectly, a true feedback control that guides the robot towards the goal, while trying at the same time to avoid collisions with the workspace obstacles that have been detected by the sensory system. To emphasize this aspect, on-line motion generation based on artificial potentials is also referred to as *reactive* planning.

In off-line motion planning, configuration space paths are generated by simulation, i.e., integrating numerically the robot dynamic model if (12.19) is used, or directly by the differential equations (12.20) and (12.21).

In general, the use of (12.19) generates smoother paths, because with this scheme the reactions to the presence of obstacles are naturally 'filtered' through the robot dynamics. On the other hand, the strategy represented by (12.21) is faster in executing the motion corrections suggested by the force field $\boldsymbol{f}_t$, and may thus be considered safer. The characteristics of scheme (12.20) are clearly intermediate between the other two. Another aspect to be considered is that using (12.21) guarantees (in the absence of local minima) the asymptotic stability of $\boldsymbol{q}_g$ (i.e., the robot reaches the goal with zero velocity), whereas this is not true for the other two motion generation strategies. To achieve asymptotic stability with (12.19) and (12.20), a damping term proportional to the robot velocity $\dot{\boldsymbol{q}}$ must be added to $\boldsymbol{f}_t$.

In view of the above discussion, it is not surprising that the most common choice is the simple numerical integration of (12.21) via the Euler method:

$$\boldsymbol{q}_{k+1} = \boldsymbol{q}_k + T\boldsymbol{f}_t(\boldsymbol{q}_k), \tag{12.22}$$

where $\boldsymbol{q}_k$ and $\boldsymbol{q}_{k+1}$ represent respectively the current and the next robot configuration, and T is the integration step. To improve the quality of the generated path, it is also possible to use a variable T, smaller when the modulus of the force field $\boldsymbol{f}_t$ is larger (in the vicinity of obstacles) or smaller (close to the destination $\boldsymbol{q}_g$). Recalling that $\boldsymbol{f}_t(\boldsymbol{q}) = -\nabla U_t(\boldsymbol{q})$, Eq. (12.22) may be easily interpreted as a numerical implementation of the *gradient method* for the minimization of $U_t(\boldsymbol{q})$, often referred to as the *algorithm of steepest descent*.

12.6.5 The Local Minima Problem

Whatever technique is used to plan motions on the basis of artificial potentials, local minima of U_t — where the total force field $\boldsymbol{f}_t$ is zero — represent a

problem. For example, if (12.22) is used and the generated path enters the basin of attraction of a local minimum, the planning process is bound to stop there, without reaching the goal configuration.[9] Actually, the same problem may occur also when (12.19) or (12.20) are used if the basin of attraction of the local minimum is sufficiently large. On the other hand, as noticed previously, the total potential U_t obtained by the superposition of an attractive and a repulsive potential invariably exhibits local minima, except for very particular cases. This means that motion planning methods based on artificial potentials are not complete in general, because it may happen that the goal configuration $\boldsymbol{q}_g$ is not reached even though a solution exists.

Best-first algorithm

A simple workaround for the local minima problem is to use a *best-first* algorithm, which is formulated under the assumption that the free configuration space $\mathcal{C}_{\text{free}}$ has been discretized using a regular grid. In general, the discretization procedure results in the loss of some boundary regions of $\mathcal{C}_{\text{free}}$, which are not represented as free in the gridmap. Each free cell of the grid is assigned the value of the total potential U_t computed at the centroid of the cell. Planning proceeds by building a tree T rooted at the start configuration $\boldsymbol{q}_s$. At each iteration, the leaf with the minimum value of U_t is selected and its adjacent[10] cells are examined. Those that are not already in T are added as children of the considered leaf. Planning is successful when the cell containing the goal is reached (in this case, the solution path is built by tracing back the arcs of the tree from $\boldsymbol{q}_g$ to $\boldsymbol{q}_s$), whereas failure is reported when all the cells accessible from $\boldsymbol{q}_s$ have been explored without reaching $\boldsymbol{q}_g$.

The above best-first algorithm evolves as a grid-discretized version of the algorithm of steepest descent (12.22), until a cell is reached which represents a minimum for the associated total potential. If it is a local minimum, the algorithm visits ('fills') its entire basin of attraction and finally leaves it, reaching a point from which planning can continue. The resulting motion planning method is resolution complete, because a solution is found only if one exists on the gridmap that represents $\mathcal{C}_{\text{free}}$ by defect. In general, increasing the grid resolution may be necessary to recover the possibility of determining a solution. In any case, the time complexity of the basin filling procedure, which is exponential in the dimension of $\mathcal{C}$ because such is the number of adjacent cells to a given one, makes the best-first algorithm applicable only in configuration spaces of low dimension (typically, not larger than 3).

---

[9] The probability of reaching a saddle point instead is extremely low; besides, any perturbation would allow the planner to exit such a point.

[10] Various definitions of adjacency may be adopted. For example, in $\mathbb{R}^2$ one may use 1-adjacency or 2-adjacency. In the first case, each cell c has four adjacent cells, while in the second they are eight. The resulting paths will obviously reflect this fact.

A more effective approach is to include in the best-first method a randomized mechanism for evading local minima. In practice, one implements a version of the best-first algorithm in which the number of iterations aimed at filling the basin of attraction of local minima is bounded. When the bound is reached, a sequence of random steps (*random walk*) is taken. This usually allows the planner to evade the basin in a much shorter time than would be needed by the complete filling procedure. As the probability of evading the basin of attraction of a local minimum approaches 1 when the number of random steps tends to infinity, this *randomized best-first* method is probabilistically complete (in addition to being resolution complete).

Navigation functions

Although the best-first algorithm (in its basic or randomized version) represents a solution to the problem of local minima, it may generate paths that are very inefficient. In fact, with this strategy the robot will still enter (and then evade) the basin of attraction of any local minimum located on its way to goal. A more radical approach is based on the use of *navigation functions*, i.e., artificial potentials that have no local minima. As already mentioned, if the $\mathcal{C}$-obstacles are spheres this property already holds for the potential U_t defined by superposition of an attractive and a repulsive field. A first possibility would then be to approximate by excess all $\mathcal{C}$-obstacles with spheres, and use the total potential U_t. Clearly, such an approximation may severely reduce the free configuration space $\mathcal{C}_{\text{free}}$, and even destroy its connectedness; for example, imagine what would happen in the case depicted in Fig. 12.4.

In principle, a mathematically elegant way to define a navigation function consists of building first a differentiable homeomorphism (a *diffeomorphism*) that maps the $\mathcal{C}$-obstacle region to a collection of spheres, then generating a classical total potential in the transformed space, and finally mapping it back to the original configuration space so as to obtain a potential free of local minima. If the $\mathcal{C}$-obstacles are *star-shaped*,[11] such a diffeomorphism actually exists, and the procedure outlined above provides in fact a navigation function. Another approach is to build the potential using *harmonic functions*, that are the solutions of a particular differential equation that describes the physical process of heat transmission or fluid dynamics.

Generating a navigation function is, however, computationally cumbersome, and the associated planning methods are thus mainly of theoretical interest. A notable exception, at least in low-dimensional configuration spaces, is the *numerical navigation function*. This is a potential built on a gridmap representation of $\mathcal{C}_{\text{free}}$ by assigning value 0 to the cell containing $\boldsymbol{q}_g$, value 1 to its adjacent cells, value 2 to the unvisited cells among those adjacent to

[11] A subset $S \subset \mathbb{R}^n$ is said to be star-shaped if it is homeomorphic to the closed unit sphere in $\mathbb{R}^n$ and has a point p (*centre*) such that any other point in S may be joined to p by a segment that is entirely contained in S.

2	1	2	3	4	5	6	7	8	9		19
1	0	1			6	7	8	9	10		18
2	1	2	3		7	8		10	11		17
3		3	4	5	6	7	8		12		16
4			5	6	7			12	13		15
5	6	7	6	7	8	9	10	11	12	13	14
6	7	8	7	8	9	10	11	12	13	14	15

Fig. 12.15. An example of numerical navigation function in a simple two-dimensional gridmap using 1-adjacency. Cells in gray denote a particular solution path obtained by following from the start (cell 12) the steepest descent of the potential on the gridmap

cells with potential 1, and so on. To understand better the procedure, one may visualize a wavefront that originates at $\boldsymbol{q}_g$ and expands according to the adopted adjacency definition (*wavefront expansion algorithm*). It is easy to realize that the obtained potential is free of local minima, and therefore its use in conjunction with the algorithm of steepest descent provides a motion planning method that is complete in resolution (see Fig. 12.15).

Finally, it should be mentioned that the use of navigation functions is limited to off-line motion planning, because their construction — be they continuous or discrete in nature — requires the a priori knowledge of the geometry and pose of the workspace obstacles. As for on-line motion planning, in which the obstacles are gradually reconstructed via sensor measurements as the robot moves, the incremental construction of a total potential by superposition of attractive and repulsive fields represents a simple, often effective method to generate collision-free motions, even though completeness cannot be claimed. In any case, motion planning based on artificial potentials belong to the single-query category, because the attractive component of the potential depends on the goal configuration $\boldsymbol{q}_g$.

12.7 The Robot Manipulator Case

Generating collision-free movements for robot manipulators is a particularly important category of motion planning problems. In general, the computational complexity associated with this problem is substantial, due to the high dimension of the configuration space (typically $n \geq 4$) and to the presence of rotational DOFs (revolute joints).

It is sometimes possible to reduce the dimension of the configuration space $\mathcal{C}$ approximating by excess the size of the robot. For example, in a six-DOF anthropomorphic manipulator one can replace the last three links of the kinematic chain (the spherical wrist) and the end-effector with the volume they

'sweep' when the corresponding joints move across their whole available range. The dimension of the configuration space becomes three, as planning concerns only the base, shoulder and elbow joints, while the wrist can move arbitrarily. Clearly, this approximation is conservative, and hence acceptable only if the aforementioned volume is small with respect to the workspace of the manipulator.

In the presence of rotational DOFs, the other complication is the shape of the $\mathcal{C}$-obstacles, which is complex even for simple workspace obstacles due to the strong nonlinearity introduced by the manipulator inverse kinematics (recall Fig. 12.4). Apart from the intrinsic difficulty of computing $\mathcal{C}$-obstacles, their non-polyhedral shape does not allow the application of the planning methods presented in Sects. 12.3 and 12.4.

The most convenient choice for off-line planning is represented by probabilistic methods, which exhibit the best performance in high-dimensional configuration spaces. Moreover, as the computation of the $\mathcal{C}$-obstacles is not required, these planners are not affected by their shape. However, it should be noted that collision checking — which is an essential tool for probabilistic motion planning — becomes more onerous as the number of DOFs is increased.

For on-line planning, the best results are obtained by a suitable adaptation of the method based on artificial potentials. In particular, to avoid the computation of $\mathcal{C}$-obstacles and at the same time plan in a space of reduced dimension, the potential is directly built in the workspace $\mathcal{W} = \mathbb{R}^N$ rather than in the configuration space $\mathcal{C}$, and acts on a set of *control points* located on the manipulator. Among these is included a point that represents the end-effector (to which is assigned the goal of the motion planning problem) and at least one point (possibly variable in time) for each body of the linkage. While the attractive potential only influences the end-effector representative point, the repulsive potential acts on all control points. As a consequence, the artificial potentials used in this scheme are actually two: an attractive-repulsive field for the end-effector, and a repulsive field for the other control points distributed on the manipulator links.

As before, different approaches may be used to convert the force fields generated by the artificial potentials to commands for the manipulator. Denote by $\boldsymbol{p}_i(\boldsymbol{q})$, $i = 1, \ldots, P$, the coordinates in $\mathcal{W}$ of the P control points in correspondence of the configuration $\boldsymbol{q}$ of the manipulator. In particular, $\boldsymbol{p}_1, \ldots, \boldsymbol{p}_{P-1}$ are the control points located on the manipulator links, subject only to the repulsive potential U_r, while $\boldsymbol{p}_P$ is the control point for the end-effector, which is subject to the total potential $U_t = U_a + U_r$.

A first possibility is to impose to the robot joints the generalized forces which would result from the combined action of the various force fields acting on the control points in the workspace, according to

$$\boldsymbol{\tau} = -\sum_{i=1}^{P-1} \boldsymbol{J}_i^T(\boldsymbol{q}) \nabla U_r(\boldsymbol{p}_i) - \boldsymbol{J}_P^T(\boldsymbol{q}) \nabla U_t(\boldsymbol{p}_P), \tag{12.23}$$

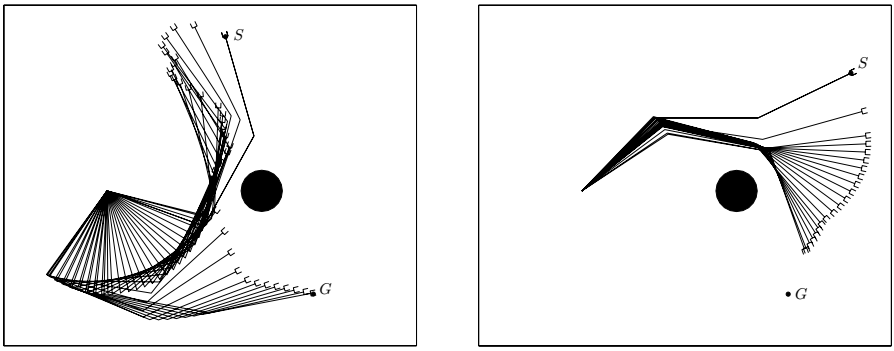

Fig. 12.16. Examples of motion planning via artificial potentials acting on control points for a planar 3R manipulator; *left*: planning is successful and leads to a collision-free motion between the start S and the goal G, *right*: a failure is reported because the manipulator is stuck at a force equilibrium

where $\boldsymbol{J}_i(\boldsymbol{q})$, $i = 1, \ldots, P$, denotes the Jacobian of the direct kinematics function associated with the control point $\boldsymbol{p}_i(\boldsymbol{q})$.

Alternatively, a purely kinematic planning scheme is obtained by letting

$$\dot{\boldsymbol{q}} = -\sum_{i=1}^{P-1} \boldsymbol{J}_i^T(\boldsymbol{q}) \nabla U_r(\boldsymbol{p}_i) - \boldsymbol{J}_P^T(\boldsymbol{q}) \nabla U_t(\boldsymbol{p}_P) \tag{12.24}$$

and feeding these joint velocities to the low-level control loops as reference signals. Note that Eq. (12.24) represents a gradient-based minimization step in the configuration space $\mathcal{C}$ of a combined potential defined in the workspace $\mathcal{W}$. In fact, a potential function acting on a control point in the workspace may be seen as a composite function of $\boldsymbol{q}$ through the associated direct kinematics relationship, and the Jacobian transpose $\boldsymbol{J}_i^T(\boldsymbol{q})$, $i = 1, \ldots, P$, maps a gradient in $\mathcal{W}$ to a gradient in $\mathcal{C}$. In formulae:

$$\nabla_{\boldsymbol{q}} U(\boldsymbol{p}_i) = \left(\frac{\partial U(\boldsymbol{p}_i(\boldsymbol{q}))}{\partial \boldsymbol{q}} \right)^T = \left(\frac{\partial U(\boldsymbol{p}_i)}{\partial \boldsymbol{p}_i} \frac{\partial \boldsymbol{p}_i}{\partial \boldsymbol{q}} \right)^T = \boldsymbol{J}_i^T(\boldsymbol{q}) \nabla U(\boldsymbol{p}_i),$$

for $i = 1, \ldots, P$.

The above two schemes can be considered respectively the transposition of (12.19) and (12.21), and therefore they inherit the same characteristics. In particular, when (12.23) is used the motion corrections prescribed by the force fields are filtered through the dynamic model of the manipulator, and smoother movements can be expected. The kinematic scheme (12.24) instead is faster in realizing such corrections.

Finally, it should be mentioned that the use of artificial potentials that are defined in the workspace may aggravate the local minima problem. In fact, the various forces (either purely attractive or repulsive-attractive) acting on the control points may neutralize each other at the joint level, blocking the

manipulator in a configuration (*force equilibrium*) where no control point is at a local minimum of the associated potential (see Fig. 12.16). As consequence, it is always advisable to use workspace potential fields in conjunction with a randomized best-first algorithm.

Bibliography

In the last three decades, the literature on motion planning has grown considerably, and this area may now be considered as a scientific discipline in itself. In the following, only a short list is given of the seminal works for the material presented in this chapter.

The systematic use of the concept of configuration space for motion planning was proposed in [138]. The method based on the retraction of the free configuration space on the generalized Voronoi diagram was originally introduced for a robot of circular shape in [170]. The technique based on trapezoidal cell decomposition is described in [122], while [197] and [33] are among the general planning methods via decomposition mentioned at the end of Sect. 12.4.1. The approach based on approximate cell decomposition was proposed in [138]. The PRM method for probabilistic planning was introduced in [107], while the RRT method with its variants is described in [125].

The use of artificial potentials for on-line motion planning was pioneered in [113]. The concept of navigation function was introduced in [185], while its numerical version on a gridmap was described in [17], together with the best-first algorithm, also in its randomized version.

For other aspects of the motion planning problem that are merely hinted at (nonholonomic motion planning) or simply ignored (managing uncertainty, mobile obstacles) in this chapter, the reader can consult many excellent books, going from the classical treatment in [122], through [123], to the most recent texts [45, 145, 124].

Problems

12.1. Describe the nature (including the dimension) of the configuration space for a mobile manipulator consisting of a unicycle-like vehicle carrying a six-DOF anthropomorphic arm, providing a choice of generalized coordinates for the system.

12.2. With reference to a 2R manipulator, modify the definition (12.2) of configuration space distance so as to take into account the fact that the manipulator posture does not change if the joint variables q_1 and q_2 are increased (or decreased) by a multiple of 2π.

12.3. Consider a polygonal robot translating at a fixed orientation in $\mathbb{R}^2$ among polygonal obstacles. Build an example showing that the same $\mathcal{C}$-obstacle region may correspond to robot and obstacles of different shapes.

12.4. With reference to the second workspace shown in Fig. 12.4, give the numerical value of three configurations of the manipulator that lie in the three connected components of $\mathcal{C}_{\text{free}}$. Moreover, sketch the manipulator posture for each of these configurations.

12.5. Discuss the basic steps of an algorithm for computing the generalized Voronoi diagram of a limited polygonal subset of $\mathbb{R}^2$. [*Hint*: a simple algorithm is obtained by considering all the possible side-side, side-vertex and vertex-vertex pairs, and generating the elementary arcs of the diagram on the basis of the intersections of the associated equidistance contours.]

12.6. For the motion planning method via exact cell decomposition, give an example in which the path obtained as a broken line joining the midpoints of the common boundary of the channel cells goes through a vertex of the $\mathcal{C}$-obstacle region, and propose an alternative procedure for extracting a free path from the channel. [*Hint*: build a situation in which the channel from c_s to c_g contains a cell for which the entrance and the exit boundary lie on the same side.]

12.7. Implement in a computer program the PRM method for a 2R planar robot moving among circular workspace obstacles. The program receives as input a geometrical description of the obstacles (centre and radius of each obstacle) as well as the start and goal configurations, and terminates after a maximum number of iterations. If a solution path has been found, it is given as output together with the associated PRM; otherwise, a failure is reported. Discuss the performance of the method with respect to the choice of configuration space distance (for example, (12.1) or (12.2)).

12.8. Implement in a computer program the RRT method for a circular-shaped unicycle moving among square obstacles, with the motion primitives defined as in (12.9). The program receives as input the start and goal configurations, in addition to a geometrical description of the obstacles (centre and side of each obstacle), and terminates after a maximum number of iterations. If a solution path has been found, it is given as output together with the associated RRT; otherwise, a failure is reported. Build a situation in which the method cannot find a solution because of the limitations inherent to the chosen motion primitives.

12.9. For the case $\mathcal{C} = \mathbb{R}^2$, build a continuously differentiable attractive potential having a paraboloidic profile inside the circle of radius ρ and a conical profile outside. [*Hint*: modify the expression (12.12) of the conical potential using a different constant k_b in place of k_a, which already characterizes the paraboloidic potential.]

12.10. For the case $\mathcal{C} = \mathbb{R}^2$, prove that the total potential U_t may exhibit a local minimum in areas where the equipotential contours of the repulsive potential U_r have lower curvature than those of the attractive potential U_a. [*Hint*: consider a polygonal $\mathcal{C}$-obstacle and use a geometric construction.]

12.11. Consider a point robot moving in a planar workspace containing three circular obstacles, respectively of radius 1, 2 and 3 and centre in $(2, 1)$, $(-1, 3)$ and $(1, -2)$. The goal is the origin of the workspace reference frame. Build the total potential U_t resulting from the superposition of the attractive potential U_a to the goal and the repulsive potential U_r from the obstacles, and derive the corresponding artificial force field. Moreover, compute the coordinates of the saddle points of U_t.

12.12. Discuss the main issues arising from the application of the artificial potential technique for planning on-line the motion of an omnidirectional circular robot. Assume that the robot is equipped with a rotating laser range finder placed at its centre that measures the distance between the sensor and the closest obstacles along each direction. If the distance is larger than the maximum measurable range R, the sensor returns R as a reading. Sketch a possible extension of the method to a unicycle-like mobile robot.

12.13. Implement in a computer program the motion planning method based on the numerical navigation function. The program receives as input a two-dimensional gridmap, in which some of the cells are labelled as 'obstacles', with the specification of the start and the goal cells. If the algorithm is successful, a sequence of free cells from the start to the goal is provided as output. With the aid of some examples, compare the average length of the solution paths obtained using 1-adjacency and 2-adjacency to build the navigation function.

Appendices

A

Linear Algebra

Since modelling and control of robot manipulators requires an extensive use of *matrices* and *vectors* as well as of matrix and vector *operations*, the goal of this appendix is to provide a brush-up of *linear algebra*.

A.1 Definitions

A *matrix* of dimensions $(m \times n)$, with m and n positive integers, is an array of elements a_{ij} arranged into m *rows* and n *columns*:

$$\boldsymbol{A} = [a_{ij}]_{\substack{i=1,\ldots,m \\ j=1,\ldots,n}} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \ldots & a_{mn} \end{bmatrix}. \tag{A.1}$$

If $m = n$, the matrix is said to be *square*; if $m < n$, the matrix has more columns than rows; if $m > n$ the matrix has more rows than columns. Further, if $n = 1$, the notation (A.1) is used to represent a (column) vector $\boldsymbol{a}$ of dimensions $(m \times 1)$;[1] the elements a_i are said to be vector components.

A square matrix $\boldsymbol{A}$ of dimensions $(n \times n)$ is said to be *upper triangular* if $a_{ij} = 0$ for $i > j$:

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ 0 & a_{22} & \ldots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & a_{nn} \end{bmatrix};$$

the matrix is said to be *lower triangular* if $a_{ij} = 0$ for $i < j$.

[1] According to standard mathematical notation, small boldface is used to denote vectors while capital boldface is used to denote matrices. Scalars are denoted by roman characters.

An $(n \times n)$ square matrix $\boldsymbol{A}$ is said to be *diagonal* if $a_{ij} = 0$ for $i \neq j$, i.e.,

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & 0 & \dots & 0 \\ 0 & a_{22} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & a_{nn} \end{bmatrix} = \text{diag}\{a_{11}, a_{22}, \dots, a_{nn}\}.$$

If an $(n \times n)$ diagonal matrix has all unit elements on the diagonal ($a_{ii} = 1$), the matrix is said to be *identity* and is denoted by $\boldsymbol{I}_n$.[2] A matrix is said to be *null* if all its elements are null and is denoted by $\boldsymbol{O}$. The null column vector is denoted by $\boldsymbol{0}$.

The *transpose* $\boldsymbol{A}^T$ of a matrix $\boldsymbol{A}$ of dimensions $(m \times n)$ is the matrix of dimensions $(n \times m)$ which is obtained from the original matrix by interchanging its rows and columns:

$$\boldsymbol{A}^T = \begin{bmatrix} a_{11} & a_{21} & \dots & a_{m1} \\ a_{12} & a_{22} & \dots & a_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \dots & a_{mn} \end{bmatrix}. \tag{A.2}$$

The transpose of a column vector $\boldsymbol{a}$ is the row vector $\boldsymbol{a}^T$.

An $(n \times n)$ square matrix $\boldsymbol{A}$ is said to be *symmetric* if $\boldsymbol{A}^T = \boldsymbol{A}$, and thus $a_{ij} = a_{ji}$:

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{12} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \dots & a_{nn} \end{bmatrix}.$$

An $(n \times n)$ square matrix $\boldsymbol{A}$ is said to be *skew-symmetric* if $\boldsymbol{A}^T = -\boldsymbol{A}$, and thus $a_{ij} = -a_{ji}$ for $i \neq j$ and $a_{ii} = 0$, leading to

$$\boldsymbol{A} = \begin{bmatrix} 0 & a_{12} & \dots & a_{1n} \\ -a_{12} & 0 & \dots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{1n} & -a_{2n} & \dots & 0 \end{bmatrix}.$$

A *partitioned* matrix is a matrix whose elements are matrices (*blocks*) of proper dimensions:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{A}_{11} & \boldsymbol{A}_{12} & \dots & \boldsymbol{A}_{1n} \\ \boldsymbol{A}_{21} & \boldsymbol{A}_{22} & \dots & \boldsymbol{A}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \boldsymbol{A}_{m1} & \boldsymbol{A}_{m2} & \dots & \boldsymbol{A}_{mn} \end{bmatrix}.$$

[2] Subscript n is usually omitted if the dimensions are clear from the context.

A partitioned matrix may be block-triangular or block-diagonal. Special partitions of a matrix are that by columns

$$\boldsymbol{A} = [\, \boldsymbol{a}_1 \quad \boldsymbol{a}_2 \quad \dots \quad \boldsymbol{a}_n \,]$$

and that by rows

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}_1^T \\ \boldsymbol{a}_2^T \\ \vdots \\ \boldsymbol{a}_m^T \end{bmatrix}.$$

Given a square matrix $\boldsymbol{A}$ of dimensions $(n \times n)$, the *algebraic complement* $\boldsymbol{A}_{(ij)}$ of element a_{ij} is the matrix of dimensions $((n-1) \times (n-1))$ which is obtained by eliminating row i and column j of matrix $\boldsymbol{A}$.

A.2 Matrix Operations

The *trace* of an $(n \times n)$ square matrix $\boldsymbol{A}$ is the sum of the elements on the diagonal:

$$\mathrm{Tr}(\boldsymbol{A}) = \sum_{i=1}^{n} a_{ii}. \tag{A.3}$$

Two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ of the same dimensions $(m \times n)$ are equal if $a_{ij} = b_{ij}$. If $\boldsymbol{A}$ and $\boldsymbol{B}$ are two matrices of the same dimensions, their *sum* is the matrix

$$\boldsymbol{C} = \boldsymbol{A} + \boldsymbol{B} \tag{A.4}$$

whose elements are given by $c_{ij} = a_{ij} + b_{ij}$. The following properties hold:

$$\begin{aligned} &\boldsymbol{A} + \boldsymbol{O} = \boldsymbol{A} \\ &\boldsymbol{A} + \boldsymbol{B} = \boldsymbol{B} + \boldsymbol{A} \\ &(\boldsymbol{A} + \boldsymbol{B}) + \boldsymbol{C} = \boldsymbol{A} + (\boldsymbol{B} + \boldsymbol{C}). \end{aligned}$$

Notice that two matrices of the same dimensions and partitioned in the same way can be summed formally by operating on the blocks in the same position and treating them like elements.

The *product of a scalar* α *by an* $(m \times n)$ *matrix* $\boldsymbol{A}$ is the matrix $\alpha\boldsymbol{A}$ whose elements are given by αa_{ij}. If $\boldsymbol{A}$ is an $(n \times n)$ diagonal matrix with all equal elements on the diagonal $(a_{ii} = a)$, it follows that $\boldsymbol{A} = a\boldsymbol{I}_n$.

If $\boldsymbol{A}$ is a square matrix, one may write

$$\boldsymbol{A} = \boldsymbol{A}_s + \boldsymbol{A}_a \tag{A.5}$$

where

$$\boldsymbol{A}_s = \frac{1}{2}(\boldsymbol{A} + \boldsymbol{A}^T) \tag{A.6}$$

is a symmetric matrix representing the *symmetric* part of $\boldsymbol{A}$, and

$$\boldsymbol{A}_a = \frac{1}{2}(\boldsymbol{A} - \boldsymbol{A}^T) \tag{A.7}$$

is a skew-symmetric matrix representing the *skew-symmetric* part of $\boldsymbol{A}$.

The row-by-column *product* of a matrix $\boldsymbol{A}$ of dimensions $(m \times p)$ by a matrix $\boldsymbol{B}$ of dimensions $(p \times n)$ is the matrix of dimensions $(m \times n)$

$$\boldsymbol{C} = \boldsymbol{A}\boldsymbol{B} \tag{A.8}$$

whose elements are given by $c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj}$. The following properties hold:

$$\begin{aligned}
&\boldsymbol{A} = \boldsymbol{A}\boldsymbol{I}_p = \boldsymbol{I}_m \boldsymbol{A} \\
&\boldsymbol{A}(\boldsymbol{B}\boldsymbol{C}) = (\boldsymbol{A}\boldsymbol{B})\boldsymbol{C} \\
&\boldsymbol{A}(\boldsymbol{B} + \boldsymbol{C}) = \boldsymbol{A}\boldsymbol{B} + \boldsymbol{A}\boldsymbol{C} \\
&(\boldsymbol{A} + \boldsymbol{B})\boldsymbol{C} = \boldsymbol{A}\boldsymbol{C} + \boldsymbol{B}\boldsymbol{C} \\
&(\boldsymbol{A}\boldsymbol{B})^T = \boldsymbol{B}^T \boldsymbol{A}^T.
\end{aligned}$$

Notice that, in general, $\boldsymbol{A}\boldsymbol{B} \neq \boldsymbol{B}\boldsymbol{A}$, and $\boldsymbol{A}\boldsymbol{B} = \boldsymbol{O}$ does not imply that $\boldsymbol{A} = \boldsymbol{O}$ or $\boldsymbol{B} = \boldsymbol{O}$; further, notice that $\boldsymbol{A}\boldsymbol{C} = \boldsymbol{B}\boldsymbol{C}$ does not imply that $\boldsymbol{A} = \boldsymbol{B}$.

If an $(m \times p)$ matrix $\boldsymbol{A}$ and a $(p \times n)$ matrix $\boldsymbol{B}$ are partitioned in such a way that the number of blocks for each row of $\boldsymbol{A}$ is equal to the number of blocks for each column of $\boldsymbol{B}$, and the blocks $\boldsymbol{A}_{ik}$ and $\boldsymbol{B}_{kj}$ have dimensions compatible with product, the matrix product $\boldsymbol{A}\boldsymbol{B}$ can be formally obtained by operating by rows and columns on the blocks of proper position and treating them like elements.

For an $(n \times n)$ *square* matrix $\boldsymbol{A}$, the *determinant* of $\boldsymbol{A}$ is the scalar given by the following expression, which holds $\forall i = 1, \ldots, n$:

$$\det(\boldsymbol{A}) = \sum_{j=1}^{n} a_{ij} (-1)^{i+j} \det\big(\boldsymbol{A}_{(ij)}\big). \tag{A.9}$$

The determinant can be computed according to any row i as in (A.9); the same result is obtained by computing it according to any column j. If $n = 1$, then $\det(a_{11}) = a_{11}$. The following property holds:

$$\det(\boldsymbol{A}) = \det(\boldsymbol{A}^T).$$

Moreover, interchanging two generic columns p and q of a matrix $\boldsymbol{A}$ yields

$$\det\big([\, \boldsymbol{a}_1 \ldots \boldsymbol{a}_p \ldots \boldsymbol{a}_q \ldots \boldsymbol{a}_n \,]\big) = -\det\big([\, \boldsymbol{a}_1 \ldots \boldsymbol{a}_q \ldots \boldsymbol{a}_p \ldots \boldsymbol{a}_n \,]\big).$$

As a consequence, if a matrix has two equal columns (rows), then its determinant is null. Also, it is $\det(\alpha \boldsymbol{A}) = \alpha^n \det(\boldsymbol{A})$.

Given an $(m \times n)$ matrix $\boldsymbol{A}$, the determinant of the square block obtained by selecting an equal number k of rows and columns is said to be k-order *minor*

of matrix $\boldsymbol{A}$. The minors obtained by taking the *first* k rows and columns of $\boldsymbol{A}$ are said to be *principal* minors.

If $\boldsymbol{A}$ and $\boldsymbol{B}$ are square matrices, then

$$\det(\boldsymbol{AB}) = \det(\boldsymbol{A})\det(\boldsymbol{B}). \tag{A.10}$$

If $\boldsymbol{A}$ is an $(n \times n)$ triangular matrix (in particular diagonal), then

$$\det(\boldsymbol{A}) = \prod_{i=1}^{n} a_{ii}.$$

More generally, if $\boldsymbol{A}$ is block-triangular with m blocks $\boldsymbol{A}_{ii}$ on the diagonal, then

$$\det(\boldsymbol{A}) = \prod_{i=1}^{m} \det(\boldsymbol{A}_{ii}).$$

A square matrix $\boldsymbol{A}$ is said to be *singular* when $\det(\boldsymbol{A}) = 0$.

The *rank* $\varrho(\boldsymbol{A})$ of a matrix $\boldsymbol{A}$ of dimensions $(m \times n)$ is the maximum integer r so that at least a non-null minor of order r exists. The following properties hold:

$$\begin{aligned}
&\varrho(\boldsymbol{A}) \leq \min\{m, n\} \\
&\varrho(\boldsymbol{A}) = \varrho(\boldsymbol{A}^T) \\
&\varrho(\boldsymbol{A}^T\boldsymbol{A}) = \varrho(\boldsymbol{A}) \\
&\varrho(\boldsymbol{AB}) \leq \min\{\varrho(\boldsymbol{A}), \varrho(\boldsymbol{B})\}.
\end{aligned}$$

A matrix so that $\varrho(\boldsymbol{A}) = \min\{m, n\}$ is said to be *full-rank*.

The *adjoint* of a square matrix $\boldsymbol{A}$ is the matrix

$$\operatorname{Adj} \boldsymbol{A} = [(-1)^{i+j}\det(\boldsymbol{A}_{(ij)})]^T_{\substack{i=1,\ldots,n \\ j=1,\ldots,n}}. \tag{A.11}$$

An $(n \times n)$ square matrix $\boldsymbol{A}$ is said to be *invertible* if a matrix $\boldsymbol{A}^{-1}$ exists, termed *inverse* of $\boldsymbol{A}$, so that

$$\boldsymbol{A}^{-1}\boldsymbol{A} = \boldsymbol{A}\boldsymbol{A}^{-1} = \boldsymbol{I}_n.$$

Since $\varrho(\boldsymbol{I}_n) = n$, an $(n \times n)$ square matrix $\boldsymbol{A}$ is invertible if and only if $\varrho(\boldsymbol{A}) = n$, i.e., $\det(\boldsymbol{A}) \neq 0$ (nonsingular matrix). The inverse of $\boldsymbol{A}$ can be computed as

$$\boldsymbol{A}^{-1} = \frac{1}{\det(\boldsymbol{A})} \operatorname{Adj} \boldsymbol{A}. \tag{A.12}$$

The following properties hold:

$$\begin{aligned}
&(\boldsymbol{A}^{-1})^{-1} = \boldsymbol{A} \\
&(\boldsymbol{A}^T)^{-1} = (\boldsymbol{A}^{-1})^T.
\end{aligned}$$

If the inverse of a square matrix is equal to its transpose

$$\boldsymbol{A}^T = \boldsymbol{A}^{-1} \tag{A.13}$$

then the matrix is said to be *orthogonal*; in this case it is

$$\boldsymbol{A}\boldsymbol{A}^T = \boldsymbol{A}^T\boldsymbol{A} = \boldsymbol{I}. \tag{A.14}$$

A square matrix $\boldsymbol{A}$ is said *idempotent* if

$$\boldsymbol{A}\boldsymbol{A} = \boldsymbol{A}. \tag{A.15}$$

If $\boldsymbol{A}$ and $\boldsymbol{B}$ are invertible square matrices of the same dimensions, then

$$(\boldsymbol{A}\boldsymbol{B})^{-1} = \boldsymbol{B}^{-1}\boldsymbol{A}^{-1}. \tag{A.16}$$

Given n square matrices $\boldsymbol{A}_{ii}$ all invertible, the following expression holds:

$$\big(\mathrm{diag}\{\boldsymbol{A}_{11}, \ldots, \boldsymbol{A}_{nn}\}\big)^{-1} = \mathrm{diag}\{\boldsymbol{A}_{11}^{-1}, \ldots, \boldsymbol{A}_{nn}^{-1}\}.$$

where $\mathrm{diag}\{\boldsymbol{A}_{11}, \ldots, \boldsymbol{A}_{nn}\}$ denotes the block-diagonal matrix.

If $\boldsymbol{A}$ and $\boldsymbol{C}$ are invertible square matrices of proper dimensions, the following expression holds:

$$(\boldsymbol{A} + \boldsymbol{B}\boldsymbol{C}\boldsymbol{D})^{-1} = \boldsymbol{A}^{-1} - \boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{D}\boldsymbol{A}^{-1}\boldsymbol{B} + \boldsymbol{C}^{-1})^{-1}\boldsymbol{D}\boldsymbol{A}^{-1},$$

where the matrix $\boldsymbol{D}\boldsymbol{A}^{-1}\boldsymbol{B} + \boldsymbol{C}^{-1}$ must be invertible.

If a block-partitioned matrix is invertible, then its inverse is given by the general expression

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{D} \\ \boldsymbol{C} & \boldsymbol{B} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{A}^{-1} + \boldsymbol{E}\boldsymbol{\Delta}^{-1}\boldsymbol{F} & -\boldsymbol{E}\boldsymbol{\Delta}^{-1} \\ -\boldsymbol{\Delta}^{-1}\boldsymbol{F} & \boldsymbol{\Delta}^{-1} \end{bmatrix} \tag{A.17}$$

where $\boldsymbol{\Delta} = \boldsymbol{B} - \boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{D}$, $\boldsymbol{E} = \boldsymbol{A}^{-1}\boldsymbol{D}$ and $\boldsymbol{F} = \boldsymbol{C}\boldsymbol{A}^{-1}$, under the assumption that the inverses of matrices $\boldsymbol{A}$ and $\boldsymbol{\Delta}$ exist. In the case of a block-triangular matrix, invertibility of the matrix requires invertibility of the blocks on the diagonal. The following expressions hold:

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{O} \\ \boldsymbol{C} & \boldsymbol{B} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{A}^{-1} & \boldsymbol{O} \\ -\boldsymbol{B}^{-1}\boldsymbol{C}\boldsymbol{A}^{-1} & \boldsymbol{B}^{-1} \end{bmatrix}$$

$$\begin{bmatrix} \boldsymbol{A} & \boldsymbol{D} \\ \boldsymbol{O} & \boldsymbol{B} \end{bmatrix}^{-1} = \begin{bmatrix} \boldsymbol{A}^{-1} & -\boldsymbol{A}^{-1}\boldsymbol{D}\boldsymbol{B}^{-1} \\ \boldsymbol{O} & \boldsymbol{B}^{-1} \end{bmatrix}.$$

The *derivative* of an $(m \times n)$ matrix $\boldsymbol{A}(t)$, whose elements $a_{ij}(t)$ are differentiable functions, is the matrix

$$\dot{\boldsymbol{A}}(t) = \frac{d}{dt}\boldsymbol{A}(t) = \left[\frac{d}{dt}a_{ij}(t)\right]_{\substack{i=1,\ldots,m \\ j=1,\ldots,n}}. \tag{A.18}$$

If an $(n \times n)$ square matrix $\boldsymbol{A}(t)$ is so that $\varrho(\boldsymbol{A}(t)) = n \ \forall t$ and its elements $a_{ij}(t)$ are differentiable functions, then the derivative of the *inverse* of $\boldsymbol{A}(t)$ is given by

$$\frac{d}{dt}\boldsymbol{A}^{-1}(t) = -\boldsymbol{A}^{-1}(t)\dot{\boldsymbol{A}}(t)\boldsymbol{A}^{-1}(t). \tag{A.19}$$

Given a scalar function $f(\boldsymbol{x})$, endowed with partial derivatives with respect to the elements x_i of the $(n \times 1)$ vector $\boldsymbol{x}$, the *gradient* of function f with respect to vector $\boldsymbol{x}$ is the $(n \times 1)$ column vector

$$\nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \left(\frac{\partial f(\boldsymbol{x})}{\partial \boldsymbol{x}}\right)^T = \left[\begin{array}{cccc} \dfrac{\partial f(\boldsymbol{x})}{\partial x_1} & \dfrac{\partial f(\boldsymbol{x})}{\partial x_2} & \ldots & \dfrac{\partial f(\boldsymbol{x})}{\partial x_n} \end{array}\right]^T. \tag{A.20}$$

Further, if $\boldsymbol{x}(t)$ is a differentiable function with respect to t, then

$$\dot{f}(\boldsymbol{x}) = \frac{d}{dt} f(\boldsymbol{x}(t)) = \frac{\partial f}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} = \nabla_{\boldsymbol{x}}^T f(\boldsymbol{x})\dot{\boldsymbol{x}}. \tag{A.21}$$

Given a vector function $\boldsymbol{g}(\boldsymbol{x})$ of dimensions $(m \times 1)$, whose elements g_i are differentiable with respect to the vector $\boldsymbol{x}$ of dimensions $(n \times 1)$, the Jacobian matrix (or simply *Jacobian*) of the function is defined as the $(m \times n)$ matrix

$$\boldsymbol{J}_g(\boldsymbol{x}) = \frac{\partial \boldsymbol{g}(\boldsymbol{x})}{\partial \boldsymbol{x}} = \left[\begin{array}{c} \dfrac{\partial g_1(\boldsymbol{x})}{\partial \boldsymbol{x}} \\ \dfrac{\partial g_2(\boldsymbol{x})}{\partial \boldsymbol{x}} \\ \vdots \\ \dfrac{\partial g_m(\boldsymbol{x})}{\partial \boldsymbol{x}} \end{array}\right]. \tag{A.22}$$

If $\boldsymbol{x}(t)$ is a differentiable function with respect to t, then

$$\dot{\boldsymbol{g}}(\boldsymbol{x}) = \frac{d}{dt}\boldsymbol{g}(\boldsymbol{x}(t)) = \frac{\partial \boldsymbol{g}}{\partial \boldsymbol{x}}\dot{\boldsymbol{x}} = \boldsymbol{J}_g(\boldsymbol{x})\dot{\boldsymbol{x}}. \tag{A.23}$$

A.3 Vector Operations

Given n vectors $\boldsymbol{x}_i$ of dimensions $(m \times 1)$, they are said to be *linearly independent* if the expression

$$k_1\boldsymbol{x}_1 + k_2\boldsymbol{x}_2 + \ldots + k_n\boldsymbol{x}_n = \boldsymbol{0}$$

holds true only when all the constants k_i vanish. A necessary and sufficient condition for the vectors $\boldsymbol{x}_1, \boldsymbol{x}_2 \ldots, \boldsymbol{x}_n$ to be linearly independent is that the matrix

$$\boldsymbol{A} = [\, \boldsymbol{x}_1 \quad \boldsymbol{x}_2 \quad \ldots \quad \boldsymbol{x}_n \,]$$

has rank n; this implies that a necessary condition for linear independence is that $n \leq m$. If instead $\varrho(\boldsymbol{A}) = r < n$, then only r vectors are linearly independent and the remaining $n - r$ vectors can be expressed as a linear combination of the previous ones.

A system of vectors $\mathcal{X}$ is a *vector space* on the field of real numbers $\mathbb{R}$ if the operations of *sum of two vectors* of $\mathcal{X}$ and *product of a scalar by a vector* of $\mathcal{X}$ have values in $\mathcal{X}$ and the following properties hold:

$$\begin{aligned}
&\boldsymbol{x} + \boldsymbol{y} = \boldsymbol{y} + \boldsymbol{x} \qquad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X} \\
&(\boldsymbol{x} + \boldsymbol{y}) + \boldsymbol{z} = \boldsymbol{x} + (\boldsymbol{y} + \boldsymbol{z}) \qquad \forall \boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z} \in \mathcal{X} \\
&\exists \mathbf{0} \in \mathcal{X} : \boldsymbol{x} + \mathbf{0} = \boldsymbol{x} \qquad \forall \boldsymbol{x} \in \mathcal{X} \\
&\forall \boldsymbol{x} \in \mathcal{X}, \quad \exists (-\boldsymbol{x}) \in \mathcal{X} : \boldsymbol{x} + (-\boldsymbol{x}) = \mathbf{0} \\
&1\boldsymbol{x} = \boldsymbol{x} \qquad \forall \boldsymbol{x} \in \mathcal{X} \\
&\alpha(\beta \boldsymbol{x}) = (\alpha\beta)\boldsymbol{x} \qquad \forall \alpha, \beta \in \mathbb{R} \quad \forall \boldsymbol{x} \in \mathcal{X} \\
&(\alpha + \beta)\boldsymbol{x} = \alpha \boldsymbol{x} + \beta \boldsymbol{x} \qquad \forall \alpha, \beta \in \mathbb{R} \quad \forall \boldsymbol{x} \in \mathcal{X} \\
&\alpha(\boldsymbol{x} + \boldsymbol{y}) = \alpha \boldsymbol{x} + \alpha \boldsymbol{y} \qquad \forall \alpha \in \mathbb{R} \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}.
\end{aligned}$$

The *dimension* of the space $\dim(\mathcal{X})$ is the maximum number of linearly independent vectors $\boldsymbol{x}$ in the space. A set $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$ of linearly independent vectors is a *basis* of vector space $\mathcal{X}$, and each vector $\boldsymbol{y}$ in the space can be uniquely expressed as a linear combination of vectors from the basis

$$\boldsymbol{y} = c_1 \boldsymbol{x}_1 + c_2 \boldsymbol{x}_2 + \ldots + c_n \boldsymbol{x}_n, \tag{A.24}$$

where the constants $c_1, c_2, \ldots, c_n$ are said to be the *components* of the vector $\boldsymbol{y}$ in the basis $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$.

A subset $\mathcal{Y}$ of a vector space $\mathcal{X}$ is a *subspace* $\mathcal{Y} \subseteq \mathcal{X}$ if it is a vector space with the operations of vector sum and product of a scalar by a vector, i.e.,

$$\alpha \boldsymbol{x} + \beta \boldsymbol{y} \in \mathcal{Y} \qquad \forall \alpha, \beta \in \mathbb{R} \quad \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{Y}.$$

According to a geometric interpretation, a subspace is a *hyperplane* passing by the origin (null element) of $\mathcal{X}$.

The *scalar product* $< \boldsymbol{x}, \boldsymbol{y} >$ of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ of dimensions $(m \times 1)$ is the scalar that is obtained by summing the products of the respective components in a given basis

$$< \boldsymbol{x}, \boldsymbol{y} >= x_1 y_1 + x_2 y_2 + \ldots + x_m y_m = \boldsymbol{x}^T \boldsymbol{y} = \boldsymbol{y}^T \boldsymbol{x}. \tag{A.25}$$

Two vectors are said to be *orthogonal* when their scalar product is null:

$$\boldsymbol{x}^T \boldsymbol{y} = 0. \tag{A.26}$$

The *norm* of a vector can be defined as

$$\|\boldsymbol{x}\| = \sqrt{\boldsymbol{x}^T \boldsymbol{x}}. \tag{A.27}$$

It is possible to show that both the *triangle inequality*

$$\|\boldsymbol{x}+\boldsymbol{y}\| \leq \|\boldsymbol{x}\| + \|\boldsymbol{y}\| \tag{A.28}$$

and the *Schwarz inequality*

$$|\boldsymbol{x}^T\boldsymbol{y}| \leq \|\boldsymbol{x}\|\,\|\boldsymbol{y}\|. \tag{A.29}$$

hold. A *unit vector* $\hat{\boldsymbol{x}}$ is a vector whose *norm* is unity, i.e., $\hat{\boldsymbol{x}}^T\hat{\boldsymbol{x}} = 1$. Given a vector $\boldsymbol{x}$, its unit vector is obtained by dividing each component by its norm:

$$\widehat{\boldsymbol{x}} = \frac{1}{\|\boldsymbol{x}\|}\boldsymbol{x}. \tag{A.30}$$

A typical example of vector space is the *Euclidean space* whose dimension is 3; in this case a basis is constituted by the unit vectors of a coordinate frame.

The *vector product* of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ in the Euclidean space is the vector

$$\boldsymbol{x}\times\boldsymbol{y} = \begin{bmatrix} x_2y_3 - x_3y_2 \\ x_3y_1 - x_1y_3 \\ x_1y_2 - x_2y_1 \end{bmatrix}. \tag{A.31}$$

The following properties hold:

$$\begin{aligned} &\boldsymbol{x}\times\boldsymbol{x} = \boldsymbol{0} \\ &\boldsymbol{x}\times\boldsymbol{y} = -\boldsymbol{y}\times\boldsymbol{x} \\ &\boldsymbol{x}\times(\boldsymbol{y}+\boldsymbol{z}) = \boldsymbol{x}\times\boldsymbol{y} + \boldsymbol{x}\times\boldsymbol{z}. \end{aligned}$$

The vector product of two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ can be expressed also as the product of a matrix operator $\boldsymbol{S}(\boldsymbol{x})$ by the vector $\boldsymbol{y}$. In fact, by introducing the *skew-symmetric* matrix

$$\boldsymbol{S}(\boldsymbol{x}) = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix} \tag{A.32}$$

obtained with the components of vector $\boldsymbol{x}$, the vector product $\boldsymbol{x}\times\boldsymbol{y}$ is given by

$$\boldsymbol{x}\times\boldsymbol{y} = \boldsymbol{S}(\boldsymbol{x})\boldsymbol{y} = -\boldsymbol{S}(\boldsymbol{y})\boldsymbol{x} \tag{A.33}$$

as can be easily verified. Moreover, the following properties hold:

$$\begin{aligned} &\boldsymbol{S}(\boldsymbol{x})\boldsymbol{x} = \boldsymbol{S}^T(\boldsymbol{x})\boldsymbol{x} = \boldsymbol{0} \\ &\boldsymbol{S}(\alpha\boldsymbol{x}+\beta\boldsymbol{y}) = \alpha\boldsymbol{S}(\boldsymbol{x}) + \beta\boldsymbol{S}(\boldsymbol{y}). \end{aligned}$$

Given three vectors $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$ in the Euclidean space, the following expressions hold for the *scalar triple products*:

$$\boldsymbol{x}^T(\boldsymbol{y}\times\boldsymbol{z}) = \boldsymbol{y}^T(\boldsymbol{z}\times\boldsymbol{x}) = \boldsymbol{z}^T(\boldsymbol{x}\times\boldsymbol{y}). \tag{A.34}$$

If any two vectors of three are equal, then the scalar triple product is null; e.g.,

$$\boldsymbol{x}^T(\boldsymbol{x}\times\boldsymbol{y}) = \boldsymbol{0}.$$

A.4 Linear Transformation

Consider a vector space $\mathcal{X}$ of dimension n and a vector space $\mathcal{Y}$ of dimension m with $m \leq n$. The *linear transformation* (or linear map) between the vectors $\boldsymbol{x} \in \mathcal{X}$ and $\boldsymbol{y} \in \mathcal{Y}$ can be defined as

$$\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x} \tag{A.35}$$

in terms of the matrix $\boldsymbol{A}$ of dimensions $(m \times n)$. The *range space* (or simply range) of the transformation is the subspace

$$\mathcal{R}(\boldsymbol{A}) = \{\boldsymbol{y} : \boldsymbol{y} = \boldsymbol{A}\boldsymbol{x},\ \boldsymbol{x} \in \mathcal{X}\} \subseteq \mathcal{Y}, \tag{A.36}$$

which is the subspace generated by the linearly independent columns of matrix $\boldsymbol{A}$ taken as a basis of $\mathcal{Y}$. It is easy to recognize that

$$\varrho(\boldsymbol{A}) = \dim(\mathcal{R}(\boldsymbol{A})). \tag{A.37}$$

On the other hand, the *null space* (or simply null) of the transformation is the subspace

$$\mathcal{N}(\boldsymbol{A}) = \{\boldsymbol{x} : \boldsymbol{A}\boldsymbol{x} = \boldsymbol{0},\ \boldsymbol{x} \in \mathcal{X}\} \subseteq \mathcal{X}. \tag{A.38}$$

Given a matrix $\boldsymbol{A}$ of dimensions $(m \times n)$, the notable result holds:

$$\varrho(\boldsymbol{A}) + \dim(\mathcal{N}(\boldsymbol{A})) = n. \tag{A.39}$$

Therefore, if $\varrho(\boldsymbol{A}) = r \leq \min\{m, n\}$, then $\dim(\mathcal{R}(\boldsymbol{A})) = r$ and $\dim(\mathcal{N}(\boldsymbol{A})) = n - r$. It follows that if $m < n$, then $\mathcal{N}(\boldsymbol{A}) \neq \emptyset$ independently of the rank of $\boldsymbol{A}$; if $m = n$, then $\mathcal{N}(\boldsymbol{A}) \neq \emptyset$ only in the case of $\varrho(\boldsymbol{A}) = r < m$.

If $\boldsymbol{x} \in \mathcal{N}(\boldsymbol{A})$ and $\boldsymbol{y} \in \mathcal{R}(\boldsymbol{A}^T)$, then $\boldsymbol{y}^T\boldsymbol{x} = 0$, i.e., the vectors in the null space of $\boldsymbol{A}$ are orthogonal to each vector in the range space of the transpose of $\boldsymbol{A}$. It can be shown that the set of vectors orthogonal to each vector of the range space of $\boldsymbol{A}^T$ coincides with the null space of $\boldsymbol{A}$, whereas the set of vectors orthogonal to each vector in the null space of $\boldsymbol{A}^T$ coincides with the range space of $\boldsymbol{A}$. In symbols:

$$\mathcal{N}(\boldsymbol{A}) \equiv \mathcal{R}^{\perp}(\boldsymbol{A}^T) \qquad \mathcal{R}(\boldsymbol{A}) \equiv \mathcal{N}^{\perp}(\boldsymbol{A}^T) \tag{A.40}$$

where $\perp$ denotes the *orthogonal complement* of a subspace.

If the matrix $\boldsymbol{A}$ in (A.35) is square and idempotent, the matrix represents the *projection* of space $\mathcal{X}$ into a subspace.

A linear transformation allows the definition of the *norm* of a matrix $\boldsymbol{A}$ induced by the norm defined for a vector $\boldsymbol{x}$ as follows. In view of the property

$$\|\boldsymbol{A}\boldsymbol{x}\| \leq \|\boldsymbol{A}\|\ \|\boldsymbol{x}\|, \tag{A.41}$$

the norm of $\boldsymbol{A}$ can be defined as

$$\|\boldsymbol{A}\| = \sup_{\boldsymbol{x} \neq \boldsymbol{0}} \frac{\|\boldsymbol{A}\boldsymbol{x}\|}{\|\boldsymbol{x}\|} \tag{A.42}$$

which can also be computed as

$$\max_{\|\boldsymbol{x}\|=1} \|\boldsymbol{A}\boldsymbol{x}\|.$$

A direct consequence of (A.41) is the property

$$\|\boldsymbol{A}\boldsymbol{B}\| \leq \|\boldsymbol{A}\| \, \|\boldsymbol{B}\|. \tag{A.43}$$

A different norm of a matrix is the *Frobenius norm* defined as

$$\|\boldsymbol{A}\|_F = \Big(\mathrm{Tr}(\boldsymbol{A}^T\boldsymbol{A})\Big)^{1/2} \tag{A.44}$$

A.5 Eigenvalues and Eigenvectors

Consider the linear transformation on a vector $\boldsymbol{u}$ established by an $(n \times n)$ square matrix $\boldsymbol{A}$. If the vector resulting from the transformation has the same direction of $\boldsymbol{u}$ (with $\boldsymbol{u} \neq \boldsymbol{0}$), then

$$\boldsymbol{A}\boldsymbol{u} = \lambda\boldsymbol{u}. \tag{A.45}$$

The equation in (A.45) can be rewritten in matrix form as

$$(\lambda\boldsymbol{I} - \boldsymbol{A})\boldsymbol{u} = \boldsymbol{0}. \tag{A.46}$$

For the homogeneous system of equations in (A.46) to have a solution different from the trivial one $\boldsymbol{u} = \boldsymbol{0}$, it must be

$$\det(\lambda\boldsymbol{I} - \boldsymbol{A}) = 0 \tag{A.47}$$

which is termed a *characteristic equation*. Its solutions $\lambda_1, \ldots, \lambda_n$ are the *eigenvalues* of matrix $\boldsymbol{A}$; they coincide with the eigenvalues of matrix $\boldsymbol{A}^T$. On the assumption of distinct eigenvalues, the n vectors $\boldsymbol{u}_i$ satisfying the equation

$$(\lambda_i\boldsymbol{I} - \boldsymbol{A})\boldsymbol{u}_i = \boldsymbol{0} \qquad i = 1, \ldots, n \tag{A.48}$$

are said to be the *eigenvectors* associated with the eigenvalues λ_i.

The matrix $\boldsymbol{U}$ formed by the column vectors $\boldsymbol{u}_i$ is invertible and constitutes a basis in the space of dimension n. Further, the *similarity transformation* established by $\boldsymbol{U}$

$$\boldsymbol{\Lambda} = \boldsymbol{U}^{-1}\boldsymbol{A}\boldsymbol{U} \tag{A.49}$$

is so that $\boldsymbol{\Lambda} = \mathrm{diag}\{\lambda_1, \ldots, \lambda_n\}$. It follows that $\det(\boldsymbol{A}) = \prod_{i=1}^{n} \lambda_i$.

If the matrix $\boldsymbol{A}$ is *symmetric*, its eigenvalues are real and $\boldsymbol{\Lambda}$ can be written as

$$\boldsymbol{\Lambda} = \boldsymbol{U}^T\boldsymbol{A}\boldsymbol{U}; \tag{A.50}$$

hence, the eigenvector matrix $\boldsymbol{U}$ is orthogonal.

A.6 Bilinear Forms and Quadratic Forms

A *bilinear form* in the variables x_i and y_j is the scalar

$$B = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} x_i y_j$$

which can be written in matrix form

$$B(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{y} = \boldsymbol{y}^T \boldsymbol{A}^T \boldsymbol{x} \tag{A.51}$$

where $\boldsymbol{x} = [\, x_1 \quad x_2 \quad \dots \quad x_m \,]^T$, $\boldsymbol{y} = [\, y_1 \quad y_2 \quad \dots \quad y_n \,]^T$, and $\boldsymbol{A}$ is the $(m \times n)$ matrix of the coefficients a_{ij} representing the core of the form.

A special case of bilinear form is the *quadratic form*

$$Q(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \tag{A.52}$$

where $\boldsymbol{A}$ is an $(n \times n)$ square matrix. Hence, for computation of (A.52), the matrix $\boldsymbol{A}$ can be replaced with its symmetric part $\boldsymbol{A}_s$ given by (A.6). It follows that if $\boldsymbol{A}$ is a *skew-symmetric* matrix, then

$$\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} = \boldsymbol{0} \qquad \forall \boldsymbol{x}.$$

The quadratic form (A.52) is said to be *positive definite* if

$$\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} > 0 \quad \forall \boldsymbol{x} \neq \boldsymbol{0} \qquad \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} = 0 \quad \boldsymbol{x} = \boldsymbol{0}. \tag{A.53}$$

The matrix $\boldsymbol{A}$ core of the form is also said to be *positive definite*. Analogously, a quadratic form is said to be *negative definite* if it can be written as $-Q(\boldsymbol{x}) = -\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x}$ where $Q(\boldsymbol{x})$ is positive definite.

A necessary condition for a square matrix to be positive definite is that its elements on the diagonal are strictly positive. Further, in view of (A.50), the eigenvalues of a positive definite matrix are all positive. If the eigenvalues are not known, a necessary and sufficient condition for a symmetric matrix to be positive definite is that its principal minors are strictly positive (*Sylvester criterion*). It follows that a positive definite matrix is full-rank and thus it is always invertible.

A symmetric positive definite matrix $\boldsymbol{A}$ can always be decomposed as

$$\boldsymbol{A} = \boldsymbol{U}^T \boldsymbol{\Lambda} \boldsymbol{U} \tag{A.54}$$

where $\boldsymbol{U}$ is an orthogonal matrix of eigenvectors ($\boldsymbol{U}^T \boldsymbol{U} = \boldsymbol{I}$) and $\boldsymbol{\Lambda}$ is the diagonal matrix of the eigenvalues of $\boldsymbol{A}$.

Let $\lambda_{\min}(\boldsymbol{A})$ and $\lambda_{\max}(\boldsymbol{A})$ respectively denote the smallest and largest eigenvalues of a positive definite matrix $\boldsymbol{A}$ ($\lambda_{\min}, \lambda_{\max} > 0$). Then, the quadratic form in (A.52) satisfies the following inequality:

$$\lambda_{\min}(\boldsymbol{A}) \|\boldsymbol{x}\|^2 \leq \boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \leq \lambda_{\max}(\boldsymbol{A}) \|\boldsymbol{x}\|^2. \tag{A.55}$$

An $(n \times n)$ square matrix $\boldsymbol{A}$ is said to be *positive semi-definite* if

$$\boldsymbol{x}^T \boldsymbol{A} \boldsymbol{x} \geq 0 \qquad \forall \boldsymbol{x}. \tag{A.56}$$

This definition implies that $\varrho(\boldsymbol{A}) = r < n$, and thus r eigenvalues of $\boldsymbol{A}$ are positive and $n - r$ are null. Therefore, a positive semi-definite matrix $\boldsymbol{A}$ has a null space of finite dimension, and specifically the form vanishes when $\boldsymbol{x} \in \mathcal{N}(\boldsymbol{A})$. A typical example of a positive semi-definite matrix is the matrix $\boldsymbol{A} = \boldsymbol{H}^T \boldsymbol{H}$ where $\boldsymbol{H}$ is an $(m \times n)$ matrix with $m < n$. In an analogous way, a *negative semi-definite* matrix can be defined.

Given the *bilinear form* in (A.51), the *gradient* of the form with respect to $\boldsymbol{x}$ is given by

$$\nabla_{\boldsymbol{x}} B(\boldsymbol{x}, \boldsymbol{y}) = \left(\frac{\partial B(\boldsymbol{x}, \boldsymbol{y})}{\partial \boldsymbol{x}} \right)^T = \boldsymbol{A} \boldsymbol{y}, \tag{A.57}$$

whereas the gradient of B with respect to $\boldsymbol{y}$ is given by

$$\nabla_{\boldsymbol{y}} B(\boldsymbol{x}, \boldsymbol{y}) = \left(\frac{\partial B(\boldsymbol{x}, \boldsymbol{y})}{\partial \boldsymbol{y}} \right)^T = \boldsymbol{A}^T \boldsymbol{x}. \tag{A.58}$$

Given the *quadratic form* in (A.52) with $\boldsymbol{A}$ *symmetric*, the *gradient* of the form with respect to $\boldsymbol{x}$ is given by

$$\nabla_{\boldsymbol{x}} Q(\boldsymbol{x}) = \left(\frac{\partial Q(\boldsymbol{x})}{\partial \boldsymbol{x}} \right)^T = 2 \boldsymbol{A} \boldsymbol{x}. \tag{A.59}$$

Further, if $\boldsymbol{x}$ and $\boldsymbol{A}$ are differentiable functions of t, then

$$\dot{Q}(x) = \frac{d}{dt} Q(\boldsymbol{x}(t)) = 2 \boldsymbol{x}^T \boldsymbol{A} \dot{\boldsymbol{x}} + \boldsymbol{x}^T \dot{\boldsymbol{A}} \boldsymbol{x}; \tag{A.60}$$

if $\boldsymbol{A}$ is constant, then the second term obviously vanishes.

A.7 Pseudo-inverse

The inverse of a matrix can be defined only when the matrix is square and nonsingular. The inverse operation can be extended to the case of non-square matrices. Consider a matrix $\boldsymbol{A}$ of dimensions $(m \times n)$ with $\varrho(\boldsymbol{A}) = \min\{m, n\}$

If $m < n$, a *right inverse* of $\boldsymbol{A}$ can be defined as the matrix $\boldsymbol{A}_r$ of dimensions $(n \times m)$ so that

$$\boldsymbol{A} \boldsymbol{A}_r = \boldsymbol{I}_m.$$

If instead $m > n$, a *left inverse* of $\boldsymbol{A}$ can be defined as the matrix $\boldsymbol{A}_l$ of dimensions $(n \times m)$ so that

$$\boldsymbol{A}_l \boldsymbol{A} = \boldsymbol{I}_n.$$

If $\boldsymbol{A}$ has more columns than rows ($m < n$) and has rank m, a special right inverse is the matrix

$$\boldsymbol{A}_r^\dagger = \boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{A}^T)^{-1} \tag{A.61}$$

which is termed *right pseudo-inverse*, since $\boldsymbol{A}\boldsymbol{A}_r^\dagger = \boldsymbol{I}_m$. If $\boldsymbol{W}_r$ is an $(n \times n)$ *positive definite* matrix, a *weighted* right pseudo-inverse is given by

$$\boldsymbol{A}_r^\dagger = \boldsymbol{W}_r^{-1}\boldsymbol{A}^T(\boldsymbol{A}\boldsymbol{W}_r^{-1}\boldsymbol{A}^T)^{-1}. \tag{A.62}$$

If $\boldsymbol{A}$ has more rows than columns ($m > n$) and has rank n, a special left inverse is the matrix

$$\boldsymbol{A}_l^\dagger = (\boldsymbol{A}^T\boldsymbol{A})^{-1}\boldsymbol{A}^T \tag{A.63}$$

which is termed *left pseudo-inverse*, since $\boldsymbol{A}_l^\dagger\boldsymbol{A} = \boldsymbol{I}_n$.[3] If $\boldsymbol{W}_l$ is an $(m \times m)$ *positive definite* matrix, a *weighted* left pseudo-inverse is given by

$$\boldsymbol{A}_l^\dagger = (\boldsymbol{A}^T\boldsymbol{W}_l\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{W}_l. \tag{A.64}$$

The pseudo-inverse is very useful to invert a linear transformation $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ with $\boldsymbol{A}$ a full-rank matrix. If $\boldsymbol{A}$ is a square nonsingular matrix, then obviously $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{y}$ and then $\boldsymbol{A}_l^\dagger = \boldsymbol{A}_r^\dagger = \boldsymbol{A}^{-1}$.

If $\boldsymbol{A}$ has more columns than rows ($m < n$) and has rank m, then the solution $\boldsymbol{x}$ for a given $\boldsymbol{y}$ is not unique; it can be shown that the expression

$$\boldsymbol{x} = \boldsymbol{A}^\dagger\boldsymbol{y} + (\boldsymbol{I} - \boldsymbol{A}^\dagger\boldsymbol{A})\boldsymbol{k}, \tag{A.65}$$

with $\boldsymbol{k}$ an arbitrary $(n \times 1)$ vector and $\boldsymbol{A}^\dagger$ as in (A.61), is a solution to the system of linear equations established by (A.35). The term $\boldsymbol{A}^\dagger\boldsymbol{y} \in \mathcal{N}^\perp(\boldsymbol{A}) \equiv \mathcal{R}(\boldsymbol{A}^T)$ minimizes the norm of the solution $\|\boldsymbol{x}\|$. The term $(\boldsymbol{I} - \boldsymbol{A}^\dagger\boldsymbol{A})\boldsymbol{k}$ is the projection of $\boldsymbol{k}$ in $\mathcal{N}(\boldsymbol{A})$ and is termed *homogeneous solution*; as $\boldsymbol{k}$ varies, all the solutions to the homogeneous equation system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{0}$ associated with (A.35) are generated.

On the other hand, if $\boldsymbol{A}$ has more rows than columns ($m > n$), the equation in (A.35) has no solution; it can be shown that an *approximate* solution is given by

$$\boldsymbol{x} = \boldsymbol{A}^\dagger\boldsymbol{y} \tag{A.66}$$

where $\boldsymbol{A}^\dagger$ as in (A.63) minimizes $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|$. If instead $\boldsymbol{y} \in \mathcal{R}(\boldsymbol{A})$, then (A.66) is a real solution.

Notice that the use of the weighted (left or right) pseudo-inverses in the solution to the linear equation systems leads to analogous results where the minimized norms are weighted according to the metrics defined by matrices $\boldsymbol{W}_r$ and $\boldsymbol{W}_l$, respectively.

The results of this section can be easily extended to the case of (square or nonsquare) matrices $\boldsymbol{A}$ not having full-rank. In particular, the expression (A.66) (with the pseudo-inverse computed by means of the singular value decomposition of $\boldsymbol{A}$) gives the minimum-norm vector among all those minimizing $\|\boldsymbol{y} - \boldsymbol{A}\boldsymbol{x}\|$.

[3] Subscripts l and r are usually omitted whenever the use of a left or right pseudo-inverse is clear from the context.

A.8 Singular Value Decomposition

For a nonsquare matrix it is not possible to define eigenvalues. An extension of the eigenvalue concept can be obtained by singular values. Given a matrix $\boldsymbol{A}$ of dimensions $(m \times n)$, the matrix $\boldsymbol{A}^T\boldsymbol{A}$ has n nonnegative eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n \geq 0$ (ordered from the largest to the smallest) which can be expressed in the form

$$\lambda_i = \sigma_i^2 \qquad \sigma_i \geq 0.$$

The scalars $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_n \geq 0$ are said to be the *singular values* of matrix $\boldsymbol{A}$. The *singular value decomposition* (SVD) of matrix $\boldsymbol{A}$ is given by

$$\boldsymbol{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{A.67}$$

where $\boldsymbol{U}$ is an $(m \times m)$ orthogonal matrix

$$\boldsymbol{U} = [\, \boldsymbol{u}_1 \quad \boldsymbol{u}_2 \quad \ldots \quad \boldsymbol{u}_m \,], \tag{A.68}$$

$\boldsymbol{V}$ is an $(n \times n)$ orthogonal matrix

$$\boldsymbol{V} = [\, \boldsymbol{v}_1 \quad \boldsymbol{v}_2 \quad \ldots \quad \boldsymbol{v}_n \,] \tag{A.69}$$

and $\boldsymbol{\Sigma}$ is an $(m \times n)$ matrix

$$\boldsymbol{\Sigma} = \begin{bmatrix} \boldsymbol{D} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} \end{bmatrix} \qquad \boldsymbol{D} = \mathrm{diag}\{\sigma_1, \sigma_2, \ldots, \sigma_r\} \tag{A.70}$$

where $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > 0$. The number of non-null singular values is equal to the rank r of matrix $\boldsymbol{A}$.

The columns of $\boldsymbol{U}$ are the eigenvectors of the matrix $\boldsymbol{A}\boldsymbol{A}^T$, whereas the columns of $\boldsymbol{V}$ are the eigenvectors of the matrix $\boldsymbol{A}^T\boldsymbol{A}$. In view of the partitions of $\boldsymbol{U}$ and $\boldsymbol{V}$ in (A.68), (A.69), it is $\boldsymbol{A}\boldsymbol{v}_i = \sigma_i\boldsymbol{u}_i$, for $i = 1, \ldots, r$ and $\boldsymbol{A}\boldsymbol{v}_i = \boldsymbol{0}$, for $i = r+1, \ldots, n$.

Singular value decomposition is useful for analysis of the linear transformation $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ established in (A.35). According to a geometric interpretation, the matrix $\boldsymbol{A}$ transforms the unit sphere in $\mathbb{R}^n$ defined by $\|\boldsymbol{x}\| = 1$ into the set of vectors $\boldsymbol{y} = \boldsymbol{A}\boldsymbol{x}$ which define an *ellipsoid* of dimension r in $\mathbb{R}^m$. The singular values are the lengths of the various axes of the ellipsoid. The *condition number* of the matrix

$$\kappa = \frac{\sigma_1}{\sigma_r}$$

is related to the eccentricity of the ellipsoid and provides a measure of ill-conditioning ($\kappa \gg 1$) for numerical solution of the system established by (A.35).

It is worth noticing that the numerical procedure of singular value decomposition is commonly adopted to compute the (right or left) pseudo-inverse $\boldsymbol{A}^\dagger$, even in the case of a matrix $\boldsymbol{A}$ not having full rank. In fact, from (A.67), (A.70) it is

$$\boldsymbol{A}^\dagger = \boldsymbol{V}\boldsymbol{\Sigma}^\dagger\boldsymbol{U}^T \tag{A.71}$$

with

$$\boldsymbol{\Sigma}^{\dagger} = \begin{bmatrix} \boldsymbol{D}^{\dagger} & \boldsymbol{O} \\ \boldsymbol{O} & \boldsymbol{O} \end{bmatrix} \qquad \boldsymbol{D}^{\dagger} = \mathrm{diag}\left\{ \frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \ldots, \frac{1}{\sigma_r} \right\}. \tag{A.72}$$

Bibliography

A reference text on linear algebra is [169]. For matrix computation see [88]. The properties of pseudo-inverse matrices are discussed in [24].

B

Rigid-body Mechanics

The goal of this appendix is to recall some fundamental concepts of *rigid body mechanics* which are preliminary to the study of manipulator *kinematics*, *statics* and *dynamics*.

B.1 Kinematics

A *rigid body* is a system characterized by the constraint that the distance between any two points is always constant.

Consider a rigid body $\mathcal{B}$ moving with respect to an orthonormal reference frame O–xyz of unit vectors $\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$, called *fixed frame*. The rigidity assumption allows the introduction of an orthonormal frame O'–$x'y'z'$ attached to the body, called *moving frame*, with respect to which the position of any point of $\mathcal{B}$ is independent of time. Let $\boldsymbol{x}'(t)$, $\boldsymbol{y}'(t)$, $\boldsymbol{z}'(t)$ be the unit vectors of the moving frame expressed in the fixed frame at time t.

The orientation of the moving frame O'–$x'y'z'$ at time t with respect to the fixed frame O–xyz can be expressed by means of the *orthogonal* (3×3) matrix

$$\boldsymbol{R}(t) = \begin{bmatrix} \boldsymbol{x}'^T(t)\boldsymbol{x} & \boldsymbol{y}'^T(t)\boldsymbol{x} & \boldsymbol{z}'^T(t)\boldsymbol{x} \\ \boldsymbol{x}'^T(t)\boldsymbol{y} & \boldsymbol{y}'^T(t)\boldsymbol{y} & \boldsymbol{z}'^T(t)\boldsymbol{y} \\ \boldsymbol{x}'^T(t)\boldsymbol{z} & \boldsymbol{y}'^T(t)\boldsymbol{z} & \boldsymbol{z}'^T(t)\boldsymbol{z} \end{bmatrix}, \tag{B.1}$$

which is termed *rotation matrix* defined in the orthonormal special group $SO(3)$ of the (3×3) matrices with orthonormal columns and determinant equal to 1. The columns of the matrix in (B.1) represent the components of the unit vectors of the moving frame when expressed in the fixed frame, whereas the rows represent the components of the unit vectors of the fixed frame when expressed in the moving frame.

Let $\boldsymbol{p}'$ be the *constant* position vector of a generic point P of $\mathcal{B}$ in the moving frame O'–$x'y'z'$. The motion of P with respect to the fixed frame O–xyz is described by the equation

$$\boldsymbol{p}(t) = \boldsymbol{p}_{O'}(t) + \boldsymbol{R}(t)\boldsymbol{p}', \tag{B.2}$$

where $\boldsymbol{p}_{O'}(t)$ is the position vector of origin O' of the moving frame with respect to the fixed frame.

Notice that a position vector is a *bound vector* since its line of application and point of application are both prescribed, in addition to its direction; the point of application typically coincides with the origin of a reference frame. Therefore, to transform a bound vector from a frame to another, both translation and rotation between the two frames must be taken into account.

If the positions of the points of $\mathcal{B}$ in the moving frame are known, it follows from (B.2) that the motion of each point of $\mathcal{B}$ with respect to the fixed frame is uniquely determined once the position of the origin and the orientation of the moving frame with respect to the fixed frame are specified in time. The origin of the moving frame is determined by *three* scalar functions of time. Since the orthonormality conditions impose six constraints on the nine elements of matrix $\boldsymbol{R}(t)$, the *orientation* of the moving frame depends only on *three* independent scalar functions, three being the minimum number of parameters to represent $SO(3)$.[1]

Therefore, a rigid body motion is described by arbitrarily specifying *six* scalar functions of time, which describe the body *pose* (position + orientation). The resulting rigid motions belong to the *special Euclidean group* $SE(3) = \mathbb{R}^3 \times SO(3)$.

The expression in (B.2) continues to hold if the position vector $\boldsymbol{p}_{O'}(t)$ of the origin of the moving frame is replaced with the position vector of any other point of $\mathcal{B}$, i.e.,

$$\boldsymbol{p}(t) = \boldsymbol{p}_Q(t) + \boldsymbol{R}(t)(\boldsymbol{p}' - \boldsymbol{p}'_Q) \tag{B.3}$$

where $\boldsymbol{p}_Q(t)$ and $\boldsymbol{p}'_Q$ are the position vectors of a point Q of $\mathcal{B}$ in the fixed and moving frames, respectively.

In the following, for simplicity of notation, the dependence on the time variable t will be dropped.

Differentiating (B.3) with respect to time gives the known velocity composition rule

$$\dot{\boldsymbol{p}} = \dot{\boldsymbol{p}}_Q + \boldsymbol{\omega} \times (\boldsymbol{p} - \boldsymbol{p}_Q), \tag{B.4}$$

where $\boldsymbol{\omega}$ is the *angular velocity* of rigid body $\mathcal{B}$. Notice that $\boldsymbol{\omega}$ is a *free vector* since its point of application is not prescribed. To transform a free vector from a frame to another, only rotation between the two frames must be taken into account.

By recalling the definition of the skew-symmetric operator $\boldsymbol{S}(\cdot)$ in (A.32), the expression in (B.4) can be rewritten as

$$\begin{aligned} \dot{\boldsymbol{p}} &= \dot{\boldsymbol{p}}_Q + \boldsymbol{S}(\boldsymbol{\omega})(\boldsymbol{p} - \boldsymbol{p}_Q) \\ &= \dot{\boldsymbol{p}}_Q + \boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{R}(\boldsymbol{p}' - \boldsymbol{p}'_Q). \end{aligned}$$

[1] The minimum number of parameters represent a special orthonormal group $SO(m)$ is equal to $m(m-1)/2$.

Comparing this equation with the formal time derivative of (B.3) leads to the result

$$\dot{\boldsymbol{R}} = \boldsymbol{S}(\boldsymbol{\omega})\boldsymbol{R}. \tag{B.5}$$

In view of (B.4), the *elementary displacement* of a point P of the rigid body $\mathcal{B}$ in the time interval $(t, t+dt)$ is

$$\begin{aligned} d\boldsymbol{p} = \dot{\boldsymbol{p}}dt &= \big(\dot{\boldsymbol{p}}_Q + \boldsymbol{\omega} \times (\boldsymbol{p} - \boldsymbol{p}_Q)\big)dt \\ &= d\boldsymbol{p}_Q + \boldsymbol{\omega}dt \times (\boldsymbol{p} - \boldsymbol{p}_Q). \end{aligned} \tag{B.6}$$

Differentiating (B.4) with respect to time yields the following expression for acceleration:

$$\ddot{\boldsymbol{p}} = \ddot{\boldsymbol{p}}_Q + \dot{\boldsymbol{\omega}} \times (\boldsymbol{p} - \boldsymbol{p}_Q) + \boldsymbol{\omega} \times \big(\boldsymbol{\omega} \times (\boldsymbol{p} - \boldsymbol{p}_Q)\big). \tag{B.7}$$

B.2 Dynamics

Let ρdV be the mass of an elementary particle of a rigid body $\mathcal{B}$, where ρ denotes the density of the particle of volume dV. Also let $V_{\mathcal{B}}$ be the body volume and $m = \int_{V_{\mathcal{B}}} \rho dV$ its *total mass* assumed to be constant. If $\boldsymbol{p}$ denotes the position vector of the particle of mass ρdV in the frame O–xyz, the *centre of mass* of $\mathcal{B}$ is defined as the point C whose position vector is

$$\boldsymbol{p}_C = \frac{1}{m} \int_{V_{\mathcal{B}}} \boldsymbol{p} \rho dV. \tag{B.8}$$

In the case when $\mathcal{B}$ is the union of n distinct parts of mass $m_1, \ldots, m_n$ and centres of mass $\boldsymbol{p}_{C1} \ldots \boldsymbol{p}_{Cn}$, the centre of mass of $\mathcal{B}$ can be computed as

$$\boldsymbol{p}_C = \frac{1}{m} \sum_{i=1}^{n} m_i \boldsymbol{p}_{Ci}$$

with $m = \sum_{i=1}^{n} m_i$.

Let r be a line passing by O and $d(\boldsymbol{p})$ the distance from r of the particle of $\mathcal{B}$ of mass ρdV and position vector $\boldsymbol{p}$. The *moment of inertia* of body $\mathcal{B}$ with respect to line r is defined as the positive scalar

$$I_r = \int_{V_{\mathcal{B}}} d^2(\boldsymbol{p}) \rho dV.$$

Let $\boldsymbol{r}$ denote the unit vector of line r; then, the moment of inertia of $\mathcal{B}$ with respect to line r can be expressed as

$$I_r = \boldsymbol{r}^T \left(\int_{V_{\mathcal{B}}} \boldsymbol{S}^T(\boldsymbol{p}) \boldsymbol{S}(\boldsymbol{p}) \rho dV \right) \boldsymbol{r} = \boldsymbol{r}^T \boldsymbol{I}_O \boldsymbol{r}, \tag{B.9}$$

where $\boldsymbol{S}(\cdot)$ is the skew-symmetric operator in (A.31), and the *symmetric*, *positive definite* matrix

$$\boldsymbol{I}_O = \begin{bmatrix} \int_{V_\mathcal{B}} (p_y^2 + p_z^2)\rho dV & -\int_{V_\mathcal{B}} p_x p_y \rho dV & -\int_{V_\mathcal{B}} p_x p_z \rho dV \\ * & \int_{V_\mathcal{B}} (p_x^2 + p_z^2)\rho dV & -\int_{V_\mathcal{B}} p_y p_z \rho dV \\ * & * & \int_{V_\mathcal{B}} (p_x^2 + p_y^2)\rho dV \end{bmatrix}$$

$$= \begin{bmatrix} I_{Oxx} & -I_{Oxy} & -I_{Oxz} \\ * & I_{Oyy} & -I_{Oyz} \\ * & * & I_{Ozz} \end{bmatrix} \tag{B.10}$$

is termed *inertia tensor* of body $\mathcal{B}$ relative to pole O.[2] The (positive) elements I_{Oxx}, I_{Oyy}, I_{Ozz} are the *inertia moments* with respect to three coordinate axes of the reference frame, whereas the elements I_{Oxy}, I_{Oxz}, I_{Oyz} (of any sign) are said to be *products of inertia*.

The expression of the inertia tensor of a rigid body $\mathcal{B}$ depends both on the pole and the reference frame. If orientation of the reference frame with origin at O is changed according to a rotation matrix $\boldsymbol{R}$, the inertia tensor $\boldsymbol{I}'_O$ in the new frame is related to $\boldsymbol{I}_O$ by the relationship

$$\boldsymbol{I}_O = \boldsymbol{R}\boldsymbol{I}'_O\boldsymbol{R}^T. \tag{B.11}$$

The way an inertia tensor is transformed when the pole is changed can be inferred by the following equation, also known as *Steiner theorem* or parallel axis theorem:

$$\boldsymbol{I}_O = \boldsymbol{I}_C + m\boldsymbol{S}^T(\boldsymbol{p}_C)\boldsymbol{S}(\boldsymbol{p}_C), \tag{B.12}$$

where $\boldsymbol{I}_C$ is the inertia tensor relative to the centre of mass of $\mathcal{B}$, when expressed in a frame parallel to the frame with origin at O and with origin at the centre of mass C.

Since the inertia tensor is a symmetric positive definite matrix, there always exists a reference frame in which the inertia tensor attains a diagonal form; such a frame is said to be a *principal frame* (relative to pole O) and its coordinate axes are said to be *principal axes*. In the case when pole O coincides with the centre of mass, the frame is said to be a *central frame* and its axes are said to be *central axes*.

Notice that if the rigid body is moving with respect to the reference frame with origin at O, then the elements of the inertia tensor $\boldsymbol{I}_O$ become a function of time. With respect to a pole and a reference frame attached to the body (moving frame), instead, the elements of the inertia tensor represent six structural constants of the body which are known once the pole and reference frame have been specified.

[2] The symbol '$*$' has been used to avoid rewriting the symmetric elements.

Let $\dot{\boldsymbol{p}}$ be the velocity of a particle of $\mathcal{B}$ of elementary mass ρdV in frame O–xyz. The *linear momentum* of body $\mathcal{B}$ is defined as the vector

$$\boldsymbol{l} = \int_{V_\mathcal{B}} \dot{\boldsymbol{p}} \rho dV = m\dot{\boldsymbol{p}}_C. \tag{B.13}$$

Let Ω be any point in space and $\boldsymbol{p}_\Omega$ its position vector in frame O–xyz; then, the *angular momentum* of body $\mathcal{B}$ relative to pole Ω is defined as the vector

$$\boldsymbol{k}_\Omega = \int_{V_\mathcal{B}} \dot{\boldsymbol{p}} \times (\boldsymbol{p}_\Omega - \boldsymbol{p}) \rho dV.$$

The pole can be either fixed or moving with respect to the reference frame. The angular momentum of a rigid body has the following notable expression:

$$\boldsymbol{k}_\Omega = \boldsymbol{I}_C \boldsymbol{\omega} + m\dot{\boldsymbol{p}}_C \times (\boldsymbol{p}_\Omega - \boldsymbol{p}_C), \tag{B.14}$$

where $\boldsymbol{I}_C$ is the inertia tensor relative to the centre of mass, when expressed in a frame parallel to the reference frame with origin at the centre of mass.

The *forces* acting on a generic system of material particles can be distinguished into *internal* forces and *external* forces.

The internal forces, exerted by one part of the system on another, have null linear and angular momentum and thus they do not influence rigid body motion.

The external forces, exerted on the system by an agency outside the system, in the case of a rigid body $\mathcal{B}$ are distinguished into *active* forces and *reaction* forces.

The active forces can be either *concentrated* forces or *body* forces. The former are applied to specific points of $\mathcal{B}$, whereas the latter act on all elementary particles of the body. An example of body force is the *gravitational force* which, for any elementary particle of mass ρdV, is equal to $\boldsymbol{g}_0 \rho dV$ where $\boldsymbol{g}_0$ is the gravity acceleration vector.

The reaction forces are those exerted because of surface contact between two or more bodies. Such forces can be distributed on the contact surfaces or they can be assumed to be concentrated.

For a rigid body $\mathcal{B}$ subject to gravitational force, as well as to active and or reaction forces $\boldsymbol{f}_1 \dots \boldsymbol{f}_n$ concentrated at points $\boldsymbol{p}_1 \dots \boldsymbol{p}_n$, the *resultant* of the external forces $\boldsymbol{f}$ and the *resultant moment* $\boldsymbol{\mu}_\Omega$ with respect to a pole Ω are respectively

$$\boldsymbol{f} = \int_{V_\mathcal{B}} \boldsymbol{g}_0 \rho dV + \sum_{i=1}^{n} \boldsymbol{f}_i = m\boldsymbol{g}_0 + \sum_{i=1}^{n} \boldsymbol{f}_i \tag{B.15}$$

$$\begin{aligned} \boldsymbol{\mu}_\Omega &= \int_{V_\mathcal{B}} \boldsymbol{g}_0 \times (\boldsymbol{p}_\Omega - \boldsymbol{p}) \rho dV + \sum_{i=1}^{n} \boldsymbol{f}_i \times (\boldsymbol{p}_\Omega - \boldsymbol{p}_i) \\ &= m\boldsymbol{g}_0 \times (\boldsymbol{p}_\Omega - \boldsymbol{p}_C) + \sum_{i=1}^{n} \boldsymbol{f}_i \times (\boldsymbol{p}_\Omega - \boldsymbol{p}_i). \end{aligned} \tag{B.16}$$

In the case when $\boldsymbol{f}$ and $\boldsymbol{\mu}_\Omega$ are known and it is desired to compute the resultant moment with respect to a point Ω' other than Ω, the following relation holds:

$$\boldsymbol{\mu}_{\Omega'} = \boldsymbol{\mu}_\Omega + \boldsymbol{f} \times (\boldsymbol{p}_{\Omega'} - \boldsymbol{p}_\Omega). \tag{B.17}$$

Consider now a generic system of material particles subject to *external forces* of resultant $\boldsymbol{f}$ and resultant moment $\boldsymbol{\mu}_\Omega$. The motion of the system in a frame O–xyz is established by the following *fundamental principles of dynamics* (Newton laws of motion):

$$\boldsymbol{f} = \dot{\boldsymbol{l}} \tag{B.18}$$

$$\boldsymbol{\mu}_\Omega = \dot{\boldsymbol{k}}_\Omega \tag{B.19}$$

where Ω is a pole fixed or coincident with the centre of mass C of the system. These equations hold for any mechanical system and can be used even in the case of variable mass. For a system with constant mass, computing the time derivative of the momentum in (B.18) gives *Newton equations of motion* in the form

$$\boldsymbol{f} = m\ddot{\boldsymbol{p}}_C, \tag{B.20}$$

where the quantity on the right-hand side represents the *resultant of inertia forces*.

If, besides the assumption of constant mass, the assumption of rigid system holds too, the expression in (B.14) of the angular momentum with (B.19) yield *Euler equations of motion* in the form

$$\boldsymbol{\mu}_\Omega = \boldsymbol{I}_\Omega \dot{\boldsymbol{\omega}} + \boldsymbol{\omega} \times (\boldsymbol{I}_\Omega \boldsymbol{\omega}), \tag{B.21}$$

where the quantity on the right-hand side represents the *resultant moment of inertia forces*.

For a system constituted by a set of rigid bodies, the external forces obviously do not include the reaction forces exerted between the bodies belonging to the same system.

B.3 Work and Energy

Given a force $\boldsymbol{f}_i$ applied at a point of position $\boldsymbol{p}_i$ with respect to frame O–xyz, the *elementary work* of the force $\boldsymbol{f}_i$ on the displacement $d\boldsymbol{p}_i = \dot{\boldsymbol{p}}_i dt$ is defined as the scalar

$$dW_i = \boldsymbol{f}_i^T d\boldsymbol{p}_i.$$

For a rigid body $\mathcal{B}$ subject to a system of forces of resultant $\boldsymbol{f}$ and resultant moment $\boldsymbol{\mu}_Q$ with respect to any point Q of $\mathcal{B}$, the elementary work on the rigid displacement (B.6) is given by

$$dW = (\boldsymbol{f}^T \dot{\boldsymbol{p}}_Q + \boldsymbol{\mu}_Q^T \boldsymbol{\omega}) dt = \boldsymbol{f}^T d\boldsymbol{p}_Q + \boldsymbol{\mu}_Q^T \boldsymbol{\omega} dt. \tag{B.22}$$

The *kinetic energy* of a body $\mathcal{B}$ is defined as the scalar quantity

$$\mathcal{T} = \frac{1}{2}\int_{V_\mathcal{B}} \dot{\boldsymbol{p}}^T \dot{\boldsymbol{p}} \rho dV$$

which, for a rigid body, takes on the notable expression

$$\mathcal{T} = \frac{1}{2} m \dot{\boldsymbol{p}}_C^T \dot{\boldsymbol{p}}_C + \frac{1}{2} \boldsymbol{\omega}^T \boldsymbol{I}_C \boldsymbol{\omega} \tag{B.23}$$

where $\boldsymbol{I}_C$ is the inertia tensor relative to the centre of mass expressed in a frame parallel to the reference frame with origin at the centre of mass.

A system of position forces, i.e., the forces depending only on the positions of the points of application, is said to be *conservative* if the work done by each force is independent of the trajectory described by the point of application of the force but it depends only on the initial and final positions of the point of application. In this case, the elementary work of the system of forces is equal to minus the total differential of a scalar function termed *potential energy*, i.e.,

$$dW = -d\mathcal{U}. \tag{B.24}$$

An example of a conservative system of forces on a rigid body is the gravitational force, with which is associated the potential energy

$$\mathcal{U} = -\int_{V_\mathcal{B}} \boldsymbol{g}_0^T \boldsymbol{p} \rho dV = -m \boldsymbol{g}_0^T \boldsymbol{p}_C. \tag{B.25}$$

B.4 Constrained Systems

Consider a system $\mathcal{B}_r$ of r rigid bodies and assume that all the elements of $\mathcal{B}_r$ can reach any position in space. In order to find uniquely the position of all the points of the system, it is necessary to assign a vector $\boldsymbol{x} = [\, x_1 \quad \dots \quad x_p \,]^T$ of $6r = p$ parameters, termed *configuration*. These parameters are termed *Lagrange* or *generalized coordinates* of the *unconstrained* system $\mathcal{B}_r$, and p determines the number of *degrees of freedom* (DOFs).

Any limitation on the mobility of the system $\mathcal{B}_r$ is termed *constraint*. A constraint acting on $\mathcal{B}_r$ is said to be *holonomic* if it is expressed by a system of equations

$$\boldsymbol{h}(\boldsymbol{x}, t) = \boldsymbol{0}, \tag{B.26}$$

where $\boldsymbol{h}$ is a vector of dimensions $(s \times 1)$, with $s < m$. On the other hand, a constraint in the form $\boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}, t) = \boldsymbol{0}$ which is nonintegrable is said to be *nonholonomic*. For simplicity, only equality (or *bilateral*) constraints are considered. If the equations in (B.26) do not explicitly depend on time, the constraint is said to be *scleronomic*.

On the assumption that $\boldsymbol{h}$ has continuous and continuously differentiable components, and its Jacobian $\partial \boldsymbol{h} / \partial \boldsymbol{x}$ has full rank, the equations in (B.26)

allow the elimination of s out of m coordinates of the system $\mathcal{B}_r$. With the remaining $n = m - s$ coordinates it is possible to determine uniquely the configurations of $\mathcal{B}_r$ satisfying the constraints (B.26). Such coordinates are the *Lagrange* or *generalized coordinates* and n is the number of *degrees of freedom* of the *unconstrained* system $\mathcal{B}_r$.[3]

The motion of a system $\mathcal{B}_r$ with n DOFs and holonomic equality constraints can be described by equations of the form

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{q}(t), t), \tag{B.27}$$

where $\boldsymbol{q}(t) = [\, q_1(t) \quad \dots \quad q_n(t) \,]^T$ is a vector of Lagrange coordinates.

The *elementary displacement* of system (B.27) relative to the interval $(t, t+dt)$ is defined as

$$d\boldsymbol{x} = \frac{\partial \boldsymbol{x}(\boldsymbol{q}, t)}{\partial \boldsymbol{q}} \dot{\boldsymbol{q}} dt + \frac{\partial \boldsymbol{x}(\boldsymbol{q}, t)}{\partial t} dt. \tag{B.28}$$

The *virtual displacement* of system (B.27) at time t, relative to an increment $\delta\boldsymbol{\lambda}$, is defined as the quantity

$$\delta \boldsymbol{x} = \frac{\partial \boldsymbol{x}(\boldsymbol{q}, t)}{\partial \boldsymbol{q}} \delta \boldsymbol{q}. \tag{B.29}$$

The difference between the elementary displacement and the virtual displacement is that the former is relative to an actual motion of the system in an interval $(t, t + dt)$ which is consistent with the constraints, while the latter is relative to an imaginary motion of the system when the constraints are made invariant and equal to those at time t.

For a system with time-invariant constraints, the equations of motion (B.27) become

$$\boldsymbol{x} = \boldsymbol{x}(\boldsymbol{q}(t)), \tag{B.30}$$

and then, by setting $\delta\boldsymbol{\lambda} = d\boldsymbol{\lambda} = \dot{\boldsymbol{\lambda}} dt$, the virtual displacements (B.29) coincide with the elementary displacements (B.28).

To the concept of virtual displacement can be associated that of *virtual work* of a system of forces, by considering a virtual displacement instead of an elementary displacement.

If external forces are distinguished into *active forces* and *reaction forces*, a direct consequence of the principles of dynamics (B.18), (B.19) applied to the system of rigid bodies $\mathcal{B}_r$ is that, for each virtual displacement, the following relation holds:

$$\delta W_m + \delta W_a + \delta W_h = 0, \tag{B.31}$$

where δW_m, δW_a, δW_h are the total virtual works done by the inertia, active, reaction forces, respectively.

[3] In general, the Lagrange coordinates of a constrained system have a local validity; in certain cases, such as the joint variables of a manipulator, they can have a global validity.

In the case of *frictionless* equality constraints, reaction forces are exerted orthogonally to the contact surfaces and the virtual work is always null. Hence, (B.31) reduces to

$$\delta W_m + \delta W_a = 0. \tag{B.32}$$

For a steady system, inertia forces are identically null. Then the condition for the equilibrium of system $\mathcal{B}_r$ is that the virtual work of the active forces is identically null on any virtual displacement, which gives the fundamental equation of *statics* of a constrained system

$$\delta W_a = 0 \tag{B.33}$$

known as *principle of virtual work*. Expressing (B.33) in terms of the increment $\delta\boldsymbol{\lambda}$ of generalized coordinates leads to

$$\delta W_a = \boldsymbol{\zeta}^T \delta \boldsymbol{q} = 0 \tag{B.34}$$

where $\boldsymbol{\zeta}$ denotes the $(n \times 1)$ vector of active *generalized* forces.

In the dynamic case, it is worth distinguishing active forces into *conservative* (that can be derived from a potential) and *nonconservative*. The virtual work of conservative forces is given by

$$\delta W_c = -\frac{\partial \mathcal{U}}{\partial \boldsymbol{q}} \delta \boldsymbol{q}, \tag{B.35}$$

where $\mathcal{U}(\boldsymbol{\lambda})$ is the total potential energy of the system. The work of nonconservative forces can be expressed in the form

$$\delta W_{nc} = \boldsymbol{\xi}^T \delta \boldsymbol{q}, \tag{B.36}$$

where $\boldsymbol{\xi}$ denotes the vector of nonconservative generalized forces. It follows that the vector of active generalized forces is

$$\boldsymbol{\zeta} = \boldsymbol{\xi} - \left(\frac{\partial \mathcal{U}}{\partial \boldsymbol{q}} \right)^T. \tag{B.37}$$

Moreover, the work of inertia forces can be computed from the total kinetic energy of system $\mathcal{T}$ as

$$\delta W_m = \left(\frac{\partial \mathcal{T}}{\partial \boldsymbol{q}} - \frac{d}{dt} \frac{\partial \mathcal{T}}{\partial \dot{\boldsymbol{q}}} \right) \delta \boldsymbol{q}. \tag{B.38}$$

Substituting (B.35), (B.36), (B.38) into (B.32) and observing that (B.32) holds true for any increment $\delta\boldsymbol{\lambda}$ leads to *Lagrange equations*

$$\frac{d}{dt} \left(\frac{\partial \mathcal{L}}{\partial \dot{\boldsymbol{q}}} \right)^T - \left(\frac{\partial \mathcal{L}}{\partial \boldsymbol{q}} \right)^T = \boldsymbol{\xi}, \tag{B.39}$$

where

$$\mathcal{L} = \mathcal{T} - \mathcal{U} \tag{B.40}$$

is the *Lagrangian* function of the system. The equations in (B.39) completely describe the dynamic behaviour of an n-DOF system with holonomic equality constraints.

The sum of kinetic and potential energy of a system with time-invariant constraints is termed *Hamiltonian* function

$$\mathcal{H} = \mathcal{T} + \mathcal{U}. \tag{B.41}$$

Conservation of energy dictates that the time derivative of the Hamiltonian must balance the power generated by the nonconservative forces acting on the system, i.e.,

$$\frac{d\mathcal{H}}{dt} = \boldsymbol{\xi}^T \dot{\boldsymbol{q}}. \tag{B.42}$$

In view of (B.37), (B.41), the equation in (B.42) becomes

$$\frac{d\mathcal{T}}{dt} = \boldsymbol{\zeta}^T \dot{\boldsymbol{q}}. \tag{B.43}$$

Bibliography

The fundamental concepts of rigid-body mechanics and constrained systems can be found in classical texts such as [87, 154, 224]. An authoritative reference on rigid-body system dynamics is [187].

C

Feedback Control

As a premise to the study of manipulator decentralized control and centralized control, the fundamental principles of *feedback control* of *linear systems* are recalled, and an approach to the determination of control laws for *nonlinear systems* based on the use of *Lyapunov functions* is presented.

C.1 Control of Single-input/Single-output Linear Systems

According to classical *automatic control* theory of *linear time-invariant single-input/single-output systems*, in order to servo the output $y(t)$ of a system to a reference $r(t)$, it is worth adopting a *negative feedback control* structure. This structure indeed allows the use of approximate mathematical models to describe the input/output relationship of the system to control, since negative feedback has a potential for reducing the effects of system parameter variations and nonmeasurable disturbance inputs $d(t)$ on the output.

This structure can be represented in the *domain of complex variable s* as in the block scheme of Fig. C.1, where $G(s)$, $H(s)$ and $C(s)$ are the transfer functions of the system to control, the transducer and the controller, respectively. From this scheme it is easy to derive

$$Y(s) = W(s)R(s) + W_D(s)D(s), \tag{C.1}$$

where

$$W(s) = \frac{C(s)G(s)}{1 + C(s)G(s)H(s)} \tag{C.2}$$

is the *closed-loop input/output transfer function* and

$$W_D(s) = \frac{G(s)}{1 + C(s)G(s)H(s)} \tag{C.3}$$

is the *disturbance/output transfer function*.

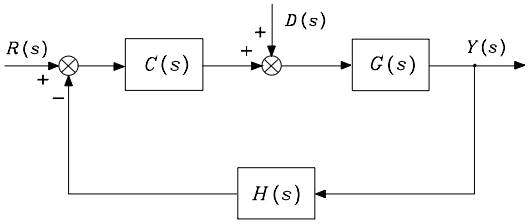

Fig. C.1. Feedback control structure

The goal of the controller design is to find a control structure $C(s)$ ensuring that the output variable $Y(s)$ tracks a reference input $R(s)$. Further, the controller should guarantee that the effects of the disturbance input $D(s)$ on the output variable are suitably reduced. The goal is then twofold, namely, *reference tracking* and *disturbance rejection*.

The basic problem for controller design consists of the determination of an action $C(s)$ which can make the system *asymptotically stable*. In the absence of positive or null real part pole/zero and zero/pole cancellation in the *open-loop* function $F(s) = C(s)G(s)H(s)$, a necessary and sufficient condition for asymptotic stability is that the *poles* of $W(s)$ and $W_D(s)$ have all *negative real parts*; such poles coincide with the zeros of the rational transfer function $1 + F(s)$. Testing for this condition can be performed by resorting to stability criteria, thus avoiding computation of the function zeros.

Routh criterion allows the determination of the sign of the real parts of the zeros of the function $1 + F(s)$ by constructing a table with the coefficients of the polynomial at the numerator of $1 + F(s)$ (*characteristic polynomial*).

Routh criterion is easy to apply for testing stability of a feedback system, but it does not provide a direct relationship between the open-loop function and stability of the closed-loop system. It is then worth resorting to *Nyquist criterion* which is based on the representation, in the complex plane, of the open-loop transfer function $F(s)$ evaluated in the *domain of real angular frequency* ($s = j\omega, -\infty < \omega < +\infty$).

Drawing of Nyquist plot and computation of the number of circles made by the vector representing the complex number $1 + F(j\omega)$ when ω continuously varies from $-\infty$ to $+\infty$ allows a test on whether or not the *closed-loop* system is asymptotically stable. It is also possible to determine the number of positive, null and negative real part roots of the characteristic polynomial, similarly to application of Routh criterion. Nonetheless, Nyquist criterion is based on the plot of the open-loop transfer function, and thus it allows the determination of a direct relationship between this function and closed-loop system stability. It is then possible from an examination of the Nyquist plot to draw suggestions on the controller structure $C(s)$ which ensures closed-loop system asymptotic stability.

If the closed-loop system is asymptotically stable, the *steady-state response* to a sinusoidal input $r(t)$, with $d(t) = 0$, is sinusoidal, too. In this case, the function $W(s)$, evaluated for $s = j\omega$, is termed *frequency response function*; the frequency response function of a feedback system can be assimilated to that of a low-pass filter with the possible occurrence of a *resonance peak* inside its *bandwidth*.

As regards the transducer, this should be chosen so that its bandwidth is much greater than the feedback system bandwidth, in order to ensure a nearly instantaneous response for any value of ω inside the bandwidth of $W(j\omega)$. Therefore, setting $H(j\omega) \approx H_0$ and assuming that the *loop gain* $|C(j\omega)G(j\omega)H_0| \gg 1$ in the same bandwidth, the expression in (C.1) for $s = j\omega$ can be approximated as

$$Y(j\omega) \approx \frac{R(j\omega)}{H_0} + \frac{D(j\omega)}{C(j\omega)H_0}.$$

Assuming $R(j\omega) = H_0 Y_d(j\omega)$ leads to

$$Y(j\omega) \approx Y_d(j\omega) + \frac{D(j\omega)}{C(j\omega)H_0}; \tag{C.4}$$

i.e., the output tracks the desired output $Y_d(j\omega)$ and the frequency components of the disturbance in the bandwidth of $W(j\omega)$ produce an effect on the output which can be reduced by increasing $|C(j\omega)H_0|$. Furthermore, if the disturbance input is a constant, the steady-state output is not influenced by the disturbance as long as $C(s)$ has at least a pole at the origin.

Therefore, a feedback control system is capable of establishing a proportional relationship between the desired output and the actual output, as evidenced by (C.4). This equation, however, requires that the frequency content of the input (desired output) be inside the frequency range for which the loop gain is much greater than unity.

The previous considerations show the advantage of including a *proportional action* and an *integral action* in the controller $C(s)$, leading to the transfer function

$$C(s) = K_I \frac{1 + sT_I}{s} \tag{C.5}$$

of a *proportional-integral controller* (PI); T_I is the time constant of the integral action and the quantity $K_I T_I$ is called proportional sensitivity.

The adoption of a PI controller is effective for low-frequency response of the system, but it may involve a reduction of *stability margins* and/or a reduction of closed-loop system bandwidth. To avoid these drawbacks, a *derivative action* can be added to the proportional and integral actions, leading to the transfer function

$$C(s) = K_I \frac{1 + sT_I + s^2 T_D T_I}{s} \tag{C.6}$$

of a *proportional-integral-derivative controller* (PID); T_D denotes the time constant of the derivative action. Notice that physical realizability of (C.6)

demands the introduction of a high-frequency pole which little influences the input/output relationship in the system bandwidth. The transfer function in (C.6) is characterized by the presence of two zeros which provide a stabilizing action and an enlargement of the closed-loop system bandwidth. Bandwidth enlargement implies shorter *response time* of the system, in terms of both variations of the reference signal and recovery action of the feedback system to output variations induced by the disturbance input.

The parameters of the adopted control structure should be chosen so as to satisfy requirements on the system behaviour at *steady state* and during the *transient*. Classical tools to determine such parameters are the *root locus* in the domain of the complex variable s or the *Nichols chart* in the domain of the real angular frequency ω. The two tools are conceptually equivalent. Their potential is different in that root locus allows a control law to be found which assigns the exact parameters of the closed-loop system time response, whereas Nichols chart allows a controller to be specified which confers good transient and steady-state behaviour to the system response.

A feedback system with strict requirements on the steady-state and transient behaviour, typically, has a response that can be assimilated to that of a *second-order system*. In fact, even for closed-loop functions of greater order, it is possible to identify a pair of complex conjugate poles whose real part absolute value is smaller than the real part absolute values of the other poles. Such a pair of poles is *dominant* in that its contribution to the transient response prevails over that of the other poles. It is then possible to approximate the input/output relationship with the transfer function

$$W(s) = \frac{k_W}{1 + \dfrac{2\zeta s}{\omega_n} + \dfrac{s^2}{\omega_n^2}} \tag{C.7}$$

which has to be realized by a proper choice of the controller. Regarding the values to assign to the parameters characterizing the transfer function in (C.7), the following remarks are in order. The constant k_W represents the input/output *steady-state gain*, which is equal to $1/H_0$ if $C(s)G(s)H_0$ has at least a pole at the origin. The *natural frequency* ω_n is the modulus of the complex conjugate poles, whose real part is given by $-\zeta\omega_n$ where ζ is the *damping ratio* of the pair of poles.

The influence of parameters ζ and ω_n on the closed-loop frequency response can be evaluated in terms of the resonance peak magnitude

$$M_r = \frac{1}{2\zeta\sqrt{1-\zeta^2}},$$

occurring at the resonant frequency

$$\omega_r = \omega_n\sqrt{1-2\zeta^2},$$

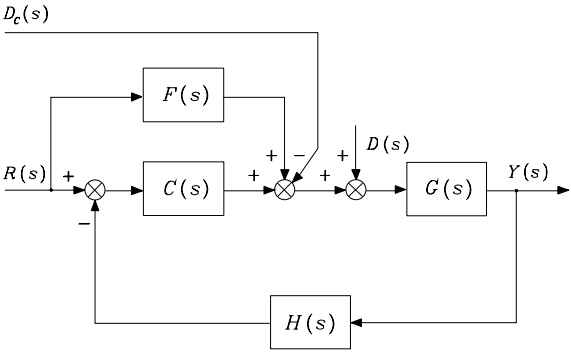

Fig. C.2. Feedback control structure with feedforward compensation

and of the 3 dB bandwidth

$$\omega_3 = \omega_n \sqrt{1 - 2\zeta^2 + \sqrt{2 - 4\zeta^2 + 4\zeta^4}}.$$

A step input is typically used to characterize the transient response in the time domain. The influence of parameters ζ and ω_n on the *step response* can be evaluated in terms of the percentage of *overshoot*

$$s\% = 100 \exp(-\pi\zeta/\sqrt{1-\zeta^2}),$$

of the *rise time*

$$t_r \approx \frac{1.8}{\omega_n}$$

and of the *settling time* within 1%

$$t_s = \frac{4.6}{\zeta\omega_n}.$$

The adoption of a *feedforward compensation* action represents a feasible solution both for tracking a time-varying reference input and for enhancing rejection of the effects of a disturbance on the output. Consider the general scheme in Fig. C.2. Let $R(s)$ denote a given input reference and $D_c(s)$ denote a computed estimate of the disturbance $D(s)$; the introduction of the feedforward action yields the input/output relationship

$$Y(s) = \left(\frac{C(s)G(s)}{1 + C(s)G(s)H(s)} + \frac{F(s)G(s)}{1 + C(s)G(s)H(s)} \right) R(s) \qquad \text{(C.8)}$$
$$+ \frac{G(s)}{1 + C(s)G(s)H(s)} \big(D(s) - D_c(s)\big).$$

By assuming that the desired output is related to the reference input by a constant factor K_d and regarding the transducer as an instantaneous system ($H(s) \approx H_0 = 1/K_d$) for the current operating conditions, the choice

$$F(s) = \frac{K_d}{G(s)} \qquad \text{(C.9)}$$

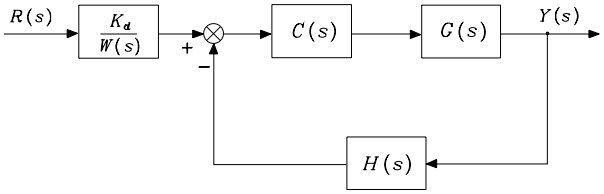

Fig. C.3. Feedback control structure with inverse model technique

yields the input/output relationship

$$Y(s) = Y_d(s) + \frac{G(s)}{1 + C(s)G(s)H_0}\big(D(s) - D_c(s)\big). \tag{C.10}$$

If $|C(j\omega)G(j\omega)H_0| \gg 1$, the effect of the disturbance on the output is further reduced by means of an accurate estimate of the disturbance.

Feedforward compensation technique may lead to a solution, termed *inverse model control*, illustrated in the scheme of Fig. C.3. It should be remarked, however, that such a solution is based on dynamics cancellation, and thus it can be employed only for a minimum-phase system, i.e., a system whose poles and zeros have all strictly negative real parts. Further, one should consider physical realizability issues as well as effects of parameter variations which prevent perfect cancellation.

C.2 Control of Nonlinear Mechanical Systems

If the system to control does not satisfy the linearity property, the control design problem becomes more complex. The fact that a *system* is qualified as *nonlinear*, whenever linearity does not hold, leads to understanding how it is not possible to resort to general techniques for control design, but it is necessary to face the problem for each class of nonlinear systems which can be defined through imposition of special properties.

On the above premise, the control design problem of nonlinear systems described by the dynamic model

$$\boldsymbol{H}(\boldsymbol{x})\ddot{\boldsymbol{x}} + \boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \boldsymbol{u} \tag{C.11}$$

is considered, where $[\,\boldsymbol{x}^T \quad \dot{\boldsymbol{x}}^T\,]^T$ denotes the $(2n \times 1)$ *state* vector of the system, $\boldsymbol{u}$ is the $(n \times 1)$ *input* vector, $\boldsymbol{H}(\boldsymbol{x})$ is an $(n \times n)$ *positive definite* (and thus invertible) matrix depending on $\boldsymbol{x}$, and $\boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}})$ is an $(n \times 1)$ vector depending on state. Several *mechanical systems* can be reduced to this class, including manipulators with rigid links and joints.

The *control* law can be found through a nonlinear compensating action obtained by choosing the following *nonlinear state feedback* law (*inverse dynamics* control):

$$\boldsymbol{u} = \widehat{\boldsymbol{H}}(\boldsymbol{x})\boldsymbol{v} + \widehat{\boldsymbol{h}}(\boldsymbol{x}, \dot{\boldsymbol{x}}) \tag{C.12}$$

where $\widehat{\boldsymbol{H}}(\boldsymbol{x})$ and $\widehat{\boldsymbol{h}}(\boldsymbol{x})$ respectively denote the *estimates* of the terms $\boldsymbol{H}(\boldsymbol{x})$ and $\boldsymbol{h}(\boldsymbol{x})$, computed on the basis of measures on the system state, and $\boldsymbol{v}$ is a new control input to be defined later. In general, it is

$$\widehat{\boldsymbol{H}}(\boldsymbol{x}) = \boldsymbol{H}(\boldsymbol{x}) + \Delta\boldsymbol{H}(\boldsymbol{x}) \tag{C.13}$$

$$\widehat{\boldsymbol{h}}(\boldsymbol{x}, \dot{\boldsymbol{x}}) = \boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}) + \Delta\boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}}) \tag{C.14}$$

because of the unavoidable modelling approximations or as a consequence of an intentional simplification in the compensating action. Substituting (C.12) into (C.11) and accounting for (C.13), (C.14) yields

$$\ddot{\boldsymbol{x}} = \boldsymbol{v} + \boldsymbol{z}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{v}) \tag{C.15}$$

where

$$\boldsymbol{z}(\boldsymbol{x}, \dot{\boldsymbol{x}}, \boldsymbol{v}) = \boldsymbol{H}^{-1}(\boldsymbol{x})\big(\Delta\boldsymbol{H}(\boldsymbol{x})\boldsymbol{v} + \Delta\boldsymbol{h}(\boldsymbol{x}, \dot{\boldsymbol{x}})\big).$$

If *tracking* of a trajectory $(\boldsymbol{x}_d(t), \dot{\boldsymbol{x}}_d(t), \ddot{\boldsymbol{x}}_d(t))$ is desired, the tracking error can be defined as

$$\boldsymbol{e} = \begin{bmatrix} \boldsymbol{x}_d - \boldsymbol{x} \\ \dot{\boldsymbol{x}}_d - \dot{\boldsymbol{x}} \end{bmatrix} \tag{C.16}$$

and it is necessary to derive the error dynamics equation to study convergence of the actual state to the desired one. To this end, the choice

$$\boldsymbol{v} = \ddot{\boldsymbol{x}}_d + \boldsymbol{w}(\boldsymbol{e}), \tag{C.17}$$

substituted into (C.15), leads to the error equation

$$\dot{\boldsymbol{e}} = \boldsymbol{F}\boldsymbol{e} - \boldsymbol{G}\boldsymbol{w}(\boldsymbol{e}) - \boldsymbol{G}\boldsymbol{z}(\boldsymbol{e}, \boldsymbol{x}_d, \dot{\boldsymbol{x}}_d, \ddot{\boldsymbol{x}}_d), \tag{C.18}$$

where the $(2n \times 2n)$ and $(2n \times n)$ matrices, respectively,

$$\boldsymbol{F} = \begin{bmatrix} \boldsymbol{O} & \boldsymbol{I} \\ \boldsymbol{O} & \boldsymbol{O} \end{bmatrix} \qquad \boldsymbol{G} = \begin{bmatrix} \boldsymbol{O} \\ \boldsymbol{I} \end{bmatrix}$$

follow from the error definition in (C.16). Control law design consists of finding the error function $\boldsymbol{w}(\boldsymbol{e})$ which makes (C.18) *globally asymptotically stable*,[1] i.e.,

$$\lim_{t \to \infty} \boldsymbol{e}(t) = \boldsymbol{0}.$$

In the case of *perfect* nonlinear compensation ($\boldsymbol{z}(\cdot) = \boldsymbol{0}$), the simplest choice of the control action is the *linear* one

$$\begin{aligned} \boldsymbol{w}(\boldsymbol{e}) &= -\boldsymbol{K}_P(\boldsymbol{x}_d - \boldsymbol{x}) - \boldsymbol{K}_D(\dot{\boldsymbol{x}}_d - \dot{\boldsymbol{x}}) \\ &= [\, -\boldsymbol{K}_P \quad -\boldsymbol{K}_D \,]\, \boldsymbol{e}, \end{aligned} \tag{C.19}$$

[1] *Global* asymptotic stability is invoked to remark that the equilibrium state is asymptotically stable for any perturbation.

where asymptotic stability of the error equation is ensured by choosing *positive definite* matrices $\boldsymbol{K}_P$ and $\boldsymbol{K}_D$. The error transient behaviour is determined by the eigenvalues of the matrix

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{O} & \boldsymbol{I} \\ -\boldsymbol{K}_P & -\boldsymbol{K}_D \end{bmatrix} \tag{C.20}$$

characterizing the error dynamics

$$\dot{\boldsymbol{e}} = \boldsymbol{A}\boldsymbol{e}. \tag{C.21}$$

If compensation is *imperfect*, then $\boldsymbol{z}(\cdot)$ cannot be neglected and the error equation in (C.18) takes on the general form

$$\dot{\boldsymbol{e}} = \boldsymbol{f}(\boldsymbol{e}). \tag{C.22}$$

It may be worth choosing the control law $\boldsymbol{w}(\boldsymbol{e})$ as the sum of a nonlinear term and a linear term of the kind in (C.19); in this case, the error equation can be written as

$$\dot{\boldsymbol{e}} = \boldsymbol{A}\boldsymbol{e} + \boldsymbol{k}(\boldsymbol{e}), \tag{C.23}$$

where $\boldsymbol{A}$ is given by (C.20) and $\boldsymbol{k}(\boldsymbol{e})$ is available to make the system globally asymptotically stable. The equations in (C.22), (C.23) express nonlinear differential equations of the error. To test for stability and obtain advise on the choice of suitable control actions, one may resort to *Lyapunov direct method* illustrated below.

C.3 Lyapunov Direct Method

The philosophy of the *Lyapunov direct method* is the same as that of most methods used in control engineering to study stability, namely, testing for stability without solving the differential equations describing the dynamic system.

This method can be presented in short on the basis of the following reasoning. If it is possible to associate an energy-based description with a (linear or nonlinear) autonomous dynamic system and, for each system state with the exception of the equilibrium state, the time rate of such energy is negative, then energy decreases along any system trajectory until it attains its minimum at the equilibrium state; this argument justifies an intuitive concept of stability.

With reference to (C.22), by setting $\boldsymbol{f}(\boldsymbol{0}) = \boldsymbol{0}$, the *equilibrium state* is $\boldsymbol{e} = \boldsymbol{0}$. A scalar function $V(\boldsymbol{e})$ of the system state, continuous together with its first derivative, is defined a *Lyapunov function* if the following properties hold:

$$V(\boldsymbol{e}) > 0 \qquad \forall \boldsymbol{e} \neq \boldsymbol{0}$$

$$\begin{aligned} V(\boldsymbol{e}) &= 0 \qquad && \boldsymbol{e} = \boldsymbol{0} \\ \dot{V}(\boldsymbol{e}) &< 0 \qquad && \forall \boldsymbol{e} \neq \boldsymbol{0} \\ V(\boldsymbol{e}) &\to \infty \qquad && \|\boldsymbol{e}\| \to \infty. \end{aligned}$$

The existence of such a function ensures *global asymptotic stability* of the equilibrium $\boldsymbol{e} = \boldsymbol{0}$. In practice, the equilibrium $\boldsymbol{e} = \boldsymbol{0}$ is globally asymptotically stable if a positive definite, radially unbounded function $\boldsymbol{V}(\boldsymbol{e})$ is found so that its time derivative along the system trajectories is negative definite.

If positive definiteness of $V(\boldsymbol{e})$ is realized by the adoption of a *quadratic form*, i.e.,

$$V(\boldsymbol{e}) = \boldsymbol{e}^T \boldsymbol{Q} \boldsymbol{e} \tag{C.24}$$

with $\boldsymbol{Q}$ a symmetric positive definite matrix, then in view of (C.22) it follows

$$\dot{V}(\boldsymbol{e}) = 2\boldsymbol{e}^T \boldsymbol{Q} \boldsymbol{f}(\boldsymbol{e}). \tag{C.25}$$

If $\boldsymbol{f}(\boldsymbol{e})$ is so as to render the function $\dot{V}(\boldsymbol{e})$ negative definite, the function $V(\boldsymbol{e})$ is a *Lyapunov function*, since the choice (C.24) allows system global asymptotic stability to be proved. If $\dot{V}(\boldsymbol{e})$ in (C.25) is not negative definite for the given $V(\boldsymbol{e})$, nothing can be inferred on the stability of the system, since the Lyapunov method gives only a *sufficient* condition. In such cases one should resort to different choices of $V(\boldsymbol{e})$ in order to find, if possible, a negative definite $\dot{V}(\boldsymbol{e})$.

In the case when the property of negative definiteness does not hold, but $\dot{V}(\boldsymbol{e})$ is only *negative semi-definite*

$$\dot{V}(\boldsymbol{e}) \leq 0,$$

global asymptotic stability of the equilibrium state is ensured if the only system trajectory for which $\dot{V}(\boldsymbol{e})$ is *identically* null ($\dot{V}(\boldsymbol{e}) \equiv 0$) is the equilibrium trajectory $\boldsymbol{e} \equiv \boldsymbol{0}$ (a consequence of *La Salle theorem*).

Finally, consider the stability problem of the nonlinear system in the form (C.23); under the assumption that $\boldsymbol{k}(\boldsymbol{0}) = \boldsymbol{0}$, it is easy to verify that $\boldsymbol{e} = \boldsymbol{0}$ is an equilibrium state for the system. The choice of a Lyapunov function candidate as in (C.24) leads to the following expression for its derivative:

$$\dot{V}(\boldsymbol{e}) = \boldsymbol{e}^T (\boldsymbol{A}^T \boldsymbol{Q} + \boldsymbol{Q} \boldsymbol{A}) \boldsymbol{e} + 2\boldsymbol{e}^T \boldsymbol{Q} \boldsymbol{k}(\boldsymbol{e}). \tag{C.26}$$

By setting

$$\boldsymbol{A}^T \boldsymbol{Q} + \boldsymbol{Q} \boldsymbol{A} = -\boldsymbol{P}, \tag{C.27}$$

the expression in (C.26) becomes

$$\dot{V}(\boldsymbol{e}) = -\boldsymbol{e}^T \boldsymbol{P} \boldsymbol{e} + 2\boldsymbol{e}^T \boldsymbol{Q} \boldsymbol{k}(\boldsymbol{e}). \tag{C.28}$$

The matrix equation in (C.27) is said to be a *Lyapunov equation*; for any choice of a symmetric positive definite matrix $\boldsymbol{P}$, the solution matrix $\boldsymbol{Q}$ exists

and is symmetric positive definite if and only if the eigenvalues of $\boldsymbol{A}$ have all negative real parts. Since matrix $\boldsymbol{A}$ in (C.20) verifies such condition, it is always possible to assign a positive definite matrix $\boldsymbol{P}$ and find a positive definite matrix solution $\boldsymbol{Q}$ to (C.27). It follows that the first term on the right-hand side of (C.28) is negative definite and the stability problem is reduced to searching a control law so that $\boldsymbol{k}(\boldsymbol{e})$ renders the total $\dot{V}(\boldsymbol{e})$ negative (semi-)definite.

It should be underlined that La Salle theorem does not hold for *time-varying* systems (also termed *non-autonomous*) in the form

$$\dot{\boldsymbol{e}} = f(\boldsymbol{e}, t).$$

In this case, a conceptually analogous result which might be useful is the following, typically referred to as *Barbalat lemma* — of which it is indeed a consequence. Given a scalar function $V(\boldsymbol{e}, t)$ so that

1. $V(\boldsymbol{e}, t)$ is lower bounded
2. $\dot{V}(\boldsymbol{e}, t) \leq 0$
3. $\dot{V}(\boldsymbol{e}, t)$ is *uniformly continuous*

then it is $\lim_{t\to\infty} \dot{V}(\boldsymbol{e}, t) = 0$. Conditions 1 and 2 imply that $V(\boldsymbol{e}, t)$ has a bounded limit for $t \to \infty$. Since it is not easy to verify the property of uniform continuity from the definition, Condition 3 is usually replaced by

3'. $\ddot{V}(\boldsymbol{e}, t)$ is bounded

which is sufficient to guarantee validity of Condition 3. Barbalat lemma can obviously be used for time-invariant (autonomous) dynamic systems as an alternative to La Salle theorem, with respect to which some conditions are relaxed; in particular, $V(\boldsymbol{e})$ needs not necessarily be positive definite.

Bibliography

Linear systems analysis can be found in classical texts such as [61]. For the control of these systems see [82, 171]. For the analysis of nonlinear systems see [109]. Control of nonlinear mechanical systems is dealt with in [215].

D

Differential Geometry

The analysis of mechanical systems subject to nonholonomic constraints, such as wheeled mobile robots, requires some basic concepts of differential geometry and nonlinear controllability theory, that are briefly recalled in this appendix.

D.1 Vector Fields and Lie Brackets

For simplicity, the case of vectors $\boldsymbol{x} \in \mathbb{R}^n$ is considered. The tangent space at $\boldsymbol{x}$ (intuitively, the space of velocities of trajectories passing through $\boldsymbol{x}$) is hence denoted by $T_{\boldsymbol{x}}(\mathbb{R}^n)$. The presented notions are however valid in the more general case in which a *differentiable manifold* (i.e., a space that is locally diffeomorphic to $\mathbb{R}^n$) is considered in place of a Euclidean space.

A *vector field* $\boldsymbol{g} : \mathbb{R}^n \mapsto T_x(\mathbb{R}^n)$ is a mapping that assigns to each point $\boldsymbol{x} \in \mathbb{R}^n$ a tangent vector $\boldsymbol{g}(\boldsymbol{x}) \in T_x(\mathbb{R}^n)$. In the following it is always assumed that vector fields are *smooth*, i.e., such that the associated mappings are of class C^∞.

If the vector field $\boldsymbol{g}(\boldsymbol{x})$ is used to define a differential equation as in

$$\dot{\boldsymbol{x}} = \boldsymbol{g}(\boldsymbol{x}), \tag{D.1}$$

the *flow* $\boldsymbol{\phi}_t^{\boldsymbol{g}}(\boldsymbol{x})$ of $\boldsymbol{g}$ is the mapping that associates to each point $\boldsymbol{x}$ the value at time t of the solution of (D.1) evolving from $\boldsymbol{x}$ at time 0, or

$$\frac{d}{dt}\boldsymbol{\phi}_t^{\boldsymbol{g}}(\boldsymbol{x}) = \boldsymbol{g}(\phi_t^{\boldsymbol{g}}(\boldsymbol{x})). \tag{D.2}$$

The family of mappings $\{\boldsymbol{\phi}_t^{\boldsymbol{g}}\}$ is a one-parameter (i.e., t) group under the composition operator

$$\boldsymbol{\phi}_{t_1}^{\boldsymbol{g}} \circ \boldsymbol{\phi}_{t_2}^{\boldsymbol{g}} = \boldsymbol{\phi}_{t_1+t_2}^{\boldsymbol{g}}.$$

For example, for time-invariant linear systems it is $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ and the flow is the linear operator $\boldsymbol{\phi}_t^{\boldsymbol{g}} = e^{\boldsymbol{A}t}$.

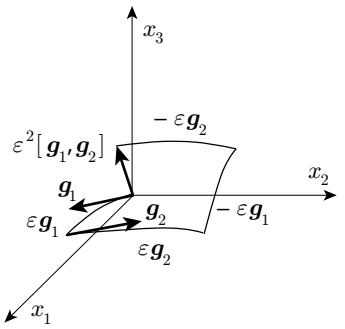

Fig. D.1. The net displacement of system (D.4) under the input sequence (D.5) is directed as the Lie bracket of the two vector fields $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$

Given two vector fields $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$, the composition of their flows is non-commutative in general:

$$\phi_t^{\boldsymbol{g}_1} \circ \phi_s^{\boldsymbol{g}_2} \neq \phi_s^{\boldsymbol{g}_2} \circ \phi_t^{\boldsymbol{g}_1}.$$

The vector field $[\boldsymbol{g}_1, \boldsymbol{g}_2]$ defined as

$$[\boldsymbol{g}_1, \boldsymbol{g}_2](\boldsymbol{x}) = \frac{\partial \boldsymbol{g}_2}{\partial x} \boldsymbol{g}_1(\boldsymbol{x}) - \frac{\partial \boldsymbol{g}_1}{\partial x} \boldsymbol{g}_2(\boldsymbol{x}) \tag{D.3}$$

is called *Lie bracket* of $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$. The two vector field $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ *commute* if $[\boldsymbol{g}_1, \boldsymbol{g}_2] = 0$.

The Lie bracket operation has an interesting interpretation. Consider the driftless dynamic system

$$\dot{\boldsymbol{x}} = \boldsymbol{g}_1(\boldsymbol{x}) u_1 + \boldsymbol{g}_2(\boldsymbol{x}) u_2 \tag{D.4}$$

associated with the vector fields $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$. If the inputs u_1 and u_2 are never active simultaneously, the solution of the differential equation (D.4) can be obtained by composing the flows of $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$. In particular, consider the following input sequence:

$$u(t) = \begin{cases} u_1(t) = +1, u_2(t) = 0 & t \in [0, \varepsilon) \\ u_1(t) = 0, u_2(t) = +1 & t \in [\varepsilon, 2\varepsilon) \\ u_1(t) = -1, u_2(t) = 0 & t \in [2\varepsilon, 3\varepsilon) \\ u_1(t) = 0, u_2(t) = -1 & t \in [3\varepsilon, 4\varepsilon), \end{cases} \tag{D.5}$$

where ε is an infinitesimal time interval. The solution of (D.4) at time $t = 4\varepsilon$ can be obtained by following first the flow of $\boldsymbol{g}_1$, then of $\boldsymbol{g}_2$, then of $-\boldsymbol{g}_1$, and finally of $-\boldsymbol{g}_2$ (see Fig. D.1). By computing $\boldsymbol{x}(\varepsilon)$ through a series expansion at $\boldsymbol{x}_0 = \boldsymbol{x}(0)$ along $\boldsymbol{g}_1$, then $\boldsymbol{x}(2\varepsilon)$ as a series expansion at $\boldsymbol{x}(\varepsilon)$ along $\boldsymbol{g}_2$, and so on, one obtains

$$\begin{aligned} \boldsymbol{x}(4\varepsilon) &= \phi_\varepsilon^{-\boldsymbol{g}_2} \circ \phi_\varepsilon^{-\boldsymbol{g}_1} \circ \phi_\varepsilon^{\boldsymbol{g}_2} \circ \phi_\varepsilon^{\boldsymbol{g}_1}(\boldsymbol{x}_0) \\ &= \boldsymbol{x}_0 + \varepsilon^2 \left(\frac{\partial \boldsymbol{g}_2}{\partial x} \boldsymbol{g}_1(\boldsymbol{x}_0) - \frac{\partial \boldsymbol{g}_1}{\partial x} \boldsymbol{g}_2(\boldsymbol{x}_0) \right) + O(\varepsilon^3). \end{aligned}$$

If $\boldsymbol{g}_1$ and $\boldsymbol{g}_2$ commute, the net displacement resulting from the input sequence (D.5) is zero.

The above expression shows that, at each point $\boldsymbol{x}$, infinitesimal motion of the driftless system (D.4) is possible not only in the directions belonging to the linear span of $\boldsymbol{g}_1(\boldsymbol{x})$ and $\boldsymbol{g}_2(\boldsymbol{x})$, but also in the direction of their Lie bracket $[\boldsymbol{g}_1, \boldsymbol{g}_2](\boldsymbol{x})$. It can be proven that more complicated input sequences can be used to generate motion in the direction of higher-order Lie brackets, such as $[\boldsymbol{g}_1, [\boldsymbol{g}_1, \boldsymbol{g}_2]]$.

Similar constructive procedures can be given for systems with a *drift*[1] vector field, such as the following:

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \boldsymbol{g}_1(\boldsymbol{x})u_1 + \boldsymbol{g}_2(\boldsymbol{x})u_2. \tag{D.6}$$

Using appropriate input sequences, it is possible to generate motion in the direction of Lie brackets involving the vector field $\boldsymbol{f}$ as well as $\boldsymbol{g}_j$, $j = 1, 2$.

Example D.1

For a single-input linear system

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\,\boldsymbol{x} + \boldsymbol{b}\,u,$$

the drift and input vector fields are $\boldsymbol{f}(\boldsymbol{x}) = \boldsymbol{A}\boldsymbol{x}$ and $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{b}$, respectively. The following Lie brackets:

$$\begin{aligned} -[\boldsymbol{f}, \boldsymbol{g}] &= \boldsymbol{A}\boldsymbol{b} \\ [\boldsymbol{f}, [\boldsymbol{f}, \boldsymbol{g}]] &= \boldsymbol{A}^2\boldsymbol{b} \\ -[\boldsymbol{f}, [\boldsymbol{f}, [\boldsymbol{f}, \boldsymbol{g}]]] &= \boldsymbol{A}^3\boldsymbol{b} \\ &\vdots \end{aligned}$$

represent well-known directions in which it is possible to move the system.

The *Lie derivative* of the scalar function $\alpha : \mathbb{R}^n \mapsto \mathbb{R}$ along vector field $\boldsymbol{g}$ is defined as

$$L_{\boldsymbol{g}}\,\alpha(\boldsymbol{x}) = \frac{\partial \alpha}{\partial \boldsymbol{x}}\,\boldsymbol{g}(\boldsymbol{x}). \tag{D.7}$$

The following properties of Lie brackets are useful in computation:

$$\begin{aligned} &[\boldsymbol{f}, \boldsymbol{g}] = -[\boldsymbol{g}, \boldsymbol{f}] && \text{(skew-symmetry)} \\ &[\boldsymbol{f}, [\boldsymbol{g}, \boldsymbol{h}]] + [\boldsymbol{h}, [\boldsymbol{f}, \boldsymbol{g}]] + [\boldsymbol{g}, [\boldsymbol{h}, \boldsymbol{f}]] = 0 && \text{(Jacobi identity)} \\ &[\alpha\boldsymbol{f}, \beta\boldsymbol{g}] = \alpha\beta[\boldsymbol{f}, \boldsymbol{g}] + \alpha(L_{\boldsymbol{f}}\beta)\boldsymbol{g} - \beta(L_{\boldsymbol{g}}\alpha)\boldsymbol{f} && \text{(chain rule)} \end{aligned}$$

[1] This term emphasizes how the presence of $\boldsymbol{f}$ will in general force the system to move ($\dot{\boldsymbol{x}} \neq \boldsymbol{0}$) even in the absence of inputs.

with $\alpha, \beta: \mathbb{R}^n \mapsto \mathbb{R}$. The vector space $\mathcal{V}(\mathbb{R}^n)$ of smooth vector fields on $\mathbb{R}^n$, equipped with the Lie bracket operation, is a *Lie algebra*.

The *distribution* Δ associated with the m vector fields $\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m\}$ is the mapping that assigns to each point $\boldsymbol{x} \in \mathbb{R}^n$ the subspace of $T_x(\mathbb{R}^n)$ defined as

$$\Delta(\boldsymbol{x}) = \text{span}\{\boldsymbol{g}_1(\boldsymbol{x}), \ldots, \boldsymbol{g}_m(\boldsymbol{x})\}. \tag{D.8}$$

Often, a shorthand notation is used:

$$\Delta = \text{span}\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m\}.$$

The distribution Δ is *nonsingular* if $\dim \Delta(\boldsymbol{x}) = r$, with r constant for all $\boldsymbol{x}$. In this case, r is called the *dimension* of the distribution. Moreover, Δ is called *involutive* if it is closed under the Lie bracket operation:

$$[\boldsymbol{g}_i, \boldsymbol{g}_j] \in \Delta \qquad \forall\, \boldsymbol{g}_i, \boldsymbol{g}_j \in \Delta.$$

The *involutive closure* $\bar{\Delta}$ of a distribution Δ is its closure under the Lie bracket operation. Hence, Δ is involutive if and only if $\bar{\Delta} = \Delta$. Note that the distribution $\Delta = \text{span}\{\boldsymbol{g}\}$ associated with a single vector field is always involutive, because $[\boldsymbol{g}, \boldsymbol{g}](\boldsymbol{x}) = \boldsymbol{0}$.

Example D.2

The distribution

$$\Delta = \text{span}\{\boldsymbol{g}_1, \boldsymbol{g}_2\} = \text{span}\left\{ \begin{bmatrix} \cos x_3 \\ \sin x_3 \\ 0 \end{bmatrix}, \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix} \right\}$$

is nonsingular and has dimension 2. It is not involutive, because the Lie bracket

$$[\boldsymbol{g}_1, \boldsymbol{g}_2](\boldsymbol{x}) = \begin{bmatrix} \sin x_3 \\ -\cos x_3 \\ 0 \end{bmatrix}$$

is always linearly independent of $\boldsymbol{g}_1(\boldsymbol{x})$ and $\boldsymbol{g}_2(\boldsymbol{x})$. Its involutive closure is therefore

$$\bar{\Delta} = \text{span}\{\boldsymbol{g}_1, \boldsymbol{g}_2, [\boldsymbol{g}_1, \boldsymbol{g}_2]\}.$$

D.2 Nonlinear Controllability

Consider a nonlinear dynamic system of the form

$$\dot{\boldsymbol{x}} = \boldsymbol{f}(\boldsymbol{x}) + \sum_{j=1}^{m} \boldsymbol{g}_j(\boldsymbol{x}) u_j, \tag{D.9}$$

that is called *affine* in the inputs u_j. The state $\boldsymbol{x}$ takes values in $\mathbb{R}^n$, while each component u_j of the control input $\boldsymbol{u} \in \mathbb{R}^m$ takes values in the class $\mathcal{U}$ of piecewise-constant functions.

Denote by $\boldsymbol{x}(t, 0, \boldsymbol{x}_0, \boldsymbol{u})$ the solution of (D.9) at time $t \geq 0$, corresponding to an input $\boldsymbol{u}\colon [0, t] \to \mathcal{U}$ and an initial condition $\boldsymbol{x}(0) = \boldsymbol{x}_0$. Such a solution exists and is unique provided that the drift vector field $\boldsymbol{f}$ and the input vector fields $\boldsymbol{g}_j$ are of class C^∞. System (D.9) is said to be *controllable* if, for any choice of $\boldsymbol{x}_1, \boldsymbol{x}_2$ in $\mathbb{R}^n$, there exists a time instant T and an input $\boldsymbol{u}\colon [0, T] \to \mathcal{U}$ such that $\boldsymbol{x}(T, 0, \boldsymbol{x}_1, \boldsymbol{u}) = \boldsymbol{x}_2$.

The *accessibility algebra* $\mathcal{A}$ of system (D.9) is the smallest subalgebra of $\mathcal{V}(\mathbb{R}^n)$ that contains $\boldsymbol{f}, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_m$. By definition, all the Lie brackets that can be generated using these vector fields belong to $\mathcal{A}$. The *accessibility distribution* $\Delta_{\mathcal{A}}$ of system (D.9) is defined as

$$\Delta_{\mathcal{A}} = \operatorname{span}\{\boldsymbol{v} | \boldsymbol{v} \in \mathcal{A}\}. \tag{D.10}$$

In other words, $\Delta_{\mathcal{A}}$ is the involutive closure of $\Delta = \operatorname{span}\{\boldsymbol{f}, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_m\}$.

The computation of $\Delta_{\mathcal{A}}$ may be organized as an iterative procedure

$$\Delta_{\mathcal{A}} = \operatorname{span}\left\{\boldsymbol{v} | \boldsymbol{v} \in \Delta_i, \forall i \geq 1\right\},$$

with

$$\begin{aligned} \Delta_1 &= \Delta = \operatorname{span}\{\boldsymbol{f}, \boldsymbol{g}_1, \ldots, \boldsymbol{g}_m\} \\ \Delta_i &= \Delta_{i-1} + \operatorname{span}\{[\boldsymbol{g}, \boldsymbol{v}] | \, \boldsymbol{g} \in \Delta_1, \boldsymbol{v} \in \Delta_{i-1}\}, \quad i \geq 2. \end{aligned}$$

This procedure stops after κ steps, where κ is the smallest integer such that $\Delta_{\kappa+1} = \Delta_\kappa = \Delta_{\mathcal{A}}$. This number is called the *nonholonomy degree* of the system and is related to the 'level' of Lie brackets that must be included in $\Delta_{\mathcal{A}}$. Since $\dim \Delta_{\mathcal{A}} \leq n$, it is $\kappa \leq n - m$ necessarily.

If system (D.9) is driftless

$$\dot{\boldsymbol{x}} = \sum_{i=1}^{m} \boldsymbol{g}_i(\boldsymbol{x}) u_i, \tag{D.11}$$

the accessibility distribution $\Delta_{\mathcal{A}}$ associated with vector fields $\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m$ characterizes its controllability. In particular, system (D.11) is controllable if and only if the following *accessibility rank condition* holds:

$$\dim \Delta_{\mathcal{A}}(\boldsymbol{x}) = n. \tag{D.12}$$

Note that for driftless systems the iterative procedure for building $\Delta_{\mathcal{A}}$ starts with $\Delta_1 = \Delta = \text{span}\{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_m\}$, and therefore $\kappa \leq n - m + 1$.

For systems in the general form (D.9), condition (D.12) is only necessary for controllability. There are, however, two notable exceptions:

- If system (D.11) is controllable, the system with drift obtained by performing a *dynamic extension* of (D.11)

$$\dot{\boldsymbol{x}} = \sum_{i=1}^{m} \boldsymbol{g}_i(\boldsymbol{x}) v_i \tag{D.13}$$

$$\dot{v}_i = u_i, \qquad i = 1, \ldots, m, \tag{D.14}$$

  i.e., by adding an integrator on each input channel, is also controllable.
- For a linear system

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \sum_{j=1}^{m} \boldsymbol{b}_j u_j = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u}$$

  (D.12) becomes

$$\varrho\left([\, \boldsymbol{B} \quad \boldsymbol{A}\boldsymbol{B} \quad \boldsymbol{A}^2\boldsymbol{B} \quad \ldots \quad \boldsymbol{A}^{n-1}\boldsymbol{B} \,]\right) = n, \tag{D.15}$$

  i.e., the well-known necessary and sufficient condition for controllability due to Kalman.

Bibliography

The concepts briefly recalled in this appendix can be studied in detail in various tests of differential geometry [94, 20] and nonlinear control theory [104, 168, 195].

E

Graph Search Algorithms

This appendix summarizes some basic concepts on algorithm complexity and graph search techniques that are useful in the study of motion planning.

E.1 Complexity

A major criterion for assessing the efficiency of an algorithm A is its *running time*, i.e., the time needed for executing the algorithm in a computational model capturing the most relevant characteristics of an actual elaboration system. In practice, one is interested in estimating the running time as a function of a single parameter n characterizing the *size* of the input within a specific class of instances of the problem. In motion planning, this parameter may be the dimension of the configuration space, or the number of vertices of the free configuration space (if it is a polygonal subset).

In *worst-case analysis*, $t(n)$ denotes the maximum running time of A in correspondence of input instances of size n. Other kinds of analyses (e.g., average-case) are possible but they are less critical or general, requiring a statistical knowledge of the input distribution that may not be available.

The exact functional expression of $t(n)$ depends on the implementation of the algorithm, and is of little practical interest because the running time in the adopted computational model is only an approximation of the actual one. More significant is the *asymptotic behaviour* of $t(n)$, i.e., the rate of growth of $t(n)$ with n. Denote by $O(f(n))$ the set of real functions $g(n)$ such that

$$c_1 f(n) \leq g(n) \leq c_2 f(n) \qquad \forall\, n \geq n_0,$$

with c_1, c_2 and n_0 positive constants. If the worst-case running time of A is $O(f(n))$, i.e., if $t(n) \in O(f(n))$, the *time complexity* of A is said to be $O(f(n))$.

A very important class is represented by algorithms whose worst-case running time is asymptotically polynomial in the size of the input. In particular, if $t(n) \in O(n^p)$, for some $p \geq 0$, the algorithm is said to have *polynomial* time

complexity. If the asymptotic behaviour of the worst-case running time is not polynomial, the time complexity of the algorithm is *exponential*. Note that here 'exponential' actually means 'not bounded by any polynomial function'.

The asymptotic behaviour of an algorithm with exponential time complexity is such that in the worst case it can only be applied to problems of 'small' size. However, there exist algorithms of exponential complexity that are very efficient on average, i.e., for the most frequent classes of input. A well known example is the simplex algorithm for solving linear programming problems. Similarly, there are algorithms with polynomial time complexity which are inefficient in practice because c_1, c_2 or p are 'large'.

The above concepts can be extended to inputs whose size is characterized by more than one parameter, or to performance criteria different from running time. For example, the memory space required by an algorithm is another important measure. The *space complexity* of an algorithm is said to be $O(f(n))$ if the memory space required for its execution is a function in $O(f(n))$.

E.2 Breadth-first and Depth-first Search

Let $G = (N, A)$ be a graph consisting of a set N of nodes and a set A of arcs, with cardinality n and a respectively. It is assumed that G is represented by an *adjacency list*: to each node N_i is associated a list of nodes that are connected to N_i by an arc. Consider the problem of searching G to find a path from a start node N_s to a goal node N_g. The simplest graph search strategies are *breadth-first search* (BFS) and *depth-first search* (DFS). These are briefly described in the following with reference to an iterative implementation.

Breadth-first search makes use of a *queue* — i.e., a FIFO (First In First Out) data structure — of nodes called OPEN. Initially, OPEN contains only the start node N_s, which is marked *visited*. All the other nodes in G are marked *unvisited*. At each iteration the first node in OPEN is extracted, and all its *unvisited* adjacent nodes are marked *visited* and inserted in OPEN. The search terminates when either N_g is inserted in OPEN or OPEN is empty (failure). During the search, the algorithm maintains the *BFS tree*, which contains only those arcs that have led to discovering *unvisited* nodes. This tree contains one and only one path connecting the start node to each visited node, and hence also a solution path from N_s to N_g, if it exists.

In depth-first search, OPEN is a *stack*, i.e., a LIFO (Last In First Out) data structure. Like in the breadth-first case, it contains initially only the start node N_s marked *visited*. When a node N_j is inserted in OPEN, the node N_i which has determined its insertion is memorized. At each iteration, the first node in OPEN is extracted. If it is *unvisited*, it is marked *visited* and the arc connecting N_i to N_j is inserted in the *DFS tree*. All *unvisited* nodes that are adjacent to N_j are inserted in OPEN. The search terminates when either N_g is inserted in OPEN or OPEN is empty (failure). Like in the BFS, the DFS tree contains the solution path from N_s to N_g, if it exists.

Both breadth-first and depth-first search have time complexity $O(a)$. Note that BFS and DFS are actually *traversal* strategies, because they do not use any information about the goal node; the graph is simply traversed until N_g is marked *visited*. Both the algorithms are *complete*, i.e., they find a solution path if it exists and report failure otherwise.

E.3 $A^\star$ Algorithm

In many applications, the arcs of G are labelled with positive numbers called *weights*. As a consequence, one may define the *cost* of a path on G as the sum of the weights of its arcs. Consider the problem of connecting N_s to N_g on G through a path of minimum cost, simply called *minimum path*. In motion planning problems, for example, the nodes generally represent points in configuration space, and it is then natural to define the weight of an arc as the length of the path that it represents. The minimum path is obviously interesting because it is the shortest among those joining N_s to N_g on G.

A widely used strategy for determining the minimum path on a graph is the $A^\star$ algorithm. $A^\star$ visits the nodes of G iteratively starting from N_s, storing only the current minimum paths from N_s to the visited nodes in a tree T. The algorithm employs a cost function $f(N_i)$ for each node N_i visited during the search. This function, which is an *estimate* of the cost of the minimum path that connects N_s to N_g passing through N_i, is computed as

$$f(N_i) = g(N_i) + h(N_i),$$

where $g(N_i)$ is the cost of the path from N_s to N_i as stored in the current tree T, and $h(N_i)$ is a *heuristic* estimate of the cost $h^\star(N_i)$ of the minimum path between N_i and N_g. While the value of $g(N_i)$ is uniquely determined by the search, any choice of $h(\cdot)$ such that

$$\forall N_i \in N : 0 \leq h(N_i) \leq h^\star(N_i) \tag{E.1}$$

is admissible. Condition (E.1) means that $h(\cdot)$ must not 'overestimate' the cost of the minimum path from N_i to N_g.

In the following, a pseudocode description of $A^\star$ is given. For its understanding, some preliminary remarks are needed:

- all the nodes are initially *unvisited*, except N_s which is *visited*;
- at the beginning, T contains only N_s;
- OPEN is a list of nodes that initially contains only N_s;
- N_{best} is the node in OPEN with the minimum value of f (in particular, it is the first node if OPEN is sorted by increasing values of f);
- ADJ(N_i) is the adjacency list of N_i;
- $c(N_i, N_j)$ is the weight of the arc connecting N_i to N_j.

```
A* algorithm
1    repeat
2      find and extract N_best from OPEN
3      if N_best = N_g then exit
4      for each node N_i in ADJ(N_best) do
5        if N_i is unvisited then
6          add N_i to T with a pointer toward N_best
7          insert N_i in OPEN; mark N_i visited
8        else if g(N_best) + c(N_best, N_i) < g(N_i) then
9          redirect the pointer of N_i in T toward N_best
10         if N_i is not in OPEN then
10           insert N_i in OPEN
10         else update f(N_i)
10         end if
11       end if
12   until OPEN is empty
```

Under condition (E.1), the $A^\star$ algorithm is complete. In particular, if the algorithm terminates with an empty OPEN, there exists no path in G from N_s to N_g (failure); otherwise, the tree T contains the minimum path from N_s to N_g, which can be reconstructed by backtracking from N_g to N_s.

The $A^\star$ algorithm with the particular (admissible) choice $h(N_i) = 0$, for each node N_i, is equivalent to the *Dijkstra algorithm*. If the nodes in G represent points in a Euclidean space, an admissible heuristic is the Euclidean distance between N and N_g. In fact, the length of the minimum path between N_i and N_g is bounded below by the Euclidean distance.

The extraction of a node from OPEN and the visit of its adjacent nodes is called node *expansion*. Given two admissible heuristic functions h_1 and h_2 such that $h_2(N_i) \geq h_1(N_i)$, for each node N_i in G, it is possible to prove that each node in G expanded by $A^\star$ using h_2 is also expanded using h_1. This means that $A^\star$ equipped with the heuristic h_2 is *at least* as efficient as $A^\star$ equipped with the heuristic h_1; h_2 is said to be *more informed* than h_1.

The $A^\star$ algorithm can be implemented with time complexity $O(a \log n)$.

Bibliography

The notions briefly recalled in this appendix are explained in detail in various texts on algorithm theory and artificial intelligence, such as [51, 189, 202].

References

1. C. Abdallah, D. Dawson, P. Dorato, M. Jamshidi, "Survey of robust control for rigid robots," *IEEE Control Systems Magazine*, vol. 11, no. 2, pp. 24–30, 1991.
2. M. Aicardi, G. Casalino, A. Bicchi, A. Balestrino, "Closed loop steering of unicycle-like vehicles via Lyapunov techniques," *IEEE Robotics and Automation Magazine*, vol. 2, no. 1, pp. 27–35, 1995.
3. J.S. Albus, H.G. McCain, R. Lumia, *NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)*, NBS tech. note 1235, Gaithersburg, MD, 1987.
4. C.H. An, C.G. Atkeson, J.M. Hollerbach, *Model-Based Control of a Robot Manipulator*, MIT Press, Cambridge, MA, 1988.
5. R.J. Anderson, M.W. Spong, "Hybrid impedance control of robotic manipulators," *IEEE Journal of Robotics and Automation*, vol. 4, pp. 549–556, 1988.
6. J. Angeles, *Spatial Kinematic Chains: Analysis, Synthesis, Optimization*, Springer-Verlag, Berlin, 1982.
7. S. Arimoto, F. Miyazaki, "Stability and robustness of PID feedback control for robot manipulators of sensory capability," in *Robotics Research: The First International Symposium*, M. Brady, R. Paul (Eds.), MIT Press, Cambridge, MA, pp. 783–799, 1984.
8. R.C. Arkin, *Behavior-Based Robotics*, MIT Press, Cambridge, MA, 1998.
9. B. Armstrong-Hélouvry, *Control of Machines with Friction*, Kluwer, Boston, MA, 1991.
10. H. Asada, J.-J.E. Slotine, *Robot Analysis and Control*, Wiley, New York, 1986.
11. H. Asada, K. Youcef-Toumi, "Analysis and design of a direct-drive arm with a five-bar-link parallel drive mechanism," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 106, pp. 225–230, 1984.
12. H. Asada, K. Youcef-Toumi, *Direct-Drive Robots*, MIT Press, Cambridge, MA, 1987.
13. C.G. Atkeson, C.H. An, J.M. Hollerbach, "Estimation of inertial parameters of manipulator loads and links," *International Journal of Robotics Research*, vol. 5, no. 3, pp. 101–119, 1986.
14. J. Baillieul, "Kinematic programming alternatives for redundant manipulators," *Proc. 1985 IEEE International Conference on Robotics and Automation*, St. Louis, MO, pp. 722–728, 1985.

15. A. Balestrino, G. De Maria, L. Sciavicco, "An adaptive model following control for robotic manipulators," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 105, pp. 143–151, 1983.
16. A. Balestrino, G. De Maria, L. Sciavicco, B. Siciliano, "An algorithmic approach to coordinate transformation for robotic manipulators," *Advanced Robotics*, vol. 2, pp. 327–344, 1988.
17. J. Barraquand, J.-C. Latombe, "Robot motion planning: A distributed representation approach," *International Journal of Robotics Research*, vol. 10, pp. 628–649, 1991.
18. G. Bastin, G. Campion, B. D'Andréa-Novel, "Structural properties and classification of kinematic and dynamic models of wheeled mobile robots," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 47–62, 1996.
19. A.K. Bejczy, *Robot Arm Dynamics and Control*, memo. TM 33-669, Jet Propulsion Laboratory, California Institute of Technology, 1974.
20. W.M. Boothby, *An Introduction to Differentiable Manifolds and Riemannian Geometry*, Academic Press, Orlando, FL, 1986.
21. J. Borenstein, H.R. Everett, L. Feng, *Navigating Mobile Robots: Systems and Techniques*, A K Peters, Wellesley, MA, 1996.
22. B.K.K. Bose, *Modern Power Electronics and AC Drives*, Prentice-Hall, Englewood Cliffs, NJ, 2001.
23. O. Bottema, B. Roth, *Theoretical Kinematics*, North Holland, Amsterdam, 1979.
24. T.L. Boullion, P.L. Odell, *Generalized Inverse Matrices*, Wiley, New York, 1971.
25. M. Brady, "Artificial intelligence and robotics," *Artificial Intelligence*, vol. 26, pp. 79–121, 1985.
26. M. Brady, J.M. Hollerbach, T.L. Johnson, T. Lozano-Pérez, M.T. Mason, (Eds.), *Robot Motion: Planning and Control*, MIT Press, Cambridge, MA, 1982.
27. H. Bruyninckx, J. De Schutter, "Specification of force-controlled actions in the "task frame formalism" — A synthesis," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 581–589, 1996.
28. H. Bruyninckx, S. Dumey, S. Dutré, J. De Schutter, "Kinematic models for model-based compliant motion in the presence of uncertainty," *International Journal of Robotics Research*, vol. 14, pp. 465–482, 1995.
29. F. Caccavale, P. Chiacchio, "Identification of dynamic parameters and feedforward control for a conventional industrial manipulator," *Control Engineering Practice*, vol. 2, pp. 1039–1050, 1994.
30. F. Caccavale, C. Natale, B. Siciliano, L. Villani, "Resolved-acceleration control of robot manipulators: A critical review with experiments," *Robotica*, vol. 16, pp. 565–573, 1998.
31. F. Caccavale, C. Natale, B. Siciliano, L. Villani, "Six-DOF impedance control based on angle/axis representations," *IEEE Transactions on Robotics and Automation*, vol. 15, pp. 289–300, 1999.
32. F. Caccavale, C. Natale, B. Siciliano, L. Villani, "Robot impedance control with nondiagonal stiffness," *IEEE Transactions on Automatic Control*, vol. 44, pp. 1943–1946, 1999.
33. J.F. Canny, *The Complexity of Robot Motion Planning*, MIT Press, Cambridge, MA, 1988.

34. C. Canudas de Wit, H. Khennouf, C. Samson, O.J. Sørdalen, "Nonlinear control design for mobile robots," in *Recent Trends in Mobile Robots*, Y.F. Zheng, (Ed.), pp. 121–156, World Scientific Publisher, Singapore, 1993.
35. F. Chaumette, "Image moments: A general and useful set of features for visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 21, pp. 1116-1127, 2005.
36. F. Chaumette, S. Hutchinson, "Visual servo control. Part I: Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, 2006.
37. P. Chiacchio, S. Chiaverini, L. Sciavicco, B. Siciliano, "Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy," *International Journal of Robotics Research*, vol. 10, pp. 410–425, 1991.
38. P. Chiacchio, S. Chiaverini, L. Sciavicco, B. Siciliano, "Influence of gravity on the manipulability ellipsoid for robot arms," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 114, pp. 723–727, 1992.
39. P. Chiacchio, F. Pierrot, L. Sciavicco, B. Siciliano, "Robust design of independent joint controllers with experimentation on a high-speed parallel robot," *IEEE Transactions on Industrial Electronics*, vol. 40, pp. 393–403, 1993.
40. S. Chiaverini, L. Sciavicco, "The parallel approach to force/position control of robotic manipulators," *IEEE Transactions on Robotics and Automation*, vol. 4, pp. 361–373, 1993.
41. S. Chiaverini, B. Siciliano, "The unit quaternion: A useful tool for inverse kinematics of robot manipulators," *Systems Analysis Modelling Simulation*, vol. 35, pp. 45–60, 1999.
42. S. Chiaverini, B. Siciliano, O. Egeland, "Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator," *IEEE Transactions on Control Systems Technology*, vol. 2, pp. 123–134, 1994.
43. S. Chiaverini, B. Siciliano, L. Villani, "Force/position regulation of compliant robot manipulators," *IEEE Transactions on Automatic Control*, vol. 39, pp. 647–652, 1994.
44. S.L. Chiu, "Task compatibility of manipulator postures," *International Journal of Robotics Research*, vol. 7, no. 5, pp. 13–21, 1988.
45. H. Choset, K.M. Lynch, S. Hutchinson, G. Kantor, W. Burgard, L.E. Kavraki, S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*, MIT Press, Cambridge, MA, 2005.
46. J.C.K. Chou, "Quaternion kinematic and dynamic differential equations. *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 53–64, 1992.
47. A.I. Comport, E. Marchand, M. Pressigout, F. Chaumette, "Real-time markerless tracking for augmented reality: The virtual visual servoing framework," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, pp. 615–628, 2006.
48. P.I. Corke, *Visual Control of Robots: High-Performance Visual Servoing*, Research Studies Press, Taunton, UK, 1996.
49. P. Corke, S. Hutchinson, "A new partitioned approach to image-based visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 17, pp. 507-515, 2001.
50. M. Corless, G. Leitmann, "Continuous state feedback guaranteeing uniform ultimate boundedness for uncertain dynamic systems," *IEEE Transactions on Automatic Control*, vol. 26, pp. 1139–1144, 1981.

51. T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, *Introduction to Algorithms*, 2nd ed., MIT Press, Cambridge, MA, 2001.
52. J.J. Craig, *Adaptive Control of Mechanical Manipulators*, Addison-Wesley, Reading, MA, 1988.
53. J.J. Craig, *Introduction to Robotics: Mechanics and Control*, 3rd ed., Pearson Prentice Hall, Upper Saddle River, NJ, 2004.
54. C. De Boor, *A Practical Guide to Splines*, Springer-Verlag, New York, 1978.
55. T.L. De Fazio, D.S. Seltzer, D.E. Whitney, "The instrumented Remote Center of Compliance," *Industrial Robot*, vol. 11, pp. 238–242, 1984.
56. A. De Luca, *A Spline Generator for Robot Arms*, tech. rep. RAL 68, Rensselaer Polytechnic Institute, Department of Electrical, Computer, and Systems Engineering, 1986.
57. A. De Luca, C. Manes, "Modeling robots in contact with a dynamic environment," *IEEE Transactions on Robotics and Automation*, vol. 10, pp. 542–548, 1994.
58. A. De Luca, G. Oriolo, C. Samson, "Feedback control of a nonholonomic car-like robot," in *Robot Motion Planning and Control*, J.-P. Laumond, (Ed.), Springer-Verlag, Berlin, Germany, 1998.
59. A. De Luca, G. Oriolo, B. Siciliano, "Robot redundancy resolution at the acceleration level," *Laboratory Robotics and Automation*, vol. 4, pp. 97–106, 1992.
60. J. Denavit, R.S. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *ASME Journal of Applied Mechanics*, vol. 22, pp. 215–221, 1955.
61. P.M. DeRusso, R.J. Roy, C.M. Close, A.A. Desrochers, *State Variables for Engineers*, 2nd ed., Wiley, New York, 1998.
62. J. De Schutter, H. Bruyninckx, S. Dutré, J. De Geeter, J. Katupitiya, S. Demey, T. Lefebvre, "Estimating first-order geometric parameters and monitoring contact transitions during force-controlled compliant motions," *International Journal of Robotics Research*, vol. 18, pp. 1161–1184, 1999.
63. J. De Schutter, H. Bruyninckx, W.-H. Zhu, M.W. Spong, "Force control: A bird's eye view," in *Control Problems in Robotics and Automation*, B. Siciliano, K.P. Valavanis, (Ed.), pp. 1–17, Springer-Verlag, London, UK, 1998.
64. J. De Schutter, H. Van Brussel, "Compliant robot motion I. A formalism for specifying compliant motion tasks," *International Journal of Robotics Research*, vol. 7, no. 4, pp. 3–17, 1988.
65. J. De Schutter, H. Van Brussel, "Compliant robot motion II. A control approach based on external control loops," *International Journal of Robotics Research*, vol. 7, no. 4, pp. 18–33, 1988.
66. K.L. Doty, C. Melchiorri, C. Bonivento, "A theory of generalized inverses applied to robotics," *International Journal of Robotics Research*, vol. 12, pp. 1–19, 1993.
67. S. Dubowsky, D.T. DesForges, "The application of model referenced adaptive control to robotic manipulators," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 101, pp. 193–200, 1979.
68. C. Edwards, L. Galloway, "A single-point calibration technique for a six-degree–of–freedom articulated arm," *International Journal of Robotics Research*, vol. 13, pp. 189–199, 1994.
69. O. Egeland, "Task-space tracking with redundant manipulators," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 471–475, 1987.

70. S.D. Eppinger, W.P. Seering, "Introduction to dynamic models for robot force control," *IEEE Control Systems Magazine*, vol. 7, no. 2, pp. 48–52, 1987.
71. B. Espiau, F. Chaumette, P. Rives, "A new approach to visual servoing in robotics," *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 313–326, 1992.
72. H.R. Everett, *Sensors for Mobile Robots: Theory and Application*, AK Peters, Wellesley, MA, 1995.
73. G.E. Farin, *Curves and Surfaces for CAGD: A Practical Guide*, 5th ed., Morgan Kaufmann Publishers, San Francisco, CA, 2001.
74. E.D. Fasse, P.C. Breedveld, "Modelling of elastically coupled bodies: Parts I–II", *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 120, pp. 496–506, 1998.
75. O. Faugeras, *Three-Dimensional Computer Vision: A Geometric Viewpoint*, MIT Press, Boston, MA, 1993.
76. R. Featherstone, "Position and velocity transformations between robot end-effector coordinates and joint angles," *International Journal of Robotics Research*, vol. 2, no. 2, pp. 35–45, 1983.
77. R. Featherstone, *Robot Dynamics Algorithms*, Kluwer, Boston, MA, 1987.
78. R. Featherstone, O. Khatib, "Load independence of the dynamically consistent inverse of the Jacobian matrix," *International Journal of Robotics Research*, vol. 16, pp. 168–170, 1997.
79. J. Feddema, O. Mitchell, "Vision-guided servoing with feature-based trajectory generation," *IEEE Transactions on Robotics and Automation*, vol. 5, pp. 691–700, 1989.
80. M. Fliess, J. Lévine, P. Martin, P. Rouchon, "Flatness and defect of nonlinear systems: Introductory theory and examples," *International Journal of Control*, vol. 61, pp. 1327–1361, 1995.
81. J. Fraden, *Handbook of Modern Sensors: Physics, Designs, and Applications*, Springer, New York, 2004.
82. G.F. Franklin, J.D. Powell, A. Emami-Naeini, *Feedback Control of Dynamic Systems*, 5th ed., Prentice-Hall, Lebanon, IN, 2005.
83. E. Freund, "Fast nonlinear control with arbitrary pole-placement for industrial robots and manipulators," *International Journal of Robotics Research*, vol. 1, no. 1, pp. 65–78, 1982.
84. L.-C. Fu, T.-L. Liao, "Globally stable robust tracking of nonlinear systems using variable structure control with an application to a robotic manipulator," *IEEE Transactions on Automatic Control*, vol. 35, pp. 1345–1350, 1990.
85. M. Gautier, W. Khalil, "Direct calculation of minimum set of inertial parameters of serial robots," *IEEE Transactions on Robotics and Automation*, vol. 6, pp. 368–373, 1990.
86. A.A. Goldenberg, B. Benhabib, R.G. Fenton, "A complete generalized solution to the inverse kinematics of robots," *IEEE Journal of Robotics and Automation*, vol. 1, pp. 14–20, 1985.
87. H. Goldstein, C.P. Poole, J.L. Safko, *Classical Mechanics*, 3rd ed., Addison-Wesley, Reading, MA, 2002.
88. G.H. Golub, C.F. Van Loan, *Matrix Computations*, 3rd ed., The Johns Hopkins University Press, Baltimore, MD, 1996.
89. M.C. Good, L.M. Sweet, K.L. Strobel, "Dynamic models for control system design of integrated robot and drive systems," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 107, pp. 53–59, 1985.

90. D.M. Gorinevski, A.M. Formalsky, A.Yu. Schneider, *Force Control of Robotics Systems*, CRC Press, Boca Raton, FL, 1997.
91. W.A. Gruver, B.I. Soroka, J.J. Craig, T.L. Turner, "Industrial robot programming languages: A comparative evaluation," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 14, pp. 565–570, 1984.
92. G. Hager, W. Chang, A. Morse, "Robot feedback control based on stereo vision: Towards calibration-free hand-eye coordination," *IEEE Control Systems Magazine*, vol. 15, no. 1, pp. 30–39, 1995.
93. R.M. Haralick, L.G. Shapiro, *Computer and Robot Vision*, vols. 1 & 2, Addison-Wesley, Reading, MA, 1993.
94. S. Helgason, *Differential Geometry and Symmetric Spaces*, Academic Press, New York, NY, 1962.
95. N. Hogan, "Impedance control: An approach to manipulation: Part I — Theory," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 107, pp. 1–7, 1985.
96. J.M. Hollerbach, "A recursive Lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 10, pp. 730–736, 1980.
97. J.M. Hollerbach, "Dynamic scaling of manipulator trajectories," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 106, pp. 102–106, 1984.
98. J.M. Hollerbach, "A survey of kinematic calibration," in *The Robotics Review 1*, O. Khatib, J.J. Craig, and T. Lozano-Pérez (Eds.), MIT Press, Cambridge, MA, pp. 207–242, 1989.
99. J.M. Hollerbach, G. Sahar, "Wrist-partitioned inverse kinematic accelerations and manipulator dynamics," *International Journal of Robotics Research*, vol. 2, no. 4, pp. 61–76, 1983.
100. R. Horowitz, M. Tomizuka, "An adaptive control scheme for mechanical manipulators — Compensation of nonlinearity and decoupling control," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 108, pp. 127–135, 1986.
101. T.C.S. Hsia, T.A. Lasky, Z. Guo, "Robust independent joint controller design for industrial robot manipulators," *IEEE Transactions on Industrial Electronics*, vol. 38, pp. 21–25, 1991.
102. P. Hsu, J. Hauser, S. Sastry, "Dynamic control of redundant manipulators," *Journal of Robotic Systems*, vol. 6, pp. 133–148, 1989.
103. S. Hutchinson, G. Hager, P. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 651–670, 1996.
104. A. Isidori, *Nonlinear Control Systems*, 3rd ed., Springer-Verlag, London, UK, 1995.
105. H. Kazerooni, P.K. Houpt, T.B. Sheridan, "Robust compliant motion of manipulators, Part I: The fundamental concepts of compliant motion," *IEEE Journal of Robotics and Automation*, vol. 2, pp. 83–92, 1986.
106. J.L. Jones, A.M. Flynn, *Mobile Robots: Inspiration to Implementation*, AK Peters, Wellesley, MA, 1993.
107. L.E. Kavraki, P. Svestka, J.-C. Latombe, M.H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 566–580, 1996.

108. R. Kelly, R. Carelli, O. Nasisi, B. Kuchen, F. Reyes, "Stable visual servoing of camera-in-hand robotic systems," *IEEE/ASME Transactions on Mechatronics*, vol. 5, pp. 39–48, 2000.
109. H.K. Khalil, *Nonlinear Systems*, Prentice-Hall, Englewood Cliffs, NJ, 2002.
110. W. Khalil, F. Bennis, "Symbolic calculation of the base inertial parameters of closed-loop robots," *International Journal of Robotics Research*, vol. 14, pp. 112–128, 1995.
111. W. Khalil, E. Dombre, *Modeling, Identification and Control of Robots*, Hermes Penton Ltd, London, 2002.
112. W. Khalil, J.F. Kleinfinger, "Minimum operations and minimum parameters of the dynamic model of tree structure robots," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 517–526, 1987.
113. O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 90–98, 1986.
114. O. Khatib, "A unified approach to motion and force control of robot manipulators: The operational space formulation," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 43–53, 1987.
115. P.K. Khosla, "Categorization of parameters in the dynamic robot model," *IEEE Transactions on Robotics and Automation*, vol. 5, pp. 261–268, 1989.
116. P.K. Khosla, T. Kanade, "Parameter identification of robot dynamics," in *Proceedings of 24th IEEE Conference on Decision and Control*, Fort Lauderdale, FL, pp. 1754–1760, 1985.
117. P.K. Khosla, T. Kanade, "Experimental evaluation of nonlinear feedback and feedforward control schemes for manipulators," *International Journal of Robotics Research*, vol. 7, no. 1, pp. 18–28, 1988.
118. C.A. Klein, C.H. Huang, "Review of pseudoinverse control for use with kinematically redundant manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, pp. 245–250, 1983.
119. D.E. Koditschek, "Natural motion for robot arms," *Proc. 23th IEEE Conference on Decision and Control*, Las Vegas, NV, pp. 733–735, 1984.
120. A.J. Koivo, *Fundamentals for Control of Robotic Manipulators*, Wiley, New York, 1989.
121. K. Kreutz, "On manipulator control by exact linearization," *IEEE Transactions on Automatic Control*, vol. 34, pp. 763–767, 1989.
122. J.-C. Latombe, *Robot Motion Planning*, Kluwer, Boston, MA, 1991.
123. J.-P. Laumond, (Ed.), *Robot Motion Planning and Control*, Springer-Verlag, Berlin, 1998.
124. S.M. LaValle, *Planning Algorithms*, Cambridge University Press, New York, 2006.
125. S.M. LaValle, J.J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *New Directions in Algorithmic and Computational Robotics*, B.R. Donald, K. Lynch, D. Rus, (Eds.), AK Peters, Wellesley, MA, pp. 293–308, 2001.
126. M.B. Leahy, G.N. Saridis, "Compensation of industrial manipulator dynamics," *International Journal of Robotics Research*, vol. 8, no. 4, pp. 73–84, 1989.
127. C.S.G. Lee, "Robot kinematics, dynamics and control," *IEEE Computer*, vol. 15, no. 12, pp. 62–80, 1982.
128. W. Leonhard, *Control of Electrical Drives*, Springer-Verlag, New York, 2001.

129. A. Liégeois, "Automatic supervisory control of the configuration and behavior of multibody mechanisms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 7, pp. 868–871, 1977.
130. K.Y. Lim, M. Eslami, "Robust adaptive controller designs for robot manipulator systems," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 54–66, 1987.
131. C.S. Lin, P.R. Chang, J.Y.S. Luh, "Formulation and optimization of cubic polynomial joint trajectories for industrial robots," *IEEE Transactions on Automatic Control*, vol. 28, pp. 1066–1073, 1983.
132. S.K. Lin, "Singularity of a nonlinear feedback control scheme for robots," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, pp. 134–139, 1989.
133. H. Lipkin, J. Duffy, "Hybrid twist and wrench control for a robotic manipulator," *ASME Journal of Mechanism, Transmissions, and Automation Design*, vol. 110, pp. 138–144, 1988.
134. V. Lippiello, B. Siciliano, L. Villani, "Position-based visual servoing in industrial multirobot cells using a hybrid camera configuration," *IEEE Transactions on Robotics and Automation*, vol. 23, pp. 73–86, 2007.
135. D.A. Lizárraga, "Obstructions to the existence of universal stabilizers for smooth control systems," *Mathematics of Control, Signals, and Systems*, vol. 16, pp. 255–277, 2004.
136. J. Lončarić, "Normal forms of stiffness and compliance matrices," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 567–572, 1987.
137. T. Lozano-Pérez, "Automatic planning of manipulator transfer movements," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, pp. 681–698, 1981.
138. T. Lozano-Pérez, "Spatial planning: A configuration space approach," *IEEE Transactions on Computing*, vol. 32, pp. 108–120, 1983.
139. T. Lozano-Pérez, "Robot programming," *Proceedings IEEE*, vol. 71, pp. 821–841, 1983.
140. T. Lozano-Pérez, M.T. Mason, R.H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *International Journal of Robotics Research*, vol. 3, no. 1, pp. 3–24, 1984.
141. J.Y.S. Luh, "Conventional controller design for industrial robots: A tutorial," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 13, pp. 298–316, 1983.
142. J.Y.S. Luh, M.W. Walker, R.P.C. Paul, "On-line computational scheme for mechanical manipulators," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 102, pp. 69–76, 1980.
143. J.Y.S. Luh, M.W. Walker, R.P.C. Paul, "Resolved-acceleration control of mechanical manipulators," *IEEE Transactions on Automatic Control*, vol. 25, pp. 468–474, 1980.
144. J.Y.S. Luh, Y.-F. Zheng, "Computation of input generalized forces for robots with closed kinematic chain mechanisms," *IEEE Journal of Robotics and Automation* vol. 1, pp. 95–103, 1985.
145. V.J. Lumelsky, *Sensing, Intelligence, Motion: How Robots and Humans Move in an Unstructured World*, Wiley, Hoboken, NJ, 2006.
146. Y. Ma, S. Soatto, J. Kosecka, S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*, Springer, New York, 2003.

147. A.A. Maciejewski, C.A. Klein, "Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments," *International Journal of Robotics Research*, vol. 4, no. 3, pp. 109–117, 1985.
148. E. Malis, F. Chaumette, S. Boudet, "2-1/2D visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 15, pp. 238–250, 1999.
149. B.R. Markiewicz, *Analysis of the Computed Torque Drive Method and Comparison with Conventional Position Servo for a Computer-Controlled Manipulator*, memo. TM 33-601, JPL, Pasadena, CA, 1973.
150. M.T. Mason, "Compliance and force control for computer controlled manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 6, pp. 418–432, 1981.
151. J.M. McCarthy, *An Introduction to Theoretical Kinematics*, MIT Press, Cambridge, MA, 1990.
152. N.H. McClamroch, D. Wang, "Feedback stabilization and tracking of constrained robots," *IEEE Transactions on Automatic Control*, vol. 33, pp. 419–426, 1988.
153. R.T. M'Closkey, R.M. Murray, "Exponential stabilization of driftless nonlinear control systems using homogeneous feedback," *IEEE Transactions on Automatic Control*, vol. 42, pp. 614–628, 1997.
154. L. Meirovitch, *Dynamics and Control of Structures*, Wiley, New York, 1990.
155. C. Melchiorri, *Traiettorie per Azionamenti Elettrici*, Progetto Leonardo, Bologna, I, 2000.
156. N. Manring, *Hydraulic Control Systems*, Wiley, New York, 2005.
157. R. Middleton, G.C. Goodwin, "Adaptive computed torque control for rigid link manipulators," *Systems & Control Letters*, vol. 10, pp. 9–16, 1988.
158. R.R. Murphy, *Introduction to AI Robotics*, MIT Press, Cambridge, MA, 2000.
159. R.M. Murray, Z. Li, S.S. Sastry, *A Mathematical Introduction to Robotic Manipulation*, CRC Press, Boca Raton, CA. 1994.
160. Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*, Addison-Wesley, Reading, MA, 1991.
161. Y. Nakamura, H. Hanafusa, "Inverse kinematic solutions with singularity robustness for robot manipulator control," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 108, pp. 163–171, 1986.
162. Y. Nakamura, H. Hanafusa, "Optimal redundancy control of robot manipulators," *International Journal of Robotics Research*, vol. 6, no. 1, pp. 32–42, 1987.
163. Y. Nakamura, H. Hanafusa, T. Yoshikawa, "Task-priority based redundancy control of robot manipulators," *International Journal of Robotics Research*, vol. 6, no. 2, pp. 3–15, 1987.
164. J.I. Neimark, F.A. Fufaev, *Dynamics of Nonholonomic Systems*, American Mathematical Society, Providence, RI, 1972.
165. I. Nevins, D.E. Whitney, "The force vector assembler concept," *Proc. First CISM-IFToMM Symposium on Theory and Practice of Robots and Manipulators*, Udine, I, 1973.
166. F. Nicolò, J. Katende, "A robust MRAC for industrial robots," *Proc. 2nd IASTED International Symposium on Robotics and Automation*, pp. 162–171, Lugano, Switzerland, 1983.
167. S. Nicosia, P. Tomei, "Model reference adaptive control algorithms for industrial robots," *Automatica*, vol. 20, pp. 635–644, 1984.

168. H. Nijmeijer, A. van de Schaft, *Nonlinear Dynamical Control Systems*, Springer-Verlag, Berlin, Germany, 1990.
169. B. Noble, *Applied Linear Algebra*, 3rd ed., Prentice-Hall, Englewood Cliffs, NJ, 1987.
170. C. O'Dúnlaing, C.K. Yap, "A retraction method for planning the motion of a disc," *Journal of Algorithms*, vol. 6, pp. 104–111, 1982.
171. K. Ogata, *Modern Control Engineering*, 4th ed., Prentice-Hall, Englewood Cliffs, NJ, 2002.
172. D.E. Orin, R.B. McGhee, M. Vukobratović, G. Hartoch, "Kinematic and kinetic analysis of open-chain linkages utilizing Newton–Euler methods," *Mathematical Biosciences* vol. 43, pp. 107–130, 1979.
173. D.E. Orin, W.W. Schrader, "Efficient computation of the Jacobian for robot manipulators," *International Journal of Robotics Research*, vol. 3, no. 4, pp. 66–75, 1984.
174. G. Oriolo, A. De Luca, M. Vendittelli, "WMR control via dynamic feedback linearization: Design, implementation and experimental validation," *IEEE Transactions on Control Systems Technology*, vol. 10, pp. 835–852, 2002.
175. R. Ortega, M.W. Spong, "Adaptive motion control of rigid robots: A tutorial," *Automatica*, vol. 25, pp. 877–888, 1989.
176. T. Patterson, H. Lipkin, "Duality of constrained elastic manipulation," *Proc. 1991 IEEE International Conference on Robotics and Automation*, pp. 2820–2825, Sacramento, CA, 1991.
177. T. Patterson, H. Lipkin, "Structure of robot compliance," *ASME Journal of Mechanical Design*, vol. 115, pp. 576–580, 1993.
178. R.P. Paul, *Modelling, Trajectory Calculation, and Servoing of a Computer Controlled Arm*, memo. AIM 177, Stanford Artificial Intelligence Laboratory, 1972.
179. R.P. Paul, "Manipulator Cartesian path control," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, pp. 702–711, 1979.
180. R.P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*, MIT Press, Cambridge, MA, 1981.
181. R.P. Paul, B.E. Shimano, G. Mayer, "Kinematic control equations for simple manipulators," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, pp. 449–455, 1981.
182. R.P. Paul, H. Zhang, "Computationally efficient kinematics for manipulators with spherical wrists based on the homogeneous transformation representation," *International Journal of Robotics Research*, vol. 5, no. 2, pp. 32–44, 1986.
183. D.L. Pieper, *The Kinematics of Manipulators Under Computer Control* memo. AIM 72, Stanford Artificial Intelligence Laboratory, 1968.
184. M.H. Raibert, J.J. Craig, "Hybrid position/force control of manipulators," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 103, pp. 126–133, 1981.
185. E. Rimon, D.E. Koditschek, "The construction of analytic diffeomorphisms for exact robot navigation on star worlds," *Proc. 1989 IEEE International Conference on Robotics and Automation*, Scottsdale, AZ, pp. 21–26, 1989.
186. E.I. Rivin, *Mechanical Design of Robots*, McGraw-Hill, New York, 1987.
187. R.E. Roberson, R. Schwertassek, *Dynamics of Multibody Systems*, Springer-Verlag, Berlin, Germany, 1988.
188. Z. Roth, B.W. Mooring, B. Ravani, "An overview of robot calibration," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 377–386, 1987.

189. S. Russell, P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed., Prentice Hall, Englewood Cliffs, NJ, 2003.
190. J.K. Salisbury, "Active stiffness control of a manipulator in Cartesian coordinates," *Proc. 19th IEEE Conference on Decision and Control*, pp. 95–100, Albuquerque, NM, 1980.
191. J.K. Salisbury, J.J. Craig, "Articulated hands: Force control and kinematic issues," *International Journal of Robotics Research*, vol. 1, no. 1, pp. 4–17, 1982.
192. C. Samson, "Robust control of a class of nonlinear systems and applications to robotics," *International Journal of Adaptive Control and Signal Processing*, vol. 1, pp. 49–68, 1987.
193. C. Samson, "Time-varying feedback stabilization of car-like wheeled mobile robots," *International Journal of Robotics Research*, vol. 12, no. 1, pp. 55–64, 1993.
194. C. Samson, M. Le Borgne, B. Espiau, *Robot Control: The Task Function Approach*, Clarendon Press, Oxford, UK, 1991.
195. S. Sastry, *Nonlinear Systems: Analysis, Stability and Control*, Springer-Verlag, Berlin, Germany, 1999.
196. V.D. Scheinman, *Design of a Computer Controlled Manipulator*, memo. AIM 92, Stanford Artificial Intelligence Laboratory, 1969.
197. J.T. Schwartz, M. Sharir, "On the 'piano movers' problem: II. General techniques for computing topological properties of real algebraic manifolds," *Advances in Applied Mathematics*, vol. 4, pp. 298–351, 1983.
198. L. Sciavicco, B. Siciliano, "Coordinate transformation: A solution algorithm for one class of robots," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, pp. 550–559, 1986.
199. L. Sciavicco, B. Siciliano, "A solution algorithm to the inverse kinematic problem for redundant manipulators," *IEEE Journal of Robotics and Automation*, vol. 4, pp. 403–410, 1988.
200. L. Sciavicco, B. Siciliano, *Modelling and Control of Robot Manipulators*, 2nd ed., Springer, London, UK, 2000.
201. L. Sciavicco, B. Siciliano, L. Villani, "Lagrange and Newton–Euler dynamic modeling of a gear-driven rigid robot manipulator with inclusion of motor inertia effects," *Advanced Robotics*, vol. 10, pp. 317–334, 1996.
202. R. Sedgewick, *Algorithms*, 2nd ed., Addison-Wesley, Reading, MA, 1988.
203. H. Seraji, "Configuration control of redundant manipulators: Theory and implementation," *IEEE Transactions on Robotics and Automation*, vol. 5, pp. 472–490, 1989.
204. S.W. Shepperd, S.W., "Quaternion from rotation matrix," *AIAA Journal of Guidance and Control*, vol. 1, pp. 223–224, 1978.
205. R. Shoureshi, M.E. Momot, M.D. Roesler, "Robust control for manipulators with uncertain dynamics," *Automatica*, vol. 26, pp. 353–359, 1990.
206. B. Siciliano, "Kinematic control of redundant robot manipulators: A tutorial," *Journal of Intelligent and Robotic Systems*, vol. 3, pp. 201–212, 1990.
207. B. Siciliano, "A closed-loop inverse kinematic scheme for on-line joint based robot control," *Robotica*, vol. 8, pp. 231–243, 1990.
208. B. Siciliano, J.-J.E. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," *Proc. 5th International Conference on Advanced Robotics*, Pisa, I, pp. 1211–1216, 1991.

209. B. Siciliano, L. Villani, *Robot Force Control*, Kluwer, Boston, MA, 2000.
210. R. Siegwart, I.R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*, MIT Press, Cambridge, MA, 2004.
211. D.B. Silver, "On the equivalence of Lagrangian and Newton–Euler dynamics for manipulators," *International Journal of Robotics Research*, vol. 1, no. 2, pp. 60–70, 1982.
212. J.-J.E. Slotine, "The robust control of robot manipulators," *International Jorunal of Robotics Research*, vol. 4, no. 2, pp. 49–64, 1985.
213. J.-J.E. Slotine, "Putting physics in control — The example of robotics," *IEEE Control Systems Magazine*, vol. 8, no. 6, pp. 12–18, 1988.
214. J.-J.E. Slotine, W. Li, "On the adaptive control of robot manipulators," *International Journal of Robotics Research*, vol. 6, no. 3, pp. 49–59, 1987.
215. J.-J.E. Slotine, W. Li, *Applied Nonlinear Control*, Prentice-Hall, Englewood Cliffs, NJ, 1991.
216. M.W. Spong, "On the robust control of robot manipulators," *IEEE Transactions on Automatic Control*, vol. 37, pp. 1782–1786, 1992.
217. M.W. Spong, S. Hutchinson, M. Vidyasagar, *Robot Modeling and Control*, Wiley, New York, 2006.
218. M.W. Spong, R. Ortega, R. Kelly, "Comments on "Adaptive manipulator control: A case study"," *IEEE Transactions on Automatic Control*, vol. 35, pp. 761–762, 1990.
219. M.W. Spong, M. Vidyasagar, "Robust linear compensator design for nonlinear robotic control," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 345–351, 1987.
220. SRI International, *Robot Design Handbook*, G.B. Andeen, (Ed.), McGraw-Hill, New York, 1988.
221. Y. Stepanenko, M. Vukobratović, "Dynamics of articulated open-chain active mechanisms," *Mathematical Biosciences*, vol. 28, pp. 137–170, 1976.
222. Y. Stepanenko, J. Yuan, "Robust adaptive control of a class of nonlinear mechanical systems with unbounded and fast varying uncertainties," *Automatica*, vol. 28, pp. 265–276, 1992.
223. S. Stramigioli, *Modeling and IPC Control of Interactive Mechanical Systems — A Coordinate Free Approach*, Springer, London, UK, 2001.
224. K.R. Symon, *Mechanics*, 3rd ed., Addison-Wesley, Reading, MA, 1971.
225. K. Takase, R. Paul, E. Berg, "A structured approach to robot programming and teaching," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 11, pp. 274–289, 1981.
226. M. Takegaki, S. Arimoto, "A new feedback method for dynamic control of manipulators," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 102, pp. 119–125, 1981.
227. T.-J. Tarn, A.K. Bejczy, X. Yun, Z. Li, "Effect of motor dynamics on nonlinear feedback robot arm control," *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 114–122, 1991.
228. T.-J. Tarn, Y. Wu, N. Xi, A. Isidori, "Force regulation and contact transition control," *IEEE Control Systems Magazine*, vol. 16, no. 1, pp. 32–40, 1996.
229. R.H. Taylor, "Planning and execution of straight line manipulator trajectories," *IBM Journal of Research and Development*, vol. 23, pp. 424–436, 1979.
230. R.H. Taylor, D.D. Grossman, "An integrated robot system architecture," *Proceedings IEEE*, vol. 71, pp. 842–856, 1983.

231. S. Thrun, W. Burgard, D. Fox, *Probabilistic Robotics*, MIT Press, Cambridge, MA, 2005.
232. L.W. Tsai, A.P. Morgan, "Solving the kinematics of the most general six- and five-degree-of-freedom manipulators by continuation methods," *ASME Journal of Mechanisms, Transmission, and Automation in Design*, vol. 107, pp. 189–200, 1985.
233. R. Tsai, "A versatile camera calibration technique for high accuracy 3-D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE Transactions on Robotics and Automation*, vol. 3, pp. 323–344, 1987.
234. J.J.Uicker, "Dynamic force analysis of spatial linkages," *ASME Journal of Applied Mechanics*, vol. 34, pp. 418–424, 1967.
235. L. Villani, C. Canudas de Wit, B. Brogliato, "An exponentially stable adaptive control for force and position tracking of robot manipulators," *IEEE Transactions on Automatic Control*, vol. 44, pp. 798–802, 1999.
236. M. Vukobratović, "Dynamics of active articulated mechanisms and synthesis of artificial motion," *Mechanism and Machine Theory*, vol. 13, pp. 1–56, 1978.
237. M.W. Walker, D.E. Orin, "Efficient dynamic computer simulation of robotic mechanisms," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 104, pp. 205–211, 1982.
238. C.W. Wampler, "Manipulator inverse kinematic solutions based on damped least-squares solutions," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 16, pp. 93–101, 1986.
239. L. Weiss, A. Sanderson, C. Neuman, "Dynamic sensor-based control of robots with visual feedback," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 404–417, 1987.
240. D.E. Whitney, "Resolved motion rate control of manipulators and human prostheses," *IEEE Transactions on Man-Machine Systems*, vol. 10, pp. 47–53, 1969.
241. D.E. Whitney, "Force feedback control of manipulator fine motions," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 99, pp. 91–97, 1977.
242. D.E. Whitney, "Quasi-static assembly of compliantly supported rigid parts," *ASME Journal of Dynamic Systems, Measurement, and Control*, vol. 104, pp. 65–77, 1982.
243. D.E. Whitney, "Historical perspective and state of the art in robot force control," *International Journal of Robotics Research*, vol. 6, no. 1, pp. 3–14, 1987.
244. W. Wilson, C. Hulls, G. Bell, "Relative end-effector control using Cartesian position based visual servoing," *IEEE Transactions on Robotics and Automation*, vol. 12, pp. 684–696, 1996.
245. T. Yoshikawa, "Manipulability of robotic mechanisms," *International Journal of Robotics Research*, vol. 4, no. 2, pp. 3–9, 1985.
246. T. Yoshikawa, "Dynamic manipulability ellipsoid of robot manipulators," *Journal of Robotic Systems*, vol. 2, pp. 113–124, 1985.
247. T. Yoshikawa, "Dynamic hybrid position/force control of robot manipulators — Description of hand constraints and calculation of joint driving force," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 386–392, 1987.
248. T. Yoshikawa, *Foundations of Robotics*, MIT Press, Boston, MA, 1990.
249. T. Yoshikawa, T. Sugie, N. Tanaka, "Dynamic hybrid position/force control of robot manipulators — Controller design and experiment," *IEEE Journal of Robotics and Automation*, vol. 4, pp. 699–705, 1988.

250. J.S.-C. Yuan, "Closed-loop manipulator control using quaternion feedback," *IEEE Journal of Robotics and Automation*, vol. 4, pp. 434–440, 1988.

Index