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, analytical *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.

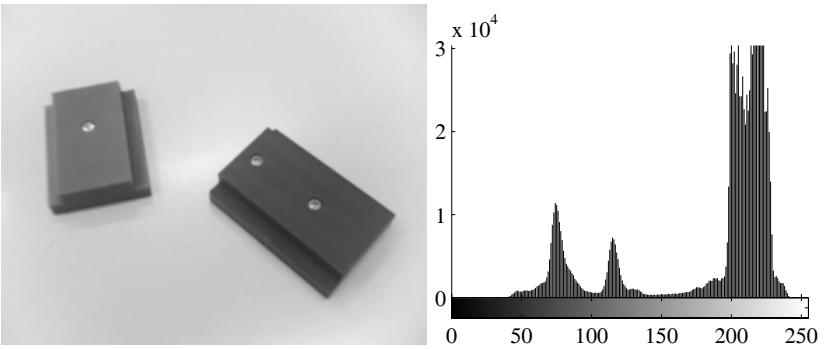

Fig. 10.1. Black-and-white image and corresponding gray-level histogram on the right

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

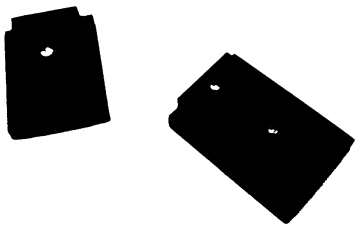

Fig. 10.2. Binary image corresponding to image of Fig. 10.1

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:

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

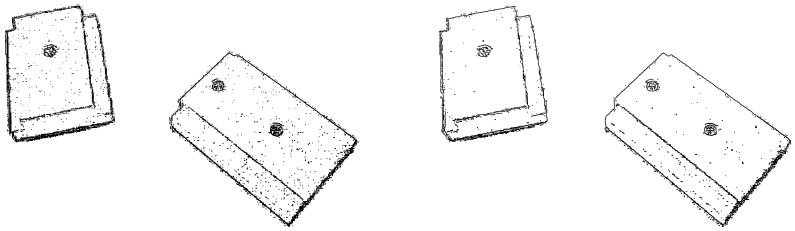

Fig. 10.3. Contours of image of Fig. 10.1 obtained using Roberts (*left*) and Sobel (*right*) operators

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

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

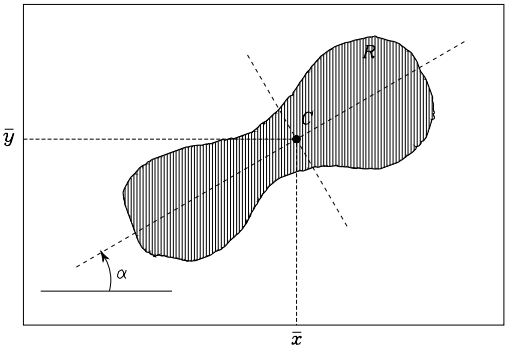

Fig. 10.4. Region of a binary image and some feature parameters

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.

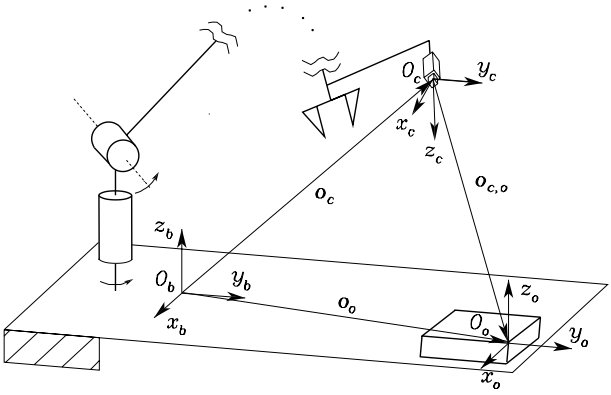

Fig. 10.5. Reference frames for an eye-in-hand camera

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 Analytical 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 analytical 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

$$\boldsymbol{S}(\widetilde{\boldsymbol{s}}_i)\boldsymbol{H}\begin{bmatrix} r_{x,i} & r_{y,i} & 1 \end{bmatrix}^T = \boldsymbol{0}, \tag{10.7}$$

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

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{o}_{c,o}^{c} \end{bmatrix}, \tag{10.8}$$

$\boldsymbol{r}_1$ 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

$$\boldsymbol{A}_i(\boldsymbol{s}_i) = \begin{bmatrix} r_{x,i}\boldsymbol{S}(\widetilde{\boldsymbol{s}}_i) & r_{y,i}\boldsymbol{S}(\widetilde{\boldsymbol{s}}_i) & \boldsymbol{S}(\widetilde{\boldsymbol{s}}_i) \end{bmatrix}. \tag{10.9}$$

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 ${\boldsymbol{R}_o^c}^T\boldsymbol{Q}$. 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.

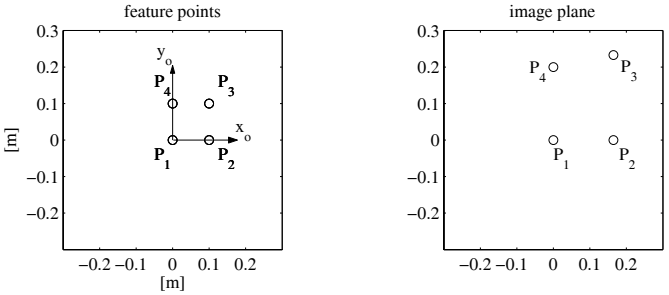

Fig. 10.6. Planar object; *left*: object frame and feature points; *right*: feature points projections on the normalized image plane of a camera in the pose of Example 10.1

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 analytical 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 analytical Jacobian of a manipulator.

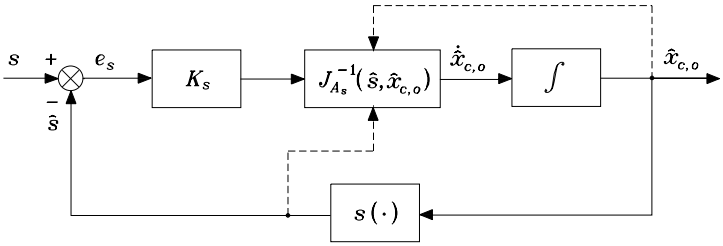

Fig. 10.7. Pose estimation algorithm based on the inverse of image Jacobian

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 analytical 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.

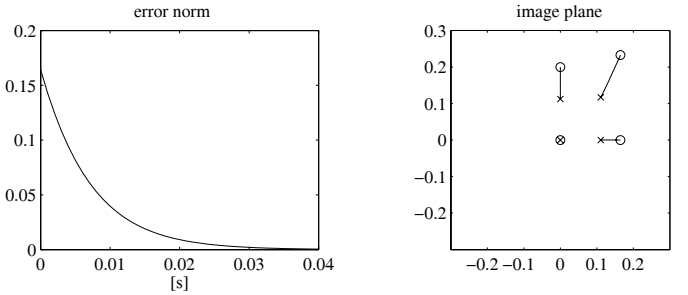

Fig. 10.8. Time history of the norm of the estimation error and corresponding paths of the feature points projections on image plane

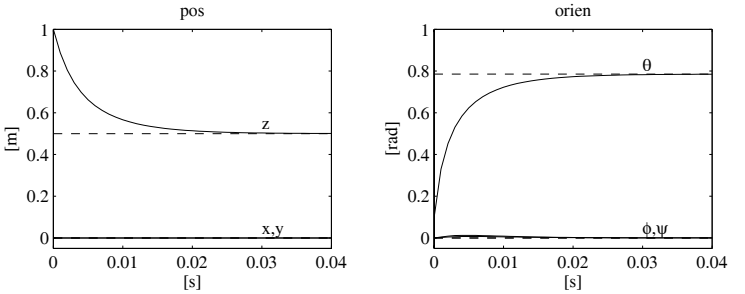

Fig. 10.9. Time history of camera pose estimate

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

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

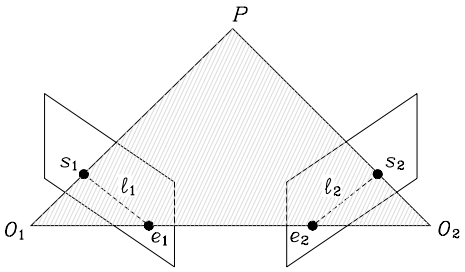

Fig. 10.10. Epipolar geometry

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 analytical 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 object 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:

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

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:

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

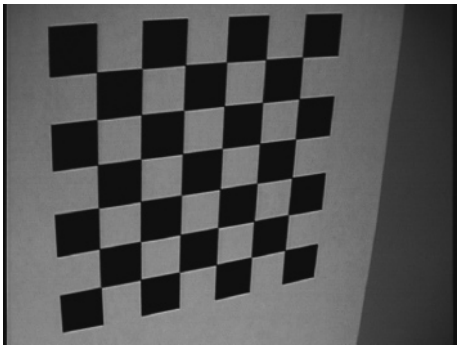

Fig. 10.11. Example of calibration plane

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

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.

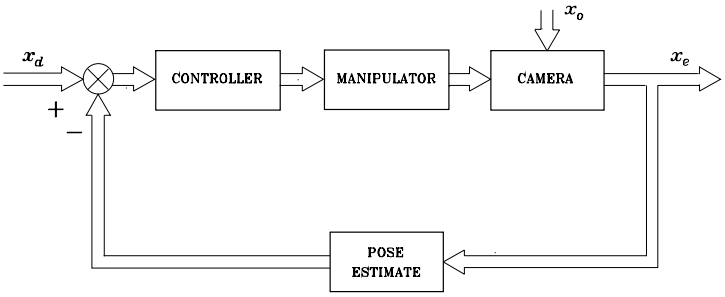

Fig. 10.12. General block scheme of position-based visual servoing

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

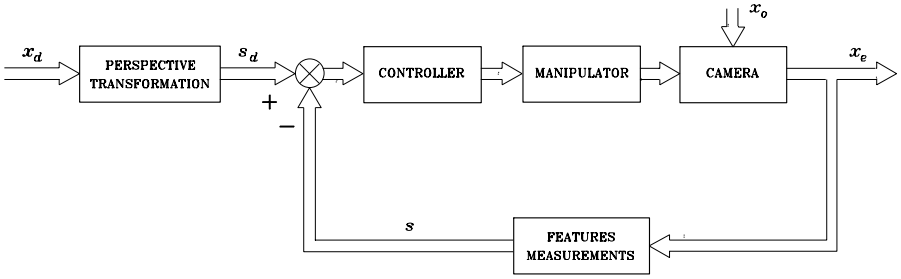

Fig. 10.13. General block scheme of image-based visual servoing

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 analytical 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

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

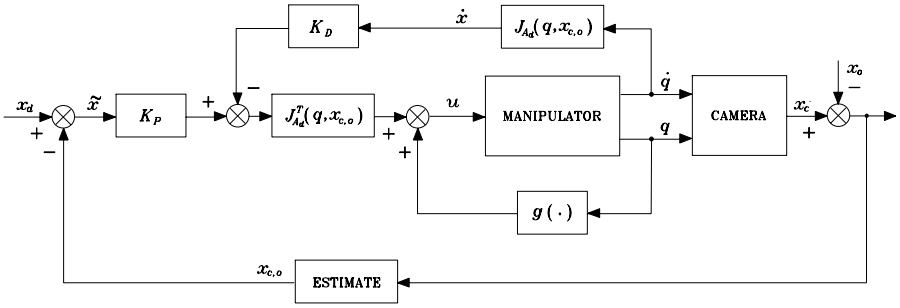

Fig. 10.14. Block scheme of position-based visual servoing of PD type with gravity compensation

similarly to the case of control law (8.110).

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 $\widetilde{\boldsymbol{x}}$ 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.

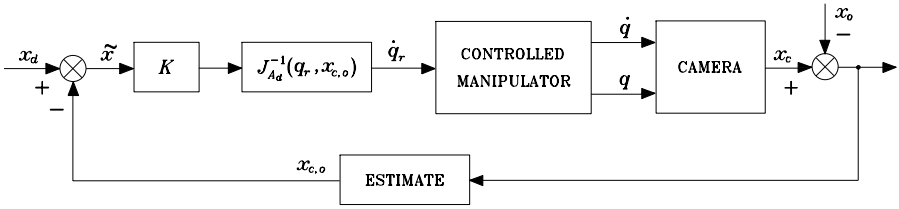

Fig. 10.15. Block scheme of resolved-velocity position-based visual servoing

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

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

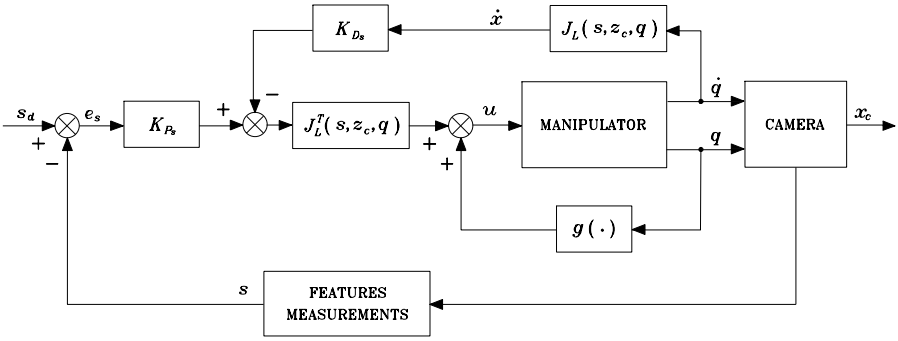

Fig. 10.16. Block scheme of image-based visual servoing of PD type with gravity compensation

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.

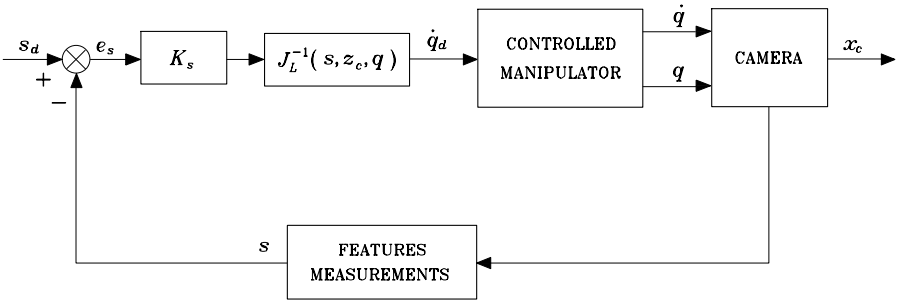

Fig. 10.17. Block scheme of resolved-velocity image-based visual servoing

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

$$\boldsymbol{x}_c(0) = [\,1 \quad 1 \quad 0.5 \quad \pi/4\,]^T$$

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

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

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

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

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 analytical 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.

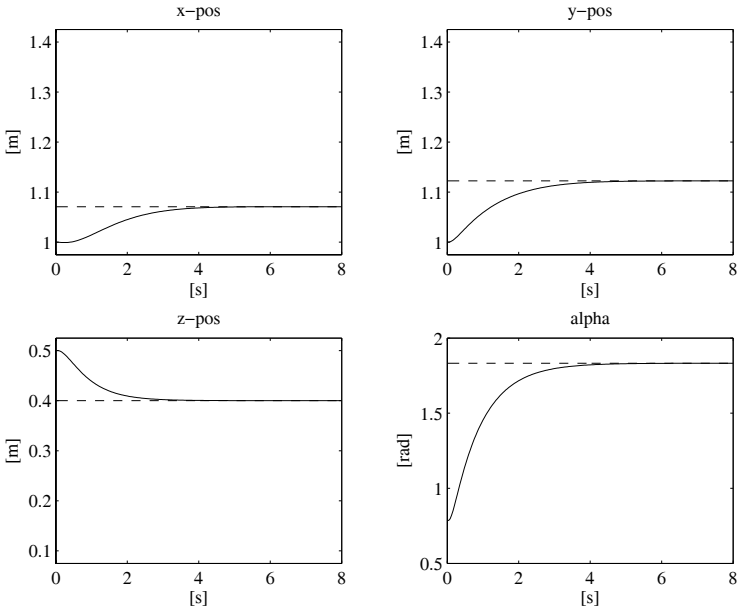

Fig. 10.18. Time history of camera frame position and orientation with control **A**

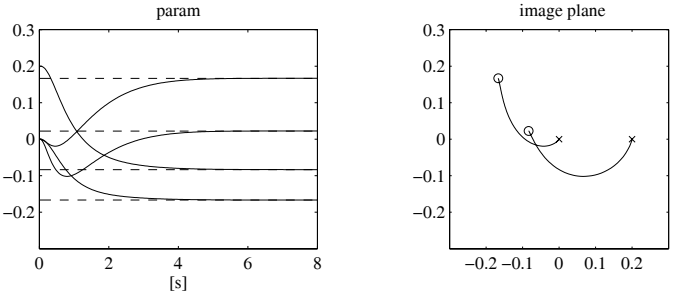

Fig. 10.19. Time history of feature parameters and corresponding path of feature points projections on image plane with control **A**

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

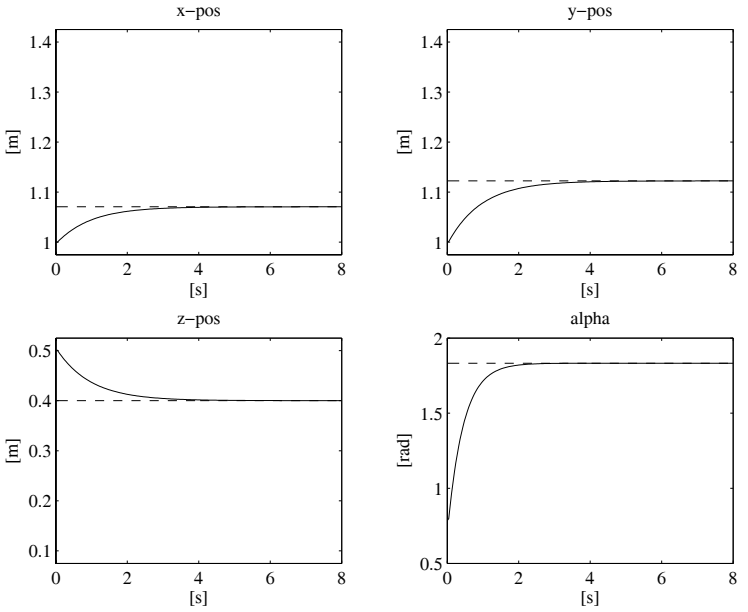

Fig. 10.20. Time history of camera frame position and orientation with control **B**

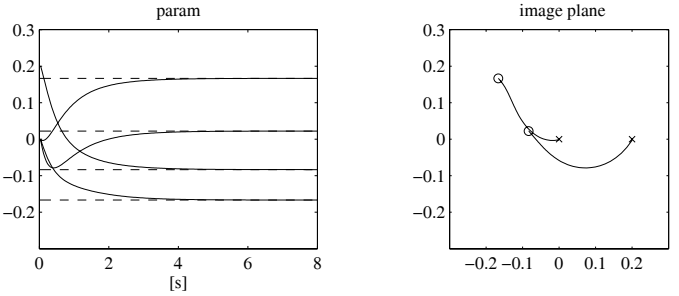

Fig. 10.21. Time history of feature parameters and corresponding path of feature points projections on image plane with control **B**

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

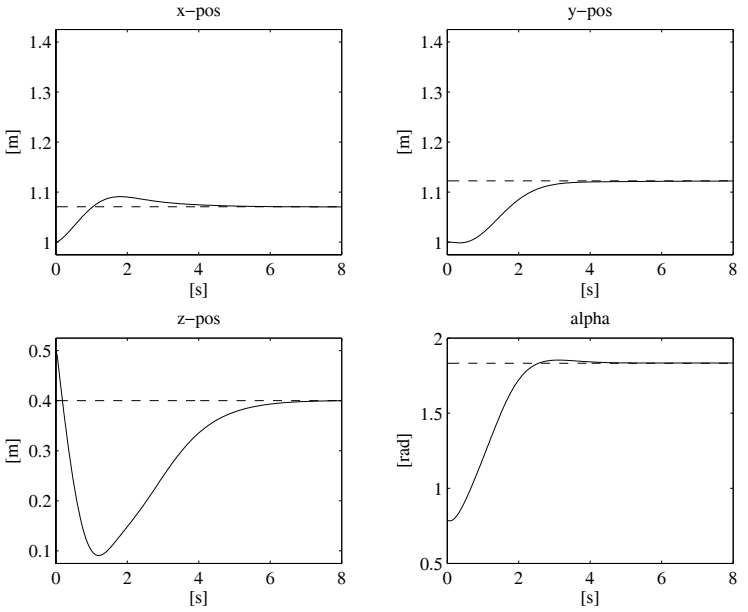

Fig. 10.22. Time history of camera frame position and orientation with control **C**

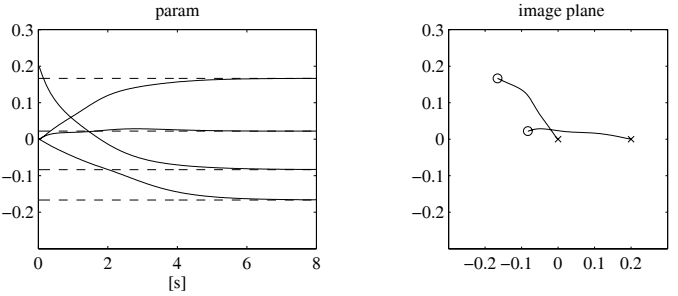

Fig. 10.23. Time history of feature parameters and corresponding path of feature points projections on image plane with control **C**

(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.

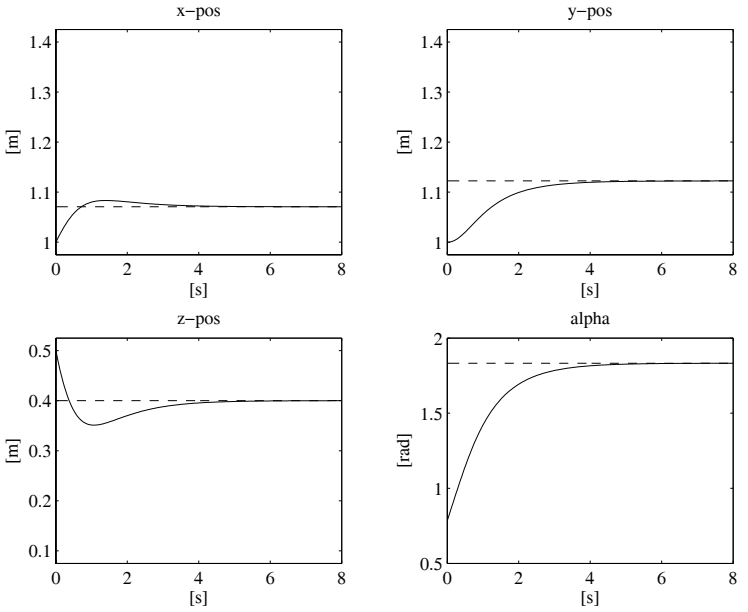

Fig. 10.24. Time history of camera frame position and orientation with control **D**

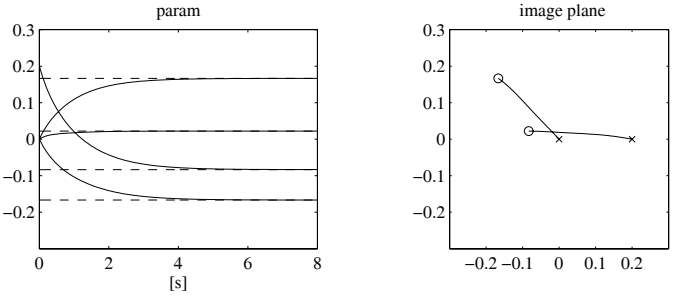

Fig. 10.25. Time history of feature parameters and corresponding path of feature points projections on image plane with control **D**

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

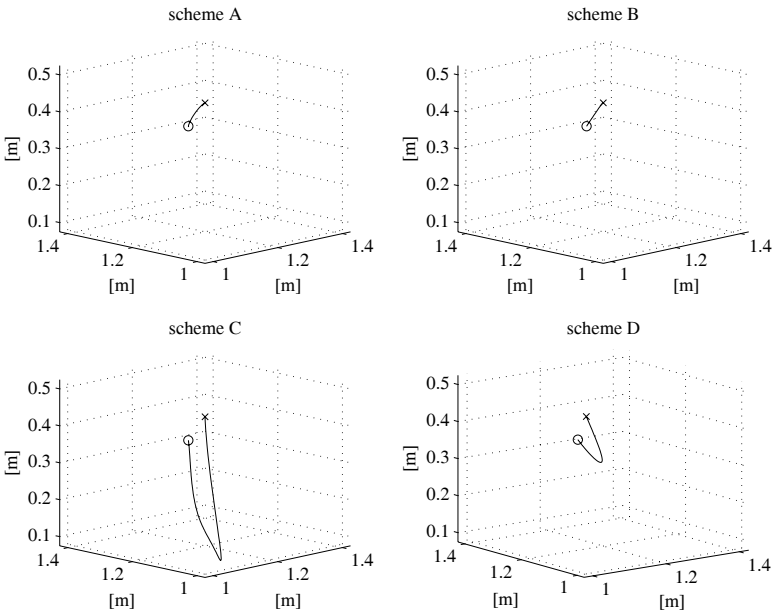

Fig. 10.26. Paths of camera frame origin in Cartesian space with the four control schemes

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

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

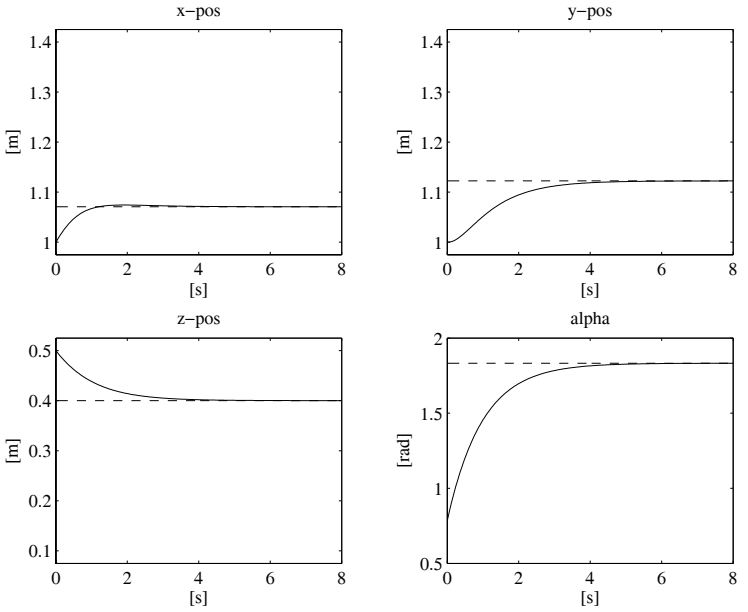

Fig. 10.27. Time history of position and orientation of camera frame with hybrid visual servoing

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:

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

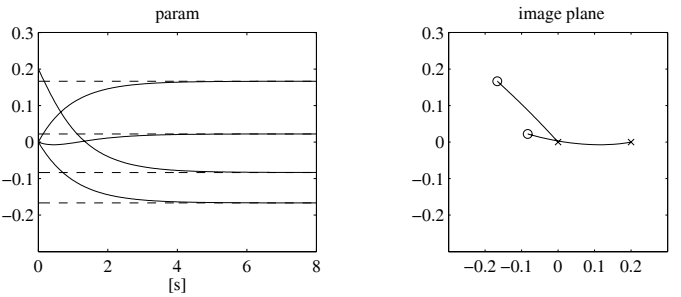

Fig. 10.28. Time history of feature parameters and corresponding path of feature points projections on image plane with hybrid visual servoing

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

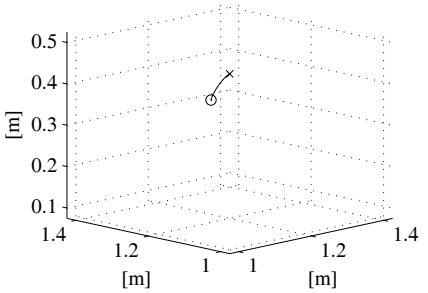

Fig. 10.29. Path of camera frame origin in Cartesian space with hybrid visual servoing

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 function (10.57) is the matrix which maximizes the trace of $\boldsymbol{R}^T\boldsymbol{K}$.

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).