Chapter 3
Image Formation

And since geometry is the right foundation of all painting, I have decided to teach its rudiments and principles to all youngsters eager for art...

– Albrecht Dürer, *The Art of Measurement, 1525*

This chapter introduces simple mathematical models of the image formation process. In a broad figurative sense, vision is the inverse problem of image formation: the latter studies how objects give rise to images, while the former attempts to use images to recover a description of objects in space. Therefore, designing vision algorithms requires first developing a suitable model of image formation. Suitable, in this context, does not necessarily mean physically accurate: the level of abstraction and complexity in modeling image formation must trade off physical constraints and mathematical simplicity in order to result in a manageable model (i.e. one that can be inverted with reasonable effort). Physical models of image formation easily exceed the level of complexity necessary and appropriate for this book, and determining the right model for the problem at hand is a form of engineering art.

It comes as no surprise, then, that the study of image formation has for centuries been in the domain of artistic reproduction and composition, more so than of mathematics and engineering. Rudimentary understanding of the geometry of image formation, which includes various models for projecting the three-dimensional world onto a plane (e.g., a canvas), is implicit in various forms of visual arts. The roots of formulating the geometry of image formation can be traced back to the work of Euclid in the fourth century B.C. Examples of partially

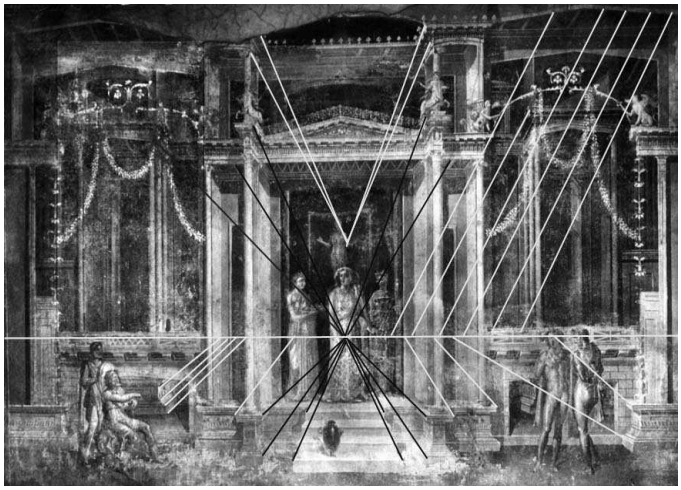

Figure 3.1. Frescoes from the first century B.C. in Pompeii. Partially correct perspective projection is visible in the paintings, although not all parallel lines converge to the vanishing point. The skill was lost during the middle ages, and it did not reappear in paintings until the Renaissance (image courtesy of C. Taylor).

correct perspective projection are visible in the frescoes and mosaics of Pompeii (Figure 3.1) from the first century B.C. Unfortunately, these skills seem to have been lost with the fall of the Roman empire, and it took over a thousand years for correct perspective projection to emerge in paintings again in the late fourteenth century. It was the early Renaissance painters who developed systematic methods for determining the perspective projection of three-dimensional landscapes. The first treatise on perspective, *Della Pictura*, was published by Leon Battista Alberti, who emphasized the "eye's view" of the world capturing correctly the geometry of the projection process. It is no coincidence that early attempts to formalize the rules of perspective came from artists proficient in architecture and engineering, such as Alberti and Brunelleschi. Geometry, however, is only a part of the image formation process: in order to obtain an image, we need to decide not only where to draw a point, but also what brightness value to assign to it. The interaction of light with matter is at the core of the studies of Leonardo Da Vinci in the 1500s, and his insights on perspective, shading, color, and even stereopsis are vibrantly expressed in his notes. Renaissance painters such as Caravaggio and Raphael exhibited rather sophisticated skills in rendering light and color that remain compelling to this day.[1]

In this book, we restrict our attention to the geometry of the scene, and therefore, we need a simple geometric model of image formation, which we derive

[1]There is some evidence that suggests that some Renaissance artists secretly used camera-like devices (*camera obscura*) [Hockney, 2001].

in this chapter. More complex photometric models are beyond the scope of this book; in the next two sections as well as in Appendix 3.A at the end of this chapter, we will review some of the basic notions of radiometry so that the reader can better evaluate the assumptions by which we are able to reduce image formation to a purely geometric process.

3.1 Representation of images

An *image*, as far as this book is concerned, is a two-dimensional brightness array.[2] In other words, it is a map I, defined on a compact region Ω of a two-dimensional surface, taking values in the positive real numbers. For instance, in the case of a camera, Ω is a planar, rectangular region occupied by the photographic medium or by the CCD sensor. So I is a function

$$I : \Omega \subset \mathbb{R}^2 \to \mathbb{R}_+; \quad (x, y) \mapsto I(x, y). \tag{3.1}$$

Such an image (function) can be represented, for instance, using the graph of I as in the example in Figure 3.2. In the case of a digital image, both the domain Ω and the range $\mathbb{R}_+$ are discretized. For instance, $\Omega = [1, 640] \times [1, 480] \subset \mathbb{Z}^2$, and $\mathbb{R}_+$ is approximated by an interval of integers $[0, 255] \subset \mathbb{Z}_+$. Such an image can be represented by an array of numbers as in Table 3.1.

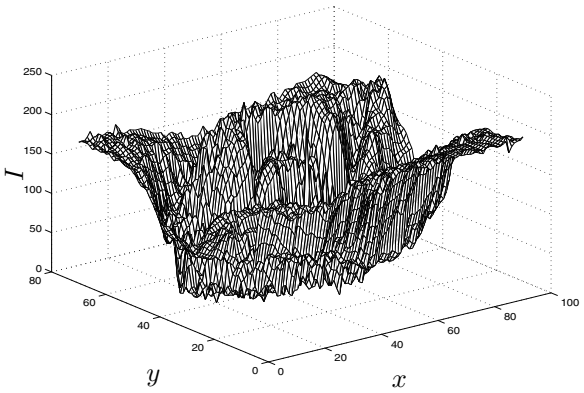

Figure 3.2. An image I represented as a two-dimensional surface, the graph of I.

The values of the image I depend upon physical properties of the scene being viewed, such as its shape, its material reflectance properties, and the distribution of the light sources. Despite the fact that Figure 3.2 and Table 3.1 do not seem very indicative of the properties of the scene they portray, this is how they are represented in a computer. A different representation of the same image that is

[2] If it is a color image, its RGB (red, green, blue) values represent three such arrays.

188	186	188	187	168	130	101	99	110	113	112	107	117	140	153	153	156	158	156	153
189	189	188	181	163	135	109	104	113	113	110	109	117	134	147	152	156	163	160	156
190	190	188	176	159	139	115	106	114	123	114	111	119	130	141	154	165	160	156	151
190	188	188	175	158	139	114	103	113	126	112	113	127	133	137	151	165	156	152	145
191	185	189	177	158	138	110	99	112	119	107	115	137	140	135	144	157	163	158	150
193	183	178	164	148	134	118	112	119	117	118	106	122	139	140	152	154	160	155	147
185	181	178	165	149	135	121	116	124	120	122	109	123	139	141	154	156	159	154	147
175	176	176	163	145	131	120	118	125	123	125	112	124	139	142	155	158	158	155	148
170	170	172	159	137	123	116	114	119	122	126	113	123	137	141	156	158	159	157	150
171	171	173	157	131	119	116	113	114	118	125	113	122	135	140	155	156	160	160	152
174	175	176	156	128	120	121	118	113	112	123	114	122	135	141	155	155	158	159	152
176	174	174	151	123	119	126	121	112	108	122	115	123	137	143	156	155	152	155	150
175	169	168	144	117	117	127	122	109	106	122	116	125	139	145	158	156	147	152	148
179	179	180	155	127	121	118	109	107	113	125	133	130	129	139	153	161	148	155	157
176	183	181	153	122	115	113	106	105	109	123	132	131	131	140	151	157	149	156	159
180	181	177	147	115	110	111	107	107	105	120	132	133	133	141	150	154	148	155	157
181	174	170	141	113	111	115	112	113	105	119	130	132	134	144	153	156	148	152	151
180	172	168	140	114	114	118	113	112	107	119	128	130	134	146	157	162	153	153	148
186	176	171	142	114	114	116	110	108	104	116	125	128	134	148	161	165	159	157	149
185	178	171	138	109	110	114	110	109	97	110	121	127	136	150	160	163	158	156	150

Table 3.1. The image I represented as a two-dimensional matrix of integers (subsampled).

better suited for interpretation by the human visual system is obtained by generating a *picture*. A picture can be thought of as a scene different from the true one that produces on the imaging sensor (the eye in this case) the same image as the true one. In this sense *pictures* are "controlled illusions": they are scenes different from the true ones (they are *flat*) that produce in the eye the same image as the original scenes. A picture of the same image I described in Figure 3.2 and Table 3.1 is shown in Figure 3.3. Although the latter seems more *informative* as to the content of the scene, it is merely a different representation and contains exactly the same information.

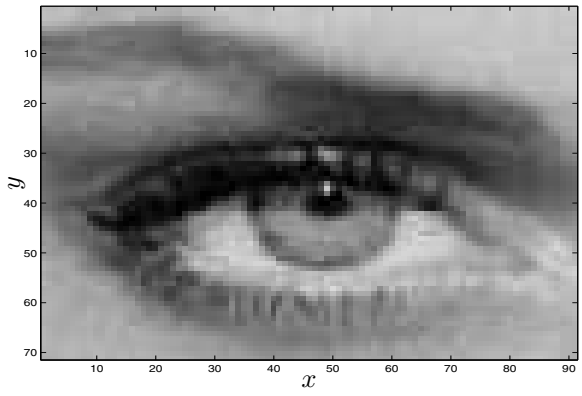

Figure 3.3. A "picture" of the image I (compare with Figure 3.2 and Table 3.1).

3.2 Lenses, light, and basic photometry

In order to describe the image formation process, we must specify the value of $I(x, y)$ at each point (x, y) in Ω. Such a value $I(x, y)$ is typically called *image*

intensity or *brightness*, or more formally *irradiance*. It has the units of power per unit area (W/m^2) and describes the energy falling onto a small patch of the imaging sensor. The irradiance at a point of coordinates (x, y) is obtained by integrating energy both in time (e.g., the shutter interval in a camera, or the integration time in a CCD array) and in a region of space. The region of space that contributes to the irradiance at (x, y) depends upon the shape of the object (surface) of interest, the optics of the imaging device, and it is by no means trivial to determine. In Appendix 3.A at the end of this chapter, we discuss some common simplifying assumptions to approximate it.

3.2.1 Imaging through lenses

A camera (or in general an optical system) is composed of a set of lenses used to "direct" light. By directing light we mean a controlled change in the direction of propagation, which can be performed by means of diffraction, refraction, and reflection. For the sake of simplicity, we neglect the effects of diffraction and reflection in a lens system, and we consider only refraction. Even so, a complete description of the functioning of a (purely refractive) lens is well beyond the scope of this book. Therefore, we will consider only the simplest possible model, that of a *thin lens*. For a more germane model of light propagation, the interested reader is referred to the classic textbook [Born and Wolf, 1999].

A *thin lens* (Figure 3.4) is a mathematical model defined by an axis, called the *optical axis*, and a plane perpendicular to the axis, called the *focal plane*, with a circular aperture centered at the *optical center*, i.e. the intersection of the focal plane with the optical axis. The thin lens has two parameters: its *focal length* f and its *diameter* d. Its function is characterized by two properties. The first property is that all rays entering the aperture parallel to the optical axis intersect on the optical axis at a distance f from the optical center. The point of intersection is called the *focus* of the lens (Figure 3.4). The second property is that all rays through the optical center are undeflected. Consider a point $p \in \mathbb{E}^3$ not too far from the optical axis at a distance Z along the optical axis from the optical center. Now draw two rays from the point p: one parallel to the optical axis, and one through the optical center (Figure 3.4). The first one intersects the optical axis at the focus; the second remains undeflected (by the defining properties of the thin lens). Call $\boldsymbol{x}$ the point where the two rays intersect, and let z be its distance from the optical center. By decomposing any other ray from p into a component ray parallel to the optical axis and one through the optical center, we can argue that all rays from p intersect at $\boldsymbol{x}$ on the opposite side of the lens. In particular, a ray from $\boldsymbol{x}$ parallel to the optical axis, must go through p. Using similar triangles, from Figure 3.4, we obtain the following *fundamental equation of the thin lens*:

$$\boxed{\frac{1}{Z} + \frac{1}{z} = \frac{1}{f}.}$$

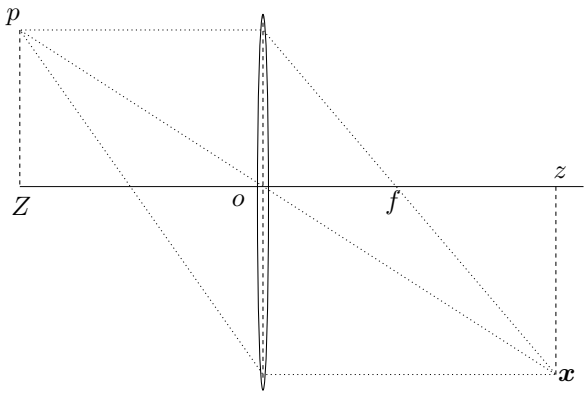

Figure 3.4. The image of the point p is the point $\boldsymbol{x}$ at the intersection of rays going parallel to the optical axis and the ray through the optical center.

The point $\boldsymbol{x}$ will be called the *image*[3] of the point p. Therefore, under the assumption of a thin lens, the irradiance $I(\boldsymbol{x})$ at the point $\boldsymbol{x}$ with coordinates (x, y) on the image plane is obtained by integrating all the energy emitted from the region of space contained in the cone determined by the geometry of the lens, as we describe in Appendix 3.A.

3.2.2 Imaging through a pinhole

If we let the aperture of a thin lens decrease to zero, all rays are forced to go through the optical center o, and therefore they remain undeflected. Consequently, the aperture of the cone decreases to zero, and the only points that contribute to the irradiance at the image point $\boldsymbol{x} = [x, y]^T$ are on a line through the center o of the lens. If a point p has coordinates $\boldsymbol{X} = [X, Y, Z]^T$ relative to a reference frame centered at the optical center o, with its z-axis being the optical axis (of the lens), then it is immediate to see from similar triangles in Figure 3.5 that the coordinates of p and its image $\boldsymbol{x}$ are related by the so-called ideal *perspective projection*

$$x = -f\frac{X}{Z}, \quad y = -f\frac{Y}{Z}, \tag{3.2}$$

where f is referred to as the *focal length*. Sometimes, we simply write the projection as a map π:

$$\pi : \mathbb{R}^3 \to \mathbb{R}^2; \quad \boldsymbol{X} \mapsto \boldsymbol{x}. \tag{3.3}$$

We also often write $\boldsymbol{x} = \pi(\boldsymbol{X})$. Note that any other point on the line through o and p projects onto the same coordinates $\boldsymbol{x} = [x, y]^T$. This imaging model is called an *ideal pinhole camera* model. It is an idealization of the thin lens model, since

[3]Here the word "image" is to be distinguished from the irradiance image $I(\boldsymbol{x})$ introduced before. Whether "image" indicates $\boldsymbol{x}$ or $I(\boldsymbol{x})$ will be made clear by the context.

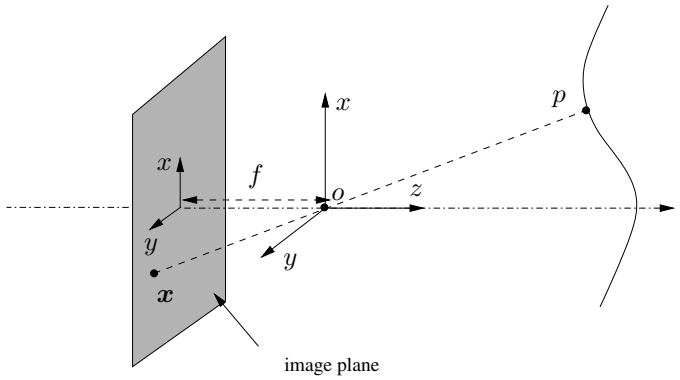

Figure 3.5. Pinhole imaging model: The image of the point p is the point $\boldsymbol{x}$ at the intersection of the ray going through the optical center o and an image plane at a distance f away from the optical center.

when the aperture decreases, diffraction effects become dominant, and therefore the (purely refractive) thin lens model does not hold [Born and Wolf, 1999]. Furthermore, as the aperture decreases to zero, the energy going through the lens also becomes zero. Although it is possible to actually build devices that approximate pinhole cameras, from our perspective the pinhole model will be just a good geometric approximation of a well-focused imaging system.

Notice that there is a negative sign in each of the formulae (3.2). This makes the image of an object appear to be upside down on the image plane (or the retina). To eliminate this effect, we can simply flip the image: $(x, y) \mapsto (-x, -y)$. This corresponds to placing the image plane $\{z = -f\}$ in front of the optical center instead $\{z = +f\}$. In this book we will adopt this more convenient "frontal" pinhole camera model, illustrated in Figure 3.6. In this case, the image $\boldsymbol{x} = [x, y]^T$ of the point p is given by

$$\boxed{x = f\frac{X}{Z}, \quad y = f\frac{Y}{Z}.} \tag{3.4}$$

We often use the same symbol, $\boldsymbol{x}$, to denote the homogeneous representation $[fX/Z, fY/Z, 1]^T \in \mathbb{R}^3$, as long as the dimension is clear from the context.[4]

In practice, the size of the image plane is usually limited, hence not every point p in space will generate an image $\boldsymbol{x}$ inside the image plane. We define the *field of view* (FOV) to be the angle subtended by the spatial extent of the sensor as seen from the optical center. If $2r$ is the largest spatial extension of the sensor (e.g., the

[4]In the homogeneous representation, it is only the direction of the vector $\boldsymbol{x}$ that is important. It is not crucial to normalize the last entry to 1 (see Appendix 3.B). In fact, $\boldsymbol{x}$ can be represented by $\lambda \boldsymbol{X}$ for any nonzero $\lambda \in \mathbb{R}$ as long as we remember that any such vector uniquely determines the intersection of the image ray and the actual image plane, in this case $\{Z = f\}$.

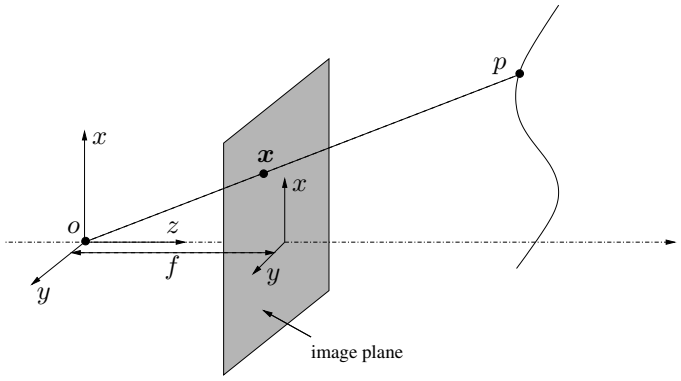

Figure 3.6. Frontal pinhole imaging model: the image of a 3-D point p is the point $\boldsymbol{x}$ at the intersection of the ray going through the optical center o and the image plane at a distance f in front of the optical center.

side of the CCD), then the field of view is $\theta = 2\arctan(r/f)$. Notice that if a flat plane is used as the image plane, the angle θ is always less than 180°.[5]

In Appendix 3.A we give a concise description of a simplified model to determine the intensity value of the image at the position $\boldsymbol{x}$, $I(\boldsymbol{x})$. This depends upon the ambient light distribution, the material properties of the visible surfaces and, their geometry. There we also show under what conditions this model can be reduced to a purely geometric one, where the intensity measured at a pixel is identical to the amount of energy radiated at the corresponding point in space, independent of the vantage point, e.g., a Lambertian surface. Under these conditions, the image formation process can be reduced to tracing rays from surfaces in space to points on the image plane. How to do so is explained in the next section.

3.3 A geometric model of image formation

As we have mentioned in the previous section and we elaborate further in Appendix 3.A, under the assumptions of a pinhole camera model and Lambertian surfaces, one can essentially reduce the process of image formation to tracing rays from points on objects to pixels. That is, knowing which point in space projects onto which point on the image plane allows one to directly associate the radiance at the point to the irradiance of its image; see equation (3.36) in Appendix 3.A. In order to establish a precise correspondence between points in 3-D space (with respect to a fixed global reference frame) and their projected images in a 2-D image plane (with respect to a local coordinate frame), a mathematical model for this process must account for three types of transformations:

[5] In case of a spherical or ellipsoidal imaging surface, common in omnidirectional cameras, the field of view can often exceed 180°.

1. coordinate transformations between the camera frame and the world frame;
2. projection of 3-D coordinates onto 2-D image coordinates;
3. coordinate transformation between possible choices of image coordinate frame.

In this section we will describe such a (simplified) image formation process as a series of transformations of coordinates. Inverting such a chain of transformations is generally referred to as "camera calibration," which is the subject of Chapter 6 and also a key step to 3-D reconstruction.

3.3.1 An ideal perspective camera

Let us consider a generic point p, with coordinates $\boldsymbol{X}_0 = [X_0, Y_0, Z_0]^T \in \mathbb{R}^3$ relative to the world reference frame.[6] As we know from Chapter 2, the coordinates $\boldsymbol{X} = [X, Y, Z]^T$ of the same point p relative to the camera frame are given by a rigid-body transformation $g = (R, T)$ of $\boldsymbol{X}_0$:

$$\boldsymbol{X} = R\boldsymbol{X}_0 + T \quad \in \mathbb{R}^3.$$

Adopting the frontal pinhole camera model introduced in the previous section (Figure 3.6), we see that the point $\boldsymbol{X}$ is projected onto the image plane at the point

$$\boldsymbol{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix}.$$

In homogeneous coordinates, this relationship can be written as

$$Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{3.5}$$

We can rewrite the above equation equivalently as

$$Z\boldsymbol{x} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \boldsymbol{X}, \tag{3.6}$$

where $\boldsymbol{X} \doteq [X, Y, Z, 1]^T$ and $\boldsymbol{x} \doteq [x, y, 1]^T$ are now in homogeneous representation. Since the coordinate Z (or the depth of the point p) is usually unknown, we may simply write it as an arbitrary positive scalar $\lambda \in \mathbb{R}_+$. Notice that in the

[6] We often indicate with $\boldsymbol{X}_0$ the coordinates of the point relative to the initial position of a moving camera frame.

above equation we can decompose the matrix into

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

Define two matrices

$$K_f \doteq \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}, \quad \Pi_0 \doteq \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \in \mathbb{R}^{3\times 4}. \tag{3.7}$$

The matrix Π_0 is often referred to as the *standard (or "canonical") projection matrix*. From the coordinate transformation we have for $\boldsymbol{X} = [X, Y, Z, 1]^T$,

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix}. \tag{3.8}$$

To summarize, using the above notation, the overall geometric model for *an ideal camera* can be described as

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix},$$

or in matrix form,

$$\lambda \boldsymbol{x} = K_f \Pi_0 \boldsymbol{X} = K_f \Pi_0 g \boldsymbol{X}_0. \tag{3.9}$$

If the focal length f is known and hence can be normalized to 1, this model reduces to a Euclidean transformation g followed by a standard projection Π_0, i.e.

$$\boxed{\lambda \boldsymbol{x} = \Pi_0 \boldsymbol{X} = \Pi_0 g \boldsymbol{X}_0.} \tag{3.10}$$

3.3.2 Camera with intrinsic parameters

The ideal model of equation (3.9) is specified relative to a very particular choice of reference frame, the "canonical retinal frame," centered at the optical center with one axis aligned with the optical axis. In practice, when one captures images with a digital camera the measurements are obtained in terms of pixels (i, j), with the origin of the image coordinate frame typically in the upper-left corner of the image. In order to render the model (3.9) usable, we need to specify the relationship between the retinal plane coordinate frame and the pixel array.

The first step consists in specifying the units along the x- and y-axes: if (x, y) are specified in terms of metric units (e.g., millimeters), and (x_s, y_s) are scaled

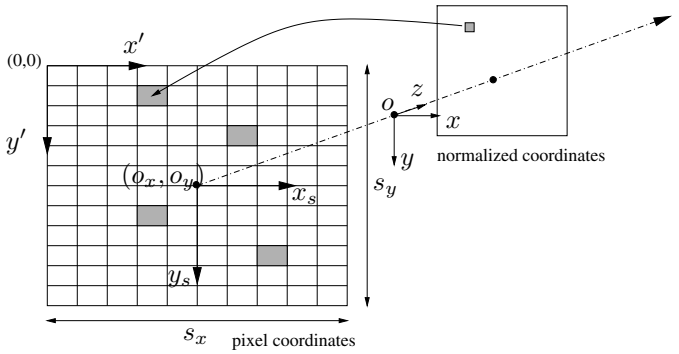

Figure 3.7. Transformation from normalized coordinates to coordinates in pixels.

versions that correspond to coordinates of the pixel, then the transformation can be described by a scaling matrix

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.11}$$

that depends on the size of the pixel (in metric units) along the x and y directions (Figure 3.7). When $s_x = s_y$, each pixel is square. In general, they can be different, and then the pixel is rectangular. However, here x_s and y_s are still specified relative to the *principal point* (where the z-axis intersects the image plane), whereas the pixel index (i, j) is conventionally specified relative to the upper-left corner, and is indicated by positive numbers. Therefore, we need to translate the origin of the reference frame to this corner (as shown in Figure 3.7),

$$\begin{aligned} x' &= x_s + o_x, \\ y' &= y_s + o_y, \end{aligned}$$

where (o_x, o_y) are the coordinates (in pixels) of the principal point relative to the image reference frame. So the actual image coordinates are given by the vector $\boldsymbol{x}' = [x', y', 1]^T$ instead of the ideal image coordinates $\boldsymbol{x} = [x, y, 1]^T$. The above steps of coordinate transformation can be written in the homogeneous representation as

$$\boldsymbol{x}' \doteq \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{3.12}$$

where x' and y' are actual image coordinates in pixels. This is illustrated in Figure 3.7. In case the pixels are not rectangular, a more general form of the scaling matrix can be considered,

$$\begin{bmatrix} s_x & s_\theta \\ 0 & s_y \end{bmatrix} \quad \in \mathbb{R}^{2\times 2},$$

where s_θ is called a *skew factor* and is proportional to $\cot(\theta)$, where θ is the angle between the image axes x_s and y_s.[7] The transformation matrix in (3.12) then takes the general form

$$K_s \doteq \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \in \mathbb{R}^{3\times 3}. \tag{3.13}$$

In many practical applications it is common to assume that $s_\theta = 0$.

Now, combining the projection model from the previous section with the scaling and translation yields a more realistic model of a transformation between homogeneous coordinates of a 3-D point relative to the camera frame and homogeneous coordinates of its image expressed in terms of pixels,

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}.$$

Notice that in the above equation, the effect of a real camera is in fact carried through two stages:

- The first stage is a standard perspective projection with respect to a *normalized coordinate system* (as if the focal length were $f = 1$). This is characterized by the standard projection matrix $\Pi_0 = [I, 0]$.
- The second stage is an additional transformation (on the obtained image $\boldsymbol{x}$) that depends on parameters of the camera such as the focal length f, the scaling factors s_x, s_y, and s_θ, and the center offsets o_x, o_y.

The second transformation is obviously characterized by the combination of the two matrices K_s and K_f:

$$K \doteq K_s K_f \doteq \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.14}$$

The coupling of K_s and K_f allows us to write the projection equation in the following way:

$$\lambda \boldsymbol{x}' = K\Pi_0 \boldsymbol{X} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{3.15}$$

The constant 3×4 matrix Π_0 represents the perspective projection. The upper triangular 3×3 matrix K collects all parameters that are "intrinsic" to a particular camera, and is therefore called the *intrinsic parameter matrix*, or the *calibration*

[7] Typically, the angle θ is very close to 90°, and hence s_θ is very close to zero.

matrix of the camera. The entries of the matrix K have the following geometric interpretation:

- o_x: x-coordinate of the principal point in pixels,
- o_y: y-coordinate of the principal point in pixels,
- $fs_x = \alpha_x$: size of unit length in horizontal pixels,
- $fs_y = \alpha_y$: size of unit length in vertical pixels,
- α_x/α_y: aspect ratio σ,
- fs_θ: skew of the pixel, often close to zero.

Note that the height of the pixel is not necessarily identical to its width unless the aspect ratio σ is equal to 1.

When the calibration matrix K is known, the *calibrated* coordinates $\boldsymbol{x}$ can be obtained from the pixel coordinates $\boldsymbol{x}'$ by a simple inversion of K:

$$\lambda \boldsymbol{x} = \lambda K^{-1}\boldsymbol{x}' = \Pi_0 \boldsymbol{X} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{3.16}$$

The information about the matrix K can be obtained through the process of camera calibration to be described in Chapter 6. With the effect of K compensated for, equation (3.16), expressed in the normalized coordinate system, corresponds to the ideal pinhole camera model with the image plane located in front of the center of projection and the focal length f equal to 1.

To summarize, the geometric relationship between a point of coordinates $\boldsymbol{X}_0 = [X_0, Y_0, Z_0, 1]^T$ relative to the world frame and its corresponding image coordinates $\boldsymbol{x}' = [x', y', 1]^T$ (in pixels) depends on the rigid-body motion (R, T) between the world frame and the camera frame (sometimes referred to as the *extrinsic calibration parameters*), an ideal projection Π_0, and the camera intrinsic parameters K. The overall model for image formation is therefore captured by the following equation:

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix}.$$

In matrix form, we write

$$\lambda \boldsymbol{x}' = K\Pi_0 \boldsymbol{X} = K\Pi_0 g \boldsymbol{X}_0, \tag{3.17}$$

or equivalently,

$$\lambda \boldsymbol{x}' = K\Pi_0 \boldsymbol{X} = [KR, KT]\boldsymbol{X}_0. \tag{3.18}$$

Often, for convenience, we call the 3×4 matrix $K\Pi_0 g = [KR, KT]$ a (general) *projection matrix* Π, to be distinguished from the standard projection matrix Π_0.

Hence, the above equation can be simply written as

$$\boxed{\lambda \boldsymbol{x}' = \Pi \boldsymbol{X}_0 = K \Pi_0 g \boldsymbol{X}_0.} \tag{3.19}$$

Compared to the ideal camera model (3.10), the only change here is the standard projection matrix Π_0 being replaced by a general one Π.

At this stage, in order to explicitly see the nonlinear nature of the perspective projection equation, we can divide equation (3.19) by the scale λ and obtain the following expressions for the image coordinates (x', y', z'),

$$x' = \frac{\pi_1^T \boldsymbol{X}_0}{\pi_3^T \boldsymbol{X}_0}, \quad y' = \frac{\pi_2^T \boldsymbol{X}_0}{\pi_3^T \boldsymbol{X}_0}, \quad z' = 1, \tag{3.20}$$

where $\pi_1^T, \pi_2^T, \pi_3^T \in \mathbb{R}^4$ are the three rows of the projection matrix Π.

Example 3.1 (Spherical perspective projection). The perspective pinhole camera model outlined above considers planar imaging surfaces. An alternative imaging surface that is also commonly used is that of a sphere, shown in Figure 3.8.

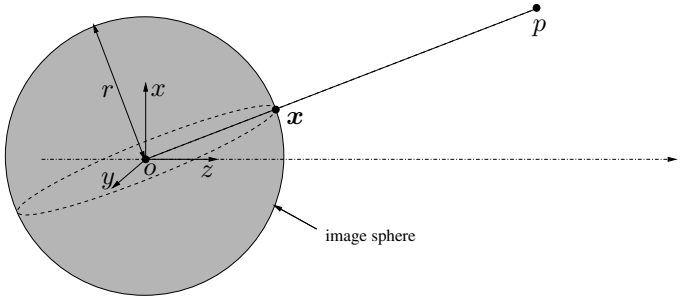

Figure 3.8. Spherical perspective projection model: the image of a 3-D point p is the point $\boldsymbol{x}$ at the intersection of the ray going through the optical center o and a sphere of radius r around the optical center. Typically r is chosen to be 1.

This choice is partly motivated by retina shapes often encountered in biological systems. For spherical projection, we simply choose the imaging surface to be the unit sphere $\mathbb{S}^2 = \{p \in \mathbb{R}^3 \mid \|\boldsymbol{X}(p)\| = 1\}$. Then, the spherical projection is defined by the map π_s from $\mathbb{R}^3$ to $\mathbb{S}^2$:

$$\pi_s : \mathbb{R}^3 \to \mathbb{S}^2; \quad \boldsymbol{X} \mapsto \boldsymbol{x} = \frac{\boldsymbol{X}}{\|\boldsymbol{X}\|}.$$

As in the case of planar perspective projection, the relationship between pixel coordinates of a point and their 3-D metric counterpart can be expressed as

$$\lambda \boldsymbol{x}' = K \Pi_0 \boldsymbol{X} = K \Pi_0 g \boldsymbol{X}_0, \tag{3.21}$$

where the scale is given by $\lambda = \sqrt{X^2 + Y^2 + Z^2}$ in the case of spherical projection while $\lambda = Z$ in the case of planar projection. Therefore, mathematically, spherical projection and planar projection can be described by the same set of equations. The only difference is that the unknown (depth) scale λ takes different values. ■

For convenience, we often write $\boldsymbol{x} \sim \boldsymbol{y}$ for two (homogeneous) vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ equal up to a scalar factor (see Appendix 3.B for more detail). From the above example, we see that for any perspective projection we have

$$\boldsymbol{x}' \sim \Pi \boldsymbol{X}_0 = K \Pi_0 g \boldsymbol{X}_0, \tag{3.22}$$

and the shape of the imaging surface chosen does not matter. The imaging surface can be any (regular) surface as long as any ray $\overrightarrow{op}$ intersects with the surface at one point at most. For example, an entire class of ellipsoidal surfaces can be used, which leads to the so-called *catadioptric model* popular in many omnidirectional cameras. In principle, all images thus obtained contain exactly the same information.

3.3.3 Radial distortion

In addition to linear distortions described by the parameters in K, if a camera with a wide field of view is used, one can often observe significant distortion along radial directions. The simplest effective model for such a distortion is:

$$\begin{aligned} x &= x_d(1 + a_1 r^2 + a_2 r^4), \\ y &= y_d(1 + a_1 r^2 + a_2 r^4), \end{aligned}$$

where (x_d, y_d) are coordinates of the distorted points, $r^2 = x_d^2 + y_d^2$ and a_1, a_2 are additional camera parameters that model the amount of distortion. Several algorithms and software packages are available for compensating radial distortion via calibration procedures. In particular, a commonly used approach is that of [Tsai, 1986a], if a calibration rig is available (see Chapter 6 for more details).

In case the calibration rig is not available, the radial distortion parameters can be estimated directly from images. A simple method suggested by [Devernay and Faugeras, 1995] assumes a more general model of radial distortion:

$$\begin{aligned} \boldsymbol{x} &= c + f(r)(\boldsymbol{x}_d - c), \\ f(r) &= 1 + a_1 r + a_2 r^2 + a_3 r^3 + a_4 r^4, \end{aligned}$$

where $\boldsymbol{x}_d = [x_d, y_d]^T$ are the distorted image coordinates, $r^2 = \|\boldsymbol{x}_d - c\|^2$, $c = [c_x, c_y]^T$ is the center of the distortion, not necessarily coincident with the center of the image, and $f(r)$ is the distortion correction factor. The method assumes a set of straight lines in the world and computes the best parameters of the radial distortion model which would transform the curved images of the lines into straight segments. One can use this model to transform Figure 3.9 (left) into 3.9 (right) via preprocessing algorithms described in [Devernay and Faugeras, 1995]. Therefore, in the rest of this book we assume that radial distortion has been compensated for, and a camera is described simply by the parameter matrix K. The interested reader may consult classical references such as [Tsai, 1986a, Tsai, 1987, Tsai, 1989, Zhang, 1998b], which are available as software packages. Some authors have shown that radial distortion

Figure 3.9. Left: image taken by a camera with a short focal length; note that the straight lines in the scene become curved on the image. Right: image with radial distortion compensated for.

can be recovered from multiple corresponding images: a simultaneous estimation of 3-D geometry and radial distortion can be found in the more recent work of [Zhang, 1996, Stein, 1997, Fitzgibbon, 2001]. For more sophisticated lens aberration models, the reader can refer to classical references in geometric optics given at the end of this chapter.

3.3.4 Image, preimage, and coimage of points and lines

The preceding sections have formally established the notion of a perspective image of a point. In principle, this allows us to define an image of any other geometric entity in 3-D that can be defined as a set of points (e.g., a line or a plane). Nevertheless, as we have seen from the example of spherical projection, even for a point, there exist seemingly different representations for its image: two vectors $\boldsymbol{x} \in \mathbb{R}^3$ and $\boldsymbol{y} \in \mathbb{R}^3$ may represent the same image point as long as they are related by a nonzero scalar factor; i.e. $\boldsymbol{x} \sim \boldsymbol{y}$ (as a result of different choices in the imaging surface). To avoid possible confusion that can be caused by such different representations for the same geometric entity, we introduce a few abstract notions related to the image of a point or a line.

Consider the perspective projection of a straight line L in 3-D onto the 2-D image plane (Figure 3.10). To specify a line in 3-D, we can typically specify a point p_o, called the base point, on the line and specify a vector v that indicates the direction of the line. Suppose that $\boldsymbol{X}_o = [X_o, Y_o, Z_o, 1]^T$ are the homogeneous coordinates of the base point p_o and $\boldsymbol{V} = [V_1, V_2, V_3, 0]^T \in \mathbb{R}^4$ is the homogeneous representation of v, relative to the camera coordinate frame. Then the (homogeneous) coordinates of any point on the line L can be expressed as

$$\boldsymbol{X} = \boldsymbol{X}_o + \mu \boldsymbol{V}, \quad \mu \in \mathbb{R}.$$

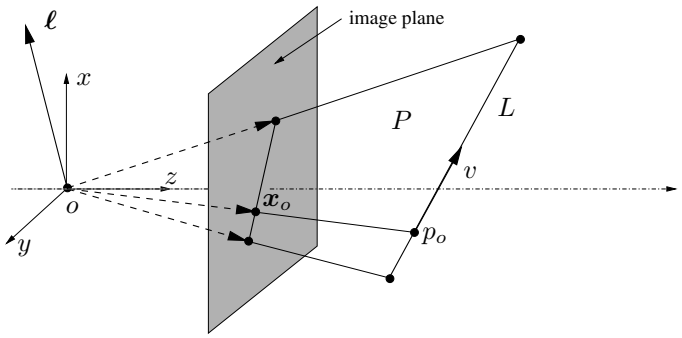

Figure 3.10. Perspective image of a line L in 3-D. The collection of images of points on the line forms a plane P. Intersection of this plane and the image plane gives a straight line ℓ which is the image of the line.

Then, the image of the line L is given by the collection of image points with homogeneous coordinates given by

$$\boldsymbol{x} \sim \Pi_0 \boldsymbol{X} = \Pi_0(\boldsymbol{X}_o + \mu \boldsymbol{V}) = \Pi_0 \boldsymbol{X}_o + \mu \Pi_0 \boldsymbol{V}.$$

It is easy to see that this collection of points $\{\boldsymbol{x}\}$, treated as vectors with origin at o, span a 2-D subspace P, shown in Figure 3.10. The intersection of this subspace with the image plane gives rise to a straight line in the 2-D image plane, also shown in Figure 3.10. This line is then the (physical) image of the line L.

Now the question is how to efficiently represent the image of the line. For this purpose, we first introduce the notion of preimage:

Definition 3.2 (Preimage). *A preimage of a point or a line in the image plane is the set of 3-D points that give rise to an image equal to the given point or line.*

Note that the given image is constrained to lie in the image plane, whereas the preimage lies in 3-D space. In the case of a point $\boldsymbol{x}$ on the image plane, its preimage is a one-dimensional subspace, spanned by the vector joining the point $\boldsymbol{x}$ to the camera center o. In the case of a line, the preimage is a plane P through o (hence a subspace) as shown in Figure 3.10, whose intersection with the image plane is exactly the given image line. Such a plane can be represented as the span of any two linearly independent vectors in the same subspace. Thus the preimage is really the largest set of 3-D points or lines that gives rise to the same image. The definition of a preimage can be given not only for points or lines in the image plane but also for curves or other more complicated geometric entities in the image plane as well. However, when the image is a point or a line, the preimage is a subspace, and we may also represent this subspace by its (unique) orthogonal complement in $\mathbb{R}^3$. For instance, a plane can be represented by its normal vector. This leads to the following notion of coimage:

Definition 3.3 (Coimage). *The coimage of a point or a line is defined to be the subspace in $\mathbb{R}^3$ that is the (unique) orthogonal complement of its preimage.*

The reader must be aware that the image, preimage, and coimage are *equivalent* representations, since they uniquely determine one another:

$$\text{image} = \text{preimage} \cap \text{image plane}, \quad \text{preimage} = \text{span}(\text{image}),$$
$$\text{preimage} = \text{coimage}^{\perp}, \quad \text{coimage} = \text{preimage}^{\perp}.$$

Since the preimage of a line L is a two-dimensional subspace, its coimage is represented as the span of the normal vector to the subspace. The notation we use for this is $\boldsymbol{\ell} = [a, b, c]^T \in \mathbb{R}^3$ (Figure 3.10). If $\boldsymbol{x}$ is the image of a point p on this line, then it satisfies the orthogonality equation

$$\boldsymbol{\ell}^T \boldsymbol{x} = 0. \tag{3.23}$$

Recall that we use $\widehat{u} \in \mathbb{R}^{3\times 3}$ to denote the skew-symmetric matrix associated with a vector $u \in \mathbb{R}^3$. Its column vectors span the subspace orthogonal to the vector u. Thus the column vectors of the matrix $\widehat{\boldsymbol{\ell}}$ span the plane that is *orthogonal* to $\boldsymbol{\ell}$; i.e. they span the preimage of the line L. In Figure 3.10, this means that $P = \text{span}(\widehat{\boldsymbol{\ell}})$. Similarly, if $\boldsymbol{x}$ is the image of a point p, its coimage is the plane orthogonal to $\boldsymbol{x}$ given by the span of the column vectors of the matrix $\widehat{\boldsymbol{x}}$. Thus, in principle, we should use the notation in Table 3.2 to represent the image, preimage, or coimage of a point and a line.

Notation	Image	Preimage	Coimage
Point	$\text{span}(\boldsymbol{x}) \cap$ image plane	$\text{span}(\boldsymbol{x}) \subset \mathbb{R}^3$	$\text{span}(\widehat{\boldsymbol{x}}) \subset \mathbb{R}^3$
Line	$\text{span}(\widehat{\boldsymbol{\ell}}) \cap$ image plane	$\text{span}(\widehat{\boldsymbol{\ell}}) \subset \mathbb{R}^3$	$\text{span}(\boldsymbol{\ell}) \subset \mathbb{R}^3$

Table 3.2. The image, preimage, and coimage of a point and a line.

Although the (physical) image of a point or a line, strictly speaking, is a notion that depends on a particular choice of imaging surface, mathematically it is more convenient to use its preimage or coimage to represent it. For instance, we will use the vector $\boldsymbol{x}$, defined up to a scalar factor, to represent the preimage (hence the image) of a point; and the vector $\boldsymbol{\ell}$, defined up to a scalar factor, to represent the coimage (hence the image) of a line. The relationships between preimage and coimage of points and lines can be expressed in terms of the vectors $\boldsymbol{x}, \boldsymbol{\ell} \in \mathbb{R}^3$ as

$$\widehat{\boldsymbol{x}}\boldsymbol{x} = 0, \quad \widehat{\boldsymbol{\ell}}\boldsymbol{\ell} = 0.$$

Often, for a simpler language, we may refer to either the preimage or coimage of points and lines as the "image" if its actual meaning is clear from the context. For instance, in Figure 3.10, we will, in future chapters, often mark in the image plane the image of the line L by the same symbol $\boldsymbol{\ell}$ as the vector typically used to denote its coimage.

3.4 Summary

In this chapter, perspective projection is introduced as a model of the image formation for a pinhole camera. In the ideal case (e.g., when the calibration matrix K is the identity), homogeneous coordinates of an image point are related to their 3-D counterparts by an unknown (depth) scale λ,

$$\lambda \boldsymbol{x} = \Pi_0 \boldsymbol{X} = \Pi_0 g \boldsymbol{X}_0.$$

If K is not the identity, the standard perspective projection is augmented by an additional linear transformation K on the image plane

$$\boldsymbol{x}' = K\boldsymbol{x}.$$

This yields the following relationship between coordinates of an (uncalibrated) image and their 3-D counterparts:

$$\lambda \boldsymbol{x}' = K\Pi_0 \boldsymbol{X} = K\Pi_0 g \boldsymbol{X}_0.$$

As equivalent representations for an image of a point or a line, we introduced the notions of image, preimage, and coimage, whose relationships were summarized in Table 3.2.

3.5 Exercises

Exercise 3.1 Show that any point on the line through o and p projects onto the same image coordinates as p.

Exercise 3.2 Consider a thin lens imaging a plane parallel to the lens at a distance z from the focal plane. Determine the region of this plane that contributes to the image I at the point $\boldsymbol{x}$. (Hint: consider first a one-dimensional imaging model, then extend to a two-dimensional image.)

Exercise 3.3 (Field of view). An important parameter of the imaging system is the *field of view* (FOV). The field of view is twice the angle between the optical axis (z-axis) and the end of the retinal plane (CCD array). Imagine having a camera system with focal length 24 mm, and retinal plane (CCD array) (16 mm $\times$ 12 mm) and that your digitizer samples your imaging surface at 500×500 pixels in the horizontal and vertical directions.

1. Compute the FOV.
2. Write down the relationship between the image coordinate and a point in 3-D space expressed in the camera coordinate system.
3. Describe how the size of the FOV is related to the focal length and how it affects the resolution in the image.
4. Write a software program (in Matlab) that simulates the geometry of the projection process; given the coordinates of an object with respect to the calibrated camera frame, create an image of that object. Experiment with changing the parameters of the imaging system.

Exercise 3.4 Under the standard perspective projection (i.e. $K = I$):

1. What is the image of a sphere?
2. Characterize the objects for which the image of the centroid is the centroid of the image.

Exercise 3.5 (Calibration matrix). Compute the calibration matrix K that represents the transformation from image I to I' as shown in Figure 3.11. Note that from the definition of the calibration matrix, you need to use homogeneous coordinates to represent image points. Suppose that the resulting image I' is further digitized into an array of 640×480 pixels and the intensity value of each pixel is quantized to an integer in $[0, 255]$. Then how many *different* digitized images can one possibly get from such a process?

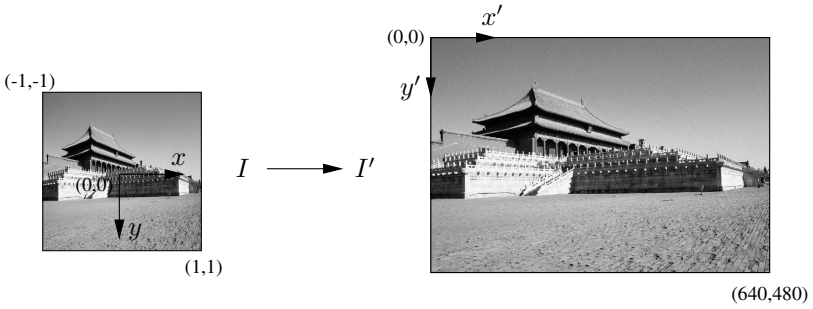

Figure 3.11. Transformation of a normalized image into pixel coordinates.

Exercise 3.6 (Image cropping). In this exercise, we examine the effect of cropping an image from a change of coordinate viewpoint. Compute the coordinate transformation between pixels (of same points) between the two images in Figure 3.12. Represent this transformation in homogeneous coordinates.

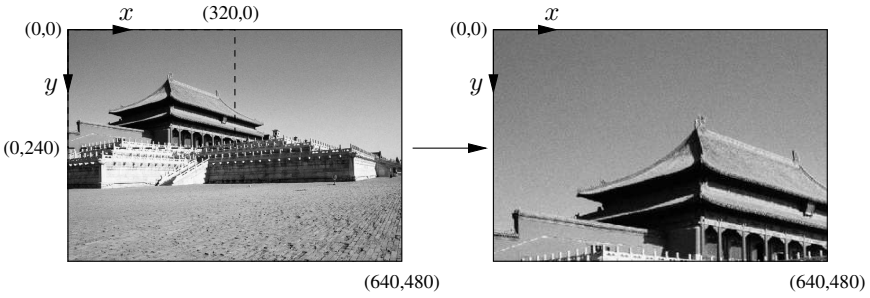

Figure 3.12. An image of size 640×480 pixels is cropped by half and then the resulting image is up-sampled and restored as a 640×480-pixel image.

Exercise 3.7 (Approximate camera models). The most commonly used approximation to the perspective projection model is *orthographic projection*. The light rays in the orthographic model travel along lines parallel to the optical axis. The relationship between

image points and 3-D points in this case is particularly simple: $x = X; y = Y$. So, the geometric model for orthographic projection can be expressed as

$$\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \tag{3.24}$$

or simply in matrix form

$$\boldsymbol{x} = \Pi_o \boldsymbol{X}, \tag{3.25}$$

where $\Pi_o \doteq [I_{2\times 2}, 0] \in \mathbb{R}^{2\times 3}$. A scaled version of the orthographic model leads to the so-called *weak-perspective* model

$$\boldsymbol{x} = s\Pi_o \boldsymbol{X}, \tag{3.26}$$

where s is a constant scalar independent of the point $\boldsymbol{x}$. Show how the (scaled) orthographic projection approximates perspective projection when the scene occupies a volume whose diameter (or depth variation of the scene) is small compared to its distance from the camera. Characterize at least one more condition under which the two projection models produce similar results (equal in the limit).

Exercise 3.8 (Scale ambiguity). It is common sense that with a perspective camera, one cannot tell an object from another object that is exactly *twice as big but twice as far*. This is a classic ambiguity introduced by the perspective projection. Use the ideal camera model to explain why this is true. Is the same also true for the orthographic projection? Explain.

Exercise 3.9 (Image of lines and their intersection). Consider the image of a line L (Figure 3.10).

1. Show that there exists a vector in $\mathbb{R}^3$, call it $\boldsymbol{\ell}$, such that
$$\boldsymbol{\ell}^T \boldsymbol{x} = 0$$
for the image $\boldsymbol{x}$ of every point on the line L. What is the geometric meaning of the vector $\boldsymbol{\ell}$? (Note that the vector $\boldsymbol{\ell}$ is defined only up to an arbitrary scalar factor.)
2. If the images of two points on the line L are given, say $\boldsymbol{x}^1, \boldsymbol{x}^2$, express the vector $\boldsymbol{\ell}$ in terms of $\boldsymbol{x}^1$ and $\boldsymbol{x}^2$.
3. Now suppose you are given two images of two lines, in the above vector form $\boldsymbol{\ell}^1, \boldsymbol{\ell}^2$. If $\boldsymbol{x}$ is the intersection of these two image lines, express $\boldsymbol{x}$ in terms of $\boldsymbol{\ell}^1, \boldsymbol{\ell}^2$.

Exercise 3.10 (Vanishing points). A straight line on the 3-D world is projected onto a straight line in the image plane. The projections of two parallel lines intersect in the image plane at the *vanishing point*.

1. Show that projections of parallel lines in 3-D space intersect at a point on the image.
2. Compute, for a given family of parallel lines, where in the image the vanishing point will be.
3. When does the vanishing point of the lines in the image plane lie at infinity (i.e. they do not intersect)?

The reader may refer to Appendix 3.B for a more formal treatment of vanishing points as well as their mathematical interpretation.

3.A Basic photometry with light sources and surfaces

In this section we give a concise description of a basic radiometric image formation model, and show that some simplifications are necessary in order to reduce the model to a purely geometric one, as described in this chapter. The idea is to describe how the intensity at a pixel on the image is generated. Under suitable assumptions, we show that such intensity depends only on the amount of energy radiated from visible surfaces in space and not on the vantage point.

Let S be a smooth visible surface in space; we denote the tangent plane to the surface at a point p by T_pS and its outward unit normal vector by ν_p. At each point $p \in S$ we can construct a local coordinate frame with its origin at p, its z-axis parallel to the normal vector ν_p, and its xy-plane parallel to T_pS (see Figure 3.13). Let L be a smooth surface that is irradiating light, which we call the *light source*. For simplicity, we may assume that L is the only source of light in space. At a point $q \in L$, we denote with T_qS and ν_q the tangent plane and the outward unit normal of L, respectively, as shown in Figure 3.13.

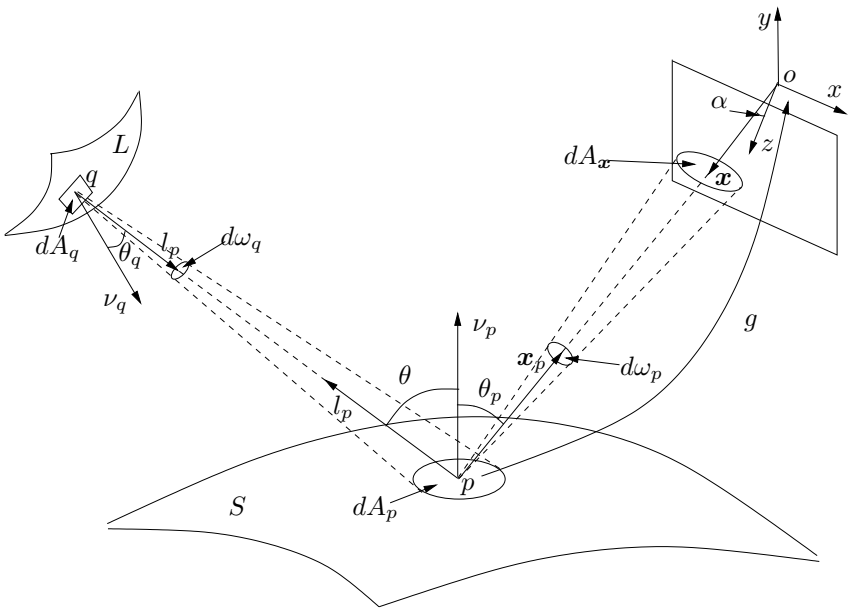

Figure 3.13. Generative model.

The change of coordinates between the local coordinate frame at p and the camera frame, which we assume coincides with the world frame, is indicated by a rigid-body transformation g; then g maps coordinates in the local coordinate

frame at p into those in the camera frame, and any vector u in the local coordinate frame to a vector $v = g_*(u)$ in the camera frame.[8]

Foreshortening and solid angle

When considering interactions between a light source and a surface, we need to introduce the notion of *foreshortening* and that of *solid angle*. Foreshortening encodes how the light distribution on a surface changes as we change the surface orientation with respect to the source of illumination. In formulas, if dA_p is the area element in T_pS, and l_p is the unit vector that indicates the direction from p to q (see Figure 3.13), then the corresponding foreshortened area as seen from q is

$$\cos(\theta) dA_p,$$

where θ is the angle between the direction l_p and the normal vector ν_p; i.e. $\cos(\theta) = \langle \nu_p, l_p \rangle$. A solid angle is defined to be the area of a cone cut out on a unit sphere. Then, the infinitesimal solid angle $d\omega_q$ seen from a point q of the infinitesimal area dA_p is

$$d\omega_q \doteq \frac{\cos(\theta) dA_p}{d(p,q)^2}, \tag{3.27}$$

where $d(p,q)$ is the distance between p and q.

Radiance and irradiance

In radiometry, *radiance* is defined to be the amount of energy emitted along a certain direction, per unit area perpendicular to the direction of emission (the foreshortening effect), per unit of solid angle, and per unit of time, following the definition in [Sillion, 1994]. According to our notation, if we denote the radiance at the point q in the direction of p by $\mathcal{R}(q, l_p)$, the energy emitted by the light L at a point q toward p on S is

$$dE(p, l_p) \doteq \mathcal{R}(q, l_p) \cos(\theta_q)\ dA_q\ d\omega_q\ dt, \tag{3.28}$$

where $\cos(\theta_q)\ dA_q$ is the foreshortened area of dA_q seen from the direction of p, and $d\omega_q$ is the solid angle given in equation (3.27), as shown in Figure 3.13. Notice that the point p on the left hand side of the equation above and the point q on the right hand side are related by the direction l_p of the vector connecting p to q.

While the radiance is used for energy that is emitted, the quantity that describes incoming energy is called *irradiance*. The irradiance is defined as the amount of energy received along a certain direction, per unit area and per unit time. Notice that in the case of the irradiance, we *do not* foreshorten the surface area as in the case of the radiance. Denote the irradiance at p received in the direction l_p by

[8] We recall from the previous chapter that if we represent the change of coordinates g with a rotation matrix $R \in SO(3)$ and a translation vector T, then the action of g on a point p of coordinates $\boldsymbol{X} \in \mathbb{R}^3$ is given by $g(\boldsymbol{X}) \doteq R\boldsymbol{X} + T$, while the action of g on a *vector* of coordinates u is given by $g_*(u) \doteq Ru$.

$dI(p, l_p)$. By energy preservation, we have $dI(p, l_p)\ dA_p\ dt = dE(p, l_p)$. Then the radiance $\mathcal{R}$ at a point q that illuminates the surface dA_p along the direction l_p with a solid angle $d\omega$ and the irradiance dI measured at the same surface dA_p received from this direction are related by

$$dI(p, l_p) = \mathcal{R}(q, l_p) \cos(\theta)\ d\omega, \tag{3.29}$$

where $d\omega = \frac{\cos(\theta_q)}{d(p,q)^2} dA_q$ is the solid angle of dA_q seen from p.

Bidirectional reflectance distribution function

For many common materials, the portion of energy coming from a direction l_p that is reflected onto a direction $\boldsymbol{x}_p$ (i.e. the direction of the vantage point) by the surface S, is described by $\beta(\boldsymbol{x}_p, l_p)$, the *bidirectional reflectance distribution function* (BRDF). Here both $\boldsymbol{x}_p$ and l_p are vectors expressed in local coordinates at p. More precisely, if $d\mathcal{R}(p, \boldsymbol{x}_p, l_p)$ is the radiance emitted in the direction $\boldsymbol{x}_p$ from the irradiance $dI(p, l_p)$, the BRDF is given by the ratio

$$\beta(\boldsymbol{x}_p, l_p) \doteq \frac{d\mathcal{R}(p, \boldsymbol{x}_p, l_p)}{dI(p, l_p)} = \frac{d\mathcal{R}(p, \boldsymbol{x}_p, l_p)}{\mathcal{R}(q, l_p) \cos(\theta)\ d\omega}. \tag{3.30}$$

To obtain the total radiance at a point p in the outgoing direction $\boldsymbol{x}_p$, we need to integrate the BRDF against all the incoming irradiance directions l_p in the hemisphere Ω at p:

$$\mathcal{R}(p, \boldsymbol{x}_p) = \int_\Omega d\mathcal{R}(p, \boldsymbol{x}_p, l_p) = \int_\Omega \beta(\boldsymbol{x}_p, l_p)\ \mathcal{R}(q, l_p) \cos(\theta)\ d\omega. \tag{3.31}$$

Lambertian surfaces

The above model can be considerably simplified if we restrict our attention to a class of materials, called *Lambertian*, that do not change appearance depending on the viewing direction. For example, matte surfaces are to a large extent well approximated by the Lambertian model, since they diffuse light almost uniformly in all directions. Metal, mirrors, and other shiny surfaces, however, do not. Figure 3.14 illustrates a few common surface properties.

For a perfect Lambertian surface, its radiance $\mathcal{R}(p, \boldsymbol{x}_p)$ only depends on how the surface faces the light source, but not on the direction $\boldsymbol{x}_p$ from which it is viewed. Therefore, $\beta(\boldsymbol{x}_p, l_p)$ is actually independent of $\boldsymbol{x}_p$, and we can think of the radiance function as being "glued," or "painted" on the surface S, so that at each point p the radiance $\mathcal{R}$ depends only on the surface. Hence, the perceived irradiance will depend only on which point on the surface is seen, not on in which direction it is seen. More precisely, for Lambertian surfaces, we have

$$\beta(\boldsymbol{x}_p, l_p) = \rho(p),$$

where $\rho(p) : \mathbb{R}^3 \mapsto \mathbb{R}_+$ is a scalar function. In this case, we can easily compute the *surface albedo* ρ_a, which is the percentage of incident irradiance reflected in

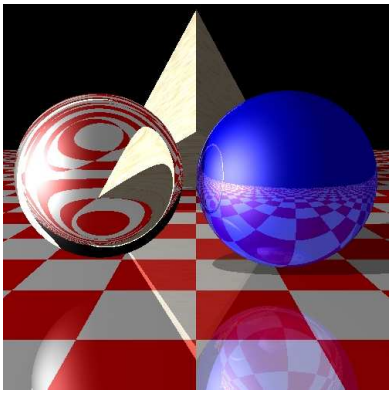

Figure 3.14. This figure demonstrates different surface properties widely used in computer graphics to model surfaces of natural objects: Lambertian, diffuse, reflective, specular (highlight), transparent with refraction, and textured. Only the (wood textured) pyramid exhibits Lambertian reflection. The ball on the right is partly ambient, diffuse, reflective and specular. The checkerboard floor is partly ambient, diffuse and reflective. The glass ball on the left is both reflective and refractive.

any direction, as

$$\begin{aligned} \rho_a(p) &= \int_\Omega \beta(\boldsymbol{x}_p, l_p)\cos(\theta_p)\, d\omega_p = \rho(p)\int_0^{2\pi}\int_0^{\frac{\pi}{2}} \cos(\theta_p)\sin(\theta_p)\, d\theta_p\, d\phi_p \\ &= \pi\rho(p), \end{aligned}$$

where $d\omega_p$, as shown in Figure 3.13, is the infinitesimal solid angle in the outgoing direction, which can be parameterized by the space angles (θ_p, ϕ_p) as $d\omega_p = \sin(\theta_p)d\theta_p d\phi_p$. Hence the radiance from the point p on a Lambertian surface S is

$$\mathcal{R}(p) = \int_\Omega \frac{1}{\pi}\rho_a(p)\, \mathcal{R}(q, l_p)\cos(\theta)\, d\omega. \tag{3.32}$$

This equation is known as *Lambertian cosine law*. Therefore, for a Lambertian surface, the radiance $\mathcal{R}$ depends only on the surface S, described by its generic point p, and on the light source L, described by its radiance $\mathcal{R}(q, l_p)$.

Image intensity for a Lambertian surface

In order to express the direction $\boldsymbol{x}_p$ in the camera frame, we consider the change of coordinates from the local coordinate frame at the point p to the camera frame: $\boldsymbol{X}(p) \doteq g(0)$ and $\boldsymbol{x} \sim g_*(\boldsymbol{x}_p)$, where we note that g_* is a rotation.[9] The reader should be aware that the transformation g itself depends on local shape of the

[9]The symbol $\sim$ indicates equivalence up to a scalar factor. Strictly speaking, $\boldsymbol{x}$ and $g_*(\boldsymbol{x}_p)$ do not represent the same vector, but only the same direction (they have opposite sign and different lengths). To obtain a rigorous expression, we would have to write $\boldsymbol{x} = \pi(-g_*(\boldsymbol{x}_p))$. However, these

surface at p, in particular its tangent plane T_pS and its normal ν_p at the point p. We now can rewrite the expression (3.31) for the radiance in terms of the camera coordinates and obtain

$$\mathcal{R}(\boldsymbol{X}) \doteq \mathcal{R}(p, g_*^{-1}(\boldsymbol{x})), \quad \text{where} \quad \boldsymbol{x} = \pi(\boldsymbol{X}). \tag{3.33}$$

If the surface is Lambertian, the above expression simplifies to

$$\mathcal{R}(\boldsymbol{X}) = \mathcal{R}(p). \tag{3.34}$$

Suppose that our imaging sensor is well modeled by a thin lens. Then, by measuring the amount of energy received along the direction $\boldsymbol{x}$, the irradiance (or image intensity) I at $\boldsymbol{x}$ can be expressed as a function of the radiance from the point p:

$$I(\boldsymbol{x}) = \mathcal{R}(\boldsymbol{X}) \frac{\pi}{4} \left(\frac{d}{f}\right)^2 \cos^4(\alpha), \tag{3.35}$$

where d is the lens diameter, f is the focal length, and α is the angle between the optical axis (i.e. the z-axis) and the image point $\boldsymbol{x}$, as shown in Figure 3.13. The quantity $\frac{d}{f}$ is called the *F-number* of the lens. A detailed derivation of the above formula can be found in [Horn, 1986] (page 208). For a Lambertian surface, we have

$$\begin{aligned} I(\boldsymbol{x}) &= \mathcal{R}(\boldsymbol{X}) \frac{\pi}{4} \left(\frac{d}{f}\right)^2 \cos^4(\alpha) = \mathcal{R}(p) \frac{\pi}{4} \left(\frac{d}{f}\right)^2 \cos^4(\alpha) \\ &= \frac{1}{4} \left(\frac{d}{f}\right)^2 \cos^4(\alpha) \int_\Omega \rho_a(p)\, \mathcal{R}(q, l_p) \cos(\theta)\, d\omega, \end{aligned}$$

where $\boldsymbol{x}$ is the image of the point p taken at the vantage point g. Notice that in the above expression, only the angle α depends on the vantage point. In general, for a thin lens with a small field of view, α is approximately constant. Therefore, in our ideal pin-hole model, we may assume that the image intensity (i.e. irradiance) is related to the surface radiance by the *irradiance equation*:

$$\boxed{I(\boldsymbol{x}) = \gamma \mathcal{R}(p),} \tag{3.36}$$

where $\gamma \doteq \frac{\pi}{4} \left(\frac{d}{f}\right)^2 \cos^4(\alpha)$ is a constant factor that is independent of the vantage point.

In all subsequent chapters we will adopt this simple model. The fact that the irradiance I does not change with the vantage point for Lambertian surfaces constitutes a fundamental condition that allows to establish correspondence across multiple images of the same object. This condition and its implications will be studied in more detail in the next chapter.

---

two vectors do represent the same ray through the camera center, and therefore we will regard them as the same.

3.B Image formation in the language of projective geometry

The perspective pinhole camera model described by (3.18) or (3.19) has retained the physical meaning of all parameters involved. In particular, the last entry of both $\boldsymbol{x}'$ and $\boldsymbol{X}$ is normalized to 1 so that the other entries may correspond to actual 2-D and 3-D coordinates (with respect to the metric unit chosen for respective coordinate frames). However, such a normalization is not always necessary as long as we know that it is the direction of those homogeneous vectors that matters. For instance, the two vectors

$$[X, Y, Z, 1]^T, \quad [XW, YW, ZW, W]^T \quad \in \mathbb{R}^4 \tag{3.37}$$

can be used to represent the same point in $\mathbb{R}^3$. Similarly, we can use $[x', y', z']^T$ to represent a point $[x, y, 1]^T$ on the 2-D image plane as long as $x'/z' = x$ and $y'/z' = y$. However, we may run into trouble if the last entry W or z' happens to be 0. To resolve this problem, we need to generalize the interpretation of homogeneous coordinates introduced in the previous chapter.

Definition 3.4 (Projective space and its homogeneous coordinates). *An n-dimensional* projective space $\mathbb{P}^n$ *is the set of one-dimensional subspaces (i.e. lines through the origin) of the vector space $\mathbb{R}^{n+1}$. A point p in $\mathbb{P}^n$ can then be assigned* homogeneous coordinates $\boldsymbol{X} = [x_1, x_2, \ldots, x_{n+1}]^T$ *among which at least one x_i is nonzero. For any nonzero $\lambda \in \mathbb{R}$ the coordinates $\boldsymbol{Y} = [\lambda x_1, \lambda x_2, \ldots, \lambda x_{n+1}]^T$ represent the same point p in $\mathbb{P}^n$. We say that $\boldsymbol{X}$ and $\boldsymbol{Y}$ are equivalent, denoted by $\boldsymbol{X} \sim \boldsymbol{Y}$.*

Example 3.5 (Topological models for the projective space $\mathbb{P}^2$). Figure 3.15 demonstrates two equivalent geometric interpretations of the 2-D projective space $\mathbb{P}^2$. According

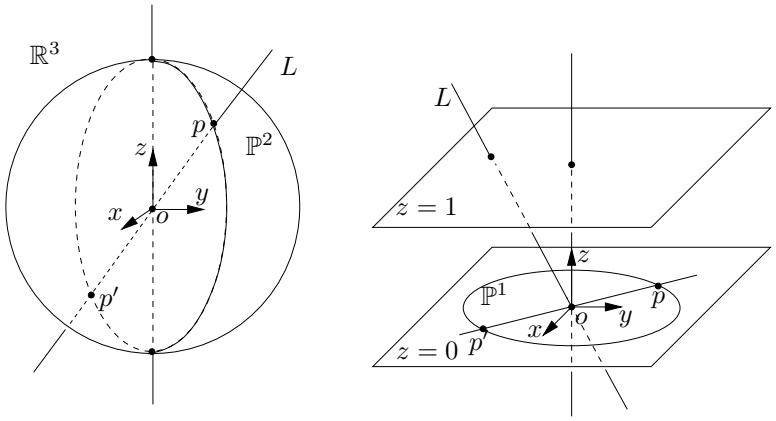

Figure 3.15. Topological models for $\mathbb{P}^2$.

to the definition, it is simply a family of 1-D lines $\{L\}$ in $\mathbb{R}^3$ through a point o (typically chosen to be the origin of the coordinate frame). Hence, $\mathbb{P}^2$ can be viewed as a 2-D sphere $\mathbb{S}^2$ with any pair of antipodal points (e.g., p and p' in the figure) identified as one point in $\mathbb{P}^2$. On the right-hand side of Figure 3.15, lines through the center o in general intersect with the plane $\{z = 1\}$ at a unique point except when they lie on the plane $\{z = 0\}$. Lines in the plane $\{z = 0\}$ simply form the 1-D projective space $\mathbb{P}^1$ (which is in fact a circle). Hence, $\mathbb{P}^2$ can be viewed as a 2-D plane $\mathbb{R}^2$ (i.e. $\{z = 1\}$) with a circle $\mathbb{P}^1$ attached. If we adopt the view that lines in the plane $\{z = 0\}$ intersect the plane $\{z = 1\}$ infinitely far, this circle $\mathbb{P}^1$ then represents a *line at infinity*. Homogeneous coordinates for a point on this circle then take the form $[x, y, 0]^T$; on the other hand, all regular points in $\mathbb{R}^2$ have coordinates $[x, y, 1]^T$. In general, any projective space $\mathbb{P}^n$ can be visualized in a similar way: $\mathbb{P}^3$ is then $\mathbb{R}^3$ with a plane $\mathbb{P}^2$ attached at infinity; and $\mathbb{P}^n$ is $\mathbb{R}^n$ with $\mathbb{P}^{n-1}$ attached at infinity, which is, however, harder to illustrate on a piece of paper. ■

Using this definition, $\mathbb{R}^n$ with its homogeneous representation can then be identified as a subset of $\mathbb{P}^n$ that includes exactly those points with coordinates $\boldsymbol{X} = [x_1, x_2, \ldots, x_{n+1}]^T$ where $x_{n+1} \neq 0$. Therefore, we can always normalize the last entry to 1 by dividing $\boldsymbol{X}$ by x_{n+1} if we so wish. Then, in the pinhole camera model described by (3.18) or (3.19), $\lambda \boldsymbol{x}'$ and $\boldsymbol{x}'$ now represent the same projective point in $\mathbb{P}^2$ and therefore the same 2-D point in the image plane. Suppose that the projection matrix is

$$\Pi = K\Pi_0 g = [KR, KT] \quad \in \mathbb{R}^{3\times 4}. \tag{3.38}$$

Then the camera model simply reduces to a projection from a three-dimensional projective space $\mathbb{P}^3$ to a two-dimensional projective space $\mathbb{P}^2$,

$$\pi : \mathbb{P}^3 \to \mathbb{P}^2; \quad \boldsymbol{X}_0 \mapsto \boldsymbol{x}' \sim \Pi \boldsymbol{X}_0, \tag{3.39}$$

where λ is omitted here, since the equivalence "$\sim$" is defined in the homogeneous sense, i.e. up to a nonzero scalar factor.

Intuitively, the remaining points in $\mathbb{P}^3$ with the fourth coordinate $x_4 = 0$ can be interpreted as points that are "infinitely far away from the origin." This is because for a very small value ϵ, if we normalize the last entry of $\boldsymbol{X} = [X, Y, Z, \epsilon]^T$ to 1, it gives rise to a point in $\mathbb{R}^3$ with 3-D coordinates $\boldsymbol{X} = [X/\epsilon, Y/\epsilon, Z/\epsilon]^T$. The smaller $|\epsilon|$ is, the farther away is the point from the origin. In fact, all points with coordinates $[X, Y, Z, 0]^T$ form a two-dimensional plane described by the equation $[0, 0, 0, 1]^T \boldsymbol{X} = 0$.[10] This plane is called *plane at infinity*. We usually denote this plane by P_∞. That is,

$$P_\infty \doteq \mathbb{P}^3 \setminus \mathbb{R}^3 \ (= \mathbb{P}^2).$$

Then the above imaging model (3.39) is well-defined on the entire projective space $\mathbb{P}^3$ including points in this plane at infinity. This slight generalization allows us to talk about images of points that are infinitely far away from the camera.

[10] It is two-dimensional because X, Y, Z are not totally free: the coordinates are determined only up to a scalar factor.

Example 3.6 (Image of points at infinity and "vanishing points"). Two parallel lines in $\mathbb{R}^3$ do not intersect. However, we can view them as intersecting at infinity. Let $\boldsymbol{V} = [V_1, V_2, V_3, 0]^T \in \mathbb{R}^4$ be a (homogeneous) vector indicating the direction of two parallel lines L^1, L^2. Let $\boldsymbol{X}_o^1 = [X_o^1, Y_o^1, Z_o^1, 1]^T$ and $\boldsymbol{X}_o^2 = [X_o^2, Y_o^2, Z_o^2, 1]^T$ be two base points on the two lines, respectively. Then (homogeneous) coordinates of points on L^1 can be expressed as

$$\boldsymbol{X}^1 = \boldsymbol{X}_o^1 + \mu \boldsymbol{V}, \quad \mu \in \mathbb{R},$$

and similarly for points on L^2. Then the two lines can be viewed as intersecting at a point at infinity with coordinates $\boldsymbol{V}$. The "image" of this intersection, traditionally called a *vanishing point*, is simply given by

$$\boldsymbol{x}' \sim \Pi \boldsymbol{V}.$$

This can be shown by considering images of points on the lines and letting $\mu \to \infty$ asymptotically. If the images of these two lines are given, the image of this intersection can be easily computed or measured. Figure 3.16 shows the intersection of images of parallel lines at the vanishing point, a concept well known to Renaissance artists. ■

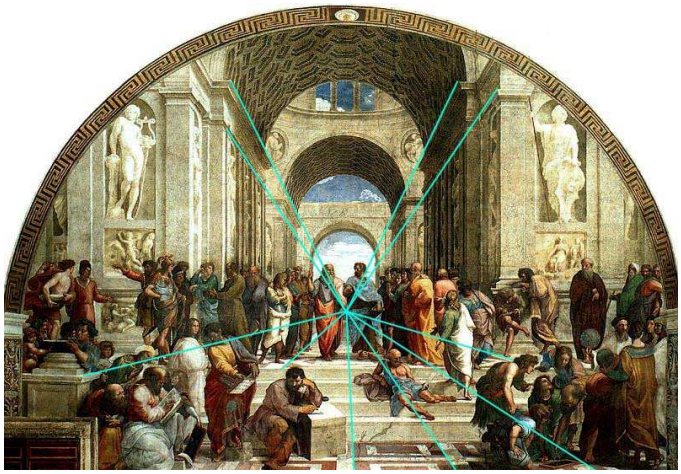

Figure 3.16. "The School of Athens" by Raphael (1518), a fine example of architectural perspective with a central vanishing point, marking the end of the classical Renaissance (courtesy of C. Taylor).

Example 3.7 (Image "outside" the image plane). Consider the standard perspective projection of a pair of parallel lines as in the previous example. We further assume that they are also parallel to the image plane, i.e. the xy-plane. In this case, we have

$$\Pi = \Pi_0 = [I, 0] \quad \text{and} \quad \boldsymbol{V} = [V_1, V_2, 0, 0]^T.$$

Hence, the "image" of the intersection is given in homogeneous coordinates as

$$\boldsymbol{x}' = [V_1, V_2, 0]^T.$$

This does not correspond to any physical point on the 2-D image plane (whose points supposedly have homogeneous coordinates of the form $[x, y, 1]^T$). It is, in fact, a vanishing point at infinity. Nevertheless, we can still treat it as a valid image point. One way is to view it as the image of a point with zero depth (i.e. with the z-coordinate zero). Such a problem will automatically go away if we choose the imaging surface to be an entire sphere rather than a flat plane. This is illustrated in Figure 3.17. ■

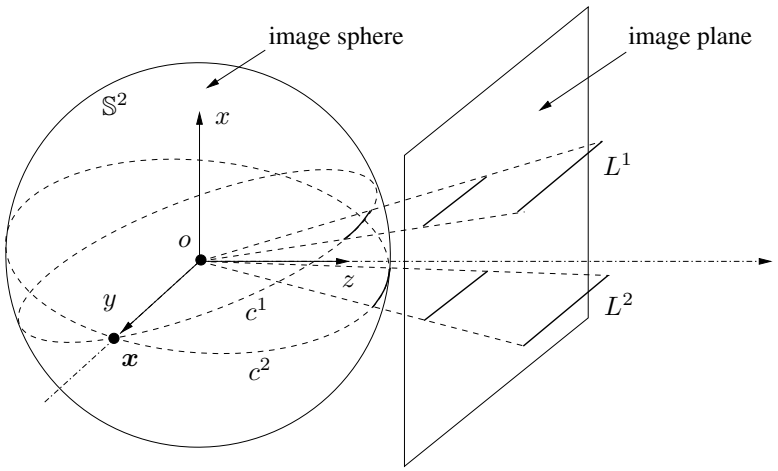

Figure 3.17. Perspective images of two parallel lines that are also parallel to the 2-D image plane. In this case they are parallel to the y-axis. The two image lines on the image plane are also parallel, and hence they do not intersect. On an image sphere, however, the two image circles c^1 and c^2 do intersect at the point $\boldsymbol{x}$. Clearly, $\boldsymbol{x}$ is the direction of the two image lines.

Further readings

Deviations from the pinhole model

As we mentioned earlier in this chapter, the analytical study of pinhole perspective imaging dates back to the Renaissance. Nevertheless, the pinhole perspective model is a rather ideal approximation to actual CCD photosensors or film-based cameras. Before the pinhole model can be applied to such cameras, a correction is typically needed to convert them to an exact perspective device; see [Brank et al., 1993] and references therein.

In general, the pinhole perspective model is not adequate for modeling complex optical systems that involve a zoom lens or multiple lenses. For a systematic introduction to photographic optics and lens systems, we recommend the classic books [Stroebel, 1999, Born and Wolf, 1999]. For a more detailed account of models for a zoom lens, the reader may refer to [Horn, 1986, Lavest et al., 1993]

and references therein. Other approaches such as using a two-plane model [Wei and Ma, 1991] have also been proposed to overcome the limitations of the pinhole model.

Other simple camera models

In the computer vision literature, besides the pinhole perspective model, there exist many other types of simple camera models that are often used for modeling various imaging systems under different practical conditions. This book will *not* cover these cases. The interested reader may refer to [Tomasi and Kanade, 1992] for the study of the orthographic projection, to [Ohta et al., 1981, Aloimonos, 1990, Poelman and Kanade, 1997, Basri, 1996] for the study of the paraperspective projection, to [Konderink and van Doorn, 1991, Mundy and Zisserman, 1992], and [Quan and Kanade, 1996, Quan, 1996] for the study of the affine camera model, and to [Geyer and Daniilidis, 2001] and references therein for catadioptric models often used for omnidirectional cameras.

Chapter 4
Image Primitives and Correspondence

Everything should be made as simple as possible, but not simpler.
– Albert Einstein

In previous chapters we have seen how geometric primitives, such as points and lines in space, can be transformed so that one can compute the coordinates of their "image," i.e. their projection onto the image plane. In practice, however, images are arrays of positive numbers that measure the amount of light incident on a sensor at a particular location (see Sections 3.1 and 3.2, and Appendix 3.A). So, how do we reconcile a geometric image formation model (Section 3.3) with the fact that what we measure with a camera is not points and lines, but light intensity? In other words, how do we go from measurements of light (photometry) to geometry? This is the subject of this chapter: we will show how geometric primitives can be extracted from photometric measurements and matched across different views, so that the rest of the book can concentrate on geometry.

The reader should be aware that although extracting geometric primitives at the outset is widely accepted and practiced, this approach has limitations. For one, in the process we throw away almost all the information (geometric primitives are a set of measure zero in the image). Moreover, as we will see, geometric primitives are extracted and matched by *local* analysis of the image, and are therefore prone to ambiguities and false matches. Nevertheless, *global* analysis of images to infer scene photometry as well as geometry would be computationally challenging, and it is not even clear that it would be meaningful. In fact, if we consider an object with arbitrary geometry and arbitrary photometry, one can always construct (infinitely many) objects with different geometry and different photometry that

give rise to the same images. One example is the image itself: It is an object different from the true scene (it is flat) that gives rise to the same image (itself).

Therefore, in what follows we will rely on *assumptions* on the photometry of the scene in order to be able to establish *correspondence* between geometric primitives in different views. Such assumptions will allow us to use measurements of light in order to discern how points and lines are "moving" in the image. Under such assumptions, the "image motion" is related to the three-dimensional structure of the scene and its motion relative to the camera in ways that we will exploit in later chapters in order to reconstruct the geometry of the scene.

4.1 Correspondence of geometric features

Suppose we have available two images of a scene taken from different vantage points, for instance those in Figure 4.1. Consider the coordinates of a specific point in the left image, for instance the one indicated by a white square. It is immediate for a human observer to establish what the "corresponding" point on the right image is. The two points correspond in the sense that, presumably, they are the projection of the same point in space. Naturally, we cannot expect the pixel coordinates of the point on the left to be identical to those of the point on the right. Therefore, *the "correspondence problem" consists in establishing which point in one image corresponds to which point in another, in the sense of being the image of the same point in space.*

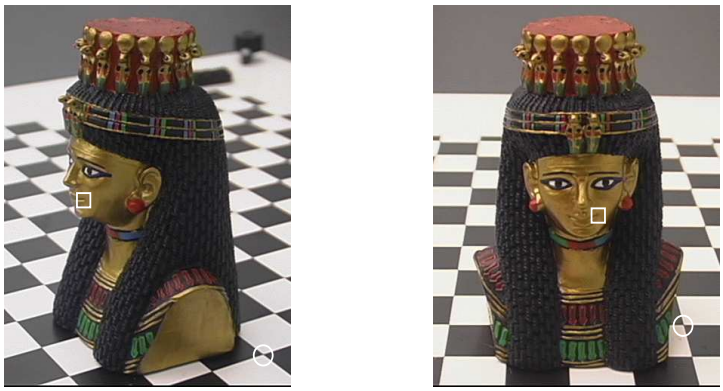

Figure 4.1. "Corresponding points" in two views are projections of the same point in space.

The fact that humans solve the correspondence problem so effortlessly should not lead us to think that this problem is trivial. On the contrary, humans exploit a remarkable amount of information in order to arrive at successfully declaring correspondence, including analyzing context and neighboring structures in the image and prior information on the content of the scene. If we were asked to establish correspondence by just looking at the small regions of the image enclosed in the

circle and square on the left, things would get much harder: which of the regions in Figure 4.2 is the right match? Hard to tell. The task is no easier for a computer.

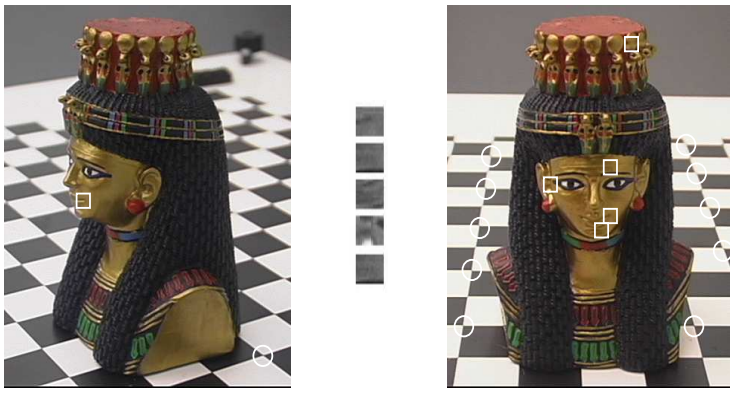

Figure 4.2. Which of these circular or square regions on the right match the ones on the left? Correspondence based on local photometric information is prone to ambiguity. The image on the right shows the corresponding positions on the image. Note that some points do not have a correspondent at all, for instance due to occlusions.

4.1.1 From photometric features to geometric primitives

Let us begin with a naive experiment. Suppose we want to establish correspondence for a pixel in position $\boldsymbol{x}_1$ in the left image in Figure 4.1. The value of the image at $\boldsymbol{x}_1$ is $I_1(\boldsymbol{x}_1)$, so we may be tempted to look for a position $\boldsymbol{x}_2$ in the right image that has the same brightness, $I_1(\boldsymbol{x}_1) = I_2(\boldsymbol{x}_2)$, which can be thought of as a "label" or "signature." Based on the discussion above, it should be obvious to the reader that this approach is doomed to failure. First, there are $307{,}200$ pixel locations in the right image (640×480), each taking a value between 0 and 255 (three for red, green and blue if in color). Therefore, we can expect to find many pixel locations in I_2 matching the value $I_1(\boldsymbol{x}_1)$. Moreover, the actual corresponding point may not even be one of them, since measuring light intensity consists in counting photons, a process intrinsically subject to uncertainty. One way to fix this is to compare not the brightness of individual pixels, but the brightness of each pixel in a small window around the point of interest (see Figure 4.2). We can think of this as attaching to each pixel, instead of a scalar label $I(\boldsymbol{x})$ denoting the brightness of that pixel, an augmented vector label that contains the brightness of each pixel in the window: $l(\boldsymbol{x}) = \{I(\tilde{\boldsymbol{x}}) \mid \tilde{\boldsymbol{x}} \in W(\boldsymbol{x})\}$, where $W(\boldsymbol{x})$ is a window around $\boldsymbol{x}$. Now matching points is carried out by matching windows, under the assumption that each point in the window moves with the same motion (Figure 4.3). Again, due to noise we cannot expect an exact matching of labels, so we

can look for the windows that minimize some discrepancy measure between their labels.

This discussion can be generalized: each point has associated with itself a support window and the value of the image at each point in the window. Both the window shape and the image values undergo *transformations* as a consequence of the change in viewpoint (e.g., the window translates, and the image intensity is corrupted by additive noise), and we look for the transformation that minimizes some discrepancy measure. We carry out this program in the next section (Section 4.1.2). Before doing so, however, we point out that this does not solve all of our problems. Consider, for instance, in Figure 4.2 the rectangular regions on the checkerboard. The value of the image at each pixel in these regions is constant, and therefore it is not possible to tell exactly which one is the corresponding region; it could be any region that fits inside the homogeneous patch of the image. This is just one manifestation of the *blank wall* or *aperture problem*, which occurs when the brightness profile within a selected region is not rich enough to allow us to recover the chosen transformation uniquely (Section 4.3.1). It will be wise, then, to restrict our attention only to those regions for which the correspondence problem can be solved. Those will be called "features," and they establish the link between photometric measurements and geometric primitives.

4.1.2 Local vs. global image deformations

In the discussion above, one can interpret matching windows, rather than points, as the local integration of intensity information, which is known to have beneficial (averaging) effects in counteracting the effects of noise. Why not, then, take this to the extreme, integrate intensity information over the entire image? After all, Chapters 2 and 3 tell us precisely how to compute the coordinates of corresponding points. Of course, the deformation undergone by the entire image cannot be captured by a simple displacement, as we will soon see. Therefore, one can envision two opposite strategies: one is to choose a complex transformation that captures the changes undergone by the entire image, or one can pick a simple transformation, and then restrict the attention to only those regions in the image whose motion can be captured, within reasonable bounds, by the chosen transformation.

As we have seen in Chapter 3, an image, for instance I_1, can be represented as a function defined on a compact two-dimensional region Ω taking irradiance values in the positive reals,

$$I_1 : \Omega \subset \mathbb{R}^2 \to \mathbb{R}_+; \quad \boldsymbol{x} \mapsto I_1(\boldsymbol{x}).$$

Under the simplifying assumptions in Appendix 3.A, the irradiance $I_1(\boldsymbol{x})$ is obtained by integrating the radiant energy in space along the ray $\{\lambda \boldsymbol{x},\ \lambda \in \mathbb{R}_+\}$.[1]

[1]We remind the reader that we do not differentiate in our notation an image point $\boldsymbol{x}$ from its homogeneous representation (with a "1" appended).

If the scene contains only opaque objects, then only one point, say p, along the projection ray contributes to the irradiance. With respect to the camera reference frame, let this point have coordinates $\boldsymbol{X} \in \mathbb{R}^3$, corresponding to a particular value of λ determined by the first intersection of the ray with a visible surface: $\lambda \boldsymbol{x} = \boldsymbol{X}$. Let $\mathcal{R} : \mathbb{R}^3 \to \mathbb{R}_+$ be the radiance distribution of the visible surface and its value $\mathcal{R}(p)$ at p, i.e. the "color" of the scene at the point p.[2] According to Appendix 3.A, we have the *irradiance equation*

$$I_1(\boldsymbol{x}) \sim \mathcal{R}(p). \tag{4.1}$$

Now suppose that a different image of the same scene becomes available, I_2, for instance one taken from a different vantage point. Naturally, we get another function

$$I_2 : \Omega \subset \mathbb{R}^2 \to \mathbb{R}_+; \quad \boldsymbol{x} \mapsto I_2(\boldsymbol{x}).$$

However, $I_2(\boldsymbol{x})$ will in general be different from $I_1(\boldsymbol{x})$ at the same image location $\boldsymbol{x}$. The first step in establishing correspondence is to understand how such a difference occurs.

Let us now use the background developed in Chapters 2 and 3. Assume for now that we are imaging empty space except for a point p with coordinates $\boldsymbol{X} \in \mathbb{R}^3$ that emits light with the same energy in all directions (i.e. the visible "surface," a point, is Lambertian, see Appendix 3.A). This is a simplifying assumption that we will relax in the next section. If I_1 and I_2 are images of the same scene, they must satisfy the same irradiance equation (4.1). Therefore, if $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are the two images of the same point p in the two views, respectively, we must have

$$I_2(\boldsymbol{x}_2) = I_1(\boldsymbol{x}_1) \ \sim \mathcal{R}(p). \tag{4.2}$$

Under these admittedly restrictive assumptions, the correspondence (or matching) problem consists in establishing the relationship between $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, i.e. verifying that the two points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are indeed images of the same 3-D point. Suppose that the displacement between the two camera viewpoints is a rigid-body motion (R, T). From the projection equations introduced in Chapter 3, the point $\boldsymbol{x}_1$ in image I_1 corresponds to the point $\boldsymbol{x}_2$ in image I_2 if

$$\boldsymbol{x}_2 = h(\boldsymbol{x}_1) = \frac{1}{\lambda_2(\boldsymbol{X})}\big(R\lambda_1(\boldsymbol{X})\boldsymbol{x}_1 + T\big), \tag{4.3}$$

where we have emphasized the fact that the scales $\lambda_i,\ i = 1, 2$, depend on the 3-D coordinates $\boldsymbol{X}$ of the point with respect to the camera frame at their respective viewpoints. Therefore, a model for the deformation between two images of the same scene is given by an image matching constraint

$$I_1(\boldsymbol{x}_1) = I_2(h(\boldsymbol{x}_1)), \quad \forall\, \boldsymbol{x}_1 \in \Omega \cap h^{-1}(\Omega) \subset \mathbb{R}^{2\times 2}. \tag{4.4}$$

[2] In the case of gray-scale images, "color" is often used inappropriately to denote intensity.

This equation is sometime called the *brightness constancy constraint*, since it expresses the fact that given a point on an image, there exists a different (transformed) point in another image that has the same brightness.

The function h describes the transformation of the domain, or "image motion," that we have described informally in the beginning of this chapter. In order to make it more suggestive of the motion of individual pixels, we could write h as

$$h(\boldsymbol{x}) = \boldsymbol{x} + \Delta\boldsymbol{x}(\boldsymbol{X}), \tag{4.5}$$

where the fact that h depends on the shape of the scene is made explicitly in the term $\Delta\boldsymbol{x}(\boldsymbol{X})$. Intuitively, $\Delta\boldsymbol{x}(\boldsymbol{X})$ is the displacement of the image of the same point from one view to another: $\Delta\boldsymbol{x} = \boldsymbol{x}_2 - \boldsymbol{x}_1$.[3] Note that the dependency of $h(\boldsymbol{x})$ on the position of the point $\boldsymbol{X}$ comes through the scales λ_1, λ_2, i.e. the depth of visible surfaces. In general, therefore, h is a function in an infinite-dimensional space (the space of all surfaces in 3-D), and solving for image correspondence is as difficult as estimating the shape of visible objects.

If the scene is not Lambertian,[4] we cannot count on equation (4.4) being satisfied at all. Therefore, as we suggested in the beginning of this subsection, modeling the transformation undergone by the entire image is an extremely hard proposition.

4.2 Local deformation models

The problem with a global model, as described in the previous section, is that the transformation undergone by the entire image is, in general, infinite-dimensional, and finding it amounts to inferring the entire 3-D structure of the scene. Therefore, in what follows we concentrate on choosing a class of simple transformations, and then restrict our attention to "regions" of the image that can be modeled as undergoing such transformations. Such transformations occur in the domain of the image, say a window $W(\boldsymbol{x})$ around $\boldsymbol{x}$, and in the intensity values $I(\tilde{\boldsymbol{x}}),\ \tilde{\boldsymbol{x}} \in W(\boldsymbol{x})$. We examine these two instances in the next two subsections.

4.2.1 Transformations of the image domain

Here we consider three cases of increasingly rich local deformation models, starting from the simplest.

Translational motion model

The simplest transformation one can conceive of is one in which each point in the window undergoes the same exact motion, i.e. $\Delta\boldsymbol{x} = \text{constant}$, no longer

[3] A precise analytical expression for $\Delta\boldsymbol{x}$ will be given in the next chapter.

[4] As we explain in Appendix 3.A, a Lambertian surface is one whose appearance does not depend on the viewing direction. In other words, the radiance of the surface at any given point is the same in all directions.

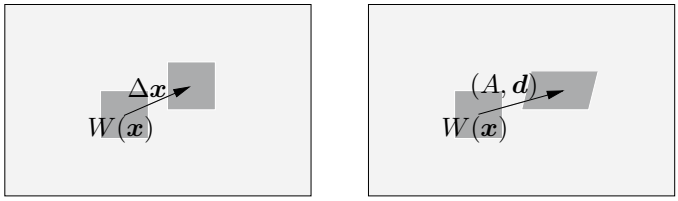

Figure 4.3. Two basic types of local domain $W(\boldsymbol{x})$ deformation. Left: translational; right: affine.

depending on $\boldsymbol{X}$, or $h(\tilde{\boldsymbol{x}}) = \tilde{\boldsymbol{x}} + \Delta\boldsymbol{x}$, $\forall\ \tilde{\boldsymbol{x}} \in W(\boldsymbol{x})$, where $\Delta\boldsymbol{x} \in \mathbb{R}^2$. This model is valid only for portions of the scene that are flat and parallel to the image plane, and moving parallel to it. While one could in principle approximate smooth portions of the scene as a collection of such planar patches, their motion will in general not satisfy the model. The model is therefore only a crude approximation, valid locally in space (small windows) and in time (adjacent time instants, or small camera motion).

Although coarse, this model is at the core of most feature matching or tracking algorithms due to its simplicity and the efficiency of the resulting implementation, which we present in Section 4.3.1.

Affine motion model

In the affine motion model, points in the window $W(\boldsymbol{x})$ do not undergo the same motion but, instead, the motion of each point depends *linearly* on its location plus a constant offset. More precisely, we have $h(\tilde{\boldsymbol{x}}) = A\tilde{\boldsymbol{x}} + \boldsymbol{d}$, $\forall\ \tilde{\boldsymbol{x}} \in W(\boldsymbol{x})$, where $A \in \mathbb{R}^{2\times 2}$ and $\boldsymbol{d} \in \mathbb{R}^2$. This model is a good approximation for small planar patches parallel to the image plane undergoing an arbitrary translation and rotation about the optical axis, and modest rotation about an axis orthogonal to it. This model represents a convenient tradeoff between simplicity and flexibility, as we will see in Section 4.3.2. The affine and translational models are illustrated in Figure 4.3.

Projective motion model

An additional generalization of the affine model occurs when we consider transformations that are linear in the homogeneous coordinates, so that $h(\tilde{\boldsymbol{x}}) \sim H\tilde{\boldsymbol{x}}$, $\forall\ \tilde{\boldsymbol{x}} \in W(\boldsymbol{x})$, where $H \in \mathbb{R}^{3\times 3}$ is defined up to a scalar factor. This model, as we will see in Section 5.3, captures an arbitrary rigid-body motion of a planar patch in the scene. Since any smooth surface can be approximated arbitrarily well by a collection of planes, this model is appropriate everywhere in the image, except at discontinuities and occluding boundaries.

Whatever the transformation h one chooses, in order to establish correspondence, it seems that one has to find h that solves equation (4.4). It turns out that the equality (4.4) is way too much to ask, as we describe in the next section. Therefore, in Section 4.3 we will describe ways to rephrase matching as an optimization problem, which lends itself to the derivation of effective algorithms.

4.2.2 Transformations of the intensity value

The basic assumption underlying the derivation in the previous section is that each point with coordinates $\boldsymbol{X}$ results in the same measured irradiance in both images, as in equation (4.4). In practice, this assumption is unrealistic due to a variety of factors. As a first approximation, one could lump all sources of uncertainty into an additive noise term n.[5] Therefore, equation (4.4) must be modified to take into account changes in the intensity value, in addition to the deformation of the domain

$$I_1(\boldsymbol{x}_1) = I_2(h(\boldsymbol{x}_1)) + n(h(\boldsymbol{x}_1)). \tag{4.6}$$

More fundamental departures from the model (4.4) occur when one considers that points that are visible from one vantage point may become occluded from another. *Occlusions* could be represented by a factor multiplying I_2 that depends upon the shape of the surface ($\boldsymbol{X}$) being imaged and the viewpoint (g): $I_1(\boldsymbol{x}_1) = f_o(\boldsymbol{X}, g)I_2(h(\boldsymbol{x}_1)) + n(h(\boldsymbol{x}_1))$. For instance, for the case where only one point on the surface is emitting light, $f_o(\boldsymbol{X}, g) = 1$ when the point is visible, and $f_o(\boldsymbol{X}, g) = 0$ when not. This equation should make very clear the fact that associating the label I_1 with the point $\boldsymbol{x}_1$ is not a good idea, since the value of I_1 depends upon the noise n and the shape of the surfaces in space $\boldsymbol{X}$, which we cannot control.

There is more: in most natural scenes, objects do not emit light of their own; rather, they reflect ambient light in a way that depends upon the properties of the material. Even in the absence of occlusions, different materials may scatter or reflect light by different amounts in different directions, violating the Lambertian assumption discussed in Appendix 3.A. In general, few materials exhibit perfect Lambertian reflection, and far more complex reflection models, such as translucent or anisotropic materials, are commonplace in natural and man-made scenes (see Figure 3.14).

4.3 Matching point features

From the discussion above one can conclude that point correspondence cannot be established for scenes with arbitrary reflectance properties. Even for relatively simple scenes, whose appearance does not depend on the viewpoint, one cannot establish correspondence among points due to the aperture problem. So how do we proceed? As we have hinted several times already, we proceed by *integrating* local photometric information. Instead of considering equation (4.2) in terms of points on an image, we can consider it defining correspondence in terms of *regions*. This can be done by integrating each side on a window $W(\boldsymbol{x})$ around

[5] This noise is often described statistically as a Poisson random variable (in emphasizing the nature of the counting process and enforcing the nonnegativity constraints on irradiance), or as a Gaussian random variable (in emphasizing the concurrence of multiple independent sources of uncertainty).

each point $\boldsymbol{x}$, and using the equation to characterize the correspondence at $\boldsymbol{x}$. Due to the presence of uncertainty, noise, deviations from Lambertian reflection, occlusions, etc. we can expect that equation (4.4) will be satisfied only up to uncertainty, as in (4.6). Therefore, we formulate correspondence as the solution of an optimization problem. We choose a class of transformations, and we look for the particular transformation $\hat{h}$ that minimizes the effects of noise (measured according to some criterion),[6] subject to equation (4.6), integrated over a window. For instance, we could have $\hat{h} = \arg\min_h \sum_{\tilde{\boldsymbol{x}} \in W(\boldsymbol{x})} \|n(\tilde{\boldsymbol{x}})\|^2$ subject to (4.6) or, writing n explicitly,

$$\hat{h} = \arg\min_h \sum_{\tilde{\boldsymbol{x}} \in W(\boldsymbol{x})} \|I_1(\tilde{\boldsymbol{x}}) - I_2(h(\tilde{\boldsymbol{x}})\|^2 \tag{4.7}$$

if we choose as a discrepancy measure the norm of the additive error. In the next subsections we explore a few choices of discrepancy criteria, which include the sum of squared differences and normalized cross-correlation. Before doing so, however, let us pause for a moment to consider whether the optimization problem just defined is well posed.

Consider equation (4.7), where the point $\boldsymbol{x}$ happens to fall within a region of constant intensity. Then $I_1(\tilde{\boldsymbol{x}}) = \text{constant}$ for all $\tilde{\boldsymbol{x}} \in W(\boldsymbol{x})$. The same is true for I_2 and therefore, the norm being minimized does not depend on h, and any choice of $\hat{h}$ would solve the equation. This is the "blank wall" effect, a manifestation of the so-called aperture problem. Therefore, it appears that in order for the problem (4.7) to be well posed, the intensity values inside a window have to be "rich enough."

Having this important fact in mind, we choose a class of transformations, h, that depends on a set of parameters α. For instance, $\alpha = \Delta\boldsymbol{x}$ for the translational model, and $\alpha = \{A, \boldsymbol{d}\}$ for the affine motion model. With an abuse of notation we indicate the dependency of h on the parameters as $h(\alpha)$. We can then define a pixel $\boldsymbol{x}$ to be a *point feature* if there exists a neighborhood $W(\boldsymbol{x})$ such that the equations

$$I_1(\tilde{\boldsymbol{x}}) = I_2(h(\tilde{\boldsymbol{x}}, \alpha)), \quad \forall \tilde{\boldsymbol{x}} \in W(\boldsymbol{x}), \tag{4.8}$$

uniquely determine the parameters α. From the example of the blank wall, it is intuitive that such conditions would require that I_1 and I_2 at least have nonzero gradient. In the sections to follow we will derive precisely what the conditions are for the translational model. Similarly, one may define a *line feature* as a line segment with a support region and a collection of labels such that the orientation and normal displacement of the transformed line can be uniquely determined from the equation above.

In the next sections we will see how to efficiently solve the problem above for the case in which the α are translational or affine parameters. We first describe

[6] Here the "hat" symbol, $\hat{(\cdot)}$, indicates an estimated quantity (see Appendix B), not to be confused with the "wide hat," $\widehat{(\cdot)}$, used to indicate a skew-symmetric matrix.

how to compute the velocity either of a moving point (feature tracking) or at a fixed location on the pixel grid (optical flow), and then give an effective algorithm to detect point features that can be easily tracked.

The definition of feature points and lines allows us to move our discussion from pixels and images to geometric entities such as points and lines. However, as we will discuss in later chapters, this separation is more conceptual than factual. Indeed, all the constraints among geometric entities that we will derive in chapters of Part II and Part III can be rephrased in terms of constraints on the irradiance values on collections of images, under the assumption of rigidity of Chapter 2 and Lambertian reflection of Chapter 3.

4.3.1 Small baseline: feature tracking and optical flow

Consider the translational model described in the previous sections, where

$$I_1(\boldsymbol{x}) = I_2(h(\boldsymbol{x})) = I_2(\boldsymbol{x} + \Delta\boldsymbol{x}). \tag{4.9}$$

If we consider the two images as being taken from infinitesimally close vantage points, we can write a continuous version of the above constraint. In order to make the notation more suggestive, we call t the time at which the first image is taken, i.e. $I_1(\boldsymbol{x}) \doteq I(\boldsymbol{x}(t), t)$ and $t + dt$ the time when the second image is taken, i.e. $I_2(\boldsymbol{x}) \doteq I(\boldsymbol{x}(t + dt), t + dt)$. The notation "$dt$" suggests an infinitesimal increment of time (and hence motion). Also to associate the displacement $\Delta\boldsymbol{x}$ with the notion of *velocity* in the infinitesimal case, we write $\Delta\boldsymbol{x} \doteq \boldsymbol{u}\, dt$ for a (velocity) vector $\boldsymbol{u} \in \mathbb{R}^2$. Thus, $h(\boldsymbol{x}(t)) = \boldsymbol{x}(t + dt) = \boldsymbol{x}(t) + \boldsymbol{u}\, dt$. With this notation, equation (4.9) can be rewritten as

$$I(\boldsymbol{x}(t), t) = I(\boldsymbol{x}(t) + \boldsymbol{u}\, dt, t + dt). \tag{4.10}$$

Applying Taylor series expansion around $\boldsymbol{x}(t)$ to the right-hand side and neglecting higher-order terms we obtain

$$\boxed{\nabla I(\boldsymbol{x}(t), t)^T \boldsymbol{u} + I_t(\boldsymbol{x}(t), t) = 0,} \tag{4.11}$$

where

$$\nabla I(\boldsymbol{x}, t) \doteq \begin{bmatrix} I_x(\boldsymbol{x}, t) \\ I_y(\boldsymbol{x}, t) \end{bmatrix} = \begin{bmatrix} \frac{\partial I}{\partial x}(\boldsymbol{x}, t) \\ \frac{\partial I}{\partial y}(\boldsymbol{x}, t) \end{bmatrix} \in \mathbb{R}^2, \quad I_t(\boldsymbol{x}, t) \doteq \frac{\partial I}{\partial t}(\boldsymbol{x}, t) \in \mathbb{R}, \tag{4.12}$$

where ∇I and I_t are the spatial and temporal derivatives of $I(\boldsymbol{x}, t)$, respectively. The spatial derivative ∇I is often called the *image gradient*.[7] We will discuss how to compute these derivatives from discretized images in Appendix 4.A of this chapter. If $\boldsymbol{x}(t) = [x(t), y(t)]^T$ is the trajectory of the image of a point moving across the image plane as time t changes, $I(\boldsymbol{x}(t), t)$ should remain constant. Thus,

[7] Be aware that, strictly speaking, the gradient of a function is a covector and should be represented as a row vector. But in this book we define it to be a column vector (see Appendix C), to be consistent with all the other vectors.

another way of deriving the above equation is in terms of the total derivative of $I(\boldsymbol{x}(t), t) = I(x(t), y(t), t)$ with respect to time,

$$\frac{dI(x(t), y(t), t)}{dt} = 0, \tag{4.13}$$

which yields

$$\boxed{\frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0.} \tag{4.14}$$

This equation is identical to (4.11) once we notice that $\boldsymbol{u} \doteq [u_x, u_y]^T = [\frac{dx}{dt}, \frac{dy}{dt}]^T \in \mathbb{R}^2$. We also call this equation the *brightness constancy constraint*. It is the continuous version of (4.4) for the simplest translational model. Depending on where the constraint is evaluated, this equation can be used to compute what is called *optical flow*, or to *track photometric features* in a sequence of moving images.

When we fix our attention at a particular image location $\bar{\boldsymbol{x}}$ and use (4.14) to compute the velocity of "particles flowing" through that pixel, $\boldsymbol{u}(\bar{\boldsymbol{x}}, t)$ is called optical flow. When the attention is on a particular particle $\boldsymbol{x}(t)$ instead, and (4.14) is computed at the location $\boldsymbol{x}(t)$ as it moves through the image domain, we refer to the computation of $\boldsymbol{u}(\boldsymbol{x}(\bar{t}), \bar{t})$ as *feature tracking*. Optical flow and feature tracking are obviously related by $\boldsymbol{x}(t+dt) = \boldsymbol{x}(t) + \boldsymbol{u}(\boldsymbol{x}(t), t)dt$. The only difference, at the conceptual level, is where the vector $\boldsymbol{u}(\boldsymbol{x}, t)$ is computed: in optical flow it is computed at a fixed location in the image, whereas in feature tracking it is computed at the point $\boldsymbol{x}(t)$.

Before we delve into the study of optical flow and feature tracking, notice that (4.11), if computed at each point, provides only one equation for two unknowns (u_x, u_y). This is the aperture problem we have hinted at earlier.

The aperture problem

We start by rewriting equation (4.11) in a more compact form as

$$\boxed{\nabla I^T \boldsymbol{u} + I_t = 0.} \tag{4.15}$$

For simplicity we omit "t" from $(x(t), y(t))$ in $I(x(t), y(t), t)$ and write only $I(x, y, t)$, or $I(\boldsymbol{x}, t)$.

The brightness constancy constraint captures the relationship between the image velocity $\boldsymbol{u}$ of an image point $\boldsymbol{x}$ and spatial and temporal derivatives $\nabla I, I_t$, which are directly measurable from images. As we have already noticed, the equation provides a single constraint for two unknowns in $\boldsymbol{u} = [u_x, u_y]^T$. From the linear-algebraic point of view there are infinitely many solutions $\boldsymbol{u}$ that satisfy this equation. All we can compute is the projection of the actual optical flow vector in the direction of the image gradient ∇I. This component is also referred to as *normal flow* and can be thought of as a minimum norm vector $\boldsymbol{u}_n \in \mathbb{R}^2$ that satisfies the brightness constancy constraint. It is given by a projection of the true

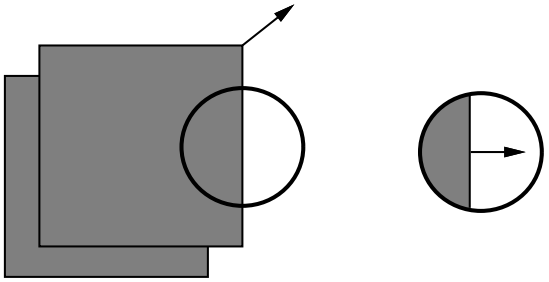

Figure 4.4. In spite of the fact that the square moves diagonally between two consecutive frames, only horizontal motion can be observed through the aperture.

motion vector $\boldsymbol{u}$ onto the gradient direction and is given by

$$\boldsymbol{u}_n \doteq \frac{\nabla I^T \boldsymbol{u}}{\|\nabla I\|} \frac{\nabla I}{\|\nabla I\|} = -\frac{I_t}{\|\nabla I\|} \frac{\nabla I}{\|\nabla I\|}. \tag{4.16}$$

This observation is a consequence of the *aperture problem* and can be easily visualized. For example, consider viewing the square in Figure 4.4 through a small aperture. In spite of the fact that the square moves diagonally between the two consecutive frames, only horizontal motion can be observed through the aperture, and nothing can be said about motion along the direction of the edge.

It is only when the brightness constancy constraint is applied to each point $\tilde{\boldsymbol{x}}$ in a region $W(\boldsymbol{x})$ that contains "sufficient texture," and the motion $\boldsymbol{u}$ is assumed to be constant in the region, that the equations provide enough constraints on $\boldsymbol{u}$. This constancy assumption enables us to integrate the constraints for all points in the region $W(\boldsymbol{x})$ and seek the best image velocity consistent with all the point constraints. In order to account for the effect of noise in the model (4.6), optical flow computation is often formulated as a minimization of the following quadratic error function based on the gradient constraint:

$$E_b(\boldsymbol{u}) = \sum_{W(\boldsymbol{x})} \left[\nabla I^T(\tilde{\boldsymbol{x}}, t)\boldsymbol{u}(\boldsymbol{x}) + I_t(\tilde{\boldsymbol{x}}, t)\right]^2, \tag{4.17}$$

where the subscript "b" indicates brightness constancy. To obtain a linear least-squares estimate of $\boldsymbol{u}(\boldsymbol{x})$ at each image location, we compute the derivative with respect to $\boldsymbol{u}$ of the error function $E_b(\boldsymbol{u})$:

$$\begin{aligned} \nabla E_b(\boldsymbol{u}) &= 2 \sum_{W(\boldsymbol{x})} \nabla I(\nabla I^T \boldsymbol{u} + I_t) \\ &= 2 \sum_{W(\boldsymbol{x})} \left(\begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \boldsymbol{u} + \begin{bmatrix} I_x I_t \\ I_y I_t \end{bmatrix} \right). \end{aligned}$$

For $\boldsymbol{u}$ that minimizes E_b, it is necessary that $\nabla E_b(\boldsymbol{u}) = 0$. This yields

$$\begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \boldsymbol{u} + \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix} = 0, \tag{4.18}$$

or, in matrix form,

$$G\boldsymbol{u} + \boldsymbol{b} = 0. \tag{4.19}$$

Solving this equation (if G is invertible) gives the least-squares estimate of image velocity

$$\boxed{\boldsymbol{u} = -G^{-1}\boldsymbol{b}.} \tag{4.20}$$

Note, however, that the matrix G is not guaranteed to be invertible. If the intensity variation in a local image window varies only along one dimension (e.g., $I_x = 0$ or $I_y = 0$) or vanishes ($I_x = 0$ and $I_y = 0$), then G is *not* invertible. These singularities have been previously mentioned as the *aperture* and *blank wall problem*, respectively. Based on these observations we see that it is the local properties of image irradiance in the window $W(\boldsymbol{x})$ that determine whether the problem is ill posed.

Since it seems that the correspondence problem can be solved, under the brightness constancy assumption, for points $\boldsymbol{x}$ where $G(\boldsymbol{x})$ is invertible, it is convenient to *define* such points as "feature points" at least according to the quadratic criterion above. As we will see shortly, this definition is also consistent with other criteria.

The "sum of squared differences" (SSD) criterion

Let us now go back to the simplest translational deformation model

$$h(\tilde{\boldsymbol{x}}) = \tilde{\boldsymbol{x}} + \Delta\boldsymbol{x}, \quad \forall\, \tilde{\boldsymbol{x}} \in W(\boldsymbol{x}). \tag{4.21}$$

In order to track a feature point $\boldsymbol{x}$ by computing its image displacement $\Delta\boldsymbol{x}$, we can seek the location $\boldsymbol{x} + \Delta\boldsymbol{x}$ on the image at time $t + dt$ whose window is "most similar" to the window $W(\boldsymbol{x})$. A common way of measuring similarity is by using the "sum of squared differences" (SSD) criterion. The SSD approach considers an image window W centered at a location (x, y) at time t and other candidate locations $(x + dx, y + dy)$ in the image at time $t + dt$, where the point could have moved between two frames. The goal is to find a displacement $\Delta\boldsymbol{x} = (dx, dy)$ at a location in the image (x, y) that minimizes the SSD criterion

$$E_t(dx, dy) \doteq \sum_{W(x,y)} \left[I(x + dx, y + dy, t + dt) - I(x, y, t)\right]^2, \tag{4.22}$$

where the subscript "t" indicates the translational deformation model. Comparing this with the error function (4.17), an advantage of the SSD criterion is that in principle we no longer need to compute derivatives of $I(x, y, t)$, although one can easily show that $\boldsymbol{u}\, dt = (-G^{-1}\boldsymbol{b})\, dt$ is the first-order approximation of the displacement $\Delta\boldsymbol{x} = (dx, dy)$. We leave this as an exercise to the reader (see Exercise 4.4). One alternative for computing the displacement is to evaluate the function at each location and choose the one that gives the minimum error. This formulation is due to [Lucas and Kanade, 1981] and was originally proposed in the context of stereo algorithms and was later refined by [Tomasi and Kanade, 1992] in a more

general feature-tracking context.

In Algorithm 4.1 we summarize a basic algorithm for feature tracking or optical flow; a more effective version of this algorithm that involves a multi-resolution representation and subpixel refinement is described in Chapter 11 (Algorithm 11.2).

Algorithm 4.1 (Basic feature tracking and optical flow).

Given an image $I(\boldsymbol{x})$ at time t, set a window W of fixed size, use the filters given in Appendix 4.A to compute the image gradient (I_x, I_y), and compute $G(\boldsymbol{x}) \doteq \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}$ at every pixel $\boldsymbol{x}$. Then, either

- (feature tracking) select a number of point features by choosing $\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots$ such that $G(\boldsymbol{x}_i)$ is invertible, or
- (optical flow) select $\boldsymbol{x}_i$ to be on a fixed grid.

An invertibility test of G that is more robust to the effects of noise will be described in Algorithm 4.2.

- Compute $\boldsymbol{b}(\boldsymbol{x}, t) \doteq \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}$.
- If $G(\boldsymbol{x})$ is invertible (which is guaranteed for point features), compute the displacement $\boldsymbol{u}(\boldsymbol{x}, t)$ from equation (4.20). If $G(\boldsymbol{x})$ is not invertible, return $\boldsymbol{u}(\boldsymbol{x}, t) = 0$.

The displacement of the pixel $\boldsymbol{x}$ at time t is therefore given by $\boldsymbol{u}(\boldsymbol{x}, t) = -G(\boldsymbol{x})^{-1}\boldsymbol{b}(\boldsymbol{x}, t)$ wherever $G(\boldsymbol{x})$ is invertible.

- (Feature tracking) at time $t + 1$, repeat the operation at $\boldsymbol{x} + \boldsymbol{u}(\boldsymbol{x}, t)$.
- (Optical flow) at time $t + 1$, repeat the operation at $\boldsymbol{x}$.

4.3.2 *Large baseline: affine model and normalized cross-correlation*

The small-baseline tracking algorithm presented in the previous section results in very efficient and fast implementations. However, when features are tracked over an extended time period, the estimation error resulting from matching templates between two adjacent frames accumulates in time. This leads to eventually losing track of the originally selected features. To avoid this problem, instead of matching image regions between adjacent frames, one could match image regions between the initial frame, say I_1, and the current frame, say I_2. On the other hand, the deformation of the image regions between the first frame and the current frame can no longer be modeled by a simple translational model. Instead, a commonly adopted model is that of affine deformation of image regions that support point

features, $I_1(\tilde{\boldsymbol{x}}) = I_2(h(\tilde{\boldsymbol{x}}))$, where the function h has the form

$$h(\tilde{\boldsymbol{x}}) = A\tilde{\boldsymbol{x}} + \boldsymbol{d} = \begin{bmatrix} a_1 & a_2 \\ a_3 & a_4 \end{bmatrix} \tilde{\boldsymbol{x}} + \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}, \quad \forall \tilde{\boldsymbol{x}} \in W(\boldsymbol{x}). \tag{4.23}$$

As in the pure translation model (4.9), we can formulate the brightness constancy constraint with this more general six-parameter affine model for the two images:

$$I_1(\tilde{\boldsymbol{x}}) = I_2(A\tilde{\boldsymbol{x}} + \boldsymbol{d}), \quad \forall\, \tilde{\boldsymbol{x}} \in W(\boldsymbol{x}). \tag{4.24}$$

Enforcing the above assumption over a region of the image, we can estimate the unknown affine parameters A and $\boldsymbol{d}$ by integrating the above constraint for all the points in the region $W(\boldsymbol{x})$,

$$E_a(A, \boldsymbol{d}) \doteq \sum_{W(\boldsymbol{x})} \left[I_2(A\tilde{\boldsymbol{x}} + \boldsymbol{d}) - I_1(\tilde{\boldsymbol{x}}) \right]^2, \tag{4.25}$$

where the subscript "a" indicates the affine deformation model. By approximating the function $I_2(A\tilde{\boldsymbol{x}} + \boldsymbol{d})$ to first order around the point $A_0 = I_{2\times 2}, \boldsymbol{d}_0 = 0_{2\times 1}$,

$$I_2(A\tilde{\boldsymbol{x}} + \boldsymbol{d}) \approx I_2(\tilde{\boldsymbol{x}}) + \nabla I_2^T(\tilde{\boldsymbol{x}})[(A - A_0)\tilde{\boldsymbol{x}} + \boldsymbol{d}],$$

the above minimization problem can be solved using linear least-squares, yielding estimates of the unknown parameters $A \in \mathbb{R}^{2\times 2}$ and $\boldsymbol{d} \in \mathbb{R}^2$ directly from measurements of spatial and temporal gradients of the image. In Exercise 4.5 we walk the reader through the steps necessary to implement such a tracking algorithm. In Chapter 11, we will combine this affine model with contrast compensation to derive a practical feature-tracking algorithm that works for a moderate baseline.

Normalized cross-correlation (NCC) criterion

In the previous sections we used the SSD as a cost function for template matching. Although the SSD allows for a linear least-squares solution in the unknowns, there are also some drawbacks to this choice. For example, the SSD is not invariant to scalings and shifts in image intensities, often caused by changing lighting conditions over time. For the purpose of template matching, a better choice is *normalized cross-correlation*. Given two nonuniform image regions $I_1(\tilde{\boldsymbol{x}})$ and $I_2(h(\tilde{\boldsymbol{x}}))$, with $\tilde{\boldsymbol{x}} \in W(\boldsymbol{x})$ and $N = |W(\boldsymbol{x})|$ (the number of pixels in the window), the normalized cross-correlation (NCC) is defined as

$$\text{NCC}(h) = \frac{\sum_{W(\boldsymbol{x})} \left(I_1(\tilde{\boldsymbol{x}}) - \bar{I}_1 \right) \left(I_2(h(\tilde{\boldsymbol{x}})) - \bar{I}_2 \right)}{\sqrt{\sum_{W(\boldsymbol{x})} (I_1(\tilde{\boldsymbol{x}}) - \bar{I}_1)^2 \sum_{W(\boldsymbol{x})} (I_2(h(\tilde{\boldsymbol{x}})) - \bar{I}_2)^2}}, \tag{4.26}$$

where $\bar{I}_1, \bar{I}_2$ are the mean intensities:

$$\begin{aligned} \bar{I}_1 &= \tfrac{1}{N} \textstyle\sum_{W(\boldsymbol{x})} I_1(\tilde{\boldsymbol{x}}), \\ \bar{I}_2 &= \tfrac{1}{N} \textstyle\sum_{W(\boldsymbol{x})} I_2(h(\tilde{\boldsymbol{x}})). \end{aligned}$$

The normalized cross-correlation value always ranges between -1 and $+1$, irrespective of the size of the window. When the normalized cross-correlation is 1, the

two image regions match perfectly. In particular, in the case of the affine model the normalized cross-correlation becomes

$$\text{NCC}(A, \boldsymbol{d}) = \frac{\sum_{W(\boldsymbol{x})} \left(I_1(\tilde{\boldsymbol{x}}) - \bar{I}_1\right) \left(I_2(A\tilde{\boldsymbol{x}} + \boldsymbol{d}) - \bar{I}_2\right)}{\sqrt{\sum_{W(\boldsymbol{x})} (I_1(\tilde{\boldsymbol{x}}) - \bar{I}_1)^2 \sum_{W(\boldsymbol{x})} (I_2(A\tilde{\boldsymbol{x}} + \boldsymbol{d}) - \bar{I}_2)^2}}. \quad (4.27)$$

So, we look for $(\hat{A}, \hat{\boldsymbol{d}}) = \arg\min_{A,\boldsymbol{d}} \text{NCC}(A, \boldsymbol{d})$. In Chapter 11, we will combine NCC with robust statistics techniques to derive a practical algorithm that can match features between two images with a large baseline.

4.3.3 Point feature selection

In previous sections we have seen how to compute the translational or affine deformation of a photometric feature, and we have distinguished the case where the computation is performed at a fixed set of locations (optical flow) from the case where point features are tracked over time (feature tracking). One issue we have not addressed in this second case is how to initially select the points to be tracked. However, we have hinted on various occasions at the possibility of selecting as "feature points" the locations that allow us to solve the correspondence problem easily. In this section we make this more precise by giving a numerical algorithm to select such features.

As the reader may have noticed, the description of any of those feature points relies on knowing the gradient of the image. Hence, before we can give any numerical algorithm for feature selection, the reader needs to know how to compute the image gradient $\nabla I = [I_x, I_y]^T$ in an accurate and robust way. The description of how to compute the gradient of a discretized image is in Appendix 4.A.

The solution to the tracking or correspondence problem for the case of pure translation relied on inverting the matrix G made of the spatial gradients of the image (4.20). For G to be invertible, the region must have nontrivial gradients along two independent directions, resembling therefore a "corner" structure, as shown in Figure 4.5. Alternatively, if we regard the corner as the "intersection"

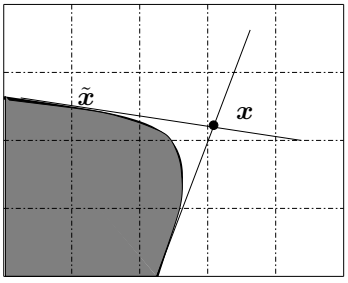

Figure 4.5. A corner feature $\boldsymbol{x}$ is the virtual intersection of local edges (within a window).

of all the edges inside the window, then the existence of at least a corner point

$\boldsymbol{x} = [x, y]^T$ means that over the window $W(\boldsymbol{x})$, the following minimization has a solution:

$$\min_{\boldsymbol{x}} E_c(\boldsymbol{x}) \doteq \sum_{\tilde{\boldsymbol{x}} \in W(\boldsymbol{x})} \left[\nabla I^T(\tilde{\boldsymbol{x}})(\tilde{\boldsymbol{x}} - \boldsymbol{x})\right]^2, \tag{4.28}$$

where $\nabla I(\tilde{\boldsymbol{x}})$ is the gradient calculated at $\tilde{\boldsymbol{x}} = [\tilde{x}, \tilde{y}]^T \in W(\boldsymbol{x})$. It is then easy to check that the existence of a local minimum for this error function is equivalent to the summation of the outer product of the gradients, i.e.

$$G(\boldsymbol{x}) = \sum_{\tilde{\boldsymbol{x}} \in W(\boldsymbol{x})} \nabla I(\tilde{\boldsymbol{x}}) \nabla I^T(\tilde{\boldsymbol{x}}) = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix} \in \mathbb{R}^{2\times 2}, \tag{4.29}$$

being nonsingular. If σ_2, the smallest singular value of G, is above a specified threshold τ, then G is invertible, (4.20) can be solved, and therefore, we say that the point $\boldsymbol{x}$ is a feature point. If both singular values of G are close to zero, the feature window has almost constant brightness. If only one of the singular values is close to zero, the brightness varies mostly along a single direction. In both cases, the point cannot be localized or matched in another image. This leads to a simple algorithm to extract point (or corner) features; see Algorithm 4.2.

Algorithm 4.2 (Corner detector).

Given an image $I(x, y)$, follow the steps to detect whether a given pixel (x, y) is a corner feature:

- set a threshold $\tau \in \mathbb{R}$ and a window W of fixed size, and compute the image gradient (I_x, I_y) using the filters given in Appendix 4.A;
- at all pixels in the window W around (x, y) compute the matrix

$$G = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}; \tag{4.30}$$

- if the smallest singular value $\sigma_2(G)$ is bigger than the prefixed threshold τ, then mark the pixel as a feature (or corner) point.

Although we have used the word "corner," the reader should observe that the test above guarantees only that the irradiance function I is "changing enough" in two independent directions within the window of interest. Another way in which this can happen is for the window to contain "sufficient texture," causing enough variation along at least two independent directions.

A variation to the above algorithm is the well-known Harris corner detector [Harris and Stephens, 1988]. The main idea is to threshold the quantity

$$C(G) = \det(G) + k \times \text{trace}^2(G), \tag{4.31}$$

where $k \in \mathbb{R}$ is a (usually small) scalar, and different choices of k may result in favoring gradient variation in one or more than one direction, or maybe both. To see this, let the two eigenvalues (which in this case coincide with the singular

Figure 4.6. An example of the response of the Harris feature detector using 5×5 integration windows and parameter $k = 0.04$. Some apparent corners around the boundary of the image are not detected due to the size of window chosen.

values) of G be σ_1, σ_2. Then

$$C(G) = \sigma_1\sigma_2 + k(\sigma_1 + \sigma_2)^2 = (1 + 2k)\sigma_1\sigma_2 + k(\sigma_1^2 + \sigma_2^2). \tag{4.32}$$

Note that if k is large and either one of the eigenvalues is large, so will be $C(G)$. That is, features with significant gradient variation in at least one direction will likely pass a threshold. If k is small, then both eigenvalues need to be big enough to make $C(G)$ pass the threshold. In this case, only the corner feature is favored. Simple thresholding operations often do not yield satisfactory results, which lead to a detection of too many corners, which are not well localized. Partial improvements can be obtained by searching for the local minima in the regions, where the response of the detector is high. Alternatively, more sophisticated techniques can be used, which utilize contour (or edge) detection techniques and indeed search for the high curvature points of the detected contours [Wuescher and Boyer, 1991]. In Chapter 11 we will explore further details that are crucial in implementing an effective feature detection and selection algorithm.

4.4 Tracking line features

As we will see in future chapters, besides point features, line (or edge) features, which typically correspond to boundaries of homogeneous regions, also provide important geometric information about the 3-D structure of objects in the scene. In this section, we study how to extract and track such features.

4.4.1 Edge features and edge detection

As mentioned above, when the matrix G in (4.29) has both singular values close to zero, it corresponds to a textureless "blank wall." When one of the singular values is large and the other one is close to zero, the brightness varies mostly along a single direction. But that does not imply a sudden change of brightness value in the direction of the gradient. For example, an image of a shaded marble sphere does vary in brightness, but the variation is smooth, and therefore the entire surface is better interpreted as one smooth region instead of one with edges everywhere. Thus, by "an edge" in an image, we typically refer to a place where there is a distinctive "peak" in the gradient. Of course, the notion of a "peak" depends on the resolution of the image and the size of the window chosen. What appears as smooth shading on a small patch in a high-resolution image may appear as a sharp discontinuity on a large patch in a subsampled image.

We therefore label a pixel $\boldsymbol{x}$ as an "edge feature" only if the gradient norm $\|\nabla I\|$ reaches a local maximum compared to its neighboring pixels. This simple idea results in the well-known Canny edge-detection algorithm [Canny, 1986].

Algorithm 4.3 (Canny edge detector).

Given an image $I(x, y)$, follow the steps to detect whether a given pixel (x, y) is on an edge

- set a threshold $\tau > 0$ and standard deviation $\sigma > 0$ for the Gaussian function g_σ used to derive the filter (see Appendix 4.A for details);
- compute the gradient vector $\nabla I = [I_x, I_y]^T$ (see Appendix 4.A);
- if $\|\nabla I(x, y)\|^2 = \nabla I^T \nabla I$ is a local maximum along the gradient and larger than the prefixed threshold τ, then mark it as an edge pixel.

Figure 4.7 demonstrates edges detected by the Canny edge detector on a gray-level image.

Figure 4.7. Original image, gradient magnitude, and detected edge pixels of an image of Einstein.

4.4.2 Composition of edge elements: line fitting

In order to compensate for the effects of digitization and thresholding that destroy the continuity of the gradient magnitude function $\|\nabla I\|$, the edge-detection stage is often followed by a *connected component analysis*, which enables us to group neighboring pixels with common gradient orientation to form a connected contour or more specifically a candidate line $\boldsymbol{\ell}$. The connected component algorithm can be found in most image processing or computer vision textbooks, and we refer the reader to [Gonzalez and Woods, 1992]. Using results from the connected component analysis, the *line fitting stage* typically involves the computation of the Hough or Radon transform, followed by a peak detection in the parameter space. Both of these techniques are well established in image processing and the algorithms are available as a part of standard image processing toolboxes (see Exercise 4.9).

Alternatively, a conceptually simpler way to obtain line feature candidates is by directly fitting lines to the segments obtained by connected component analysis. Each connected component C^k is a list of edge pixels $\{(x_i, y_i)\}_{i=1}^n$, which are connected and grouped based on their gradient orientation, forming a line support region, say $W(\boldsymbol{\ell})$. The line parameters can then be directly computed from the eigenvalues λ_1, λ_2 and eigenvectors v_1, v_2 of the matrix D^k associated with the line support region:

$$D^k \doteq \begin{bmatrix} \sum_i \tilde{x}_i^2 & \sum_i \tilde{x}_i\tilde{y}_i \\ \sum_i \tilde{x}_i\tilde{y}_i & \sum_i \tilde{y}_i^2 \end{bmatrix} \quad \in \mathbb{R}^{2\times 2}, \tag{4.33}$$

where $\tilde{x} = x_i - \bar{x}$ and $\tilde{y} = y_i - \bar{y}$ are the mean-corrected pixel coordinates of every pixel (x_i, y_i) in the connected component, and $\bar{x} = \frac{1}{n}\sum_i x_i$ and $\bar{y} = \frac{1}{n}\sum_i y_i$ are the means. In the case of an ideal line, one of the eigenvalues should be zero. The quality of the line fit is characterized by the ratio of the two eigenvalues $\frac{\lambda_1}{\lambda_2}$ (with $\lambda_1 > \lambda_2$) of D^k.

On the 2-D image plane, any point (x, y) on a line must satisfy an equation of the form

$$\sin(\theta)x - \cos(\theta)y = \rho. \tag{4.34}$$

Geometrically, θ is the angle between the line $\boldsymbol{\ell}$ and the x-axis, and ρ is the distance from the origin to the line $\boldsymbol{\ell}$ (Figure 4.8). In this notation, the unit eigenvector v_1 (corresponding to the larger eigenvalue λ_1) is of the form $v_1 = [\cos(\theta), \sin(\theta)]^T$. Then, parameters of the line $\boldsymbol{\ell} : (\rho, \theta)$ are determined from v_1 as

$$\theta = \arctan(v_1(2)/v_1(1)), \tag{4.35}$$

$$\rho = \bar{x}\sin(\theta) - \bar{y}\cos(\theta), \tag{4.36}$$

where $(\bar{x}, \bar{y})$ is the midpoint of the line segment. We leave the derivation of these formulae to the reader as an exercise (see Exercise 4.7).

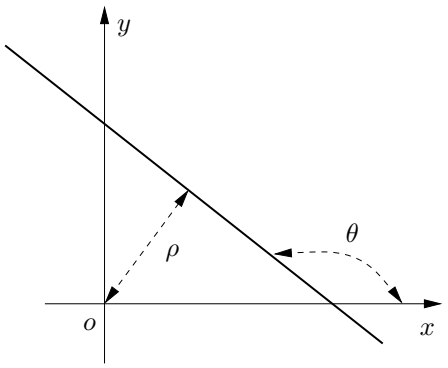

Figure 4.8. Parameterization of a line in 2-D.

Figure 4.9. Edge detection and line fitting results.

4.4.3 Tracking and matching line segments

The techniques for associating line features across multiple frames depend, as in the point feature case, on the baseline between the views. The simplest image-based line tracking technique starts with associating a window support region $W(\boldsymbol{\ell})$, containing the edge pixels forming a line support region.[8] The selected window is first transformed to a canonical image coordinate frame, making the line orientation vertical. At sample points (x_i, y_i) along the line support region, the displacement $d\rho$ in the direction perpendicular to the line is computed. Once this has been done for some number of sample points, the parameters of the new line segment can be obtained, giving rise to the change of the line orientation by $d\theta$. The remaining points can then be updated using the computed parameters $d\rho$

[8]The size of the region can vary depending on whether extended lines are being tracked or just small pieces of connected contours.

and $d\theta$ in the following way:

$$x^{k+1} = x^k + d\rho \sin(\theta^k + d\theta), \tag{4.37}$$
$$y^{k+1} = y^k - d\rho \cos(\theta^k + d\theta), \tag{4.38}$$
$$\theta^{k+1} = \theta^k + d\theta. \tag{4.39}$$

Note that this method suffers from the previously discussed aperture problem. Unless additional constraints are present, the displacement along the edge direction cannot be measured. During the tracking process, the more costly line detection is done only in the initialization stage.

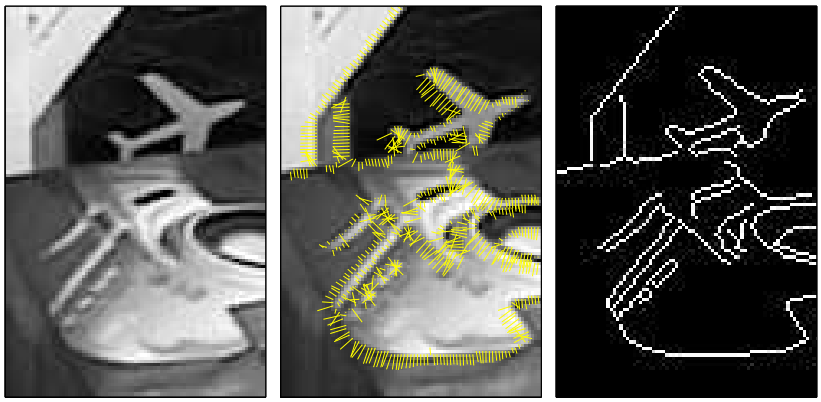

Figure 4.10. Edge tracking by computing the normal displacement of the edge between adjacent frames.

In case of line matching across wide baselines, the support regions $W(\ell)$ associated with the candidate line features subtend the entire extent of the line. Due to the fact that the line support regions automatically contain orientation information, standard window matching criteria (such as SSD and NCC), introduced in Section 4.3, can be used.

4.5 Summary

This chapter describes the crucial step of going from measurements of light intensity to geometric primitives. The notions of "point feature" and "line feature" are introduced, and basic algorithms for feature detection, tracking, and matching are described. Further refinements of these algorithms, e.g., affine tracking, subpixel iterations, and multiscale implementation, are explored in the exercises; practical issues associated with their implementation will be discussed in Chapter 11.

4.6 Exercises

Exercise 4.1 (Motion model). Consider measuring image motion $h(\boldsymbol{x})$ and noticing that $h(\boldsymbol{x}) = \boldsymbol{x} + \Delta\boldsymbol{x}$, $\forall\ \boldsymbol{x} \in \Omega$; i.e. each point on the image translates by the same amount $\Delta\boldsymbol{x}$. What particular motion (R, T) and 3-D structure $\boldsymbol{X}$ must the scene undergo to satisfy this model?

Exercise 4.2 Repeat the exercise above for an affine motion model, $h(\boldsymbol{x}) = A\boldsymbol{x} + \boldsymbol{d}$.

Exercise 4.3 Repeat the exercise above for a general linear motion model, $h(\boldsymbol{x}) = H\boldsymbol{x}$, in homogeneous coordinates.

Exercise 4.4 (LLS approximation of translational flow). Consider the problem of finding a displacement (dx, dy) at a location in the image (x, y) that minimizes the SSD criterion

$$\text{SSD}(dx, dy) \doteq \sum_{W(x,y)} [I(x + dx, y + dy, t + dt) - I(x, y, t))]^2.$$

If we approximate the function $I(x + dx, y + dy, t + dt)$ up to the first order term of its Taylor expansion,

$$I(x + dx, y + dy, t + dt) \approx I(x, y, t) + I_t(x, y, t)dt + \nabla I^T(x, y, t)[dx, dy]^T,$$

we can find a solution to the above minimization problem. Explain under what conditions a unique solution to (dx, dy) exists. Compare the solution to the optical flow solution $\boldsymbol{u} = -G^{-1}\boldsymbol{b}$.

Exercise 4.5 (LLS approximation of affine flow). To obtain an approximate solution to $(A, \boldsymbol{d})$ that minimizes the function

$$E_a(A, \boldsymbol{d}) \doteq \sum_{W(\boldsymbol{x})} [I(A\tilde{\boldsymbol{x}} + \boldsymbol{d}, t + dt) - I(\tilde{\boldsymbol{x}}, t)]^2, \tag{4.40}$$

follow the steps outlined below:

- Approximate the function $I(A\boldsymbol{x} + \boldsymbol{d}, t + dt)$ to first order as
$$I(A\tilde{\boldsymbol{x}} + \boldsymbol{d}, t + dt) \approx I(\tilde{\boldsymbol{x}}, t + dt) + \nabla I^T(\tilde{\boldsymbol{x}}, t + dt)[(A - I_{2\times 2})\tilde{\boldsymbol{x}} + \boldsymbol{d}].$$
- Consider the matrix $D = A - I_{2\times 2} \in \mathbb{R}^{2\times 2}$ and vector $\boldsymbol{d} \in \mathbb{R}^2$ as new unknowns. Collect the unknowns $(D, \boldsymbol{d})$ into the vector $\boldsymbol{y} = [d_{11}, d_{12}, d_{21}, d_{22}, d_1, d_2]^T$ and set $I_t = I(\tilde{\boldsymbol{x}}, t + dt) - I(\tilde{\boldsymbol{x}}, t)$.
- Compute the derivative of the objective function $E_a(D, \boldsymbol{d})$ with respect to the unknowns and set it to zero. Show that the resulting estimate of $\boldsymbol{y} \in \mathbb{R}^6$ is equivalent to the solution of the following systems of linear equations,
$$\sum_{W(\boldsymbol{x})} \begin{bmatrix} G_1 & G_2 \\ G_2^T & G_3 \end{bmatrix} \boldsymbol{y} = \sum_{W(\boldsymbol{x})} \boldsymbol{b}, \quad \text{where} \quad G_3 \doteq \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix},$$
$\boldsymbol{b} \doteq [xI_tI_x, xI_tI_y, yI_tI_x, yI_tI_y, I_tI_x, I_tI_y]^T$, and G_1, G_2 are
$$G_1 \doteq \begin{bmatrix} x^2I_x^2 & x^2I_xI_y & xyI_x^2 & xyI_xI_y \\ x^2I_xI_y & x^2I_y^2 & xyI_xI_y & xyI_y^2 \\ xyI_x^2 & xyI_xI_y & y^2I_x^2 & y^2I_xI_y \\ xyI_xI_y & xyI_y^2 & y^2I_xI_y & y^2I_y^2 \end{bmatrix}, \; G_2 \doteq \begin{bmatrix} xI_xI_y & xI_x^2 \\ xI_y^2 & xI_xI_y \\ yI_xI_y & yI_x^2 \\ yI_y^2 & yI_xI_y \end{bmatrix}.$$

- Write down the linear least-squares estimate of $\boldsymbol{y}$ and discuss under what condition the solution is well-defined.

Exercise 4.6 (Eigenvalues of the sum of outer products). Given a set of vectors $u_1, u_2, \ldots, u_m \in \mathbb{R}^n$, prove that all eigenvalues of the matrix

$$G = \sum_{i=1}^{m} u_i u_i^T \in \mathbb{R}^{n \times n} \tag{4.41}$$

are nonnegative. This shows that the eigenvalues of G are the same as the singular values of G. (Note: you may take it for granted that all the eigenvalues are real, since G is a real symmetric matrix.)

Exercise 4.7 Suppose $\{(x_i, y_i)\}_{i=1}^n$ are coordinates of n sample points from a straight line in $\mathbb{R}^2$. Show that the matrix D defined in (4.33) has rank 1. What is the geometric interpretation of the two eigenvectors v_1, v_2 of D in terms of the line? Since every line in $\mathbb{R}^2$ can be expressed in terms of an equation, $ax + by + c = 0$, derive an expression for the parameters a, b, c in terms of v_1 and v_2.

Exercise 4.8 (Programming: implementation of the corner detector). Implement a version of the corner detector using Matlab. Mark the most distinctive, say 20 to 50, feature points in a given image.

After you are done, you may try to play with it. Here are some suggestions:

- Identify and compare practical methods to choose the threshold τ or other parameters. Evaluate the choice by altering the levels of brightness, saturation, and contrast in the image.
- In practice, you may want to select only one pixel around a corner feature. Devise a method to choose the "best" pixel (or subpixel location) within a window, instead of every pixel above the threshold.
- Devise some quality measure for feature points and sort them according to that measure. Note that the threshold has only "soft" control on the number of features selected. With such a quality measure, however, you can specify any number of features you want.

Exercise 4.9 (Programming: implementation of the line feature detector). Implement a version of the line feature detector using Matlab. Select the line segments whose length exceeds the predefined threshold l. Here are some guidelines on how to proceed:

1. Run the edge-detection algorithm implemented by the function $\texttt{BW} = \texttt{egde}(\texttt{I}, \texttt{param})$ in Matlab.
   - Experiment with different choices of thresholds. Alternatively, you can implement individual steps of the Canny edge detector. Visualize both gradient magnitude $M = \sqrt{I_x^2 + I_y^2}$ and gradient orientation $\Theta = \text{atan2}(I_y, I_x)$.
   - Run the connected component algorithm $\texttt{L} = \texttt{bwlabel}(\texttt{BW})$ in Matlab and group the pixels with similar gradient orientations as described in Section 4.4.
   - Estimate the line parameters of each linear connected group based on equations (4.35) and (4.36), and visualize the results.

2. On the same image, experiment with the function $\mathtt{L} = \mathtt{radon}(\mathtt{I}, \theta)$ and suggest how to use the function to detect line segments in the image. Discuss the advantages and disadvantages of these two methods.

Exercise 4.10 (Programming: subpixel iteration). Both the linear and the affine model for point feature tracking can be refined by subpixel iterations as well as by using multiscale deformation models that allow handling larger deformations. In order to achieve subpixel accuracy, implement the following iteration:

- $\delta^0 = -G^{-1}\boldsymbol{e}^0$,
- $\delta^{i+1} = -G^{-1}\boldsymbol{e}^i$,
- $\boldsymbol{d}^{i+1} \leftarrow \boldsymbol{d}^i + \delta^{i+1}$,

where we define following quantities based on equation (4.19),

- $\boldsymbol{e}^0 \doteq \boldsymbol{b}$,
- $\boldsymbol{e}^{i+1} \doteq \sum_{W(\boldsymbol{x})} \nabla I(\tilde{\boldsymbol{x}})\big(I(\tilde{\boldsymbol{x}} + \boldsymbol{d}^i, t + dt) - I(\tilde{\boldsymbol{x}}, t)\big)$.

At each step, $\tilde{\boldsymbol{x}} + \boldsymbol{d}^i$ is in general not on the pixel grid, so it is necessary to interpolate the brightness values to obtain image intensity at that location.

Exercise 4.11 (Programming: multiscale implementation). One problem common to all differential techniques is that they fail if the displacement across frames is bigger than a few pixels. One possible way to overcome this inconvenience is to use a coarse-to-fine strategy:

- Build a pyramid of images by smoothing and subsampling the original images (see, for instance, [Burt and Adelson, 1983]).
- Select features at the desired level of definition and then propagate the selection up the pyramid.
- Track the features at the coarser level.
- Propagate the displacement to finer resolutions and use that displacement as an initial step for the subpixel iteration described in the previous section.

4.A Computing image gradients

Let us neglect for the moment the discrete nature of digital images. Conceptually, the image gradient $\nabla I(x, y) = [I_x(x, y), I_y(x, y)]^T \in \mathbb{R}^2$ is defined by a vector whose individual components are given by the two partial derivatives

$$I_x(x, y) = \frac{\partial I}{\partial x}(x, y), \quad I_y(x, y) = \frac{\partial I}{\partial y}(x, y). \tag{4.42}$$

In order to simplify the notation, we will omit the argument (x, y) and simply write $\nabla I = [I_x, I_y]^T$. While the notion of derivative is well defined for smooth functions, additional steps need to be taken in computing the derivatives of digital images.

Sampling of a continuous signal

The starting point of this development lies in the relationship between continuous and sampled discrete signals and the theory of sampling and reconstruction [Oppenheim et al., 1999]. Let us assume that we have a sampled version of a continuous signal $f(x), x \in \mathbb{R}$, denoted by

$$f[x] = f(xT), \quad x \in \mathbb{Z}, \tag{4.43}$$

where $f[x]$ is the value of the continuous function $f(x)$ sampled at integer values of x with T being the sampling period of the signal (Figure 4.11). We will adopt the notation of discretely sampled signals with the argument in square brackets.

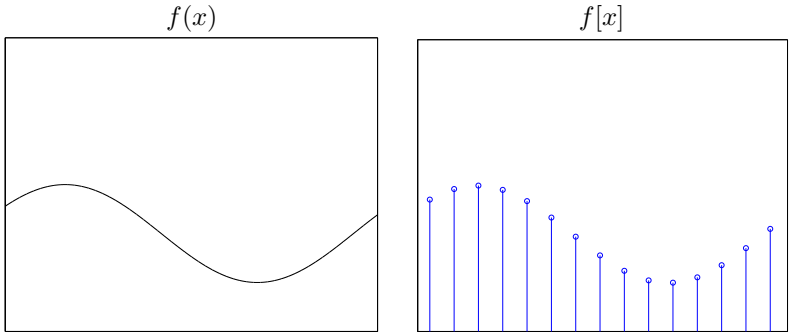

Figure 4.11. Continuous signal $f(x)$ and its discrete sampled version $f[x]$.

Consider a continuous signal $f(x)$ and its Fourier transform $F(\omega)$. The well-known *Nyquist sampling theorem* states that if the continuous signal $f(x)$ is band-limited, i.e. $|F(\omega)| = 0$ for all $\omega > \omega_n$, it can be reconstructed exactly from a set of discrete samples, provided that the sampling frequency $\omega_s > 2\omega_n$; ω_n is called the Nyquist frequency. The relationship between the sampling period and the sampling frequency is $\omega_s = \frac{2\pi}{T}$. Once the above relationship is satisfied, the original signal $f(x)$ can be reconstructed by multiplication of its sampled signal $f[x]$ in the frequency domain with an ideal reconstruction filter, denoted by $h(x)$, whose Fourier transform $H(\omega)$ is 1 between the frequencies $-\pi/T$ and π/T, and 0 elsewhere. That is, the reconstruction filter $h(x)$ is a sync function:

$$h(x) = \frac{\sin(\pi x/T)}{\pi x/T}, \quad x \in \mathbb{R}. \tag{4.44}$$

A multiplication in the frequency domain corresponds to a convolution in the spatial domain. Therefore,

$$f(x) = f[x] * h(x), \quad x \in \mathbb{R}, \tag{4.45}$$

as long as $\omega_n(f) < \frac{\pi}{T}$.

Derivative of a sampled signal

Knowing the relationship between the sampled function $f[x]$ and its continuous version $f(x)$, one can approach the computation of the derivative of the sampled function by first computing the derivative of the continuous function $f(x)$ and then sampling the result. We will outline this process for 1-D signals and then describe how to carry out the computation for 2-D images. Applying the derivative operator to both sides of equation (4.45) yields

$$D\{f(x)\} = D\{f[x] * h(x)\}. \tag{4.46}$$

Expressing the right hand side in terms of the convolution, and using the fact that both the derivative and the convolution are linear operators, we can bring the derivative operator inside of the convolution and write

$$\begin{aligned} D\{f(x)\} &= D\left\{\sum_{k=-\infty}^{k=\infty} f[x]h(x-k)\right\} \\ &= \sum_{k=-\infty}^{k=\infty} f[x]D\{h(x-k)\} = f[x] * D\{h(x)\}. \end{aligned}$$

Notice that the derivative operation is being applied only to continuous entities. Once the derivative of the continuous function has been computed, all we need to do is to sample the result. Denoting the sampling operator as $S\{\cdot\}$ and $D\{f(x)\}$ as $f'(x)$ we have

$$S\{f'(x)\} = S\{f[x] * D\{h(x)\}\} = f[x] * S\{h'(x)\} = f[x] * h'[x]. \tag{4.47}$$

Hence in an ideal situation the derivative of the sampled function can be computed as a convolution of the sampled signal with the sampled derivative of an ideal sync $h'(x)$ (Figure 4.12), where

$$h'(x) = \frac{(\pi^2 x/T^2)\cos(\pi x/T) - \pi/T \sin(\pi x/T)}{(\pi x/T)^2}, \quad x \in \mathbb{R}.$$

Note that in general the value of the function $f'[x]$ receives contribution from all samples of $h'[x]$. However, since the extent of $h[x]$ is infinite and the functions falls off very slowly far away from the origin, the convolution is not practically feasible and simple truncating would yield undesirable artifacts. In practice the computation of derivatives is accomplished by convolving the signal with a finite filter. In case of 1-D signals, a commonly used approximation to the ideal sync and its derivative is a Gaussian and its derivative, respectively, defined as

$$g(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}}, \quad g'(x) = -\frac{x}{\sigma^2\sqrt{2\pi}\sigma} e^{\frac{-x^2}{2\sigma^2}}. \tag{4.48}$$

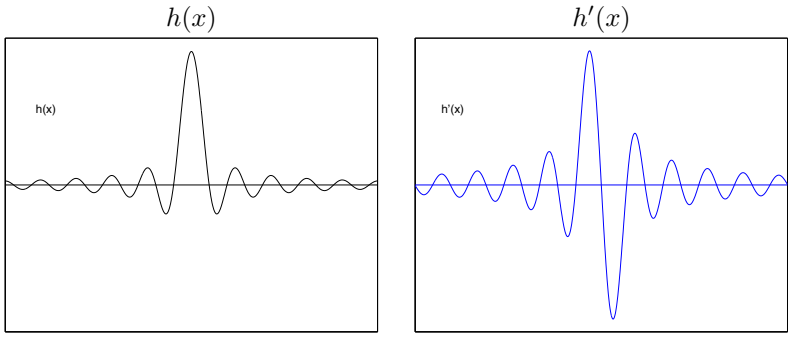

Figure 4.12. Ideal sync function and its derivative.

Note that the Gaussian, like the sync, extends to infinity, and therefore it needs to be truncated.[9] The derivative of a 1-D signal can then be simply computed by convolution with a finite-size filter, which is obtained by sampling and truncating the continuous derivative of the Gaussian. The number of samples w needed is typically related to the variance σ. An adequate relationship between the two is $w = 5\sigma$, imposing the fact that the window subtends 98.76% of the area under the curve. In such a case the convolution becomes

$$f'[x] = f[x] * g'[x] = \sum_{k=-\frac{w}{2}}^{k=\frac{w}{2}} f[x]g'[x-k], \quad x, k \in \mathbb{Z}. \tag{4.49}$$

Examples of the Gaussian filter and its derivative are shown in Figure 4.13.

Image gradient

In order to compute the derivative of a 2-D signal defined by equation (4.42) we have to revisit the relationship between the continuous and sampled versions of the signal

$$I(x, y) = I[x, y] * h(x, y), \quad x, y \in \mathbb{R}, \tag{4.50}$$

where

$$h(x, y) = \frac{\sin(\pi x/T)\sin(\pi y/T)}{\pi^2 xy/T^2}, \quad x, y \in \mathbb{R}, \tag{4.51}$$

is a 2-D ideal sync. Notice that this function is separable, namely $h(x, y) = h(x)h(y)$. Without loss of generality consider the derivative with respect to x. Applying again the derivative operator to both sides we obtain

$$D_x\{I(x, y)\} = D_x\{I[x, y] * h(x, y)\}. \tag{4.52}$$

[9] Nevertheless, the value of a Gaussian function drops to zero exponentially and much faster than a sync, although its Fourier transform, also a Gaussian function, is not an ideal band-pass filter like the sync.

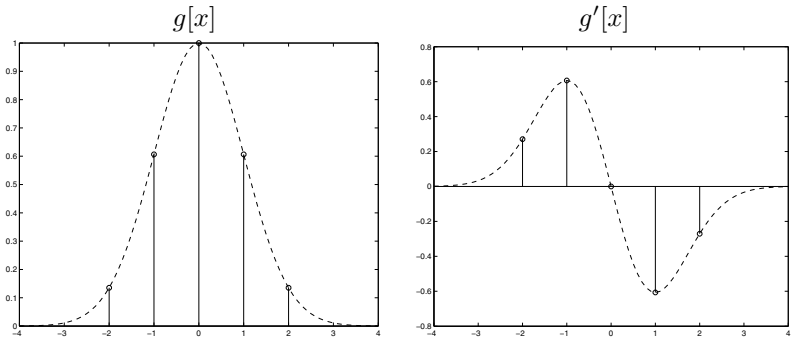

Figure 4.13. Examples of a 1-D five-tap Gaussian filter and its derivative, sampled from a continuous Gaussian function with a variance $\sigma = 1$. The numerical values of the samples are $g[x] = [0.1353, 0.6065, 1.0000, 0.6065, 0.1353]$, and $g'[x] = [0.2707, 0.6065, 0, -0.6065, -0.2707]$, respectively.

Since an ideal sync is separable we can write

$$D_x\{I(x,y)\} = I[x,y] * D_x\{h(x,y)\} = I[x,y] * D_x\{h(x)\} * h(y). \quad (4.53)$$

At last sampling the result we obtain the expression for I_x component of the image gradient

$$\begin{aligned} S\{D_x\{I(x,y)\}\} &= S\{I[x,y] * D_x\{h(x) * h(y)\}\}, \\ I_x[x,y] &= I[x,y] * h'[x] * h[y]. \end{aligned}$$

Similarly the partial derivative I_y is given by

$$I_y[x,y] = I[x,y] * h[x] * h'[y]. \quad (4.54)$$

Note that when computing the partial image derivatives, the image is convolved in one direction with the derivative filter and in the other direction with the interpolation filter. By the same argument as in the 1-D case, we approximate the ideal sync function with a Gaussian function, which falls off faster, and we sample from it and truncate it to obtain a finite-size filter. The computation of image derivatives is then accomplished as a pair of 1-D convolutions with filters obtained by sampling the continuous Gaussian function and its derivative, as shown in Figure 4.13. The image gradient at the pixel $[x,y]^T \in \mathbb{Z}^2$ is then given by

$$I_x[x,y] = I[x,y] * g'[x] * g[y] = \sum_{k=-\frac{w}{2}}^{\frac{w}{2}} \sum_{l=-\frac{w}{2}}^{\frac{w}{2}} I[k,l] g'[x-k] g[y-l],$$

$$I_y[x,y] = I[x,y] * g[x] * g'[y] = \sum_{k=-\frac{w}{2}}^{\frac{w}{2}} \sum_{l=-\frac{w}{2}}^{\frac{w}{2}} I[k,l] g[x-k] g'[y-l].$$

Recall that our choice is only an approximation to the ideal derivative filter. More systematic means of designing such approximations is a subject of

optimal filter design and is in the context of derivative filters, as described in [Farid and Simoncelli, 1997]. Alternative choices have been exploited in image processing and computer vision. One commonly used approximation comes from numerical analysis, where the derivative is approximated by finite differences. In such a case the derivative filter is of the simple form $h'[x] = \frac{1}{2}[1, -1]$, and the interpolation filter is simply $h[x] = \frac{1}{2}[1, 1]$. Another commonly used derivative operator is the so-called Sobel derivative filter where the pair of filters have the following form $h[x] = [1, \sqrt{2}, 1]/(2 + \sqrt{2})$ and $h'[x] = [1, 0, -1]/3$. Note that in both cases the filters are separable.

For the image shown in Figure 4.14, the I_x and I_y components of its gradient ∇I were computed via convolution with a five-tap Gaussian derivative filter shown in Figure 4.13.

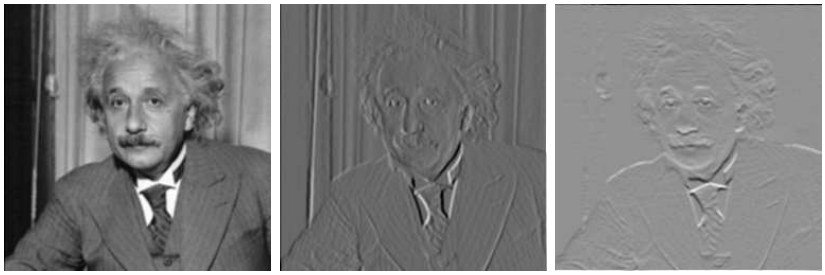

Figure 4.14. Left: original image. Middle and right: Horizontal component I_x and vertical component I_y of the image gradient ∇I.

Image smoothing

In many instances due to the presence of noise in the image formation process, it is often desirable to smooth the image in order to suppress the high frequency component. For this purpose the Gaussian filter is again suitable choice. The image smoothing is then simply accomplished by two 1-D convolutions with the Gaussian. Convolution with the Gaussian can be done efficiently due to the fact that the Gaussian is again separable. The smoothed image then becomes

$$\tilde{I}(x, y) = I(x, y) * g(x, y) = I(x, y) * g(x) * g(y). \tag{4.55}$$

The same expression written in terms of convolution with the filter size w is

$$\tilde{I}[x, y] = I[x, y] * g[x, y] = \sum_{k=-\frac{w}{2}}^{\frac{w}{2}} \sum_{l=-\frac{w}{2}}^{\frac{w}{2}} I[k, l]g[x - k]g[y - l]. \tag{4.56}$$

Figure 4.15 demonstrates the effect of smoothing a noisy image via convolution with a Gaussian.

Figure 4.15. Left: the image "Lena" corrupted by white noise. Right: the corrupted image smoothed by convolution with a 2-D Gaussian.

Further readings

Extraction of corners, edges, and contours

Gradient-based edge detectors like the Canny edge detector [Canny, 1986] and the Harris corner detector [Harris and Stephens, 1988] introduced in this chapter are widely available publicly, for instance, [Meer and Georgescu, www]. Further studies on the extraction of edge elements can be found in the work of [Casadei and Mitter, 1998, Parent and Zucker, 1989, Medioni et al., 2000]. Since lines are special curves with a constant curvature zero, they can also be extracted using curve extraction techniques based on the *constant curvature criterion* [Wuescher and Boyer, 1991]. Besides gradient-based edge detection methods, edges as boundaries of homogeneous regions can also be extracted using the *active contour methods* [Kass et al., 1987, Cohen, 1991, Menet et al., 1990]. One advantage of active contour is its robustness in generating continuous boundaries, but it typically involves solving partial differential equations; see [Kichenassamy et al., 1996, Sapiro, 2001, Osher and Sethian, 1988], and [Chan and Vese, 1999]. In time-critical applications, such as robot navigation, the gradient-based edge detection methods are more commonly used. The Hough transformation is another popular tool, which in principle enables one to extract any type of geometric primitives, including corners, edges, and curves. But its limitation is that one usually needs to specify the size of the primitive a priori.

Feature tracking and optical flow

Image motion (either feature tracking or optical flow) refers to the motion of brightness patterns on the image. It is only under restrictive assumptions on the photometry of the scene, which we discussed in Appendix 3.A, that such image motion is actually related to the motion of the scene. For instance, one can imagine a painted marble sphere rotating in space, and a static spherical mirror where

the ambient light is moved to match the image motion of the first sphere. The distinction between motion field (the motion of the projection of points in the scene onto the image) and optical flow (the motion of brightness patterns on the image) has been elucidated by [Verri and Poggio, 1989].

The feature-tracking schemes given in this chapter mainly follow the work of [Lucas and Kanade, 1981, Tomasi and Kanade, 1992]. The affine flow tracking method was due to [Shi and Tomasi, 1994]. Multiscale estimation methods of global affine flow fields have been introduced by [Bergen et al., 1992]. The use robust estimation techniques in the context of optical flow computation have been proposed by [Black and Anandan, 1993]. Feature extracting and tracking, as we have described it, is an intrinsically local operation in space and time. Therefore, it is extremely difficult to maintain tracking of a feature over extended lengths of time. Typically, a feature point becomes occluded or changes its appearance up to the point of not passing the SSD test. This does not mean that we cannot integrate motion information over time. In fact, it is possible that even if individual features appear and disappear, their motion is consistent with a global 3-D interpretation, as we will see in later parts of the book. Alternative to feature tracking include deformable contours [Blake and Isard, 1998], learning-based approaches [Yacoob and Davis, 1998] and optical flow [Verri and Poggio, 1989, Weickert et al., 1998, Nagel, 1987]. Computation of qualitative ego-motion from normal flow was addressed in [Fermüller and Aloimonos, 1995].

Part II

Geometry of Two Views

Chapter 5

Reconstruction from Two Calibrated Views

We see because we move; we move because we see.
– James J. Gibson, *The Perception of the Visual World*

In this chapter we begin unveiling the basic geometry that relates images of points to their 3-D position. We start with the simplest case of two calibrated cameras, and describe an algorithm, first proposed by the British psychologist H.C. Longuet-Higgins in 1981, to reconstruct the relative pose (i.e. position and orientation) of the cameras as well as the locations of the points in space from their projection onto the two images.

It has been long known in photogrammetry that the coordinates of the projection of a point and the two camera optical centers form a triangle (Figure 5.1), a fact that can be written as an algebraic constraint involving the camera poses and image coordinates but *not* the 3-D position of the points. Given enough points, therefore, this constraint can be used to solve for the camera poses. Once those are known, the 3-D position of the points can be obtained easily by triangulation. The interesting feature of the constraint is that although it is nonlinear in the unknown camera poses, it can be solved by two linear steps in closed form. Therefore, in the absence of any noise or uncertainty, given two images taken from calibrated cameras, one can in principle recover camera pose and position of the points in space with a few steps of simple linear algebra.

While we have not yet indicated how to calibrate the cameras (which we will do in Chapter 6), this chapter serves to introduce the basic building blocks of the geometry of two views, known as "epipolar geometry." The simple algorithms to

be introduced in this chapter, although merely conceptual,[1] allow us to introduce the basic ideas that will be revisited in later chapters of the book to derive more powerful algorithms that can deal with uncertainty in the measurements as well as with uncalibrated cameras.

5.1 Epipolar geometry

Consider two images of the same scene taken from two distinct vantage points. If we assume that the camera is *calibrated*, as described in Chapter 3 (the calibration matrix K is the identity), the homogeneous image coordinates $\boldsymbol{x}$ and the spatial coordinates $\boldsymbol{X}$ of a point p, with respect to the camera frame, are related by[2]

$$\lambda \boldsymbol{x} = \Pi_0 \boldsymbol{X}, \tag{5.1}$$

where $\Pi_0 = [I, 0]$. That is, the image $\boldsymbol{x}$ differs from the actual 3-D coordinates of the point by an unknown (depth) scale $\lambda \in \mathbb{R}_+$. For simplicity, we will assume that the scene is *static* (that is, there are no moving objects) and that the position of corresponding feature points across images is available, for instance from one of the algorithms described in Chapter 4. If we call $\boldsymbol{x}_1, \boldsymbol{x}_2$ the corresponding points in two views, they will then be related by a precise geometric relationship that we describe in this section.

5.1.1 The epipolar constraint and the essential matrix

Following Chapter 3, an orthonormal reference frame is associated with each camera, with its origin o at the optical center and the z-axis aligned with the optical axis. The relationship between the 3-D coordinates of a point in the inertial "world" coordinate frame and the camera frame can be expressed by a rigid-body transformation. Without loss of generality, we can assume the world frame to be one of the cameras, while the other is positioned and oriented according to a Euclidean transformation $g = (R, T) \in SE(3)$. If we call the 3-D coordinates of a point p relative to the two camera frames $\boldsymbol{X}_1 \in \mathbb{R}^3$ and $\boldsymbol{X}_2 \in \mathbb{R}^3$, they are related by a rigid-body transformation in the following way:

$$\boldsymbol{X}_2 = R\boldsymbol{X}_1 + T.$$

Now let $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^3$ be the homogeneous coordinates of the projection of *the same* point p in the two image planes. Since $\boldsymbol{X}_i = \lambda_i \boldsymbol{x}_i, i = 1, 2$, this equation

[1] They are not suitable for real images, which are typically corrupted by noise. In Section 5.2.3 of this chapter, we show how to modify them so as to minimize the effect of noise and obtain an optimal solution.

[2] We remind the reader that we do not distinguish between ordinary and homogeneous coordinates; in the former case $\boldsymbol{x} \in \mathbb{R}^2$, whereas in the latter $\boldsymbol{x} \in \mathbb{R}^3$ with the last component being 1. Similarly, $\boldsymbol{X} \in \mathbb{R}^3$ or $\boldsymbol{X} \in \mathbb{R}^4$ depends on whether ordinary or homogeneous coordinates are used.

can be written in terms of the image coordinates $\boldsymbol{x}_i$ and the depths λ_i as

$$\lambda_2 \boldsymbol{x}_2 = R\lambda_1 \boldsymbol{x}_1 + T.$$

In order to eliminate the depths λ_i in the preceding equation, premultiply both sides by $\widehat{T}$ to obtain

$$\lambda_2 \widehat{T}\boldsymbol{x}_2 = \widehat{T}R\lambda_1 \boldsymbol{x}_1.$$

Since the vector $\widehat{T}\boldsymbol{x}_2 = T \times \boldsymbol{x}_2$ is perpendicular to the vector $\boldsymbol{x}_2$, the inner product $\langle \boldsymbol{x}_2, \widehat{T}\boldsymbol{x}_2 \rangle = {\boldsymbol{x}_2}^T \widehat{T}\boldsymbol{x}_2$ is zero. Premultiplying the previous equation by $\boldsymbol{x}_2^T$ yields that the quantity $\boldsymbol{x}_2^T \widehat{T}R\lambda_1 \boldsymbol{x}_1$ is zero. Since $\lambda_1 > 0$, we have proven the following result:

Theorem 5.1 (Epipolar constraint). *Consider two images $\boldsymbol{x}_1, \boldsymbol{x}_2$ of the same point p from two camera positions with relative pose (R, T), where $R \in SO(3)$ is the relative orientation and $T \in \mathbb{R}^3$ is the relative position. Then $\boldsymbol{x}_1, \boldsymbol{x}_2$ satisfy*

$$\langle \boldsymbol{x}_2, T \times R\boldsymbol{x}_1 \rangle = 0, \quad \text{or} \quad \boxed{\boldsymbol{x}_2^T \widehat{T}R\boldsymbol{x}_1 = 0.} \tag{5.2}$$

The matrix

$$E \doteq \widehat{T}R \quad \in \mathbb{R}^{3\times 3}$$

in the epipolar constraint equation (5.2) is called the *essential matrix*. It encodes the relative pose between the two cameras. The epipolar constraint (5.2) is also called the *essential constraint*. Since the epipolar constraint is bilinear in each of its arguments $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$, it is also called the *bilinear constraint*. We will revisit this bilinear nature in later chapters.

In addition to the preceding algebraic derivation, this constraint follows immediately from geometric considerations, as illustrated in Figure 5.1. The vector connecting the first camera center o_1 and the point p, the vector connecting o_2

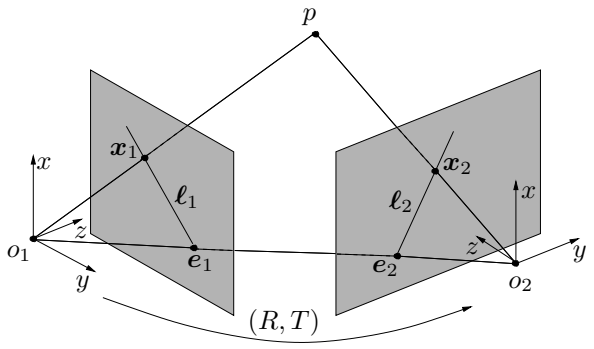

Figure 5.1. Two projections $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^3$ of a 3-D point p from two vantage points. The Euclidean transformation between the two cameras is given by $(R, T) \in SE(3)$. The intersections of the line (o_1, o_2) with each image plane are called *epipoles* and denoted by $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. The lines $\boldsymbol{\ell}_1, \boldsymbol{\ell}_2$ are called *epipolar lines*, which are the intersection of the plane (o_1, o_2, p) with the two image planes.

and p, and the vector connecting the two optical centers o_1 and o_2 clearly form a triangle. Therefore, the three vectors lie on the same plane. Their triple product,[3] which measures the volume of the parallelepiped determined by the three vectors, is therefore zero. This is true for the coordinates of the points $\boldsymbol{X}_i,\ i = 1, 2$, as well as for the homogeneous coordinates of their projection $\boldsymbol{x}_i,\ i = 1, 2$, since $\boldsymbol{X}_i$ and $\boldsymbol{x}_i$ (as vectors) differ only be a scalar factor. The constraint (5.2) is just the triple product written in the second camera frame; $R\boldsymbol{x}_1$ is simply the direction of the vector $\overrightarrow{o_1 p}$, and T is the vector $\overrightarrow{o_2 o_1}$ with respect to the second camera frame. The translation T between the two camera centers o_1 and o_2 is also called the *baseline*.

Associated with this picture, we define the following set of geometric entities, which will facilitate our future study:

Definition 5.2 (Epipolar geometric entities).

1. *The plane (o_1, o_2, p) determined by the two centers of projection o_1, o_2 and the point p is called an* epipolar plane *associated with the camera configuration and point p. There is one epipolar plane for each point p.*
2. *The projection $\boldsymbol{e}_1(\boldsymbol{e}_2)$ of one camera center onto the image plane of the other camera frame is called an* epipole*. Note that the projection may occur outside the physical boundary of the imaging sensor.*
3. *The intersection of the epipolar plane of p with one image plane is a line $\boldsymbol{\ell}_1(\boldsymbol{\ell}_2)$, which is called the* epipolar line *of p. We usually use the normal vector $\boldsymbol{\ell}_1(\boldsymbol{\ell}_2)$ to the epipolar plane to denote this line.*[4]

From the definitions, we immediately have the following relations among epipoles, epipolar lines, and image points:

Proposition 5.3 (Properties of epipoles and epipolar lines). *Given an essential matrix $E = \widehat{T}R$ that defines an epipolar relation between two images $\boldsymbol{x}_1, \boldsymbol{x}_2$, we have:*

1. *The two epipoles $\boldsymbol{e}_1, \boldsymbol{e}_2 \in \mathbb{R}^3$, with respect to the first and second camera frames, respectively, are the left and right null spaces of E, respectively:*

$$\boldsymbol{e}_2^T E = 0, \quad E\boldsymbol{e}_1 = 0. \tag{5.3}$$

   That is, $\boldsymbol{e}_2 \sim T$ and $\boldsymbol{e}_1 \sim R^T T$. We recall that $\sim$ indicates equality up to a scalar factor.

2. *The (coimages of) epipolar lines $\boldsymbol{\ell}_1, \boldsymbol{\ell}_2 \in \mathbb{R}^3$ associated with the two image points $\boldsymbol{x}_1, \boldsymbol{x}_2$ can be expressed as*

$$\boldsymbol{\ell}_2 \sim E\boldsymbol{x}_1, \quad \boldsymbol{\ell}_1 \sim E^T\boldsymbol{x}_2 \quad \in \mathbb{R}^3, \tag{5.4}$$

[3] As we have seen in Chapter 2, the triple product of three vectors is the inner product of one with the cross product of the other two.

[4] Hence the vector $\boldsymbol{\ell}_1(\boldsymbol{\ell}_2)$ is in fact the coimage of the epipolar line.

where $\boldsymbol{\ell}_1, \boldsymbol{\ell}_2$ are in fact the normal vectors to the epipolar plane expressed with respect to the two camera frames, respectively.

3. *In each image, both the image point and the epipole lie on the epipolar line*

$$\boldsymbol{\ell}_i^T \boldsymbol{e}_i = 0, \quad \boldsymbol{\ell}_i^T \boldsymbol{x}_i = 0, \quad i = 1, 2. \tag{5.5}$$

The proof is simple, and we leave it to the reader as an exercise. Figure 5.2 illustrates the relationships among 3-D points, images, epipolar lines, and epipoles.

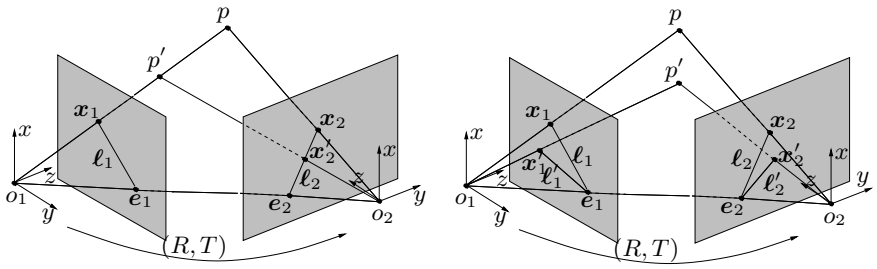

Figure 5.2. Left: the essential matrix E associated with the epipolar constraint maps an image point $\boldsymbol{x}_1$ in the first image to an epipolar line $\boldsymbol{\ell}_2 = E\boldsymbol{x}_1$ in the second image; the precise location of its corresponding image ($\boldsymbol{x}_2$ or $\boldsymbol{x}_2'$) depends on where the 3-D point (p or p') lies on the ray $(o_1, \boldsymbol{x}_1)$; Right: When (o_1, o_2, p) and (o_1, o_2, p') are two different planes, they intersect at the two image planes at two pairs of epipolar lines $(\boldsymbol{\ell}_1, \boldsymbol{\ell}_2)$ and $(\boldsymbol{\ell}_1', \boldsymbol{\ell}_2')$, respectively, and these epipolar lines always pass through the pair of epipoles $(\boldsymbol{e}_1, \boldsymbol{e}_2)$.

5.1.2 Elementary properties of the essential matrix

The matrix $E = \widehat{T}R \in \mathbb{R}^{3\times 3}$ in equation (5.2) contains information about the relative position T and orientation $R \in SO(3)$ between the two cameras. Matrices of this form belong to a very special set of matrices in $\mathbb{R}^{3\times 3}$ called the *essential space* and denoted by $\mathcal{E}$:

$$\mathcal{E} \doteq \left\{ \widehat{T}R \,\middle|\, R \in SO(3), T \in \mathbb{R}^3 \right\} \quad \subset \mathbb{R}^{3\times 3}.$$

Before we study the structure of essential matrices, we introduce a useful lemma from linear algebra.

Lemma 5.4 (The hat operator). *For a vector $T \in \mathbb{R}^3$ and a matrix $K \in \mathbb{R}^{3\times 3}$, if $\det(K) = +1$ and $T' = KT$, then $\widehat{T} = K^T \widehat{T'} K$.*

Proof. Since both $K^T\widehat{(\cdot)}K$ and $\widehat{K^{-1}(\cdot)}$ are linear maps from $\mathbb{R}^3$ to $\mathbb{R}^{3\times 3}$, one may directly verify that these two linear maps agree on the basis vectors $[1, 0, 0]^T, [0, 1, 0]^T$, and $[0, 0, 1]^T$ (using the fact that $\det(K) = 1$). □

The following theorem, due to [Huang and Faugeras, 1989], captures the algebraic structure of essential matrices in terms of their singular value decomposition (see Appendix A for a review on the SVD):

Theorem 5.5 (Characterization of the essential matrix). *A nonzero matrix $E \in \mathbb{R}^{3\times 3}$ is an essential matrix* if and only if *E has a singular value decomposition (SVD) $E = U\Sigma V^T$ with*

$$\Sigma = \mathit{diag}\{\sigma, \sigma, 0\}$$

for some $\sigma \in \mathbb{R}_+$ and $U, V \in SO(3)$.

Proof. We first prove the necessity. By definition, for any essential matrix E, there exists (at least one pair) $(R, T), R \in SO(3), T \in \mathbb{R}^3$, such that $\widehat{T}R = E$. For T, there exists a rotation matrix R_0 such that $R_0 T = [0, 0, \|T\|]^T$. Define $a = R_0 T \in \mathbb{R}^3$. Since $\det(R_0) = 1$, we know that $\widehat{T} = R_0^T \widehat{a} R_0$ from Lemma 5.4. Then $EE^T = \widehat{T}RR^T\widehat{T}^T = \widehat{T}\widehat{T}^T = R_0^T\widehat{a}\widehat{a}^T R_0$. It is immediate to verify that

$$\widehat{a}\widehat{a}^T = \begin{bmatrix} 0 & -\|T\| & 0 \\ \|T\| & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & \|T\| & 0 \\ -\|T\| & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \|T\|^2 & 0 & 0 \\ 0 & \|T\|^2 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

So, the singular values of the essential matrix $E = \widehat{T}R$ are $(\|T\|, \|T\|, 0)$. However, in the standard SVD of $E = U\Sigma V^T$, U and V are only orthonormal, and their determinants can be ± 1.[5] We still need to prove that $U, V \in SO(3)$ (i.e. they have determinant $+1$) to establish the theorem. We already have $E = \widehat{T}R = R_0^T\widehat{a}R_0 R$. Let $R_Z(\theta)$ be the matrix that represents a rotation around the Z-axis by an angle of θ radians; i.e. $R_Z(\theta) \doteq e^{\widehat{e}_3\theta}$ with $e_3 = [0, 0, 1]^T \in \mathbb{R}^3$. Then

$$R_Z\left(+\frac{\pi}{2}\right) = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Then $\widehat{a} = R_Z(+\frac{\pi}{2})R_Z^T(+\frac{\pi}{2})\widehat{a} = R_Z(+\frac{\pi}{2})\ \mathrm{diag}\{\|T\|, \|T\|, 0\}$. Therefore,

$$E = \widehat{T}R = R_0^T R_Z\left(+\frac{\pi}{2}\right)\ \mathrm{diag}\big\{\|T\|, \|T\|, 0\big\} R_0 R.$$

So, in the SVD of $E = U\Sigma V^T$, we may choose $U = R_0^T R_Z(+\frac{\pi}{2})$ and $V^T = R_0 R$. Since we have constructed both U and V as products of matrices in $SO(3)$, they are in $SO(3)$, too; that is, both U and V are rotation matrices.

We now prove sufficiency. If a given matrix $E \in \mathbb{R}^{3\times 3}$ has SVD $E = U\Sigma V^T$ with $U, V \in SO(3)$ and $\Sigma = \mathrm{diag}\{\sigma, \sigma, 0\}$, define $(R_1, T_1) \in SE(3)$ and $(R_2, T_2) \in SE(3)$ to be

$$\begin{cases} (\widehat{T}_1, R_1) &= \big(UR_Z(+\frac{\pi}{2})\Sigma U^T, UR_Z^T(+\frac{\pi}{2})V^T\big), \\ (\widehat{T}_2, R_2) &= \big(UR_Z(-\frac{\pi}{2})\Sigma U^T, UR_Z^T(-\frac{\pi}{2})V^T\big). \end{cases} \tag{5.6}$$

[5] Interested readers can verify this using the Matlab routine: `SVD`.

It is now easy to verify that $\widehat{T}_1 R_1 = \widehat{T}_2 R_2 = E$. Thus, E is an essential matrix. □

Given a rotation matrix $R \in SO(3)$ and a translation vector $T \in \mathbb{R}^3$, it is immediate to construct an essential matrix $E = \widehat{T}R$. The inverse problem, that is how to retrieve T and R from a given essential matrix E, is less obvious. In the sufficiency proof for the above theorem, we have used the SVD to construct two solutions for (R, T). Are these the only solutions? Before we can answer this question in the upcoming Theorem 5.7, we need the following lemma.

Lemma 5.6. *Consider an arbitrary nonzero skew-symmetric matrix $\widehat{T} \in so(3)$ with $T \in \mathbb{R}^3$. If for a rotation matrix $R \in SO(3)$, $\widehat{T}R$ is also a skew-symmetric matrix, then $R = I$ or $R = e^{\widehat{u}\pi}$, where $u = \frac{T}{\|T\|}$. Further, $\widehat{T}e^{\widehat{u}\pi} = -\widehat{T}$.*

Proof. Without loss of generality, we assume that T is of unit length. Since $\widehat{T}R$ is also a skew-symmetric matrix, $(\widehat{T}R)^T = -\widehat{T}R$. This equation gives

$$R\widehat{T}R = \widehat{T}. \tag{5.7}$$

Since R is a rotation matrix, there exist $\omega \in \mathbb{R}^3, \|\omega\| = 1$, and $\theta \in \mathbb{R}$ such that $R = e^{\widehat{\omega}\theta}$. If $\theta = 0$ the lemma is proved. Hence consider the case $\theta \neq 0$. Then, (5.7) is rewritten as $e^{\widehat{\omega}\theta}\widehat{T}e^{\widehat{\omega}\theta} = \widehat{T}$. Applying this equation to ω, we get $e^{\widehat{\omega}\theta}\widehat{T}e^{\widehat{\omega}\theta}\omega = \widehat{T}\omega$. Since $e^{\widehat{\omega}\theta}\omega = \omega$, we obtain $e^{\widehat{\omega}\theta}\widehat{T}\omega = \widehat{T}\omega$. Since ω is the only eigenvector associated with the eigenvalue 1 of the matrix $e^{\widehat{\omega}\theta}$, and $\widehat{T}\omega$ is orthogonal to ω, $\widehat{T}\omega$ has to be zero. Thus, ω is equal to either $\frac{T}{\|T\|}$ or $-\frac{T}{\|T\|}$; i.e. $\omega = \pm u$. Then R has the form $e^{\widehat{\omega}\theta}$, which commutes with $\widehat{T}$. Thus from (5.7), we get

$$e^{2\widehat{\omega}\theta}\widehat{T} = \widehat{T}. \tag{5.8}$$

According to *Rodrigues' formula* (2.16) from Chapter 2, we have

$$e^{2\widehat{\omega}\theta} = I + \widehat{\omega}\sin(2\theta) + \widehat{\omega}^2(1 - \cos(2\theta)),$$

and (5.8) yields

$$\widehat{\omega}^2\sin(2\theta) + \widehat{\omega}^3(1 - \cos(2\theta)) = 0.$$

Since $\widehat{\omega}^2$ and $\widehat{\omega}^3$ are linearly independent (we leave this as an exercise to the reader), we have $\sin(2\theta) = 1 - \cos(2\theta) = 0$. That is, θ is equal to $2k\pi$ or $2k\pi + \pi$, $k \in \mathbb{Z}$. Therefore, R is equal to I or $e^{\widehat{\omega}\pi}$. Now if $\omega = u = \frac{T}{\|T\|}$, then it is direct from the geometric meaning of the rotation $e^{\widehat{\omega}\pi}$ that $e^{\widehat{\omega}\pi}\widehat{T} = -\widehat{T}$. On the other hand, if $\omega = -u = -\frac{T}{\|T\|}$, then it follows that $e^{\widehat{\omega}}\widehat{T} = -\widehat{T}$. Thus, in any case the conclusions of the lemma follow. □

The following theorem shows exactly how many rotation and translation pairs (R, T) one can extract from an essential matrix, and the solutions are given in closed form by equation (5.9).

Theorem 5.7 (Pose recovery from the essential matrix). *There exist exactly two relative poses (R, T) with $R \in SO(3)$ and $T \in \mathbb{R}^3$ corresponding to a nonzero essential matrix $E \in \mathcal{E}$.*

Proof. Assume that $(R_1, T_1) \in SE(3)$ and $(R_2, T_2) \in SE(3)$ are both solutions for the equation $\widehat{T}R = E$. Then we have $\widehat{T}_1 R_1 = \widehat{T}_2 R_2$. This yields $\widehat{T}_1 = \widehat{T}_2 R_2 R_1^T$. Since $\widehat{T}_1, \widehat{T}_2$ are both skew-symmetric matrices and $R_2 R_1^T$ is a rotation matrix, from the preceding lemma, we have that either $(R_2, T_2) = (R_1, T_1)$ or $(R_2, T_2) = (e^{\widehat{u}_1 \pi} R_1, -T_1)$ with $u_1 = T_1 / \|T_1\|$. Therefore, given an essential matrix E there are exactly *two* pairs of (R, T) such that $\widehat{T}R = E$. Further, if E has the SVD: $E = U \Sigma V^T$ with $U, V \in SO(3)$, the following formulae give the two distinct solutions (recall that $R_Z(\theta) \doteq e^{\widehat{e}_3 \theta}$ with $e_3 = [0, 0, 1]^T \in \mathbb{R}^3$)

$$\boxed{\begin{array}{rcl} (\widehat{T}_1, R_1) & = & \big(U R_Z(+\frac{\pi}{2}) \Sigma U^T, U R_Z^T(+\frac{\pi}{2}) V^T\big), \\ (\widehat{T}_2, R_2) & = & \big(U R_Z(-\frac{\pi}{2}) \Sigma U^T, U R_Z^T(-\frac{\pi}{2}) V^T\big). \end{array}} \tag{5.9}$$

□

Example 5.8 (Two solutions to an essential matrix). It is immediate to verify that $\widehat{e_3} R_Z\left(+\frac{\pi}{2}\right) = \widehat{-e_3} R_Z\left(-\frac{\pi}{2}\right)$, since

$$\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

These two solutions together are usually referred to as a "twisted pair," due to the manner in which the two solutions are related geometrically, as illustrated in Figure 5.3. A physically correct solution can be chosen by enforcing that the reconstructed points be visible, i.e. they have positive depth. We discuss this issue further in Exercise 5.11. ■

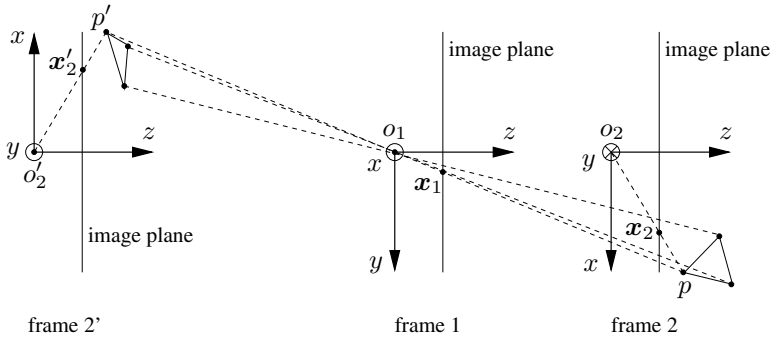

Figure 5.3. Two pairs of camera frames, i.e. $(1, 2)$ and $(1, 2')$, generate the same essential matrix. The frame 2 and frame $2'$ differ by a translation and a 180° rotation (a twist) around the z-axis, and the two pose pairs give rise to the same image coordinates. For the same set of image pairs $\boldsymbol{x}_1$ and $\boldsymbol{x}_2 = \boldsymbol{x}_2'$, the recovered structures p and p' might be different. Notice that with respect to the camera frame 1, the point p' has a negative depth.

5.2 Basic reconstruction algorithms

In the previous section, we have seen that images of corresponding points are related by the epipolar constraint, which involves the unknown relative pose between the cameras. Therefore, given a number of corresponding points, we could use the epipolar constraints to try to recover camera pose. In this section, we show a simple closed-form solution to this problem. It consists of two steps: First a matrix E is recovered from a number of epipolar constraints; then relative translation and orientation are extracted from E. However, since the matrix E recovered using correspondence data in the epipolar constraint may not be an essential matrix, it needs to be projected into the space of essential matrices prior to extraction of the relative pose of the cameras using equation (5.9).

Although the linear algorithm that we propose here is suboptimal when the measurements are corrupted by noise, it is important for illustrating the geometric structure of the space of essential matrices. We leave the more practical issues with noise and optimality to Section 5.2.3.

5.2.1 The eight-point linear algorithm

Let $E = \widehat{T}R$ be the essential matrix associated with the epipolar constraint (5.2). The entries of this 3×3 matrix are denoted by

$$E = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \quad \in \mathbb{R}^{3\times 3} \tag{5.10}$$

and stacked into a vector $E^s \in \mathbb{R}^9$, which is typically referred to as the *stacked version* of the matrix E (Appendix A.1.3):

$$E^s \doteq [e_{11}, e_{21}, e_{31}, e_{12}, e_{22}, e_{32}, e_{13}, e_{23}, e_{33}]^T \quad \in \mathbb{R}^9.$$

The inverse operation from E^s to its matrix version is then called *unstacking*. We further denote the *Kronecker product* $\otimes$ (also see Appendix A.1.3) of two vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ by

$$\boldsymbol{a} \doteq \boldsymbol{x}_1 \otimes \boldsymbol{x}_2. \tag{5.11}$$

Or, more specifically, if $\boldsymbol{x}_1 = [x_1, y_1, z_1]^T \in \mathbb{R}^3$ and $\boldsymbol{x}_2 = [x_2, y_2, z_2]^T \in \mathbb{R}^3$, then

$$\boldsymbol{a} = [x_1x_2, x_1y_2, x_1z_2, y_1x_2, y_1y_2, y_1z_2, z_1x_2, z_1y_2, z_1z_2]^T \quad \in \mathbb{R}^9. \tag{5.12}$$

Since the epipolar constraint $\boldsymbol{x}_2^T E \boldsymbol{x}_1 = 0$ is linear in the entries of E, using the above notation we can rewrite it as the inner product of $\boldsymbol{a}$ and E^s:

$$\boxed{\boldsymbol{a}^T E^s = 0.}$$

This is just another way of writing equation (5.2) that emphasizes the linear dependence of the epipolar constraint on the elements of the essential matrix. Now,

given a set of corresponding image points $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$, $j = 1, 2, \ldots, n$, define a matrix $\chi \in \mathbb{R}^{n \times 9}$ associated with these measurements to be

$$\chi \doteq [\boldsymbol{a}^1, \boldsymbol{a}^2, \ldots, \boldsymbol{a}^n]^T, \tag{5.13}$$

where the jth row $\boldsymbol{a}^j$ is the Kronecker product of each pair $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$ using (5.12). In the absence of noise, the vector E^s satisfies

$$\chi E^s = 0. \tag{5.14}$$

This linear equation may now be solved for the vector E^s. For the solution to be unique (up to a scalar factor, ruling out the trivial solution $E^s = 0$), the rank of the matrix $\chi \in \mathbb{R}^{9 \times n}$ needs to be exactly eight. This should be the case given $n \geq 8$ "ideal" corresponding points, as shown in Figure 5.4. In general, however, since correspondences may be prone to errors, there may be no solution to (5.14). In such a case, one can choose the E^s that minimizes the least-squares error function $\|\chi E^s\|^2$. This is achieved by choosing E^s to be the eigenvector of $\chi^T \chi$ that corresponds to its smallest eigenvalue, as we show in Appendix A. We would also like to draw attention to the case when the rank of χ is less then eight even for number of points greater than nine. In this instance there are multiple solutions to (5.14). This happens when the feature points are not in "general position," for example when they all lie on a plane. We will specifically deal with the planar case in the next section.

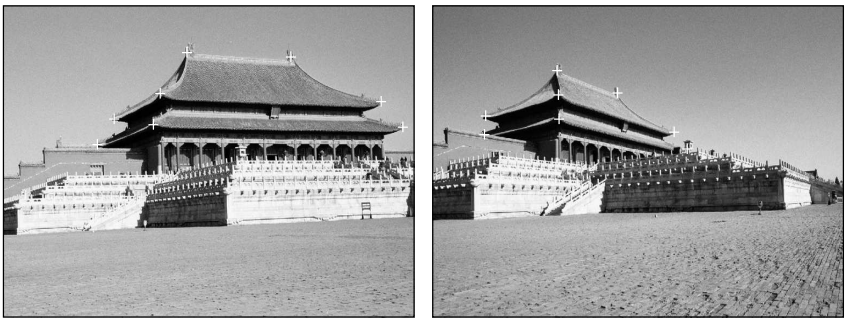

Figure 5.4. Eight pairs of corresponding image points in two views of the Tai-He palace in the Forbidden City, Beijing, China (photos courtesy of Jie Zhang).

However, even in the absence of noise, for a vector E^s to be the solution of our problem, it is not sufficient that it be in the null space of χ. In fact, it has to satisfy an additional constraint, that its matrix form E belong to the space of essential matrices. Enforcing this structure in the determination of the null space of χ is difficult. Therefore, as a first cut, we estimate the null space of χ, *ignoring the internal structure of essential matrix*, obtaining a matrix, say F, that probably does not belong to the essential space $\mathcal{E}$, and then "orthogonally" project the matrix thus obtained onto the essential space. This process is illustrated in Figure 5.5. The following theorem says precisely what this projection is.

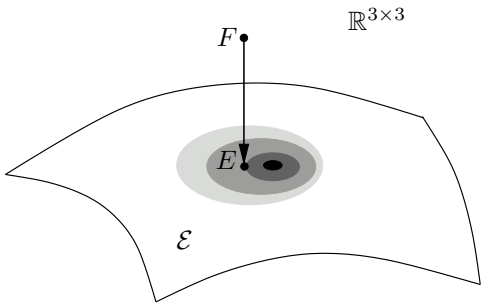

Figure 5.5. Among all points in the essential space $\mathcal{E} \subset \mathbb{R}^{3\times3}$, E has the shortest Frobenius distance to F. However, the least-square error may not be the smallest for so-obtained E among all points in $\mathcal{E}$.

Theorem 5.9 (Projection onto the essential space). *Given a real matrix $F \in \mathbb{R}^{3\times3}$ with SVD $F = U\text{diag}\{\lambda_1, \lambda_2, \lambda_3\}V^T$ with $U, V \in SO(3)$, $\lambda_1 \geq \lambda_2 \geq \lambda_3$, then the essential matrix $E \in \mathcal{E}$ that minimizes the error $\|E - F\|_f^2$ is given by $E = U\text{diag}\{\sigma, \sigma, 0\}V^T$ with $\sigma = (\lambda_1 + \lambda_2)/2$. The subscript "$f$" indicates the Frobenius norm of a matrix. This is the square norm of the sum of the squares of all the entries of the matrix (see Appendix A).*

Proof. For any fixed matrix $\Sigma = \text{diag}\{\sigma, \sigma, 0\}$, we define a subset $\mathcal{E}_\Sigma$ of the essential space $\mathcal{E}$ to be the set of all essential matrices with SVD of the form $U_1\Sigma V_1^T$, $U_1, V_1 \in SO(3)$. To simplify the notation, define $\Sigma_\lambda = \text{diag}\{\lambda_1, \lambda_2, \lambda_3\}$. We now prove the theorem in two steps:

Step 1: We prove that for a fixed Σ, the essential matrix $E \in \mathcal{E}_\Sigma$ that minimizes the error $\|E - F\|_f^2$ has a solution $E = U\Sigma V^T$ (not necessarily unique). Since $E \in \mathcal{E}_\Sigma$ has the form $E = U_1\Sigma V_1^T$, we get

$$\|E - F\|_f^2 = \|U_1\Sigma V_1^T - U\Sigma_\lambda V^T\|_f^2 = \|\Sigma_\lambda - U^T U_1 \Sigma V_1^T V\|_f^2.$$

Define $P = U^T U_1, Q = V^T V_1 \in SO(3)$, which have the form

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix}, \quad Q = \begin{bmatrix} q_{11} & q_{12} & q_{13} \\ q_{21} & q_{22} & q_{23} \\ q_{31} & q_{32} & q_{33} \end{bmatrix}. \tag{5.15}$$

Then

$$\begin{aligned} \|E - F\|_f^2 &= \|\Sigma_\lambda - U^T U_1 \Sigma V_1^T V\|_f^2 \\ &= \text{trace}(\Sigma_\lambda^2) - 2\text{trace}(P\Sigma Q^T \Sigma_\lambda) + \text{trace}(\Sigma^2). \end{aligned}$$

Expanding the second term, using $\Sigma = \text{diag}\{\sigma, \sigma, 0\}$ and the notation p_{ij}, q_{ij} for the entries of P, Q, we have

$$\text{trace}(P\Sigma Q^T \Sigma_\lambda) = \sigma\big(\lambda_1(p_{11}q_{11} + p_{12}q_{12}) + \lambda_2(p_{21}q_{21} + p_{22}q_{22})\big).$$

Since P, Q are rotation matrices, $p_{11}q_{11} + p_{12}q_{12} \leq 1$ and $p_{21}q_{21} + p_{22}q_{22} \leq 1$. Since Σ, Σ_λ are fixed and $\lambda_1, \lambda_2 \geq 0$, the error $\|E - F\|_f^2$ is minimized when

$p_{11}q_{11} + p_{12}q_{12} = p_{21}q_{21} + p_{22}q_{22} = 1$. This can be achieved when P, Q are of the general form

$$P = Q = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Obviously, $P = Q = I$ is one of the solutions. That implies $U_1 = U, V_1 = V$.

Step 2: From Step 1, we need to minimize the error function only over the matrices of the form $U\Sigma V^T \in \mathcal{E}$, where Σ may vary. The minimization problem is then converted to one of minimizing the error function

$$\|E - F\|_f^2 = (\lambda_1 - \sigma)^2 + (\lambda_2 - \sigma)^2 + (\lambda_3 - 0)^2.$$

Clearly, the σ that minimizes this error function is given by $\sigma = (\lambda_1 + \lambda_2)/2$. □

As we have already pointed out, the epipolar constraint allows us to recover the essential matrix only up to a scalar factor (since the epipolar constraint (5.2) is homogeneous in E, it is not modified by multiplying it by any nonzero constant). A typical choice to fix this ambiguity is to assume a unit translation, that is, $\|T\| = \|E\| = 1$. We call the resulting essential matrix *normalized*.

Remark 5.10. *The reader may have noticed that the above theorem relies on a special assumption that in the SVD of E both matrices U and V are rotation matrices in $SO(3)$. This is not always true when E is estimated from noisy data. In fact, standard SVD routines do* not *guarantee that the computed U and V have positive determinant. The problem can be easily resolved once one notices that the sign of the essential matrix E is also arbitrary (even after normalization). The above projection can operate either on $+E$ or $-E$. We leave it as an exercise to the reader that one of the (noisy) matrices $\pm E$ will always have an SVD that satisfies the conditions of Theorem 5.9.*

According to Theorem 5.7, each normalized essential matrix E gives two possible poses (R, T). So from $\pm E$, we can recover the pose up to four solutions. In fact, three of the solutions can be eliminated by imposing the positive depth constraint. We leave the details to the reader as an exercise (see Exercise 5.11).

The overall algorithm, which is due to [Longuet-Higgins, 1981], can then be summarized as Algorithm 5.1.

To account for the possible sign change $\pm E$, in the last step of the algorithm, the "+" and "−" signs in the equations for R and T should be arbitrarily combined so that all four solutions can be obtained.

Example 5.11 (A numerical example). Suppose that

$$R = \begin{bmatrix} \cos(\pi/4) & 0 & \sin(\pi/4) \\ 0 & 1 & 0 \\ -\sin(\pi/4) & 0 & \cos(\pi/4) \end{bmatrix} = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \\ 0 & 1 & 0 \\ -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \end{bmatrix}, \quad T = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}.$$

Algorithm 5.1 (The eight-point algorithm).

For a given set of image correspondences $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$, $j = 1, 2, \ldots, n$ ($n \geq 8$), this algorithm recovers $(R, T) \in SE(3)$, which satisfy

$$\boldsymbol{x}_2^{jT} \widehat{T} R \boldsymbol{x}_1^j = 0, \quad j = 1, 2, \ldots, n.$$

1. **Compute a first approximation of the essential matrix**
   Construct $\chi = [\boldsymbol{a}^1, \boldsymbol{a}^2, \ldots, \boldsymbol{a}^n]^T \in \mathbb{R}^{n \times 9}$ from correspondences $\boldsymbol{x}_1^j$ and $\boldsymbol{x}_2^j$ as in (5.12), namely,

   $$\boldsymbol{a}^j = \boldsymbol{x}_1^j \otimes \boldsymbol{x}_2^j \quad \in \mathbb{R}^9.$$

   Find the vector $E^s \in \mathbb{R}^9$ of unit length such that $\|\chi E^s\|$ is minimized as follows: compute the SVD of $\chi = U_\chi \Sigma_\chi V_\chi^T$ and define E^s to be the ninth column of V_χ. Unstack the nine elements of E^s into a square 3×3 matrix E as in (5.10). Note that this matrix will in general *not* be in the essential space.

2. **Project onto the essential space**
   Compute the singular value decomposition of the matrix E recovered from data to be

   $$E = U \text{diag}\{\sigma_1, \sigma_2, \sigma_3\} V^T,$$

   where $\sigma_1 \geq \sigma_2 \geq \sigma_3 \geq 0$ and $U, V \in SO(3)$. In general, since E may not be an essential matrix, $\sigma_1 \neq \sigma_2$ and $\sigma_3 \neq 0$. But its projection onto the normalized essential space is $U \Sigma V^T$, where $\Sigma = \text{diag}\{1, 1, 0\}$.

3. **Recover the displacement from the essential matrix**
   We now need only U and V to extract R and T from the essential matrix as

   $$R = U R_Z^T \left(\pm \frac{\pi}{2}\right) V^T, \quad \widehat{T} = U R_Z \left(\pm \frac{\pi}{2}\right) \Sigma U^T.$$

   where $R_Z^T \left(\pm \frac{\pi}{2}\right) \doteq \begin{bmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$.

Then the essential matrix is

$$E = \widehat{T} R = \begin{bmatrix} 0 & 0 & 0 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 0 & 2 & 0 \end{bmatrix}.$$

Since $\|T\| = 2$, the E obtained here is not normalized. It is also easy to see this from its SVD,

$$E = U \Sigma V^T \doteq \begin{bmatrix} 0 & 0 & -1 \\ -1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} -\frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \\ 0 & 1 & 0 \\ \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \end{bmatrix}^T,$$

where the nonzero singular values are 2 instead of 1. Normalizing E is equivalent to replacing the above Σ by

$$\Sigma = \text{diag}\{1, 1, 0\}.$$

It is then easy to compute the four possible decompositions $(R, \widehat{T})$ for E:

1. $UR_Z^T\left(\frac{\pi}{2}\right)V^T = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \\ 0 & -1 & 0 \\ \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \end{bmatrix}$, $UR_Z\left(\frac{\pi}{2}\right)\Sigma U^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$;
2. $UR_Z^T\left(\frac{\pi}{2}\right)V^T = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \\ 0 & -1 & 0 \\ \frac{\sqrt{2}}{2} & 0 & -\frac{\sqrt{2}}{2} \end{bmatrix}$, $UR_Z\left(-\frac{\pi}{2}\right)\Sigma U^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$;
3. $UR_Z^T\left(-\frac{\pi}{2}\right)V^T = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \\ 0 & 1 & 0 \\ -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \end{bmatrix}$, $UR_Z\left(-\frac{\pi}{2}\right)\Sigma U^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}$;
4. $UR_Z^T\left(-\frac{\pi}{2}\right)V^T = \begin{bmatrix} \frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \\ 0 & 1 & 0 \\ -\frac{\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} \end{bmatrix}$, $UR_Z\left(\frac{\pi}{2}\right)\Sigma U^T = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix}$.

Clearly, the third solution is exactly the original motion $(R, \widehat{T})$ except that the translation T is recovered up to a scalar factor (i.e. it is normalized to unit norm). ■

Despite its simplicity, the above algorithm, when used in practice, suffers from some shortcomings that are discussed below.

Number of points

The number of points, eight, assumed by the algorithm, is mostly for convenience and simplicity of presentation. In fact, the matrix E (as a function of (R, T)) has only a total of five degrees of freedom: three for rotation and two for translation (up to a scalar factor). By utilizing some additional algebraic properties of E, we may reduce the necessary number of points. For instance, knowing $\det(E) = 0$, we may relax the condition $\text{rank}(\chi) = 8$ to $\text{rank}(\chi) = 7$, and get two solutions E_1^s and $E_2^s \in \mathbb{R}^9$ from the null space of χ. Nevertheless, there is usually only one $\alpha \in \mathbb{R}$ such that

$$\det(E_1 + \alpha E_2) = 0.$$

Therefore, seven points is all we need to have a relatively simpler algorithm. As shown in Exercise 5.13, in fact, a linear algorithm exists for only six points if more complicated algebraic properties of the essential matrix are used. Hence, it should not be a surprise, as shown by [Kruppa, 1913], that one needs only five points in general position to recover (R, T). It can be shown that there are up to ten (possibly complex) solutions, though the solutions are not obtainable in closed form. Furthermore, for many special motions, one needs only up to four points to determine the associated essential matrix. For instance, planar motions (Exercise 5.6) and motions induced from symmetry (Chapter 10) have this nice property.

Number of solutions and positive depth constraint

Since both E and $-E$ satisfy the same set of epipolar constraints, they in general give rise to $2 \times 2 = 4$ possible solutions for (R, T). However, this does not pose a problem, because only one of the solutions guarantees that the depths of all the 3-D points reconstructed are *positive* with respect to both camera frames. That is, in general, three out of the four solutions will be physically impossible and hence may be discarded (see Exercise 5.11).

Structure requirement: general position

In order for the above algorithm to work properly, the condition that the given eight points be in "general position" is very important. It can be easily shown that if these points form certain degenerate configurations, called critical surfaces, the algorithm will fail (see Exercise 5.14). A case of some practical importance occurs when all the points happen to lie on the same 2-D plane in $\mathbb{R}^3$. We will discuss the geometry for the planar case in Section 5.3, and also later within the context of multiple-view geometry (Chapter 9).

Motion requirement: sufficient parallax

In the derivation of the epipolar constraint we have implicitly assumed that $E \neq 0$, which allowed us to derive the eight-point algorithm where the essential matrix is normalized to $\|E\| = 1$. Due to the structure of the essential matrix, $E = 0 \Leftrightarrow T = 0$. Therefore, the eight-point algorithm requires that the translation (or baseline) $T \neq 0$. The translation T induces parallax in the image plane. In practice, due to noise, the algorithm will likely return an answer even when there is no translation. However, in this case the estimated direction of translation will be meaningless. Therefore, one needs to exercise caution to make sure that there is "sufficient parallax" for the algorithm to be well conditioned. It has been observed experimentally that even for purely rotational motion, i.e. $T = 0$, the "spurious" translation created by noise in the image measurements is sufficient for the eight-point algorithm to return a correct estimate of R.

Infinitesimal viewpoint change

It is often the case in applications that the two views described in this chapter are taken by a moving camera rather than by two static cameras. The derivation of the epipolar constraint and the associated eight-point algorithm does not change, as long as the two vantage points are distinct. In the limit that the two viewpoints come infinitesimally close, the epipolar constraint takes a related but different form called the continuous epipolar constraint, which we will study in Section 5.4. The continuous case is typically of more significance for applications in robot vision, where one is often interested in recovering the linear and angular velocities of the camera.

Multiple motion hypotheses

In the case of multiple moving objects in the scene, image points may no longer satisfy the same epipolar constraint. For example, if we know that there are two independent moving objects with motions, say (R^1, T^1) and (R^2, T^2), then the two images $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ of a point p on one of these objects should satisfy instead the equation

$$(\boldsymbol{x}_2^T E^1 \boldsymbol{x}_1)(\boldsymbol{x}_2^T E^2 \boldsymbol{x}_1) = 0, \tag{5.16}$$

corresponding to the fact that the point p moves according to either motion 1 or motion 2. Here $E^1 = \widehat{T^1} R^1$ and $E^2 = \widehat{T^2} R^2$. As we will see, from this equation it is still possible to recover E^1 and E^2 if enough points are visible on either object. Generalizing to more than two independent motions requires some attention; we will study the multiple-motion problem in Chapter 7.

5.2.2 Euclidean constraints and structure reconstruction

The eight-point algorithm just described uses as input a set of eight or more point correspondences and returns the relative pose (rotation and translation) between the two cameras up to an arbitrary scale $\gamma \in \mathbb{R}^+$. Without loss of generality, we may assume this scale to be $\gamma = 1$, which is equivalent to scaling translation to unit length. Relative pose and point correspondences can then be used to retrieve the position of the points in 3-D by recovering their depths relative to each camera frame.

Consider the basic rigid-body equation, where the pose (R, T) has been recovered, with the translation T defined up to the scale γ. In terms of the images and the depths, it is given by

$$\lambda_2^j \boldsymbol{x}_2^j = \lambda_1^j R \boldsymbol{x}_1^j + \gamma T, \quad j = 1, 2, \ldots, n. \tag{5.17}$$

Notice that since (R, T) are known, the equations given by (5.17) are linear in both the structural scale λ's and the motion scale γ's, and therefore they can be easily solved. For each point, λ_1, λ_2 are its depths with respect to the first and second camera frames, respectively. One of them is therefore redundant; for instance, if λ_1 is known, λ_2 is simply a function of (R, T). Hence we can eliminate, say, λ_2 from the above equation by multiplying both sides by $\widehat{\boldsymbol{x}_2}$, which yields

$$\lambda_1^j \widehat{\boldsymbol{x}_2^j} R \boldsymbol{x}_1^j + \gamma \widehat{\boldsymbol{x}_2^j} T = 0, \quad j = 1, 2, \ldots, n. \tag{5.18}$$

This is equivalent to solving the linear equation

$$M^j \bar{\lambda}^j \doteq \left[\widehat{\boldsymbol{x}_2^j} R \boldsymbol{x}_1^j, \ \widehat{\boldsymbol{x}_2^j} T \right] \begin{bmatrix} \lambda_1^j \\ \gamma \end{bmatrix} = 0, \tag{5.19}$$

where $M^j = \left[\widehat{\boldsymbol{x}_2^j} R \boldsymbol{x}_1^j, \ \widehat{\boldsymbol{x}_2^j} T \right] \in \mathbb{R}^{3 \times 2}$ and $\bar{\lambda}^j = [\lambda_1^j, \gamma]^T \in \mathbb{R}^2$, for $j = 1, 2, \ldots, n$. In order to have a unique solution, the matrix M^j needs to be of

rank 1. This is not the case only when $\widehat{\boldsymbol{x}_2}T = 0$, i.e. when the point p lies on the line connecting the two optical centers o_1 and o_2.

Notice that all the n equations above share the same γ; we define a vector $\vec{\lambda} = [\lambda_1^1, \lambda_1^2, \ldots, \lambda_1^n, \gamma]^T \in \mathbb{R}^{n+1}$ and a matrix $M \in \mathbb{R}^{3n\times(n+1)}$ as

$$M \doteq \begin{bmatrix} \widehat{\boldsymbol{x}_2^1}R\boldsymbol{x}_1^1 & 0 & 0 & 0 & 0 & \widehat{\boldsymbol{x}_2^1}T \\ 0 & \widehat{\boldsymbol{x}_2^2}R\boldsymbol{x}_1^2 & 0 & 0 & 0 & \widehat{\boldsymbol{x}_2^2}T \\ 0 & 0 & \ddots & 0 & 0 & \vdots \\ 0 & 0 & 0 & \widehat{\boldsymbol{x}_2^{n-1}}R\boldsymbol{x}_1^{n-1} & 0 & \widehat{\boldsymbol{x}_2^{n-1}}T \\ 0 & 0 & 0 & 0 & \widehat{\boldsymbol{x}_2^n}R\boldsymbol{x}_1^n & \widehat{\boldsymbol{x}_2^n}T \end{bmatrix}. \quad (5.20)$$

Then the equation

$$M\vec{\lambda} = 0 \quad (5.21)$$

determines all the unknown depths *up to a single universal scale*. The linear least-squares estimate of $\vec{\lambda}$ is simply the eigenvector of M^TM that corresponds to its smallest eigenvalue. Note that this scale ambiguity is intrinsic, since without any prior knowledge about the scene and camera motion, one cannot disambiguate whether the camera moved twice the distance while looking at a scene twice larger but two times further away.

5.2.3 Optimal pose and structure

The eight-point algorithm given in the previous section assumes that *exact* point correspondences are given. In the presence of noise in image correspondences, we have suggested possible ways of estimating the essential matrix by solving a least-squares problem followed by a projection onto the essential space. But in practice, this will not be satisfying in at least two respects:

1. There is no guarantee that the estimated pose (R, T), is as close as possible to the true solution.
2. Even if we were to accept such an (R, T), a noisy image pair, say $(\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2)$, would not necessarily give rise to a consistent 3-D reconstruction, as shown in Figure 5.6.

At this stage of development, we do not want to bring in all the technical details associated with optimal estimation, since they would bury the geometric intuition. We will therefore discuss only the key ideas, and leave the technical details to Appendix 5.A as well as Chapter 11, where we will address more practical issues.

Choice of optimization objectives

Recall from Chapter 3 that a calibrated camera can be described as a plane perpendicular to the z-axis at a distance 1 from the origin; therefore, the coordinates of image points $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ are of the form $[x, y, 1]^T \in \mathbb{R}^3$. In practice, we cannot

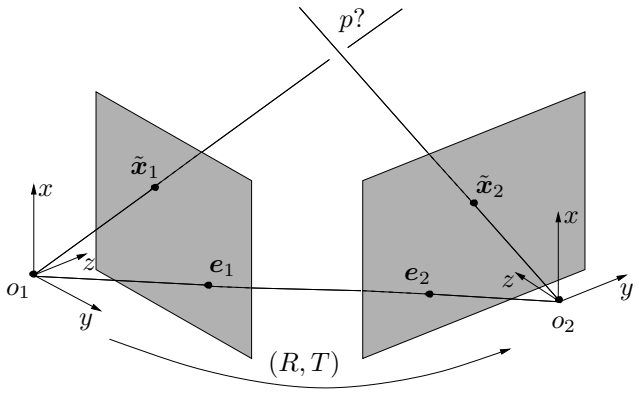

Figure 5.6. Rays extended from a noisy image pair $\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2 \in \mathbb{R}^3$ do not intersect at any point p in 3-D if they do not satisfy the epipolar constraint precisely.

measure the actual coordinates but only their noisy versions, say

$$\tilde{\boldsymbol{x}}_1^j = \boldsymbol{x}_1^j + w_1^j, \quad \tilde{\boldsymbol{x}}_2^j = \boldsymbol{x}_2^j + w_2^j, \quad j = 1, 2, \ldots, n, \tag{5.22}$$

where $\boldsymbol{x}_1^j$ and $\boldsymbol{x}_2^j$ denote the "ideal" image coordinates and $w_1^j = [w_{11}^j,\ w_{12}^j,\ 0]^T$ and $w_2^j = [w_{21}^j,\ w_{22}^j,\ 0]^T$ are localization errors in the correspondence. Notice that it is the (unknown) ideal image coordinates $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$ that satisfy the epipolar constraint $\boldsymbol{x}_2^{jT} \widehat{T} R \boldsymbol{x}_1^j = 0$, and *not* the (measured) noisy ones $(\tilde{\boldsymbol{x}}_1^j, \tilde{\boldsymbol{x}}_2^j)$. One could think of the ideal coordinates as a "model," and w_i^j as the discrepancy between the model and the measurements: $\tilde{\boldsymbol{x}}_i^j = \boldsymbol{x}_i^j + w_i^j$. Therefore, in general, we seek the parameters $(\boldsymbol{x}, R, T)$ that minimize the discrepancy between the model and the data, i.e. w_i^j. In order to do so, we first need to decide how to *evaluate* the discrepancy, which determines the choice of optimization objective.

Unfortunately, there is no "correct," uncontroversial, universally accepted objective function, and the choice of discrepancy measure is part of the design process, since it depends on what assumptions are made on the *residuals* w_i^j. Different assumptions result in different choices of discrepancy measures, which eventually result in different "optimal" solutions $(\boldsymbol{x}^*, R^*, T^*)$.

For instance, one may assume that $w = \{w_i^j\}$ are samples from a distribution that depends on the unknown *parameters* $(\boldsymbol{x}, R, T)$, which are considered deterministic but unknown. In this case, based on the model generating the data, one can derive an expression of the *likelihood function* $p(w|\boldsymbol{x}, R, T)$ and choose to maximize it (or, more conveniently, its logarithm) with respect to the unknown parameters. Then the "optimal solution," in the sense of *maximum likelihood*, is given by

$$(\boldsymbol{x}^*, R^*, T^*) = \arg\max \phi_{ML}(\boldsymbol{x}, R, T) \doteq \sum_{i,j} \log p\big((\tilde{\boldsymbol{x}}_i^j - \boldsymbol{x}_i^j)|\boldsymbol{x}, R, T\big).$$

Naturally, different likelihood functions can result in very different optimal solutions. Indeed, there is no guarantee that the maximum is unique, since p can

be multimodal, and therefore there may be several choices of parameters that achieve the maximum. Constructing the likelihood function for the location of point features from first principles, starting from the noise characteristics of the photosensitive elements of the sensor, is difficult because of the many nonlinear steps involved in feature detection and tracking. Therefore, it is common to assume that the likelihood belongs to a family of density functions, the most popular choice being the normal (or Gaussian) distribution.

Sometimes, however, one may have reasons to believe that $(\boldsymbol{x}, R, T)$ are not just unknown parameters that can take any value. Instead, even before any measurement is gathered, one can say that some values are more probable than others, a fact that can be described by a joint *a priori* probability density (or *prior*) $p(\boldsymbol{x}, R, T)$. For instance, for a robot navigating on a flat surface, rotation about the horizontal axis may be very improbable, as would translation along the vertical axis. When combined with the likelihood function, the prior can be used to determine the *a posteriori density* , or *posterior* $p(\boldsymbol{x}, R, T|\{\tilde{\boldsymbol{x}}_i^j\})$ using Bayes rule. In this case, one may seek the maximum of the posterior *given* the value of the measurements. This is the *maximum a posteriori* estimate

$$(\boldsymbol{x}^*, R^*, T^*) = \arg\max \phi_{MAP}(\boldsymbol{x}, R, T) \doteq p(\boldsymbol{x}, R, T|\{\tilde{\boldsymbol{x}}_i^j\}).$$

Although this choice has several advantages, in our case it requires defining a probability density on the space of camera poses $SO(3) \times \mathbb{S}^2$, which has a nontrivial geometric structure. This is well beyond the scope of this book, and we will therefore not discuss this criterion further here.

In what follows, we will take a more minimalistic approach to optimality, and simply assume that $\{w_i^j\}$ are unknown values ("errors," or "residuals") whose norms need to be minimized. In this case, we do not postulate any probabilistic description, and we simply seek $(\boldsymbol{x}^*, R^*, T^*) = \arg\min \phi(\boldsymbol{x}, R, T)$, where ϕ is, for instance, the squared 2-norm:

$$\phi(\boldsymbol{x}, R, T) \doteq \sum_j \|w_1^j\|_2^2 + \|w_2^j\|_2^2 = \sum_j \|\tilde{\boldsymbol{x}}_1^j - \boldsymbol{x}_1^j\|_2^2 + \|\tilde{\boldsymbol{x}}_2^j - \boldsymbol{x}_2^j\|_2^2.$$

This corresponds to a *least-squares estimator*. Since $\boldsymbol{x}_1^j$ and $\boldsymbol{x}_2^j$ are the recovered 3-D points projected back onto the image planes, the above criterion is often called the "reprojection error."

However, the unknowns for the above minimization problem are not completely free; for example, they need to satisfy the epipolar constraint $\boldsymbol{x}_2^T \widehat{T} R \boldsymbol{x}_1 = 0$. Hence, with the choice of the least-squares criterion, we can pose the problem of reconstruction as a constrained optimization: given $\tilde{\boldsymbol{x}}_i^j, i = 1, 2,\ j = 1, 2, \ldots, n$, minimize

$$\phi(\boldsymbol{x}, R, T) \doteq \sum_{j=1}^{n} \sum_{i=1}^{2} \|\tilde{\boldsymbol{x}}_i^j - \boldsymbol{x}_i^j\|_2^2 \tag{5.23}$$

subject to

$$\boldsymbol{x}_2^{jT} \widehat{T} R \boldsymbol{x}_1^j = 0, \quad \boldsymbol{x}_1^{jT} e_3 = 1, \quad \boldsymbol{x}_2^{jT} e_3 = 1, \quad j = 1, 2, \ldots, n. \tag{5.24}$$

Using Lagrange multipliers (Appendix C), we can convert this constrained optimization problem to an unconstrained one. Details on how to carry out the optimization are outlined in Appendix 5.A.

Remark 5.12 (Equivalence to bundle adjustment). *The reader may have noticed that the depth parameters λ_i, despite being unknown, are missing from the optimization problem of equation (5.24). This is not an oversight: indeed, the depth parameters play the role of Lagrange multipliers in the constrained optimization problem described above, and therefore they enter the optimization problem indirectly. Alternatively, one can write the optimization problem in unconstrained form:*

$$\sum_{j=1}^{n} \left\|\tilde{\boldsymbol{x}}_1^j - \pi_1(\boldsymbol{X}^j)\right\|_2^2 + \left\|\tilde{\boldsymbol{x}}_2^j - \pi_2(\boldsymbol{X}^j)\right\|_2^2, \tag{5.25}$$

where π_1 and π_2 denote the projection of a point $\boldsymbol{X}$ in space onto the first and second images, respectively. If we choose the first camera frame as the reference, then the above expression can be simplified to[6]

$$\phi(\boldsymbol{x}_1, R, T, \lambda) = \sum_{j=1}^{n} \left\|\tilde{\boldsymbol{x}}_1^j - \boldsymbol{x}_1^j\right\|_2^2 + \left\|\tilde{\boldsymbol{x}}_2^j - \pi(R\lambda_1^j \boldsymbol{x}_1^j + T)\right\|_2^2. \tag{5.26}$$

Minimizing the above expression with respect to the unknowns $(R, T, \boldsymbol{x}_1, \lambda)$ is known in the literature as bundle adjustment*. Bundle adjustment and the constrained optimization described above are simply two different ways to parameterize the same optimization objective. As we will see in Appendix 5.A, the constrained form better highlights the geometric structure of the problem, and serves as a guide to develop effective approximations.*

In the remainder of this section, we limit ourselves to describing a simplified cost functional that approximates the reprojection error resulting in simpler optimization algorithms, while retaining a strong geometric interpretation. In this approximation, the unknown $\boldsymbol{x}$ is approximated by the measured $\tilde{\boldsymbol{x}}$, so that the cost function ϕ depends only on camera pose (R, T) (see Appendix 5.A for more details):

$$\phi(R, T) \doteq \sum_{j=1}^{n} \frac{(\tilde{\boldsymbol{x}}_2^{jT} \widehat{T} R \tilde{\boldsymbol{x}}_1^j)^2}{\|\widehat{e}_3 \widehat{T} R \tilde{\boldsymbol{x}}_1^j\|^2} + \frac{(\tilde{\boldsymbol{x}}_2^{jT} \widehat{T} R \tilde{\boldsymbol{x}}_1^j)^2}{\|\tilde{\boldsymbol{x}}_2^{jT} \widehat{T} R \widehat{e}_3^T\|^2}. \tag{5.27}$$

Geometrically, this expression can be interpreted as distances from the image points $\tilde{\boldsymbol{x}}_1^j$ and $\tilde{\boldsymbol{x}}_2^j$ to corresponding epipolar lines in the two image planes, respectively, as shown in Figure 5.7. For instance, the reader can verify as an exercise

[6] Here we use π to denote the standard planar projection introduced in Chapter 3: $[X, Y, Z]^T \mapsto [X/Z, Y/Z, 1]^T$.

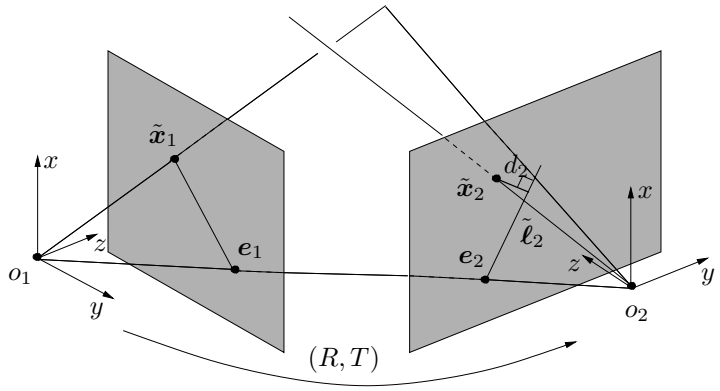

Figure 5.7. Two noisy image points $\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2 \in \mathbb{R}^3$. Here $\tilde{\boldsymbol{\ell}}_2$ is an epipolar line that is the intersection of the second image plane with the epipolar plane. The distance d_2 is the geometric distance between the second image point $\tilde{\boldsymbol{x}}_2$ and the epipolar line. Symmetrically, one can define a similar geometric distance d_1 in the first image plane.

(Exercise 5.12) that following the notation in the figure, we have

$$d_2^2 = \frac{(\tilde{\boldsymbol{x}}_2^T \widehat{T} R \tilde{\boldsymbol{x}}_1)^2}{\|\widehat{e}_3 \widehat{T} R \tilde{\boldsymbol{x}}_1\|^2}.$$

In the presence of noise, minimizing the above objective function, although more difficult, improves the results of the linear eight-point algorithm.

Example 5.13 (Comparison with the linear algorithm). Figure 5.8 demonstrates the effect of the optimization: numerical simulations were run for both the linear eight-point algorithm and the nonlinear optimization. Values of the objective function $\phi(R, T)$ at different T are plotted (with R fixed at the ground truth); "+" denotes the true translation T, "$*$" is the estimated T from the linear eight-point algorithm, and "$\circ$" is the estimated T by upgrading the linear algorithm result with the optimization. ■

Structure triangulation

If we were given the optimal estimate of camera pose (R, T), obtained, for instance, from Algorithm 5.5 in Appendix 5.A, we can find a pair of images $(\boldsymbol{x}_1^*, \boldsymbol{x}_2^*)$ that satisfy the epipolar constraint $\boldsymbol{x}_2^T \widehat{T} R \boldsymbol{x}_1 = 0$ and minimize the (reprojection) error

$$\phi(\boldsymbol{x}) = \|\tilde{\boldsymbol{x}}_1 - \boldsymbol{x}_1\|^2 + \|\tilde{\boldsymbol{x}}_2 - \boldsymbol{x}_2\|^2. \tag{5.28}$$

This is called the *triangulation problem*. The key to its solution is to find what exactly the reprojection error depends on, which can be more easily explained geometrically by Figure 5.9. As we see from the figure, the value of the reprojection error depends only on the position of the epipolar plane P: when the plane P rotates around the baseline (o_1, o_2), the image pair $(\boldsymbol{x}_1, \boldsymbol{x}_2)$, which minimizes the distance $\|\tilde{\boldsymbol{x}}_1 - \boldsymbol{x}_1\|^2 + \|\tilde{\boldsymbol{x}}_2 - \boldsymbol{x}_2\|^2$, changes accordingly, and so does the error. To

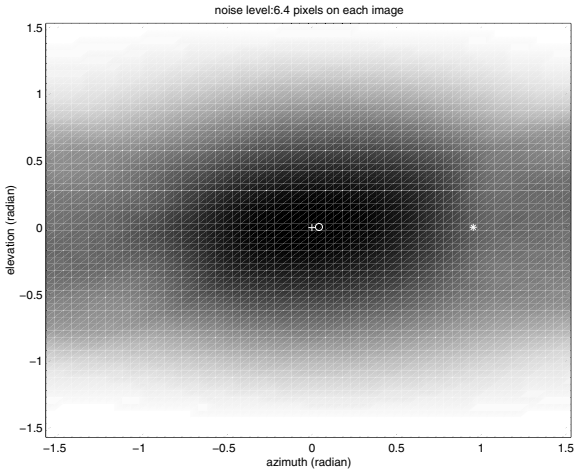

Figure 5.8. Improvement by nonlinear optimization. A two-dimensional projection of the five-dimensional residual function $\phi(R,T)$ is shown in greyscale. The residual corresponds to the two-dimensional function $\phi(\hat{R},T)$ with rotation fixed at the true value. The location of the solution found by the linear algorithm is shown as "$*$," and it can be seen that it is quite far from the true minimum (darkest point in the center of the image, marked by "+").The solution obtained by nonlinear optimization is marked by "$\circ$," which shows a significant improvement.

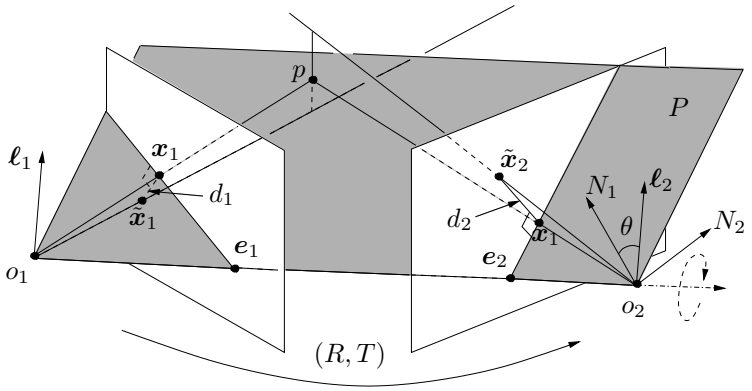

Figure 5.9. For a fixed epipolar plane P, the pair of images $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ that minimize the reprojection error $d_1^2 + d_2^2$ must be points on the two epipolar lines and closest to $\tilde{\boldsymbol{x}}_1, \tilde{\boldsymbol{x}}_2$, respectively. Hence the reprojection error is a function only of the position of the epipolar plane P.

parameterize the position of the epipolar plane, let $(\boldsymbol{e}_2, N_1, N_2)$ be an orthonormal basis in the second camera frame. Then P is determined by its normal vector $\boldsymbol{\ell}_2$ (with respect to the second camera frame), which in turn is determined by the angle θ between $\boldsymbol{\ell}_2$ and N_1 (Figure 5.9). Hence the reprojection error ϕ should be

a function that depends only on θ. There is typically only one θ^* that minimizes the error $\phi(\theta)$. Once it is found, the corresponding image pair $(\boldsymbol{x}_1^*, \boldsymbol{x}_2^*)$ and 3-D point p are determined. Details of the related algorithm can be found in Appendix 5.A.

5.3 Planar scenes and homography

In order for the eight-point algorithm to give a unique solution (up to a scalar factor) for the camera motion, it is crucial that the feature points in 3-D be in general position. When the points happen to form certain degenerate configurations, the solution might no longer be unique. Exercise 5.14 explains why this may occur when all the feature points happen to lie on certain 2-D surfaces, called critical surfaces.[7] Many of these critical surfaces occur rarely in practice, and their importance is limited. However, 2-D planes, which happen to be a special case of critical surfaces, are ubiquitous in man-made environments and in aerial imaging.

Therefore, if one applies the eight-point algorithm to images of points all lying on the same 2-D plane, the algorithm will fail to provide a unique solution (as we will soon see why). On the other hand, in many applications, a scene can indeed be approximately planar (e.g., the landing pad for a helicopter) or piecewise planar (e.g., the corridors inside a building). We therefore devote this section to this special but important case.

5.3.1 Planar homography

Let us consider two images of points p on a 2-D plane P in 3-D space. For simplicity, we will assume throughout the section that the optical center of the camera never passes through the plane.

Now suppose that two images $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ are given for a point $p \in P$ with respect to two camera frames. Let the coordinate transformation between the two frames be

$$\boldsymbol{X}_2 = R\boldsymbol{X}_1 + T, \tag{5.29}$$

where $\boldsymbol{X}_1, \boldsymbol{X}_2$ are the coordinates of p relative to camera frames 1 and 2, respectively. As we have already seen, the two images $\boldsymbol{x}_1, \boldsymbol{x}_2$ of p satisfy the epipolar constraint

$$\boldsymbol{x}_2^T E \boldsymbol{x}_1 = \boldsymbol{x}_2^T \widehat{T} R \boldsymbol{x}_1 = 0.$$

However, for points on the same plane P, their images will share an extra constraint that makes the epipolar constraint alone no longer sufficient.

[7] In general, such critical surfaces can be described by certain quadratic equations in the X, Y, Z coordinates of the point, hence are often referred to as quadratic surfaces.

Let $N = [n_1, n_2, n_2]^T \in \mathbb{S}^2$ be the unit normal vector of the plane P with respect to the first camera frame, and let $d > 0$ denote the distance from the plane P to the optical center of the first camera. Then we have

$$N^T \boldsymbol{X}_1 = n_1 X + n_2 Y + n_3 Z = d \quad \Leftrightarrow \quad \frac{1}{d} N^T \boldsymbol{X}_1 = 1, \quad \forall \boldsymbol{X}_1 \in P. \tag{5.30}$$

Substituting equation (5.30) into equation (5.29) gives

$$\boldsymbol{X}_2 = R\boldsymbol{X}_1 + T = R\boldsymbol{X}_1 + T\frac{1}{d} N^T \boldsymbol{X}_1 = \left(R + \frac{1}{d} T N^T \right) \boldsymbol{X}_1. \tag{5.31}$$

We call the matrix

$$H \doteq R + \frac{1}{d} T N^T \quad \in \mathbb{R}^{3\times 3} \tag{5.32}$$

the *(planar) homography matrix*, since it denotes a linear transformation from $\boldsymbol{X}_1 \in \mathbb{R}^3$ to $\boldsymbol{X}_2 \in \mathbb{R}^3$ as

$$\boldsymbol{X}_2 = H\boldsymbol{X}_1.$$

Note that the matrix H depends on the motion parameters $\{R, T\}$ as well as the structure parameters $\{N, d\}$ of the plane P. Due to the inherent scale ambiguity in the term $\frac{1}{d}T$ in equation (5.32), one can at most expect to recover from H the ratio of the translation T scaled by the distance d. From

$$\lambda_1 \boldsymbol{x}_1 = \boldsymbol{X}_1, \quad \lambda_2 \boldsymbol{x}_2 = \boldsymbol{X}_2, \quad \boldsymbol{X}_2 = H\boldsymbol{X}_1, \tag{5.33}$$

we have

$$\lambda_2 \boldsymbol{x}_2 = H\lambda_1 \boldsymbol{x}_1 \quad \Leftrightarrow \quad \boldsymbol{x}_2 \sim H\boldsymbol{x}_1, \tag{5.34}$$

where we recall that $\sim$ indicates equality up to a scalar factor. Often, the equation

$$\boxed{\boldsymbol{x}_2 \sim H\boldsymbol{x}_1} \tag{5.35}$$

itself is referred to as a *(planar) homography* mapping induced by a plane P. Despite the scale ambiguity, as illustrated in Figure 5.10, H introduces a special map between points in the first image and those in the second in the following sense:

1. For any point $\boldsymbol{x}_1$ in the first image that is the image of some point, say p on the plane P, its corresponding second image $\boldsymbol{x}_2$ is uniquely determined as $\boldsymbol{x}_2 \sim H\boldsymbol{x}_1$, since for any other point, say $\boldsymbol{x}_2'$, on the same epipolar line $\boldsymbol{\ell}_2 \sim E\boldsymbol{x}_1 \in \mathbb{R}^3$, the ray $o_2\boldsymbol{x}_2'$ will intersect the ray $o_1\boldsymbol{x}_1$ at a point p' out of the plane.
2. On the other hand, if $\boldsymbol{x}_1$ is the image of some point, say p', not on the plane P, then $\boldsymbol{x}_2 \sim H\boldsymbol{x}_1$ is only a point that is on the same epipolar line $\boldsymbol{\ell}_2 \sim E\boldsymbol{x}_1$ as its actual corresponding image $\boldsymbol{x}_2'$. That is, $\boldsymbol{\ell}_2^T \boldsymbol{x}_2 = \boldsymbol{\ell}_2^T \boldsymbol{x}_2' = 0$.

We hence have the following result:

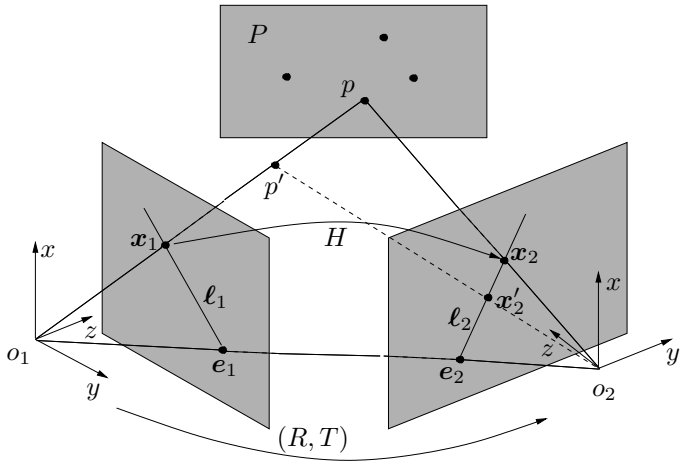

Figure 5.10. Two images $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^3$ of a 3-D point p on a plane P. They are related by a homography H that is induced by the plane.

Proposition 5.14 (Homography for epipolar lines). *Given a homography H (induced by plane P in 3-D) between two images, for any pair of corresponding images $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ of a 3-D point p that is not necessarily on P, the associated epipolar lines are*

$$\boxed{\boldsymbol{\ell}_2 \sim \widehat{\boldsymbol{x}}_2 H \boldsymbol{x}_1, \quad \boldsymbol{\ell}_1 \sim H^T \boldsymbol{\ell}_2.} \tag{5.36}$$

Proof. If p is not on P, the first equation is true from point 2 in above discussion. Note that for points on the plane P, $\boldsymbol{x}_2 = H\boldsymbol{x}_1$ implies $\widehat{\boldsymbol{x}}_2 H \boldsymbol{x}_1 = 0$, and the first equation is still true as long as we adopt the convention that $v \sim 0$, $\forall v \in \mathbb{R}^3$. The second equation is easily proven using the definition of a line $\boldsymbol{\ell}^T \boldsymbol{x} = 0$. □

This property of the homography allows one to compute epipolar lines without knowing the essential matrix. We will explore further the relationships between the essential matrix and the planar homography in Section 5.3.4.

In addition to the fact that the homography matrix H encodes information about the camera motion and the scene structure, knowing it directly facilitates establishing correspondence between points in the first and the second images. As we will see soon, H can be computed in general from a small number of corresponding image pairs. Once H is known, correspondence between images of other points on the same plane can then be fully established, since the corresponding location $\boldsymbol{x}_2$ for an image point $\boldsymbol{x}_1$ is simply $H\boldsymbol{x}_1$. Proposition 5.14 suggests that correspondence between images of points not on the plane can also be established, since H contains information about the epipolar lines.

5.3.2 Estimating the planar homography matrix

In order to further eliminate the unknown scale in equation (5.35), multiplying both sides by the skew-symmetric matrix $\widehat{\boldsymbol{x}_2} \in \mathbb{R}^{3\times 3}$, we obtain the equation

$$\boxed{\widehat{\boldsymbol{x}_2} H \boldsymbol{x}_1 = 0.} \tag{5.37}$$

We call this equation the *planar epipolar constraint*, or also the *(planar) homography constraint*.

Remark 5.15 (Plane as a critical surface). *In the planar case, since $\boldsymbol{x}_2 \sim H\boldsymbol{x}_1$, for any vector $u \in \mathbb{R}^3$, we have that $u \times \boldsymbol{x}_2 = \widehat{u}\boldsymbol{x}_2$ is orthogonal to $H\boldsymbol{x}_1$. Hence we have*

$$\boldsymbol{x}_2^T \widehat{u} H \boldsymbol{x}_1 = 0, \quad \forall u \in \mathbb{R}^3.$$

That is, $\boldsymbol{x}_2^T E \boldsymbol{x}_1 = 0$ for a family of matrices $E = \widehat{u}H \in \mathbb{R}^{3\times 3}$ besides the essential matrix $E = \widehat{T}R$. This explains why the eight-point algorithm does not apply to feature points from a planar scene.

Example 5.16 (Homography from a pure rotation). The homographic relation $\boldsymbol{x}_2 \sim H\boldsymbol{x}_1$ also shows up when the camera is purely rotating, i.e. $\boldsymbol{X}_2 = R\boldsymbol{X}_1$. In this case, the homography matrix H becomes $H = R$, since $T = 0$. Consequently, we have the constraint

$$\widehat{\boldsymbol{x}_2} R \boldsymbol{x}_1 = 0.$$

One may view this as a special planar scene case, since without translation, information about the depth of the scene is completely lost in the images, and one might as well interpret the scene to be planar (e.g., all the points lie on a plane infinitely far away). As the distance of the plane d goes to infinity, $\lim_{d\to\infty} H = R$.

The homography from purely rotational motion can be used to construct image mosaics of the type shown in Figure 5.11. For additional references on how to construct panoramic mosaics the reader can refer to [Szeliski and Shum, 1997, Sawhney and Kumar, 1999], where the latter includes compensation for radial distortion. ■

Figure 5.11. Mosaic from the rotational homography.

Since equation (5.37) is *linear* in H, by stacking the entries of H as a vector,

$$H^s \doteq [H_{11}, H_{21}, H_{31}, H_{12}, H_{22}, H_{32}, H_{13}, H_{23}, H_{33}]^T \quad \in \mathbb{R}^9, \tag{5.38}$$

we may rewrite equation (5.37) as

$$\boldsymbol{a}^T H^s = 0,$$

where the matrix $\boldsymbol{a} \doteq \boldsymbol{x}_1 \otimes \widehat{\boldsymbol{x}_2} \in \mathbb{R}^{9\times 3}$ is the Kronecker product of $\widehat{\boldsymbol{x}_2}$ and $\boldsymbol{x}_1$ (see Appendix A.1.3).

Since the matrix $\widehat{\boldsymbol{x}_2}$ is only of rank 2, so is the matrix $\boldsymbol{a}$. Thus, even though the equation $\widehat{\boldsymbol{x}_2} H \boldsymbol{x}_1 = 0$ has three rows, it only imposes two independent constraints on H. With this notation, given n pairs of images $\{(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)\}_{j=1}^n$ from points on the same plane P, by defining $\chi \doteq [\boldsymbol{a}^1, \boldsymbol{a}^2, \ldots, \boldsymbol{a}^n]^T \in \mathbb{R}^{3n\times 9}$, we may combine all the equations (5.37) for all the image pairs and rewrite them as

$$\chi H^s = 0. \tag{5.39}$$

In order to solve uniquely (up to a scalar factor) for H^s, we must have $\text{rank}(\chi) = 8$. Since each pair of image points gives two constraints, we expect that at least four point correspondences would be necessary for a unique estimate of H. We leave the proof of the following statement as an exercise to the reader.

Proposition 5.17 (Four-point homography). *We have $rank(\chi) = 8$ if and only if there exists a set of four points (out of the n) such that no three of them are collinear; i.e. they are in a general configuration in the plane.*

Thus, if there are more than four image correspondences of which no three in each image are collinear, we may apply standard linear least-squares estimation to find $\min \|\chi H^s\|^2$ to recover H up to a scalar factor. That is, we are able to recover H of the form

$$H_L \doteq \lambda H = \lambda \left(R + \frac{1}{d} T N^T \right) \quad \in \mathbb{R}^{3\times 3} \tag{5.40}$$

for some (unknown) scalar factor λ.

Knowing H_L, the next thing is obviously to determine the scalar factor λ by taking into account the structure of H.

Lemma 5.18 (Normalization of the planar homography). *For a matrix of the form $H_L = \lambda \left(R + \frac{1}{d} T N^T\right)$, we have*

$$|\lambda| = \sigma_2(H_L), \tag{5.41}$$

where $\sigma_2(H_L) \in \mathbb{R}$ is the second largest singular value of H_L.

Proof. Let $u = \frac{1}{d} R^T T \in \mathbb{R}^3$. Then we have

$$H_L^T H_L = \lambda^2 (I + u N^T + N u^T + \|u\|^2 N N^T).$$

Obviously, the vector $u \times N = \widehat{u} N \in \mathbb{R}^3$, which is orthogonal to both u and N, is an eigenvector and $H_L^T H_L (\widehat{u} N) = \lambda^2 (\widehat{u} N)$. Hence $|\lambda|$ is a singular value of H_L. We only have to show that it is the second largest. Let $v = \|u\| N, w = u/\|u\| \in \mathbb{R}^3$. We have

$$Q = u N^T + N u^T + \|u\|^2 N N^T = (w + v)(w + v)^T - w w^T.$$

The matrix Q has a positive, a negative, and a zero eigenvalue, except that when $u \sim N$, Q will have two repeated zero eigenvalues. In any case, $H_L^T H_L$ has λ^2 as its second-largest eigenvalue. □

Then, if $\{\sigma_1, \sigma_2, \sigma_3\}$ are the singular values of H_L recovered from linear least-squares estimation, we set a new

$$H = H_L/\sigma_2(H_L).$$

This recovers H up to the form $H = \pm\left(R + \frac{1}{d}TN^T\right)$. To get the correct sign, we may use $\lambda_2^j \boldsymbol{x}_2^j = H\lambda_1^j \boldsymbol{x}_1^j$ and the fact that $\lambda_1^j, \lambda_2^j > 0$ to impose the positive depth constraint

$$(\boldsymbol{x}_2^j)^T H \boldsymbol{x}_1^j > 0, \quad \forall j = 1, 2, \ldots, n.$$

Thus, if the points $\{p\}_{j=1}^n$ are in general configuration on the plane, then the matrix $H = \left(R + \frac{1}{d}TN^T\right)$ can be uniquely determined from the image pair.

5.3.3 Decomposing the planar homography matrix

After we have recovered H of the form $H = \left(R + \frac{1}{d}TN^T\right)$, we now study how to decompose such a matrix into its motion and structure parameters, namely $\left\{R, \frac{T}{d}, N\right\}$.

Theorem 5.19 (Decomposition of the planar homography matrix). *Given a matrix $H = \left(R + \frac{1}{d}TN^T\right)$, there are at most two physically possible solutions for a decomposition into parameters $\left\{R, \frac{1}{d}T, N\right\}$ given in Table 5.1.*

Proof. First notice that H preserves the length of any vector orthogonal to N, i.e. if $N \perp a$ for some $a \in \mathbb{R}^3$, we have $\|Ha\|^2 = \|Ra\|^2 = \|a\|^2$. Also, if we know the plane spanned by the vectors that are orthogonal to N, we then know N itself. Let us first recover the vector N based on this knowledge.

The symmetric matrix $H^T H$ will have three eigenvalues $\sigma_1^2 \geq \sigma_2^2 \geq \sigma_3^2 \geq 0$, and from Lemma 5.18 we know that $\sigma_2 = 1$. Since $H^T H$ is symmetric, it can be diagonalized by an orthogonal matrix $V \in SO(3)$ such that

$$H^T H = V \Sigma V^T, \tag{5.42}$$

where $\Sigma = \text{diag}\{\sigma_1^2, \sigma_2^2, \sigma_3^2\}$. If $[v_1, v_2, v_3]$ are the three column vectors of V, we have

$$H^T H v_1 = \sigma_1^2 v_1, \quad H^T H v_2 = v_2, \quad H^T H v_3 = \sigma_3^2 v_3. \tag{5.43}$$

Hence v_2 is orthogonal to both N and T, and its length is preserved under the map H. Also, it is easy to check that the length of two other unit-length vectors defined as

$$u_1 \doteq \frac{\sqrt{1-\sigma_3^2}v_1 + \sqrt{\sigma_1^2 - 1}v_3}{\sqrt{\sigma_1^2 - \sigma_3^2}}, \quad u_2 \doteq \frac{\sqrt{1-\sigma_3^2}v_1 - \sqrt{\sigma_1^2 - 1}v_3}{\sqrt{\sigma_1^2 - \sigma_3^2}} \tag{5.44}$$

is also preserved under the map H. Furthermore, it is easy to verify that H preserves the length of any vectors inside each of the two subspaces

$$S_1 = \text{span}\{v_2, u_1\}, \quad S_2 = \text{span}\{v_2, u_2\}. \tag{5.45}$$

Since v_2 is orthogonal to u_1 and u_2, $\widehat{v_2}u_1$ is a unit normal vector to S_1, and $\widehat{v_2}u_2$ a unit normal vector to S_2. Then $\{v_2, u_1, \widehat{v_2}u_1\}$ and $\{v_2, u_2, \widehat{v_2}u_2\}$ form two sets of orthonormal bases for $\mathbb{R}^3$. Notice that we have

$$Rv_2 = Hv_2, \quad Ru_i = Hu_i, \quad R(\widehat{v_2}u_i) = \widehat{Hv_2}Hu_i$$

if N is the normal to the subspace $S_i, i = 1, 2$, as shown in Figure 5.12.

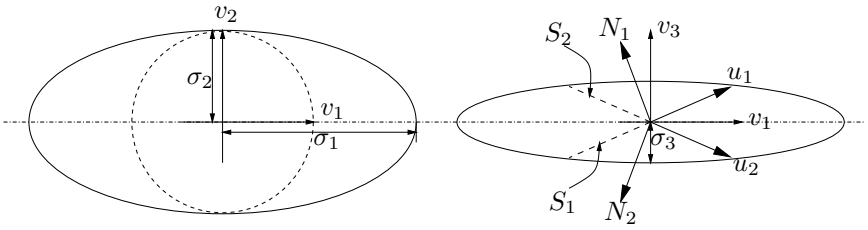

Figure 5.12. In terms of singular vectors (v_1, v_2, v_3) and singular values $(\sigma_1, \sigma_2, \sigma_3)$ of the matrix H, there are two candidate subspaces S_1 and S_2 on which the vectors' length is preserved by the homography matrix H.

Define the matrices

$$U_1 = [v_2, u_1, \widehat{v_2}u_1], \quad W_1 = [Hv_2, Hu_1, \widehat{Hv_2}Hu_1];$$
$$U_2 = [v_2, u_2, \widehat{v_2}u_2], \quad W_2 = [Hv_2, Hu_2, \widehat{Hv_2}Hu_2].$$

We then have

$$RU_1 = W_1, \quad RU_2 = W_2.$$

This suggests that each subspace S_1, or S_2 may give rise to a solution to the decomposition. By taking into account the extra sign ambiguity in the term $\frac{1}{d}TN^T$, we then obtain four solutions for decomposing $H = R + \frac{1}{d}TN^T$ to $\{R, \frac{1}{d}T, N\}$. They are given in Table 5.1.

Solution 1	R_1	=	$W_1U_1^T$	Solution 3	R_3	=	R_1
	N_1	=	$\widehat{v_2}u_1$		N_3	=	$-N_1$
	$\frac{1}{d}T_1$	=	$(H - R_1)N_1$		$\frac{1}{d}T_3$	=	$-\frac{1}{d}T_1$
Solution 2	R_2	=	$W_2U_2^T$	Solution 4	R_4	=	R_2
	N_2	=	$\widehat{v_2}u_2$		N_4	=	$-N_2$
	$\frac{1}{d}T_2$	=	$(H - R_2)N_2$		$\frac{1}{d}T_4$	=	$-\frac{1}{d}T_2$

Table 5.1. Four solutions for the planar homography decomposition, only two of which satisfy the positive depth constraint.

In order to reduce the number of physically possible solutions, we may impose the positive depth constraint (Exercise 5.11); since the camera can see only points that are in front of it, we must have $N^T e_3 = n_3 > 0$. Suppose that solution 1

is the true one; this constraint will then eliminate solution 3 as being physically impossible. Similarly, one of solutions 2 or 4 will be eliminated. For the case that $T \sim N$, we have $\sigma_3^2 = 0$ in the above proof. Hence $u_1 = u_2$, and solutions 1 and 2 are equivalent. Imposing the positive depth constraint leads to a unique solution for all motion and structure parameters. □

Example 5.20 (A numerical example). Suppose that

$$R = \begin{bmatrix} \cos(\frac{\pi}{10}) & 0 & \sin(\frac{\pi}{10}) \\ 0 & 1 & 0 \\ -\sin(\frac{\pi}{10}) & 0 & \cos(\frac{\pi}{10}) \end{bmatrix} = \begin{bmatrix} 0.951 & 0 & 0.309 \\ 0 & 1 & 0 \\ -0.309 & 0 & 0.951 \end{bmatrix}, \quad T = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}, \quad N = \begin{bmatrix} 1 \\ 0 \\ 2 \end{bmatrix},$$

and $d = 5, \lambda = 4$. Here, we deliberately choose $\|N\| \neq 1$, and we will see how this will affect the decomposition. Then the homography matrix is

$$H_L = \lambda \left(R + \frac{1}{d} T N^T \right) = \begin{bmatrix} 5.404 & 0 & 4.436 \\ 0 & 4 & 0 \\ -1.236 & 0 & 3.804 \end{bmatrix}.$$

The singular values of H_L are $\{7.197, 4.000, 3.619\}$. The middle one is exactly the scale λ. Hence for the normalized homography matrix $H_L/4 \rightarrow H$, the matrix $H^T H$ has the SVD[8]

$$V \Sigma V^T \doteq \begin{bmatrix} 0.675 & 0 & -0.738 \\ 0 & 1 & 0 \\ 0.738 & 0 & 0.675 \end{bmatrix} \begin{bmatrix} 3.237 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0.819 \end{bmatrix} \begin{bmatrix} 0.675 & 0 & -0.738 \\ 0 & 1 & 0 \\ 0.738 & 0 & 0.675 \end{bmatrix}^T.$$

Then the two vectors u_1 and u_2 are given by

$$u_1 = [-0.525, 0, 0.851]^T; \quad u_2 = [0.894, 0, -0.447]^T.$$

The four solutions to the decomposition are

$$R_1 = \begin{bmatrix} 0.704 & 0 & 0.710 \\ 0 & 1 & 0 \\ -0.710 & 0 & 0.704 \end{bmatrix}, \quad N_1 = \begin{bmatrix} 0.851 \\ 0 \\ 0.525 \end{bmatrix}, \quad \frac{1}{d} T_1 = \begin{bmatrix} 0.760 \\ 0 \\ 0.471 \end{bmatrix};$$

$$R_2 = \begin{bmatrix} 0.951 & 0 & 0.309 \\ 0 & 1 & 0 \\ -0.309 & 0 & 0.951 \end{bmatrix}, \quad N_2 = \begin{bmatrix} -0.447 \\ 0 \\ -0.894 \end{bmatrix}, \quad \frac{1}{d} T_2 = \begin{bmatrix} -0.894 \\ 0 \\ 0 \end{bmatrix};$$

$$R_3 = \begin{bmatrix} 0.704 & 0 & 0.710 \\ 0 & 1 & 0 \\ -0.710 & 0 & 0.704 \end{bmatrix}, \quad N_3 = \begin{bmatrix} -0.851 \\ 0 \\ -0.525 \end{bmatrix}, \quad \frac{1}{d} T_3 = \begin{bmatrix} -0.760 \\ 0 \\ -0.471 \end{bmatrix};$$

$$R_4 = \begin{bmatrix} 0.951 & 0 & 0.309 \\ 0 & 1 & 0 \\ -0.309 & 0 & 0.951 \end{bmatrix}, \quad N_4 = \begin{bmatrix} 0.447 \\ 0 \\ 0.894 \end{bmatrix}, \quad \frac{1}{d} T_4 = \begin{bmatrix} 0.894 \\ 0 \\ 0 \end{bmatrix}.$$

Obviously, the fourth solution is the correct one: The original $\|N\| \neq 1$, and N is recovered up to a scalar factor (with its length normalized to 1), and hence in the solution we should expect $\frac{1}{d} T_4 = \frac{\|N\|}{d} T$. Notice that the first solution also satisfies $N_1^T e_3 > 0$,

[8] The Matlab routine SVD does not always guarantee that $V \in SO(3)$. When using the routine, if one finds that $\det(V) = -1$, replace both V's by $-V$.

which indicates a plane in front of the camera. Hence it corresponds to another physically possible solution (from the decomposition). ■

We will investigate the geometric relation between the remaining two physically possible solutions in the exercises (see Exercise 5.19). We conclude this section by presenting the following four-point Algorithm 5.2 for motion estimation from a planar scene. Examples of use of this algorithm on real images are shown in Figure 5.13.

Algorithm 5.2 (The four-point algorithm for a planar scene).

For a given set of image pairs $(\boldsymbol{x}_1^j, \boldsymbol{x}_2^j)$, $j = 1, 2, \ldots, n$ ($n \geq 4$), of points on a plane $N^T \boldsymbol{X} = d$, this algorithm finds $\{R, \frac{1}{d}T, N\}$ that solves

$$\widehat{\boldsymbol{x}_2^j}^T \left(R + \frac{1}{d} T N^T \right) \boldsymbol{x}_1^j = 0, \quad j = 1, 2, \ldots, n.$$

1. **Compute a first approximation of the homography matrix**
   Construct $\chi = [\boldsymbol{a}^1, \boldsymbol{a}^2, \ldots, \boldsymbol{a}^n]^T \in \mathbb{R}^{3n \times 9}$ from correspondences $\boldsymbol{x}_1^j$ and $\boldsymbol{x}_2^j$, where $\boldsymbol{a}^j = \boldsymbol{x}_1^j \otimes \widehat{\boldsymbol{x}_2^j} \in \mathbb{R}^{9 \times 3}$. Find the vector $H_L^s \in \mathbb{R}^9$ of unit length that solves
   $$\chi H_L^s = 0$$
   as follows: compute the SVD of $\chi = U_\chi \Sigma_\chi V_\chi^T$ and define H_L^s to be the ninth column of V_χ. Unstack the nine elements of H_L^s into a square 3×3 matrix H_L.
2. **Normalization of the homography matrix**
   Compute the eigenvalues $\{\sigma_1, \sigma_2, \sigma_3\}$ of the matrix H_L and normalize it as
   $$H = H_L / \sigma_2.$$
   Correct the sign of H according to sign $\left((\boldsymbol{x}_2^j)^T H \boldsymbol{x}_1^j \right)$ for $j = 1, 2, \ldots, n$.
3. **Decomposition of the homography matrix**
   Compute the singular value decomposition of
   $$H^T H = V \Sigma V^T$$
   and compute the four solutions for a decomposition $\{R, \frac{1}{d}T, N\}$ as in the proof of Theorem 5.19. Select the two physically possible ones by imposing the positive depth constraint $N^T e_3 > 0$.

5.3.4 *Relationships between the homography and the essential matrix*

In practice, especially when the scene is piecewise planar, we often need to compute the essential matrix E with a given homography H computed from some four points known to be planar; or in the opposite situation, the essential matrix E may have been already estimated using points in general position, and we then want to compute the homography for a particular (usually smaller) set of coplanar

Figure 5.13. Homography between the left and middle images is determined by the building facade on the top, and the ground plane on the bottom. The right image is the warped image overlayed on the first image based on the estimated homography H. Note that all points on the reference plane are aligned, whereas points outside the reference plane are offset by an amount that is proportional to their distance from the reference plane.

points. We hence need to understand the relationship between the essential matrix E and the homography H.

Theorem 5.21 (Relationships between the homography and essential matrix). *For a matrix $E = \widehat{T}R$ and a matrix $H = R + Tu^T$ for some nonsingular $R \in \mathbb{R}^{3\times3}$, $T, u \in \mathbb{R}^3$, with $\|T\| = 1$, we have:*

1. $E = \widehat{T}H$;
2. $H^TE + E^TH = 0$;
3. $H = \widehat{T}^TE + Tv^T$, *for some* $v \in \mathbb{R}^3$.

Proof. The proof of item 1 is easy, since $\widehat{T}T = 0$. For item 2, notice that $H^TE = (R + Tu^T)^T\widehat{T}R = R^T\widehat{T}R$ is a skew-symmetric matrix, and hence $H^TE = -E^TH$. For item 3, notice that

$$\widehat{T}H = \widehat{T}R = \widehat{T}\widehat{T}^T\widehat{T}R = \widehat{T}\widehat{T}^TE,$$

since $\widehat{T}\widehat{T}^Tv = (I - TT^T)v$ represents an orthogonal projection of v onto the subspace (a plane) orthogonal to T (see Exercise 5.3). Therefore, $\widehat{T}(H - \widehat{T}^TE) = 0$. That is, all the columns of $H - \widehat{T}^TE$ are parallel to T, and hence we have $H - \widehat{T}^TE = Tv^T$ for some $v \in \mathbb{R}^3$. □

Notice that neither the statement nor the proof of the theorem assumes that R is a rotation matrix. Hence, the results will also be applicable to the case in which the camera is not calibrated, which will be discussed in the next chapter.

This theorem directly implies two useful corollaries stated below that allow us to easily compute E from H as well as H from E with minimum extra information from images.[9] The first corollary is a direct consequence of the above theorem and Proposition 5.14:

Corollary 5.22 (From homography to the essential matrix). *Given a homography H and two pairs of images $(\boldsymbol{x}_1^i, \boldsymbol{x}_2^i), i = 1, 2$, of two points not on the plane P from which H is induced, we have*

$$E = \widehat{T}H, \tag{5.46}$$

where $T \sim \widehat{\boldsymbol{\ell}_2^1}\boldsymbol{\ell}_2^2$ and $\|T\| = 1$.

Proof. According to Proposition 5.14, $\boldsymbol{\ell}_2^i$ is the epipolar line $\boldsymbol{\ell}_2^i \sim \widehat{\boldsymbol{x}_2^i}H\boldsymbol{x}_1^i,\ i = 1, 2$. Both epipolar lines $\boldsymbol{\ell}_2^1, \boldsymbol{\ell}_2^2$ pass through the epipole $\boldsymbol{e}_2 \sim T$. This can be illustrated by Figure 5.14. □

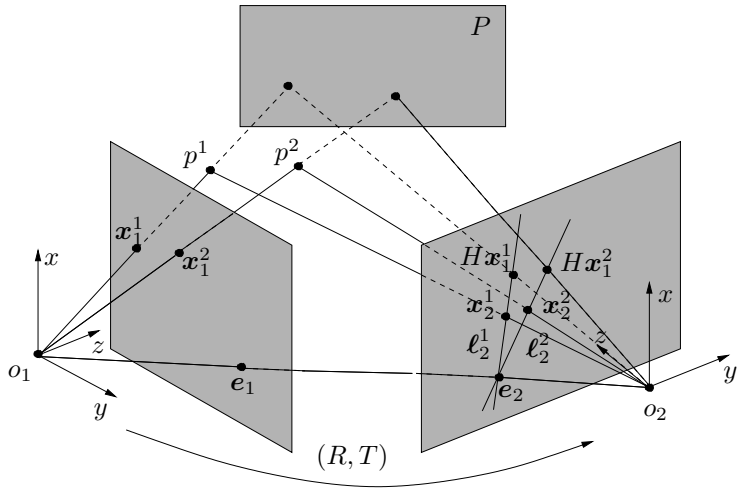

Figure 5.14. A homography H transfers two points $\boldsymbol{x}_1^1$ and $\boldsymbol{x}_1^2$ in the first image to two points $H\boldsymbol{x}_1^1$ and $H\boldsymbol{x}_1^2$ on the same epipolar lines as the respective true images $\boldsymbol{x}_2^1$ and $\boldsymbol{x}_2^2$ if the corresponding 3-D points p^1 and p^2 are not on the plane P from which H is induced.

Now consider the opposite situation that an essential matrix E is given and we want to compute the homography for a set of coplanar points. Note that once E is known, the vector T is also known (up to a scalar factor) as the left null space of E. We may typically choose T to be of unit length.

[9] Although in principle, to compute E from H, one does not need any extra information but only has to decompose H and find R and T using Theorem 5.19, the corollary will allow us to bypass that by much simpler techniques, which, unlike Theorem 5.19, will also be applicable to the uncalibrated case.

Corollary 5.23 (From essential matrix to homography). *Given an essential matrix E and three pairs of images $(\boldsymbol{x}_1^i, \boldsymbol{x}_2^i), i = 1, 2, 3$, of three points in 3-D, the homography H induced by the plane specified by the three points then is*

$$H = \widehat{T}^T E + T v^T, \tag{5.47}$$

where $v = [v_1, v_2, v_3]^T \in \mathbb{R}^3$ solves the system of three linear equations

$$\widehat{\boldsymbol{x}_2^i}(\widehat{T}^T E + T v^T)\boldsymbol{x}_1^i = 0, \quad i = 1, 2, 3. \tag{5.48}$$

Proof. We leave the proof to the reader as an exercise. □

5.4 Continuous motion case

As we pointed out in Section 5.1, the limit case where the two viewpoints are infinitesimally close requires extra attention. From the practical standpoint, this case is relevant to the analysis of a video stream where the camera motion is slow relative to the sampling frequency. In this section,[10] we follow the steps of the previous section by giving a parallel derivation of the geometry of points in space as seen from a moving camera, and deriving a conceptual algorithm for reconstructing camera motion and scene structure. In light of the fact that the camera motion is slow relative to the sampling frequency, we will treat the motion of the camera as continuous. While the derivations proceed in parallel, we will highlight some subtle but significant differences.

5.4.1 Continuous epipolar constraint and the continuous essential matrix

Let us assume that camera motion is described by a smooth (i.e. continuously differentiable) trajectory $g(t) = (R(t), T(t)) \in SE(3)$ with body velocities $(\omega(t), v(t)) \in se(3)$ as defined in Chapter 2. For a point $p \in \mathbb{R}^3$, its coordinates as a function of time $\boldsymbol{X}(t)$ satisfy

$$\dot{\boldsymbol{X}}(t) = \widehat{\omega}(t)\boldsymbol{X}(t) + v(t). \tag{5.49}$$

The image of the point p taken by the camera is the vector $\boldsymbol{x}$ that satisfies $\lambda(t)\boldsymbol{x}(t) = \boldsymbol{X}(t)$. From now on, for convenience, we will drop the time dependency from the notation. Denote the velocity of the image point $\boldsymbol{x}$ by $\boldsymbol{u} \doteq \dot{\boldsymbol{x}} \in \mathbb{R}^3$. The velocity $\boldsymbol{u}$ is also called *image motion field*, which under the brightness constancy assumption discussed in Chapter 4 can be approximated by the *optical*

[10] This section can be skipped without loss of continuity if the reader is not interested in the continuous-motion case.

flow. To obtain an explicit expression for $\boldsymbol{u}$, we notice that

$$\boldsymbol{X} = \lambda \boldsymbol{x}, \quad \dot{\boldsymbol{X}} = \dot{\lambda}\boldsymbol{x} + \lambda \dot{\boldsymbol{x}}.$$

Substituting this into equation (5.49), we obtain

$$\dot{\boldsymbol{x}} = \widehat{\omega}\boldsymbol{x} + \frac{1}{\lambda} v - \frac{\dot{\lambda}}{\lambda}\dot{\boldsymbol{x}}. \tag{5.50}$$

Then the image velocity $\boldsymbol{u} = \dot{\boldsymbol{x}}$ depends not only on the camera motion but also on the depth scale λ of the point. For the planar perspective projection and the spherical perspective projection, the expression for $\boldsymbol{u}$ will be slightly different. We leave the detail to the reader as an exercise (see Exercise 5.20).

To eliminate the depth scale λ, consider now the inner product of the vectors in (5.50) with the vector $(v \times \boldsymbol{x})$. We obtain

$$\dot{\boldsymbol{x}}^T \widehat{v}\boldsymbol{x} = \boldsymbol{x}^T \widehat{\omega}^T \widehat{v}\boldsymbol{x}.$$

We can rewrite the above equation in an equivalent way:

$$\boxed{\boldsymbol{u}^T \widehat{v}\boldsymbol{x} + \boldsymbol{x}^T \widehat{\omega}\widehat{v}\boldsymbol{x} = 0.} \tag{5.51}$$

This constraint plays the same role for the case of continuous-time images as the epipolar constraint for two discrete image, in the sense that it does not depend on the position of the point in space, but only on its projection and the motion parameters. We call it the *continuous epipolar constraint*.

Before proceeding with an analysis of equation (5.51), we state a lemma that will become useful in the remainder of this section.

Lemma 5.24. *Consider the matrices $M_1, M_2 \in \mathbb{R}^{3\times 3}$. Then $\boldsymbol{x}^T M_1 \boldsymbol{x} = \boldsymbol{x}^T M_2 \boldsymbol{x}$ for all $\boldsymbol{x} \in \mathbb{R}^3$ if and only if $M_1 - M_2$ is a skew-symmetric matrix, i.e. $M_1 - M_2 \in so(3)$.*

We leave the proof of this lemma as an exercise. Following the lemma, for any skew-symmetric matrix $M \in \mathbb{R}^{3\times 3}$, $\boldsymbol{x}^T M \boldsymbol{x} = 0$. Since $\frac{1}{2}(\widehat{\omega}\widehat{v} - \widehat{v}\widehat{\omega})$ is a skew-symmetric matrix, $\boldsymbol{x}^T \frac{1}{2}(\widehat{\omega}\widehat{v} - \widehat{v}\widehat{\omega})\boldsymbol{x} = 0$. If we define the *symmetric epipolar component* to be the matrix

$$s \doteq \frac{1}{2}(\widehat{\omega}\widehat{v} + \widehat{v}\widehat{\omega}) \quad \in \mathbb{R}^{3\times 3},$$

then we have that

$$\boldsymbol{x}^T s \boldsymbol{x} = \boldsymbol{x}^T \widehat{\omega}\widehat{v}\boldsymbol{x},$$

so that the continuous epipolar constraint may be rewritten as

$$\boldsymbol{u}^T \widehat{v}\boldsymbol{x} + \boldsymbol{x}^T s \boldsymbol{x} = 0. \tag{5.52}$$

This equation shows that for the matrix $\widehat{\omega}\widehat{v}$, only its symmetric component $s = \frac{1}{2}(\widehat{\omega}\widehat{v} + \widehat{v}\widehat{\omega})$ can be recovered from the epipolar equation (5.51) or equivalently

(5.52).[11] This structure is substantially different from that of the discrete case, and it cannot be derived by a first-order approximation of the essential matrix $\widehat{T}R$. In fact, a naive discretization of the discrete epipolar equation may lead to a constraint involving only a matrix of the form $\widehat{v}\widehat{\omega}$, whereas in the true continuous case we have to deal with only its symmetric component $s = \frac{1}{2}(\widehat{\omega}\widehat{v} + \widehat{v}\widehat{\omega})$ plus another term as given in (5.52). The set of matrices of interest in the case of continuous motions is thus the space of 6×3 matrices of the form

$$\mathcal{E}' \doteq \left\{ \begin{bmatrix} \widehat{v} \\ \frac{1}{2}(\widehat{\omega}\widehat{v} + \widehat{v}\widehat{\omega}) \end{bmatrix} \middle| \omega, v \in \mathbb{R}^3 \right\} \subset \mathbb{R}^{6\times 3},$$

which we call the *continuous essential space*. A matrix in this space is called a *continuous essential matrix*. Note that the continuous epipolar constraint (5.52) is homogeneous in the linear velocity v. Thus v may be recovered only up to a constant scalar factor. Consequently, in motion recovery, we will concern ourselves with matrices belonging to the *normalized continuous essential space* with v scaled to unit norm:

$$\mathcal{E}'_1 = \left\{ \begin{bmatrix} \widehat{v} \\ \frac{1}{2}(\widehat{\omega}\widehat{v} + \widehat{v}\widehat{\omega}) \end{bmatrix} \middle| \omega \in \mathbb{R}^3, v \in \mathbb{S}^2 \right\} \subset \mathbb{R}^{6\times 3}.$$

5.4.2 Properties of the continuous essential matrix

The skew-symmetric part of a continuous essential matrix simply corresponds to the velocity v. The characterization of the (normalized) essential matrix focuses only on the symmetric matrix part $s = \frac{1}{2}(\widehat{\omega}\widehat{v} + \widehat{v}\widehat{\omega})$. We call the space of all the matrices of this form the *symmetric epipolar space*

$$\mathcal{S} \doteq \left\{ \frac{1}{2}(\widehat{\omega}\widehat{v} + \widehat{v}\widehat{\omega}) \middle| \omega \in \mathbb{R}^3, v \in \mathbb{S}^2 \right\} \subset \mathbb{R}^{3\times 3}.$$

The motion estimation problem is now reduced to that of *recovering the velocity* (ω, v) *with* $\omega \in \mathbb{R}^3$ *and* $v \in \mathbb{S}^2$ *from a given symmetric epipolar component* s.

The characterization of symmetric epipolar components depends on a characterization of matrices of the form $\widehat{\omega}\widehat{v} \in \mathbb{R}^{3\times 3}$, which is given in the following lemma. Of use in the lemma is the matrix $R_Y(\theta)$ defined to be the rotation around the Y-axis by an angle $\theta \in \mathbb{R}$, i.e. $R_Y(\theta) = e^{\widehat{e}_2\theta}$ with $e_2 = [0, 1, 0]^T \in \mathbb{R}^3$.

Lemma 5.25. *A matrix* $Q \in \mathbb{R}^{3\times 3}$ *has the form* $Q = \widehat{\omega}\widehat{v}$ *with* $\omega \in \mathbb{R}^3$, $v \in \mathbb{S}^2$ *if and only if*

$$Q = -VR_Y(\theta)diag\{\lambda, \lambda\cos(\theta), 0\}V^T \tag{5.53}$$

[11] This redundancy is the reason why different forms of the continuous epipolar constraint exist in the literature [Zhuang and Haralick, 1984, Ponce and Genc, 1998, Vieville and Faugeras, 1995, Maybank, 1993, Brooks et al., 1997], and accordingly, various approaches have been proposed to recover ω and v (see [Tian et al., 1996]).

for some rotation matrix $V \in SO(3)$, the positive scalar $\lambda = \|\omega\|$, and $\cos(\theta) = \omega^T v/\lambda$.

Proof. We first prove the necessity. The proof follows from the geometric meaning of $\widehat{\omega}\widehat{v}$ multiplied by any vector $q \in \mathbb{R}^3$:

$$\widehat{\omega}\widehat{v}q = \omega \times (v \times q).$$

Let $b \in \mathbb{S}^2$ be the unit vector perpendicular to both ω and v. That is, $b = \frac{v \times \omega}{\|v \times \omega\|}$. (If $v \times \omega = 0$, b is not uniquely defined. In this case, ω, v are parallel, and the rest of the proof follows if one picks any vector b orthogonal to v and ω.) Then $\omega = \lambda \exp(\widehat{b}\theta)v$ (according to this definition, θ is the angle between ω and v, and $0 \leq \theta \leq \pi$). It is easy to check that if the matrix V is defined to be

$$V = \left(e^{\widehat{b}\frac{\pi}{2}}v, b, v\right),$$

then Q has the given form (5.53).

We now prove the sufficiency. Given a matrix Q that can be decomposed into the form (5.53), define the orthogonal matrix $U = -VR_Y(\theta) \in O(3)$. (Recall that $O(3)$ represents the space of all orthogonal matrices of determinant ± 1.) Let the two skew-symmetric matrices $\widehat{\omega}$ and $\widehat{v}$ be given by

$$\widehat{\omega} = UR_Z\left(\pm\frac{\pi}{2}\right)\Sigma_\lambda U^T, \quad \widehat{v} = VR_Z\left(\pm\frac{\pi}{2}\right)\Sigma_1 V^T, \tag{5.54}$$

where $\Sigma_\lambda = \text{diag}\{\lambda, \lambda, 0\}$ and $\Sigma_1 = \text{diag}\{1, 1, 0\}$. Then

$$\begin{aligned}
\widehat{\omega}\widehat{v} &= UR_Z\left(\pm\frac{\pi}{2}\right)\Sigma_\lambda U^T V R_Z\left(\pm\frac{\pi}{2}\right)\Sigma_1 V^T \\
&= UR_Z\left(\pm\frac{\pi}{2}\right)\Sigma_\lambda(-R_Y^T(\theta))R_Z\left(\pm\frac{\pi}{2}\right)\Sigma_1 V^T \\
&= U\text{diag}\{\lambda, \lambda\cos(\theta), 0\}V^T \\
&= Q.
\end{aligned} \tag{5.55}$$

Since ω and v have to be, respectively, the left and the right zero eigenvectors of Q, the reconstruction given in (5.54) is unique up to a sign. □

Based on the above lemma, the following theorem reveals the structure of the symmetric epipolar component.

Theorem 5.26 (Characterization of the symmetric epipolar component). *A real symmetric matrix $s \in \mathbb{R}^{3\times 3}$ is a symmetric epipolar component if and only if s can be diagonalized as $s = V\Sigma V^T$ with $V \in SO(3)$ and*

$$\Sigma = \textit{diag}\{\sigma_1, \sigma_2, \sigma_3\}$$

with $\sigma_1 \geq 0, \sigma_3 \leq 0$, and $\sigma_2 = \sigma_1 + \sigma_3$.

Proof. We first prove the necessity. Suppose s is a symmetric epipolar component. Thus there exist $\omega \in \mathbb{R}^3, v \in \mathbb{S}^2$ such that $s = \frac{1}{2}(\widehat{\omega}\widehat{v} + \widehat{v}\widehat{\omega})$. Since s is a symmetric matrix, it is diagonalizable, all its eigenvalues are real, and all

the eigenvectors are orthogonal to each other. It then suffices to check that its eigenvalues satisfy the given conditions.

Let the unit vector b, the rotation matrix V, θ, and λ be the same as in the proof of Lemma 5.25. According to the lemma, we have

$$\widehat{\omega}\widehat{v} = -VR_Y(\theta)\text{diag}\{\lambda, \lambda\cos(\theta), 0\}V^T.$$

Since $(\widehat{\omega}\widehat{v})^T = \widehat{v}\widehat{\omega}$, we have

$$s = \frac{1}{2}V\left(-R_Y(\theta)\text{diag}\{\lambda, \lambda\cos(\theta), 0\} - \text{diag}\{\lambda, \lambda\cos(\theta), 0\}R_Y^T(\theta)\right)V^T.$$

Define the matrix $D(\lambda, \theta) \in \mathbb{R}^{3\times 3}$ to be

$$\begin{aligned} D(\lambda,\theta) &= -R_Y(\theta)\text{diag}\{\lambda, \lambda\cos(\theta), 0\} - \text{diag}\{\lambda, \lambda\cos(\theta), 0\}R_Y^T(\theta) \\ &= \lambda\begin{bmatrix} -2\cos(\theta) & 0 & \sin(\theta) \\ 0 & -2\cos(\theta) & 0 \\ \sin(\theta) & 0 & 0 \end{bmatrix}. \end{aligned}$$

Directly calculating its eigenvalues and eigenvectors, we obtain that $D(\lambda, \theta)$ is equal to

$$R_Y\left(\frac{\theta-\pi}{2}\right)\text{diag}\left\{\lambda(1-\cos(\theta)), -2\lambda\cos(\theta), \lambda(-1-\cos(\theta))\right\}R_Y^T\left(\frac{\theta-\pi}{2}\right). \tag{5.56}$$

Thus $s = \frac{1}{2}VD(\lambda, \theta)V^T$ has eigenvalues

$$\left\{\frac{1}{2}\lambda(1-\cos(\theta)), \quad -\lambda\cos(\theta), \quad \frac{1}{2}\lambda(-1-\cos(\theta))\right\}, \tag{5.57}$$

which satisfy the given conditions.

We now prove the sufficiency. Given $s = V_1\text{diag}\{\sigma_1, \sigma_2, \sigma_3\}V_1^T$ with $\sigma_1 \geq 0, \sigma_3 \leq 0$, $\sigma_2 = \sigma_1 + \sigma_3$, and $V_1^T \in SO(3)$, these three eigenvalues uniquely determine $\lambda, \theta \in \mathbb{R}$ such that the σ_i's have the form given in (5.57):

$$\begin{aligned} \lambda &= \sigma_1 - \sigma_3, & \lambda \geq 0, \\ \theta &= \arccos(-\sigma_2/\lambda), & \theta \in [0, \pi]. \end{aligned}$$

Define a matrix $V \in SO(3)$ to be $V = V_1R_Y^T\left(\frac{\theta}{2} - \frac{\pi}{2}\right)$. Then $s = \frac{1}{2}VD(\lambda, \theta)V^T$. According to Lemma 5.25, there exist vectors $v \in \mathbb{S}^2$ and $\omega \in \mathbb{R}^3$ such that

$$\widehat{\omega}\widehat{v} = -VR_Y(\theta)\text{diag}\{\lambda, \lambda\cos(\theta), 0\}V^T.$$

Therefore, $\frac{1}{2}(\widehat{\omega}\widehat{v} + \widehat{v}\widehat{\omega}) = \frac{1}{2}VD(\lambda, \theta)V^T = s$. □

Figure 5.15 gives a geometric interpretation of the three eigenvectors of the symmetric epipolar component s for the case in which both ω, v are of unit length. The constructive proof given above is important since it gives an explicit decomposition of the symmetric epipolar component s, which will be studied in more detail next.

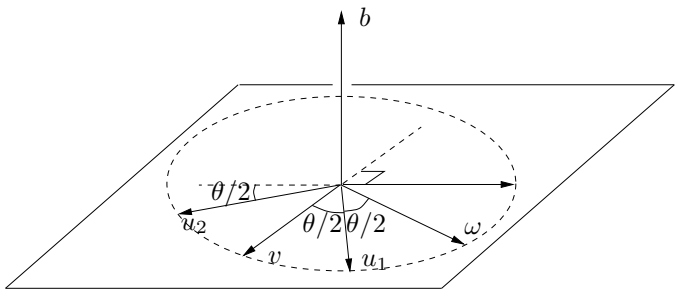

Figure 5.15. Vectors u_1, u_2, b are the three eigenvectors of a symmetric epipolar component $\frac{1}{2}(\widehat{\omega}\widehat{v}+\widehat{v}\widehat{\omega})$. In particular, b is the normal vector to the plane spanned by ω and v, and u_1, u_2 are both in this plane. The vector u_1 is the average of ω and v, and u_2 is orthogonal to both b and u_1.

Following the proof of Theorem 5.26, if we already know the eigenvector decomposition of a symmetric epipolar component s, we certainly can find at least one solution (ω, v) such that $s = \frac{1}{2}(\widehat{\omega}\widehat{v} + \widehat{v}\widehat{\omega})$. We now discuss uniqueness, i.e. how many solutions exist for $s = \frac{1}{2}(\widehat{\omega}\widehat{v} + \widehat{v}\widehat{\omega})$.

Theorem 5.27 (Velocity recovery from the symmetric epipolar component). *There exist exactly four 3-D velocities (ω, v) with $\omega \in \mathbb{R}^3$ and $v \in \mathbb{S}^2$ corresponding to a nonzero $s \in \mathcal{S}$.*

Proof. Suppose (ω_1, v_1) and (ω_2, v_2) are both solutions for $s = \frac{1}{2}(\widehat{\omega}\widehat{v} + \widehat{v}\widehat{\omega})$. Then we have

$$\widehat{v}_1\widehat{\omega}_1 + \widehat{\omega}_1\widehat{v}_1 = \widehat{v}_2\widehat{\omega}_2 + \widehat{\omega}_2\widehat{v}_2. \tag{5.58}$$

From Lemma 5.25, we may write

$$\begin{array}{rcl} \widehat{\omega}_1\widehat{v}_1 & = & -V_1 R_Y(\theta_1)\text{diag}\{\lambda_1, \lambda_1\cos(\theta_1), 0\}V_1^T, \\ \widehat{\omega}_2\widehat{v}_2 & = & -V_2 R_Y(\theta_2)\text{diag}\{\lambda_2, \lambda_2\cos(\theta_2), 0\}V_2^T. \end{array} \tag{5.59}$$

Let $W = V_1^T V_2 \in SO(3)$. Then from (5.58),

$$D(\lambda_1, \theta_1) = W D(\lambda_2, \theta_2) W^T. \tag{5.60}$$

Since both sides of (5.60) have the same eigenvalues, according to (5.56), we have

$$\lambda_1 = \lambda_2, \quad \theta_2 = \theta_1.$$

We can then denote both θ_1 and θ_2 by θ. It is immediate to check that the only possible rotation matrix W that satisfies (5.60) is given by $I_{3\times 3}$,

$$\begin{bmatrix} -\cos(\theta) & 0 & \sin(\theta) \\ 0 & -1 & 0 \\ \sin(\theta) & 0 & \cos(\theta) \end{bmatrix}, \quad \text{or} \quad \begin{bmatrix} \cos(\theta) & 0 & -\sin(\theta) \\ 0 & -1 & 0 \\ -\sin(\theta) & 0 & -\cos(\theta) \end{bmatrix}.$$

From the geometric meaning of V_1 and V_2, all the cases give either $\widehat{\omega}_1\widehat{v}_1 = \widehat{\omega}_2\widehat{v}_2$ or $\widehat{\omega}_1\widehat{v}_1 = \widehat{v}_2\widehat{\omega}_2$. Thus, according to the proof of Lemma 5.25, if (ω, v) is one

solution and $\widehat{\omega}\widehat{v} = U\text{diag}\{\lambda, \lambda\cos(\theta), 0\}V^T$, then all the solutions are given by

$$\begin{array}{rcl} \widehat{\omega} & = & UR_Z(\pm\frac{\pi}{2})\Sigma_\lambda U^T, \quad \widehat{v} = VR_Z(\pm\frac{\pi}{2})\Sigma_1 V^T, \\ \widehat{\omega} & = & VR_Z(\pm\frac{\pi}{2})\Sigma_\lambda V^T, \quad \widehat{v} = UR_Z(\pm\frac{\pi}{2})\Sigma_1 U^T, \end{array} \tag{5.61}$$

where $\Sigma_\lambda = \text{diag}\{\lambda, \lambda, 0\}$ and $\Sigma_1 = \text{diag}\{1, 1, 0\}$. □

Given a nonzero continuous essential matrix $E \in \mathcal{E}'$, according to (5.61), its symmetric component gives four possible solutions for the 3-D velocity (ω, v). However, in general, only one of them has the same linear velocity v as the skew-symmetric part of E. Hence, compared to the discrete case, where there are two 3-D motions (R, T) associated with an essential matrix, the velocity (ω, v) corresponding to a continuous essential matrix is unique. This is because in the continuous case, the *twisted-pair ambiguity*, which occurs in the discrete case and is caused by a 180° rotation of the camera around the translation direction, see Example 5.8, is now avoided.

5.4.3 The eight-point linear algorithm

Based on the preceding study of the continuous essential matrix, this section describes an algorithm to recover the 3-D velocity of the camera from a set of (possibly noisy) optical flow measurements.

Let $E = \begin{bmatrix} \widehat{v} \\ s \end{bmatrix} \in \mathcal{E}'_1$ with $s = \frac{1}{2}(\widehat{\omega}\widehat{v} + \widehat{v}\widehat{\omega})$ be the essential matrix associated with the continuous epipolar constraint (5.52). Since the submatrix $\widehat{v}$ is skew-symmetric and s is symmetric, they have the following form

$$\widehat{v} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}, \quad s = \begin{bmatrix} s_1 & s_2 & s_3 \\ s_2 & s_4 & s_5 \\ s_3 & s_5 & s_6 \end{bmatrix}. \tag{5.62}$$

Define the continuous version of the "stacked" vector $E^s \in \mathbb{R}^9$ to be

$$E^s \doteq [v_1, v_2, v_3, s_1, s_2, s_3, s_4, s_5, s_6]^T. \tag{5.63}$$

Define a vector $\boldsymbol{a} \in \mathbb{R}^9$ associated with the optical flow $(\boldsymbol{x}, \boldsymbol{u})$ with $\boldsymbol{x} = [x, y, z]^T \in \mathbb{R}^3, \boldsymbol{u} = [u_1, u_2, u_3]^T \in \mathbb{R}^3$ to be[12]

$$\boldsymbol{a} \doteq [u_3 y - u_2 z, u_1 z - u_3 x, u_2 x - u_1 y, x^2, 2xy, 2xz, y^2, 2yz, z^2]^T. \tag{5.64}$$

The continuous epipolar constraint (5.52) can be then rewritten as

$$\boldsymbol{a}^T E^s = 0.$$

Given a set of (possibly noisy) optical flow vectors $(\boldsymbol{x}^j, \boldsymbol{u}^j)$, $j = 1, 2, \ldots, n$, generated by the same motion, define a matrix $\chi \in \mathbb{R}^{n\times 9}$ associated with these

[12]For a planar perspective projection, $z = 1$ and $u_3 = 0$; thus the expression for $\boldsymbol{a}$ can be simplified.

measurements to be

$$\chi \doteq [\boldsymbol{a}^1, \boldsymbol{a}^2, \ldots, \boldsymbol{a}^n]^T, \tag{5.65}$$

where $\boldsymbol{a}^j$ are defined for each pair $(\boldsymbol{x}^j, \boldsymbol{u}^j)$ using (5.64). In the absence of noise, the vector E^s has to satisfy

$$\chi E^s = 0. \tag{5.66}$$

In order for this equation to have a unique solution for E^s, the rank of the matrix χ has to be eight. Thus, *for this algorithm, the optical flow vectors of at least eight points are needed to recover the 3-D velocity, i.e.* $n \geq 8$, although the minimum number of optical flow vectors needed for a finite number of solutions is actually five, as discussed by [Maybank, 1993].

When the measurements are noisy, there may be no solution to $\chi E^s = 0$. As in the discrete case, one may approximate the solution by minimizing the least-squares error function $\|\chi E^s\|^2$.

Since the vector E^s is recovered from noisy measurements, the symmetric part s of E directly recovered from unstacking E^s is not necessarily a symmetric epipolar component. Thus one cannot directly use the previously derived results for symmetric epipolar components to recover the 3-D velocity. In analogy to the discrete case, we can project the symmetric matrix s onto the space of symmetric epipolar components.

Theorem 5.28 (Projection onto the symmetric epipolar space). *If a real symmetric matrix $F \in \mathbb{R}^{3\times 3}$ is diagonalized as $F = V diag\{\lambda_1, \lambda_2, \lambda_3\} V^T$ with $V \in SO(3)$, $\lambda_1 \geq 0, \lambda_3 \leq 0$, and $\lambda_1 \geq \lambda_2 \geq \lambda_3$, then the symmetric epipolar component $E \in \mathcal{S}$ that minimizes the error $\|E - F\|_f^2$ is given by $E = V diag\{\sigma_1, \sigma_2, \sigma_2\} V^T$ with*

$$\sigma_1 = \frac{2\lambda_1 + \lambda_2 - \lambda_3}{3}, \quad \sigma_2 = \frac{\lambda_1 + 2\lambda_2 + \lambda_3}{3}, \quad \sigma_3 = \frac{2\lambda_3 + \lambda_2 - \lambda_1}{3}. \tag{5.67}$$

Proof. Define $\mathcal{S}_\Sigma$ to be the subspace of $\mathcal{S}$ whose elements have the same eigenvalues $\Sigma = \text{diag}\{\sigma_1, \sigma_2, \sigma_3\}$. Thus every matrix $E \in \mathcal{S}_\Sigma$ has the form $E = V_1 \Sigma V_1^T$ for some $V_1 \in SO(3)$. To simplify the notation, define $\Sigma_\lambda = \text{diag}\{\lambda_1, \lambda_2, \lambda_3\}$. We now prove this theorem in two steps.

Step 1: We prove that the matrix $E \in \mathcal{S}_\Sigma$ that minimizes the error $\|E - F\|_f^2$ is given by $E = V \Sigma V^T$. Since $E \in \mathcal{S}_\Sigma$ has the form $E = V_1 \Sigma V_1^T$, we get

$$\|E - F\|_f^2 \quad = \quad \|V_1 \Sigma V_1^T - V \Sigma_\lambda V^T\|_f^2 = \|\Sigma_\lambda - V^T V_1 \Sigma V_1^T V\|_f^2.$$

Define $W = V^T V_1 \in SO(3)$ and denote its entries by

$$W = \begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix}. \tag{5.68}$$

Then

$$\begin{aligned} \|E - F\|_f^2 &= \|\Sigma_\lambda - W\Sigma W^T\|_f^2 \\ &= \text{trace}(\Sigma_\lambda^2) - 2\text{trace}(W\Sigma W^T\Sigma_\lambda) + \text{trace}(\Sigma^2). \quad (5.69) \end{aligned}$$

Substituting (5.68) into the second term, and using the fact that $\sigma_2 = \sigma_1 + \sigma_3$ and W is a rotation matrix, we get

$$\begin{aligned} \text{trace}(W\Sigma W^T\Sigma_\lambda) &= \sigma_1(\lambda_1(1 - w_3^2) + \lambda_2(1 - w_6^2) + \lambda_3(1 - w_9^2)) \\ &\quad + \sigma_3(\lambda_1(1 - w_1^2) + \lambda_2(1 - w_4^2) + \lambda_3(1 - w_7^2)). \end{aligned}$$

Minimizing $\|E - F\|_f^2$ is equivalent to maximizing $\text{trace}(W\Sigma W^T\Sigma_\lambda)$. From the above equation, $\text{trace}(W\Sigma W^T\Sigma_\lambda)$ is maximized if and only if $w_3 = w_6 = 0$, $w_9^2 = 1$, $w_4 = w_7 = 0$, and $w_1^2 = 1$. Since W is a rotation matrix, we also have $w_2 = w_8 = 0$, and $w_5^2 = 1$. All possible W give a unique matrix in $\mathcal{S}_\Sigma$ that minimizes $\|E - F\|_f^2$: $E = V\Sigma V^T$.

Step 2: From step one, we need only to minimize the error function over the matrices that have the form $V\Sigma V^T \in \mathcal{S}$. The optimization problem is then converted to one of minimizing the error function

$$\|E - F\|_f^2 = (\lambda_1 - \sigma_1)^2 + (\lambda_2 - \sigma_2)^2 + (\lambda_3 - \sigma_3)^2$$

subject to the constraint

$$\sigma_2 = \sigma_1 + \sigma_3.$$

The formulae (5.67) for $\sigma_1, \sigma_2, \sigma_3$ are directly obtained from solving this minimization problem. □

Remark 5.29. *In the preceding theorem, for a symmetric matrix F that does not satisfy the conditions $\lambda_1 \geq 0$ and $\lambda_3 \leq 0$, one chooses $\lambda_1' = \max\{\lambda_1, 0\}$ and $\lambda_3' = \min\{\lambda_3, 0\}$ prior to applying the above theorem.*

Finally, we outline an eigenvalue-decomposition algorithm, Algorithm 5.3, for estimating 3-D velocity from optical flows of eight points, which serves as a continuous counterpart of the eight-point algorithm given in Section 5.2.

Remark 5.30. *Since both $E, -E \in \mathcal{E}_1'$ satisfy the same set of continuous epipolar constraints, both $(\omega, \pm v)$ are possible solutions for the given set of optical flow vectors. However, as in the discrete case, one can get rid of the ambiguous solution by enforcing the positive depth constraint (Exercise 5.11).*

In situations where the motion of the camera is partially constrained, the above linear algorithm can be further simplified. The following example illustrates such a scenario.

Example 5.31 (Constrained motion estimation). This example shows how to utilize constraints on motion to be estimated in order to simplify the proposed linear motion estimation algorithm in the continuous case. Let $g(t) \in SE(3)$ represent the position and orientation of an aircraft relative to the spatial frame; the inputs $\omega_1, \omega_2, \omega_3 \in \mathbb{R}$ stand for

Algorithm 5.3 (The continuous eight-point algorithm).

For a given set of images and optical flow vectors $(\boldsymbol{x}^j, \boldsymbol{u}^j),\ j = 1, 2, \ldots, n$, this algorithm finds $(\omega, v) \in SE(3)$ that solves

$$\boldsymbol{u}^{jT} \widehat{v} \boldsymbol{x}^j + \boldsymbol{x}^{jT} \widehat{\omega} \widehat{v} \boldsymbol{x}^j = 0, \quad j = 1, 2, \ldots, n.$$

1. **Estimate the essential vector**
   Define a matrix $\chi \in \mathbb{R}^{n \times 9}$ whose jth row is constructed from $\boldsymbol{x}^j$ and $\boldsymbol{u}^j$ as in (5.64). Use the SVD to find the vector $E^s \in \mathbb{R}^9$ such that $\chi E^s = 0$: $\chi = U_\chi \Sigma_\chi V_\chi^T$ and $E^s = V_\chi(:, 9)$. Recover the vector $v_0 \in \mathbb{S}^2$ from the first three entries of E^s and a symmetric matrix $s \in \mathbb{R}^{3\times 3}$ from the remaining six entries as in (5.63). Multiply E^s with a scalar such that the vector v_0 becomes of unit norm.

2. **Recover the symmetric epipolar component**
   Find the eigenvalue decomposition of the symmetric matrix s:

   $$s = V_1 \text{diag}\{\lambda_1, \lambda_2, \lambda_3\} V_1^T,$$

   with $\lambda_1 \geq \lambda_2 \geq \lambda_3$. Project the symmetric matrix s onto the symmetric epipolar space $\mathcal{S}$. We then have the new $s = V_1 \text{diag}\{\sigma_1, \sigma_2, \sigma_3\} V_1^T$ with

   $$\sigma_1 = \frac{2\lambda_1 + \lambda_2 - \lambda_3}{3},\ \sigma_2 = \frac{\lambda_1 + 2\lambda_2 + \lambda_3}{3},\ \sigma_3 = \frac{2\lambda_3 + \lambda_2 - \lambda_1}{3}.$$

3. **Recover the velocity from the symmetric epipolar component**
   Define

   $$\begin{aligned} \lambda &= \sigma_1 - \sigma_3, \quad \lambda \geq 0, \\ \theta &= \arccos(-\sigma_2/\lambda), \quad \theta \in [0, \pi]. \end{aligned}$$

   Let $V = V_1 R_Y^T \left(\frac{\theta}{2} - \frac{\pi}{2}\right) \in SO(3)$ and $U = -V R_Y(\theta) \in O(3)$. Then the four possible 3-D velocities corresponding to the matrix s are given by

   $$\begin{aligned} \widehat{\omega} &= U R_Z(\pm\tfrac{\pi}{2}) \Sigma_\lambda U^T, \quad \widehat{v} = V R_Z(\pm\tfrac{\pi}{2}) \Sigma_1 V^T, \\ \widehat{\omega} &= V R_Z(\pm\tfrac{\pi}{2}) \Sigma_\lambda V^T, \quad \widehat{v} = U R_Z(\pm\tfrac{\pi}{2}) \Sigma_1 U^T, \end{aligned}$$

   where $\Sigma_\lambda = \text{diag}\{\lambda, \lambda, 0\}$ and $\Sigma_1 = \text{diag}\{1, 1, 0\}$.

4. **Recover velocity from the continuous essential matrix**
   From the four velocities recovered from the matrix s in step 3, choose the pair (ω^*, v^*) that satisfies

   $$v^{*T} v_0 = \max_i \{v_i^T v_0\}.$$

   Then the estimated 3-D velocity (ω, v) with $\omega \in \mathbb{R}^3$ and $v \in \mathbb{S}^2$ is given by

   $$\omega = \omega^*, \quad v = v_0.$$

the rates of the rotation about the axes of the aircraft, and $v_1 \in \mathbb{R}$ is the velocity of the aircraft. Using the standard homogeneous representation for g (see Chapter 2), the kinematic

equations of the aircraft motion are given by

$$\dot{g} = \begin{bmatrix} 0 & -\omega_3 & \omega_2 & v_1 \\ \omega_3 & 0 & -\omega_1 & 0 \\ -\omega_2 & \omega_1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} g,$$

where ω_1 stands for pitch rate, ω_2 for roll rate, ω_3 for yaw rate, and v_1 for the velocity of the aircraft. Then the 3-D velocity (ω, v) in the continuous epipolar constraint (5.52) has the form $\omega = [\omega_1, \omega_2, \omega_3]^T, v = [v_1, 0, 0]^T$. For Algorithm 5.3, we have extra constraints on the symmetric matrix $s = \frac{1}{2}(\widehat{\omega}\widehat{v} + \widehat{v}\widehat{\omega})$: $s_1 = s_5 = 0$ and $s_4 = s_6$. Then there are only four different essential parameters left to determine, and we can redefine the motion parameter vector $E^s \in \mathbb{R}^4$ to be $E^s \doteq [v_1, s_2, s_3, s_4]^T$. Then the measurement vector $\boldsymbol{a} \in \mathbb{R}^4$ is given by $\boldsymbol{a} = [u_3 y - u_2 z, 2xy, 2xz, y^2 + z^2]^T$. The continuous epipolar constraint can then be rewritten as

$$\boldsymbol{a}^T E^s = 0.$$

If we define the matrix χ from $\boldsymbol{a}$ as in (5.65), the matrix $\chi^T\chi$ is a 4×4 matrix rather than a 9×9. For estimating the velocity (ω, v), the dimension of the problem is then reduced from nine to four. In this special case, the minimum number of optical flow measurements needed to guarantee a unique solution of E^s is reduced to four instead of eight. Furthermore, the symmetric matrix s recovered from E^s is automatically in the space $\mathcal{S}$, and the remaining steps of the algorithm can thus be dramatically simplified. From this simplified algorithm, the angular velocity $\omega = [\omega_1, \omega_2, \omega_3]^T$ can be fully recovered from the images. The velocity information can then be used for controlling the aircraft. ■

As in the discrete case, the linear algorithm proposed above is not optimal, since it does not enforce the structure of the parameter space during the minimization. Therefore, the recovered velocity does not necessarily minimize the originally chosen error function $\|\chi E^s(\omega, v)\|^2$ on the space $\mathcal{E}_1'$.

Additionally, as in the discrete case, we have to assume that translation is not zero. If the motion is purely rotational, then one can prove that there are infinitely many solutions to the epipolar constraint-related equations. We leave this as an exercise to the reader.

5.4.4 Euclidean constraints and structure reconstruction

As in the discrete case, the purpose of exploiting Euclidean constraints is to reconstruct the scales of the motion and structure. From the above linear algorithm, we know that we can recover the linear velocity v only up to an arbitrary scalar factor. Without loss of generality, we may assume that the velocity of the camera motion to be $(\omega, \eta v)$ with $\|v\| = 1$ and $\eta \in \mathbb{R}$. By now, only the scale factor η is unknown. Substituting $\boldsymbol{X}(t) = \lambda(t)\boldsymbol{x}(t)$ into the equation

$$\dot{\boldsymbol{X}}(t) = \widehat{\omega}\boldsymbol{X}(t) + \eta v(t),$$

we obtain for the image $\boldsymbol{x}^j$ of each point $p^j \in \mathbb{E}^3, j = 1, 2, \ldots, n$,

$$\dot{\lambda}^j \boldsymbol{x}^j + \lambda^j \dot{\boldsymbol{x}}^j = \widehat{\omega}(\lambda^j \boldsymbol{x}^j) + \eta v \quad \Leftrightarrow \quad \dot{\lambda}^j \boldsymbol{x}^j + \lambda^j(\dot{\boldsymbol{x}}^j - \widehat{\omega}\boldsymbol{x}^j) - \eta v = 0. \quad (5.70)$$

As one may expect, in the continuous case, the scale information is then encoded in $\lambda, \dot{\lambda}$ for the location of the 3-D point, and $\eta \in \mathbb{R}^+$ for the linear velocity v. Knowing $\boldsymbol{x}, \dot{\boldsymbol{x}}, \omega$, and v, we see that these constraints are all linear in $\lambda^j, \dot{\lambda}^j, 1 \leq j \leq n$, and η. Also, if $\boldsymbol{x}^j, 1 \leq j \leq n$ are linearly independent of v, i.e. the feature points do not line up with the direction of translation, it can be shown that these linear constraints are not degenerate; hence the unknown scales are determined up to a universal scalar factor. We may then arrange all the unknown scalars into a single vector $\vec{\lambda}$:

$$\vec{\lambda} = [\lambda^1, \lambda^2 \ldots, \lambda^n, \dot{\lambda}^1, \dot{\lambda}^2, \ldots, \dot{\lambda}^n, \eta]^T \quad \in \mathbb{R}^{2n+1}.$$

For n optical flow vectors, $\vec{\lambda}$ is a $(2n+1)$-dimensional vector. (5.70) gives $3n$ (scalar) linear equations. The problem of solving $\vec{\lambda}$ from (5.70) is usually over determined. It is easy to check that in the absence of noise the set of equations given by (5.70) uniquely determines $\vec{\lambda}$ if the configuration is noncritical. We can therefore write all the equations in the matrix form

$$M\vec{\lambda} = 0,$$

with $M \in \mathbb{R}^{3n \times (2n+1)}$ a matrix depending on ω, v, and $\{(\boldsymbol{x}^j, \dot{\boldsymbol{x}}^j)\}_{j=1}^n$. Then, in the presence of noise, the linear least-squares estimate of $\vec{\lambda}$ is simply the eigenvector of $M^T M$ corresponding to the smallest eigenvalue.

Notice that the time derivative of the scales $\{\dot{\lambda}^j\}_{j=1}^n$ can also be estimated. Suppose we have done the above recovery for a time interval, say (t_0, t_f). Then we have the estimate $\vec{\lambda}(t)$ as a function of time t. But $\vec{\lambda}(t)$ at each time t is determined only up to an arbitrary scalar factor. Hence $\rho(t)\vec{\lambda}(t)$ is also a valid estimation for any positive function $\rho(t)$ defined on (t_0, t_f). However, since $\rho(t)$ is multiplied by both $\lambda(t)$ and $\dot{\lambda}(t)$, their ratio

$$r(t) = \dot{\lambda}(t)/\lambda(t)$$

is independent of the choice of $\rho(t)$. Notice that $\frac{d}{dt}(\ln \lambda) = \dot{\lambda}/\lambda$. Let the logarithm of the structural scale λ be $y = \ln \lambda$. Then a time-consistent estimation $\lambda(t)$ needs to satisfy the following ordinary differential equation, which we call the *dynamical scale ODE*

$$\dot{y}(t) = r(t).$$

Given $y(t_0) = y_0 = \ln(\lambda(t_0))$, we solve this ODE and obtain $y(t)$ for $t \in [t_0, t_f]$. Then we can recover a consistent scale $\lambda(t)$ given by

$$\lambda(t) = \exp(y(t)).$$

Hence (structure and motion) scales estimated at different time instances now are all relative to the same scale at time t_0. Therefore, in the continuous case, we are also able to recover all the scales as functions of time up to a universal scalar factor. The reader must be aware that the above scheme is only *conceptual*. In reality, the ratio function $r(t)$ would never be available for *every* time instant in $[t_0, t_f]$.

Universal scale ambiguity

In both the discrete and continuous cases, in principle, the proposed schemes can reconstruct both the Euclidean structure and motion up to a universal scalar factor. This ambiguity is intrinsic, since one can scale the entire world up or down with a scaling factor while all the images obtained remain the same. In all the algorithms proposed above, this factor is fixed (rather arbitrarily, in fact) by imposing the translation scale to be 1. In practice, this scale and its unit can also be chosen to be directly related to some known length, size, distance, or motion of an object in space.

5.4.5 Continuous homography for a planar scene

In this section, we consider the continuous version of the case that we have studied in Section 5.3, where all the feature points of interest are lying on a plane P. Planar scenes are a degenerate case for the discrete epipolar constraint, and also for the continuous case. Recall that in the continuous scenario, instead of having image pairs, we measure the image point $\boldsymbol{x}$ and its optical flow $\boldsymbol{u} = \dot{\boldsymbol{x}}$. Other assumptions are the same as in Section 5.3.

Suppose the camera undergoes a rigid-body motion with body angular and linear velocities ω, v. Then the time derivative of coordinates $\boldsymbol{X} \in \mathbb{R}^3$ of a point p (with respect to the camera frame) satisfies[13]

$$\dot{\boldsymbol{X}} = \widehat{\omega}\boldsymbol{X} + v. \tag{5.71}$$

Let $N \in \mathbb{R}^3$ be the surface normal to P (with respect to the camera frame) at time t. Then, if $d(t) > 0$ is the distance from the optical center of the camera to the plane P at time t, then

$$N^T\boldsymbol{X} = d \quad \Leftrightarrow \quad \frac{1}{d}N^T\boldsymbol{X} = 1, \quad \forall \boldsymbol{X} \in P. \tag{5.72}$$

Substituting equation (5.72) into equation (5.71) yields the relation

$$\dot{\boldsymbol{X}} = \widehat{\omega}\boldsymbol{X} + v = \widehat{\omega}\boldsymbol{X} + v\frac{1}{d}N^T\boldsymbol{X} = \left(\widehat{\omega} + \frac{1}{d}vN^T\right)\boldsymbol{X}. \tag{5.73}$$

As in the discrete case, we call the matrix

$$H \doteq \left(\widehat{\omega} + \frac{1}{d}vN^T\right) \in \mathbb{R}^{3\times 3} \tag{5.74}$$

the *continuous homography matrix*. For simplicity, here we use the same symbol H to denote it, and it really is a continuous (or infinitesimal) version of the (discrete) homography matrix $H = R + \frac{1}{d}TN^T$ studied in Section 5.3.

[13] Here, as in previous cases, we assume implicitly that time dependency of $\boldsymbol{X}$ on t is smooth so that we can take derivatives whenever necessary. However, for simplicity, we drop the dependency of $\boldsymbol{X}$ on t in the notation $\boldsymbol{X}(t)$.

Note that the matrix H depends both on the continuous motion parameters $\{\omega, v\}$ and structure parameters $\{N, d\}$ that we wish to recover. As in the discrete case, there is an inherent scale ambiguity in the term $\frac{1}{d}v$ in equation (5.74). Thus, in general, knowing H, one can recover only the ratio of the camera translational velocity scaled by the distance to the plane.

From the relation

$$\lambda \boldsymbol{x} = \boldsymbol{X}, \quad \dot{\lambda}\boldsymbol{x} + \lambda \boldsymbol{u} = \dot{\boldsymbol{X}}, \quad \dot{\boldsymbol{X}} = H\boldsymbol{X}, \tag{5.75}$$

we have

$$\boxed{\boldsymbol{u} = H\boldsymbol{x} - \frac{\dot{\lambda}}{\lambda}\boldsymbol{x}.} \tag{5.76}$$

This is indeed the continuous version of the planar homography.

5.4.6 Estimating the continuous homography matrix

In order to further eliminate the depth scale λ in equation (5.76), multiplying both sides by the skew-symmetric matrix $\widehat{\boldsymbol{x}} \in \mathbb{R}^{3\times 3}$, we obtain the equation

$$\boxed{\widehat{\boldsymbol{x}} H \boldsymbol{x} = \widehat{\boldsymbol{x}}\boldsymbol{u}.} \tag{5.77}$$

We may call this the *continuous homography constraint* or the *continuous planar epipolar constraint* as a continuous version of the discrete case.

Since this constraint is linear in H, by stacking the entries of H as

$$H^s = [H_{11}, H_{21}, H_{31}, H_{12}, H_{22}, H_{32}, H_{13}, H_{23}, H_{33}]^T \quad \in \mathbb{R}^9,$$

we may rewrite (5.77) as

$$\boldsymbol{a}^T H^s = \widehat{\boldsymbol{x}}\boldsymbol{u},$$

where $\boldsymbol{a} \in \mathbb{R}^{9\times 3}$ is the Kronecker product $\boldsymbol{x} \otimes \widehat{\boldsymbol{x}}$. However, since the skew-symmetric matrix $\widehat{\boldsymbol{x}}$ is only of rank 2, the equation imposes only two constraints on the entries of H. Given a set of n image point and velocity pairs $\{(\boldsymbol{x}^j, \boldsymbol{u}^j)\}_{j=1}^n$ of points on the plane, we may stack all equations $\boldsymbol{a}^{jT} H^s = \widehat{\boldsymbol{x}^j}\boldsymbol{u}^j, j = 1, 2, \ldots, n$, into a single equation

$$\chi H^s = B, \tag{5.78}$$

where $\chi \doteq \left[\boldsymbol{a}^1, \ldots, \boldsymbol{a}^n\right]^T \in \mathbb{R}^{3n\times 9}$ and $B \doteq \left[(\widehat{\boldsymbol{x}^1}\boldsymbol{u}^1)^T, \ldots, (\widehat{\boldsymbol{x}^j}\boldsymbol{u}^j)^T\right]^T \in \mathbb{R}^{3n}$.

In order to solve uniquely (up to a scalar factor) for H^s, we must have $\text{rank}(\chi) = 8$. Since each pair of image points gives two constraints, we expect that at least four optical flow pairs would be necessary for a unique estimate of H (up to a scalar factor). In analogy with the discrete case, we have the following statement, the proof of which we leave to the reader as a linear-algebra exercise.

Proposition 5.32 (Four-point continuous homography). *We have $rank(\chi) = 8$ if and only if there exists a set of four points (out of the n) such that any three of them are not collinear; i.e. they are in general configuration in the plane.*

Then, if optical flow at more than four points in general configuration in the plane is given, using linear least-squares techniques, equation (5.78) can be used to recover H^s up to one dimension, since χ has a one-dimensional null space. That is, we can recover $H_L = H - \xi H_K$, where H_L corresponds to the minimum-norm linear least-squares estimate of H solving $\min \|\chi H^s - B\|^2$, and H_K corresponds to a vector in null(χ) and $\xi \in \mathbb{R}$ is an unknown scalar factor.

By inspection of equation (5.77) one can see that $H_K = I$, since $\widehat{\boldsymbol{x}} I \boldsymbol{x} = \widehat{\boldsymbol{x}} \boldsymbol{x} = 0$. Then we have

$$H = H_L + \xi I. \tag{5.79}$$

Thus, in order to recover H, we need only to identify the unknown ξ. So far, we have not considered the special structure of the matrix H. Next, we give constraints imposed by the structure of H that will allow us to identify ξ, and thus uniquely recover H.

Lemma 5.33. *Suppose $u, v \in \mathbb{R}^3$, and $\|u\|^2 = \|v\|^2 = \alpha$. If $u \neq v$, the matrix $D = uv^T + vu^T \in \mathbb{R}^{3\times 3}$ has eigenvalues $\{\lambda_1, 0, \lambda_3\}$, where $\lambda_1 > 0$, and $\lambda_3 < 0$. If $u = \pm v$, the matrix D has eigenvalues $\{\pm 2\alpha, 0, 0\}$.*

Proof. Let $\beta = u^T v$. If $u \neq \pm v$, we have $-\alpha < \beta < \alpha$. We can solve the eigenvalues and eigenvectors of D by

$$\begin{aligned} D(u+v) &= (\beta+\alpha)(u+v), \\ D(u\times v) &= 0, \\ D(u-v) &= (\beta-\alpha)(u-v). \end{aligned}$$

Clearly, $\lambda_1 = (\beta+\alpha) > 0$ and $\lambda_3 = \beta - \alpha < 0$. It is easy to check the conditions on D when $u = \pm v$. □

Lemma 5.34 (Normalization of the continuous homography matrix). *Given the H_L part of a continuous planar homography matrix of the form $H = H_L + \xi I$, we have*

$$\xi = -\frac{1}{2}\gamma_2\big(H_L + H_L^T\big), \tag{5.80}$$

where $\gamma_2\big(H_L + H_L^T\big) \in \mathbb{R}$ is the second-largest eigenvalue of $H_L + H_L^T$.

Proof. In this proof, we will work with sorted eigenvalues; that is, if $\{\lambda_1, \lambda_2, \lambda_3\}$ are eigenvalues of some matrix, then $\lambda_1 \geq \lambda_2 \geq \lambda_3$. If the points are not in general configuration, then rank$(\chi) < 7$, and the problem is under constrained. Now suppose the points are in general configuration. Then by least-squares estimation we may recover $H_L = H - \xi I$ for some unknown $\xi \in \mathbb{R}$. By Lemma 5.33, $H + H^T = \frac{1}{d} v N^T + \frac{1}{d} N v^T$ has eigenvalues $\{\lambda_1, \lambda_2, \lambda_3\}$, where $\lambda_1 \geq 0$, $\lambda_2 = 0$, and $\lambda_3 \leq 0$. So compute the eigenvalues of $H_L + H_L^T$ and denote them

by $\{\gamma_1, \gamma_2, \gamma_3\}$. Since we have $H = H_L + \xi I$, then $\lambda_i = \gamma_i + 2\xi$, for $i = 1, 2, 3$. Since we must have $\lambda_2 = 0$, we have $\xi = -\frac{1}{2}\gamma_2$. □

Therefore, knowing H_L, we can fully recover the continuous homography matrix as $H = H_L - \frac{1}{2}\gamma_2 I$.

5.4.7 Decomposing the continuous homography matrix

We now address the task of decomposing the recovered $H = \widehat{\omega} + \frac{1}{d} v N^T$ into its motion and structure parameters $\{\omega, \frac{v}{d}, N\}$. The following constructive proof provides an algebraic technique for the recovery of motion and structure parameters.

Theorem 5.35 (Decomposition of continuous homography matrix). *Given a matrix $H \in \mathbb{R}^{3\times 3}$ in the form $H = \widehat{\omega} + \frac{1}{d} v N^T$, one can recover the motion and structure parameters $\{\widehat{\omega}, \frac{1}{d}v, N\}$ up to at most two physically possible solutions. There is a unique solution if $v = 0$, $v \times N = 0$, or $e_3^T v = 0$, where $e_3 = [0, 0, 1]^T$ is the optical axis.*

Proof. Compute the eigenvalue/eigenvector pairs of $H + H^T$ and denote them by $\{\lambda_i, u_i\}$, $i = 1, 2, 3$. If $\lambda_i = 0$ for $i = 1, 2, 3$, then we have $v = 0$ and $\widehat{\omega} = H$. In this case we cannot recover the normal of the plane N. Otherwise, if $\lambda_1 > 0$, and $\lambda_3 < 0$, then we have $v \times N \neq 0$. Let $\alpha = \|v/d\| > 0$, let $\tilde{v} = v/\sqrt{\alpha}$ and $\tilde{N} = \sqrt{\alpha} N$, and let $\beta = \tilde{v}^T \tilde{N}$. According to Lemma 5.33, the eigenvalue/eigenvector pairs of $H + H^T$ are given by

$$\begin{aligned} &\lambda_1 = \beta + \alpha > 0, \qquad u_1 = \tfrac{1}{\|\tilde{v}+\tilde{N}\|}\big(\tilde{v} + \tilde{N}\big), \\ &\lambda_3 = \beta - \alpha < 0, \qquad u_3 = \pm\tfrac{1}{\|\tilde{v}-\tilde{N}\|}\big(\tilde{v} - \tilde{N}\big). \end{aligned} \tag{5.81}$$

Then $\alpha = \frac{1}{2}(\lambda_1 - \lambda_3)$. It is easy to check that $\|\tilde{v} + \tilde{N}\|^2 = 2\lambda_1$, $\|\tilde{v} - \tilde{N}\|^2 = -2\lambda_3$. Together with (5.81), we have two solutions (due to the two possible signs for u_3):

$$\begin{aligned} \tilde{v}_1 &= \tfrac{1}{2}\big(\sqrt{2\lambda_1}\, u_1 + \sqrt{-2\lambda_3}\, u_3\big), & \tilde{v}_2 &= \tfrac{1}{2}\big(\sqrt{2\lambda_1}\, u_1 - \sqrt{-2\lambda_3}\, u_3\big), \\ \tilde{N}_1 &= \tfrac{1}{2}\big(\sqrt{2\lambda_1}\, u_1 - \sqrt{-2\lambda_3}\, u_3\big), & \tilde{N}_2 &= \tfrac{1}{2}\big(\sqrt{2\lambda_1}\, u_1 + \sqrt{-2\lambda_3}\, u_3\big), \\ \widehat{\omega}_1 &= H - \tilde{v}_1 \tilde{N}_1^T, & \widehat{\omega}_2 &= H - \tilde{v}_2 \tilde{N}_2^T. \end{aligned}$$

In the presence of noise, the estimate of $\widehat{\omega} = H - \tilde{v}\tilde{N}^T$ is not necessarily an element in $so(3)$. In algorithms, one may take its skew-symmetric part,

$$\widehat{\omega} = \frac{1}{2}\left((H - \tilde{v}\tilde{N}^T) - (H - \tilde{v}\tilde{N}^T)^T\right).$$

There is another sign ambiguity, since $(-\tilde{v})(-\tilde{N})^T = \tilde{v}\tilde{N}^T$. This sign ambiguity leads to a total of four possible solutions for decomposing H back to $\{\widehat{\omega}, \frac{1}{d}v, N\}$ given in Table 5.2.

Solution 1	$\frac{1}{d}v_1$	=	$\sqrt{\alpha}\tilde{v}_1$	Solution 3	$\frac{1}{d}v_3$	=	$-\frac{1}{d}v_1$
	N_1	=	$\frac{1}{\sqrt{\alpha}}\tilde{N}_1$		N_3	=	$-N_1$
	$\widehat{\omega}_1$	=	$H - \tilde{v}_1\tilde{N}_1^T$		$\widehat{\omega}_3$	=	$\widehat{\omega}_1$
Solution 2	$\frac{1}{d}v_2$	=	$\sqrt{\alpha}\tilde{v}_2$	Solution 4	$\frac{1}{d}v_4$	=	$-\frac{1}{d}v_2$
	N_2	=	$\frac{1}{\sqrt{\alpha}}\tilde{N}_2$		v_4	=	$-N_2$
	$\widehat{\omega}_2$	=	$H - \tilde{v}_2\tilde{N}_2^T$		$\widehat{\omega}_4$	=	$\widehat{\omega}_2$

Table 5.2. Four solutions for continuous planar homography decomposition. Here α is computed as before as $\alpha = \frac{1}{2}(\lambda_1 - \lambda_3)$.

In order to reduce the number of physically possible solutions, we impose the positive depth constraint: since the camera can only see points that are in front of it, we must have $N^T e_3 > 0$. Therefore, if solution 1 is the correct one, this constraint will eliminate solution 3 as being physically impossible. If $v^T e_3 \neq 0$, one of solutions 2 or 4 will be eliminated, whereas if $v^T e_3 = 0$, both solutions 2 and 4 will be eliminated. For the case that $v \times N = 0$, it is easy to see that solutions 1 and 2 are equivalent, and that imposing the positive depth constraint leads to a unique solution. □

Despite the fact that as in the discrete case, there is a close relationship between the continuous epipolar constraint and continuous homography, we will not develop the details here. Basic intuition and necessary technical tools have already been established in this chapter, and at this point interested readers may finish that part of the story with ease, or more broadly, apply these techniques to solve other special problems that one may encounter in real-world applications.

We summarize Sections 5.4.6 and 5.4.7 by presenting the continuous four-point Algorithm 5.4 for motion estimation from a planar scene.

5.5 Summary

Given corresponding points in two images $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ of a point p, or, in continuous time, optical flow $(\boldsymbol{u}, \boldsymbol{x})$, we summarize the constraints and relations between the image data and the unknown motion parameters in Table 5.3.

Despite the similarity between the discrete and the continuous case, one must be aware that there are indeed important subtle differences between these two cases, since the differentiation with respect to time t changes the algebraic relation between image data and unknown motion parameters.

In the presence of noise, the motion recovery problem in general becomes a problem of minimizing a cost function associated with statistical optimality or geometric error criteria subject to the above constraints. Once the camera motion is recovered, an overall 3-D reconstruction of both the camera motion and scene structure can be obtained up to a global scaling factor.

Algorithm 5.4 (The continuous four-point algorithm for a planar scene).

For a given set of optical flow vectors $(\boldsymbol{u}^j, \boldsymbol{x}^j)$, $j = 1, 2, \ldots, n$ ($n \geq 4$), of points on a plane $N^T \boldsymbol{X} = d$, this algorithm finds $\{\widehat{\omega}, \frac{1}{d}v, N\}$ that solves

$$\widehat{\boldsymbol{x}^j}^T \left(\widehat{\omega} + \frac{1}{d} v N^T \right) \boldsymbol{x}^j = \widehat{\boldsymbol{x}^j} \boldsymbol{u}^j, \quad j = 1, 2, \ldots, n.$$

1. **Compute a first approximation of the continuous homography matrix**
   Construct the matrix $\chi = [\boldsymbol{a}^1, \boldsymbol{a}^2, \ldots, \boldsymbol{a}^n]^T \in \mathbb{R}^{3n \times 9}$, $B = [\boldsymbol{b}^{1T}, \boldsymbol{b}^{2T}, \ldots, \boldsymbol{b}^{nT}]^T \in \mathbb{R}^{3n}$ from the optical flow $(\boldsymbol{u}^j, \boldsymbol{x}^j)$, where $\boldsymbol{a}^j = \boldsymbol{x}^j \otimes \widehat{\boldsymbol{x}^j} \in \mathbb{R}^{9 \times 3}$ and $\boldsymbol{b} = \widehat{\boldsymbol{x}}\boldsymbol{u} \in \mathbb{R}^3$. Find the vector $H_L^s \in \mathbb{R}^9$ as
   $$H_L^s = \chi^\dagger B,$$
   where $\chi^\dagger \in \mathbb{R}^{9 \times 3n}$ is the pseudo-inverse of χ. Unstack H_L^s to obtain the 3×3 matrix H_L.
2. **Normalization of the continuous homography matrix**
   Compute the eigenvalue values $\{\gamma_1, \gamma_2, \gamma_3\}$ of the matrix $H_L^T + H_L$ and normalize it as
   $$H = H_L - \frac{1}{2}\gamma_2 I.$$
3. **Decomposition of the continuous homography matrix**
   Compute the eigenvalue decomposition of
   $$H^T + H = U \Lambda U^T$$
   and compute the four solutions for a decomposition $\{\widehat{\omega}, \frac{1}{d}v, N\}$ as in the proof of Theorem 5.35. Select the two physically possible ones by imposing the positive depth constraint $N^T e_3 > 0$.

5.6 Exercises

Exercise 5.1 (Linear equation). Solve $x \in \mathbb{R}^n$ from the linear equation

$$Ax = b,$$

where $A \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. In terms of conditions on the matrix A and vector b, describe when a solution exists and when it is unique. In case the solution is not unique, describe the entire solution set.

Exercise 5.2 (Properties of skew-symmetric matrices).

1. Prove Lemma 5.4.
2. Prove Lemma 5.24.

Exercise 5.3 (Skew-symmetric matrix continued). Given a vector $T \in \mathbb{R}^3$ with unit length, i.e. $\|T\| = 1$, show that:

1. The identity holds: $\widehat{T}^T \widehat{T} = \widehat{T}\widehat{T}^T = I - TT^T$ (note that the superscript ^{T} stands for matrix transpose).

	Epipolar constraint	(Planar) homography
Discrete motion	$\boldsymbol{x}_2^T \widehat{T} R \boldsymbol{x}_1 = 0$	$\widehat{\boldsymbol{x}_2}(R + \frac{1}{d} T N^T)\boldsymbol{x}_1 = 0$
Matrices	$E = \widehat{T} R$	$H = R + \frac{1}{d} T N^T$
Relation	$\exists v \in \mathbb{R}^3,\ H = \widehat{T}^T E + T v^T$	
Continuous motion	$\boldsymbol{x}^T \widehat{\omega} \widehat{v} \boldsymbol{x} + \boldsymbol{u}^T \widehat{v} \boldsymbol{x} = 0$	$\widehat{\boldsymbol{x}}(\widehat{\omega} + \frac{1}{d} v N^T)\boldsymbol{x} = \widehat{\boldsymbol{u}}\boldsymbol{x}$
Matrices	$E = \begin{bmatrix} \frac{1}{2}(\widehat{\omega}\widehat{v} + \widehat{v}\widehat{\omega}) \\ \widehat{v} \end{bmatrix}$	$H = \widehat{\omega} + \frac{1}{d} v N^T$
Linear algorithms	8 points	4 points
Decomposition	1 solution	2 solutions

Table 5.3. Here the number of points is required by corresponding linear algorithms, and we count only the number of physically possible solutions from corresponding decomposition algorithms *after* applying the positive depth constraint.

2. Explain the effect of multiplying a vector $u \in \mathbb{R}^3$ by the matrix $P = I - TT^T$. Show that $P^n = P$ for any integer n.
3. Show that $\widehat{T}^T \widehat{T} \widehat{T} = \widehat{T} \widehat{T}^T \widehat{T} = \widehat{T}$. Explain geometrically why this is true.
4. How do the above statements need to be changed if the vector T is not of unit length?

Exercise 5.4 (A rank condition for the epipolar constraint). Show that $\boldsymbol{x}_2^T \widehat{T} R \boldsymbol{x}_1 = 0$ if and only if

$$\text{rank}\,[\widehat{\boldsymbol{x}_2} R \boldsymbol{x}_1,\ \ \widehat{\boldsymbol{x}_2} T] \leq 1.$$

Exercise 5.5 (Parallel epipolar lines). Explain under what conditions the family of epipolar lines in at least one of the image planes will be parallel to each other. Where is the corresponding epipole (in terms of its homogeneous coordinates)?

Exercise 5.6 (Essential matrix for planar motion). Suppose we know that the camera always moves on a plane, say the XY plane. Show that:

1. The essential matrix $E = \widehat{T} R$ is of the special form
$$E = \begin{bmatrix} 0 & 0 & a \\ 0 & 0 & b \\ c & d & 0 \end{bmatrix}, \quad a, b, c, d \in \mathbb{R}. \tag{5.82}$$
2. Without using the SVD-based decomposition introduced in this chapter, find a solution to (R, T) in terms of a, b, c, d.

Exercise 5.7 (Rectified essential matrix). Suppose that using the linear algorithm, you obtain an essential matrix E of the form

$$E = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & a \\ 0 & -a & 0 \end{bmatrix}, \quad a \in \mathbb{R}. \tag{5.83}$$

What type of motion (R, T) does the camera undergo? How many solutions exist exactly?

Exercise 5.8 (Triangulation). Given two images $\boldsymbol{x}_1, \boldsymbol{x}_2$ of a point p together with the relative camera motion (R, T), $\boldsymbol{X}_2 = R\boldsymbol{X}_1 + T$:

1. express the depth of p with respect to the first image, i.e. λ_1 in terms of $\boldsymbol{x}_1, \boldsymbol{x}_2$, and (R, T);
2. express the depth of p with respect to the second image, i.e. λ_2 in terms of $\boldsymbol{x}_1, \boldsymbol{x}_2$, and (R, T).

Exercise 5.9 (Rotational motion). Assume that the camera undergoes pure rotational motion; i.e. it rotates around its center. Let $R \in SO(3)$ be the rotation of the camera and $\omega \in so(3)$ be the angular velocity. Show that in this case, we have:

1. discrete case: $\boldsymbol{x}_2^T \widehat{T} R \boldsymbol{x}_1 \equiv 0, \quad \forall T \in \mathbb{R}^3$;
2. continuous case: $\boldsymbol{x}^T \widehat{\omega} \widehat{v} \boldsymbol{x} + \boldsymbol{u}^T \widehat{v} \boldsymbol{x} \equiv 0, \quad \forall v \in \mathbb{R}^3$.

Exercise 5.10 (Projection onto $O(3)$). Given an arbitrary 3×3 matrix $M \in \mathbb{R}^{3\times 3}$ with positive singular values, find the orthogonal matrix $R \in O(3)$ such that the error $\|R - M\|_f^2$ is minimized. Is the solution unique? Note: Here we allow $\det(R) = \pm 1$.

Exercise 5.11 (Four motions related to an epipolar constraint). Suppose $E = \widehat{T}R$ is a solution to the epipolar constraint $\boldsymbol{x}_2^T E \boldsymbol{x}_1 = 0$. Then $-E$ is also an essential matrix, which obviously satisfies the same epipolar constraint (for given corresponding images).

1. Explain geometrically how these four motions are related. [Hint: Consider a pure translation case. If R is a rotation about T by an angle π, then $\widehat{T}R = -\widehat{T}$, which is in fact the *twisted pair* ambiguity.]
2. Show that in general, for three out of the four solutions, the equation $\lambda_2 \boldsymbol{x}_2 = \lambda_1 R \boldsymbol{x}_1 + T$ will yield either negative λ_1 or negative λ_2 or both. Hence only one solution satisfies the positive depth constraint.

Exercise 5.12 (Geometric distance to an epipolar line). Given two image points $\boldsymbol{x}_1, \tilde{\boldsymbol{x}}_2$ with respect to camera frames with their relative motion (R, T), show that the geometric distance d_2 defined in Figure 5.7 is given by the formula

$$d_2^2 = \frac{(\tilde{\boldsymbol{x}}_2^T \widehat{T} R \boldsymbol{x}_1)^2}{\|\widehat{e}_3 \widehat{T} R \boldsymbol{x}_1\|^2},$$

where $e_3 = [0, 0, 1]^T \in \mathbb{R}^3$.

Exercise 5.13 (A six-point algorithm). In this exercise, we show how to use some of the (algebraic) structure of the essential matrix to reduce the number of matched pairs of points from 8 to 6.

1. Show that if a matrix E is an essential matrix, then it satisfies the identity
$$EE^T E = \frac{1}{2}\text{trace}(EE^T)E.$$
2. Show that the dimension of the space of matrices $\{F\} \subset \mathbb{R}^{3\times 3}$ that satisfy the epipolar constraints
$$(\boldsymbol{x}_2^j)^T F \boldsymbol{x}_1^j = 0, \quad j = 1, 2, \ldots, 6,$$

is three. Hence the essential matrix E can be expressed as a linear combination $E = \alpha_1 F_1 + \alpha_2 F_2 + \alpha_3 F_3$ for some linearly independent matrices F_1, F_2, F_3 that satisfy the above equations.

3. To further determine the coefficients $\alpha_1, \alpha_2, \alpha_3$, show that the identity in (a) gives nine scalar equations linearly in the nine unknowns $\{\alpha_1^i \alpha_2^j \alpha_3^k\}$, $i + j + k = 3$, $0 \leq i, j, k \leq 3$. (Why nine?) Hence, the essential matrix E can be determined from six pairs of matched points.

Exercise 5.14 (Critical surfaces). To have a unique solution (up to a scalar factor), it is very important for the points considered in the above six-point or eight-point algorithms to be in general position. If a (dense) set of points whose images allow at least two distinct essential matrices, we say that they are "critical," Let $\boldsymbol{X} \in \mathbb{R}^3$ be coordinates of such a point and (R, T) be the motion of a camera. Let $\boldsymbol{x}_1 \sim \boldsymbol{X}$ and $\boldsymbol{x}_2 \sim (R\boldsymbol{X} + T)$ be two images of the point.

1. Show that if
$$(R\boldsymbol{X} + T)^T \widehat{T'} R' \boldsymbol{X} = 0,$$
then
$$\boldsymbol{x}_2^T \widehat{T} R \boldsymbol{x}_1 = 0, \quad \boldsymbol{x}_2^T \widehat{T'} R' \boldsymbol{x}_1 = 0.$$

2. Show that for points $\boldsymbol{X} \in \mathbb{R}^3$ that satisfy the equation $(R\boldsymbol{X} + T)^T \widehat{T'} R' \boldsymbol{X} = 0$, their homogeneous coordinates $\bar{\boldsymbol{X}} = [X, 1]^T \in \mathbb{R}^4$ satisfy the quadratic equation
$$\bar{\boldsymbol{X}}^T \begin{bmatrix} R^T \widehat{T'} R' + {R'}^T \widehat{T'}^T R & {R'}^T \widehat{T'}^T T \\ T^T \widehat{T'} R' & 0 \end{bmatrix} \bar{\boldsymbol{X}} = 0.$$
This quadratic surface is denoted by $C_1 \subset \mathbb{R}^3$ and is called a *critical surface*. So no matter how many points one chooses on such a surface, their two corresponding images always satisfy epipolar constraints for at least two different essential matrices.

3. Symmetrically, points defined by the equation $(R'\boldsymbol{X} + T')^T \widehat{T} R \boldsymbol{X} = 0$ will have similar properties. This gives another quadratic surface,
$$C_2 : \ \bar{\boldsymbol{X}}^T \begin{bmatrix} {R'}^T \widehat{T} R + R^T \widehat{T}^T R' & R^T \widehat{T}^T T' \\ {T'}^T \widehat{T} R & 0 \end{bmatrix} \bar{\boldsymbol{X}} = 0.$$
Argue that a set of points on the surface C_1 observed from two vantage points related by (R, T) could be interpreted as a corresponding set of points on the surface C_2 observed from two vantage points related by (R', T').

Exercise 5.15 (Estimation of the homography). We say that two images are related by a *homography* if the homogeneous coordinates of the two images $\boldsymbol{x}_1, \boldsymbol{x}_2$ of every point satisfy
$$\boldsymbol{x}_2 \sim H \boldsymbol{x}_1$$
for some nonsingular matrix $H \in \mathbb{R}^{3\times 3}$. Show that in general one needs four pairs of $(\boldsymbol{x}_1, \boldsymbol{x}_2)$ to determine the matrix H (up to a scalar factor).

Exercise 5.16 Under a homography $H \in \mathbb{R}^{3\times 3}$ from $\mathbb{R}^2$ to $\mathbb{R}^2$, a standard unit square with the homogeneous coordinates for the four corners

$$(0,0,1),\ (1,0,1),\ (1,1,1),\ (0,1,1)$$

is mapped to

$$(6,5,1),\ (4,3,1),\ (6,4.5,1),\ (10,8,1),$$

respectively. Determine the matrix H with its last entry H_{33} normalized to 1.

Exercise 5.17 (Epipolar line homography from an essential matrix). From the geometric interpretation of epipolar lines in Figure 5.2, we know that there is a one-to-one map between the family of epipolar lines $\{\boldsymbol{\ell}_1\}$ in the first image plane (through the epipole $\boldsymbol{e}_1$) and the family of epipolar lines $\{\boldsymbol{\ell}_2\}$ in the second. Suppose that the essential matrix E is known. Show that this map is in fact a homography. That is, there exists a nonsingular matrix $H \in \mathbb{R}^{3\times 3}$ such that

$$\boldsymbol{\ell}_2 \sim H\boldsymbol{\ell}_1$$

for any pair of corresponding epipolar lines $(\boldsymbol{\ell}_1, \boldsymbol{\ell}_2)$. Find an explicit form for H in terms of E.

Exercise 5.18 (Homography with respect to the second camera frame). In the chapter, we have learned that for a transformation $\boldsymbol{X}_2 = R\boldsymbol{X}_1 + T$ on a plane $N^T\boldsymbol{X}_1 = 1$ (expressed in the first camera frame), we have a homography $H = R + TN^T$ such that $\boldsymbol{x}_2 \sim H\boldsymbol{x}_1$ relates the two images of the plane.

1. Now switch roles of the first and the second camera frames and show that the new homography matrix becomes
$$\tilde{H} = \left(R^T + \frac{-R^T T}{1 + N^T R^T T} N^T R^T\right). \tag{5.84}$$
2. What is the relationship between H and $\tilde{H}$? Provide a formal proof to your answer. Explain why this should be expected.

Exercise 5.19 (Two physically possible solutions for the homography decomposition). Let us study in the nature of the two physically possible solutions for the homography decomposition. Without loss of generality, suppose that the true homography matrix is $H = I + ab^T$ with $\|a\| = 1$.

1. Show that $R' = -I + 2aa^T$ is a rotation matrix.
2. Show that $H' = R' + (-a)(b + 2a)^T$ is equal to $-H$.
3. Since $(H')^T H' = H^T H$, conclude that both $\{I, a, b\}$ and $\{R', -a, (b + 2a)\}$ are solutions from the homography decomposition of H.
4. Argue that, under certain conditions on the relationship between a and b, the second solution is also physically possible.
5. What is the geometric relationship between these two solutions? Draw a figure to illustrate your answer.

Exercise 5.20 (Various expressions for the image motion field). In the continuous-motion case, suppose that the camera motion is (ω, v), and $\boldsymbol{u} = \dot{\boldsymbol{x}}$ is the velocity of the image $\boldsymbol{x}$ of a point $\boldsymbol{X} = [X, Y, Z]^T$ in space. Show that:

1. For a spherical perspective projection; i.e. $\lambda = \|\boldsymbol{X}\|$, we have
$$\boldsymbol{u} = -\widehat{\boldsymbol{x}}\omega + \frac{1}{\lambda}\widehat{\boldsymbol{x}}^2 v. \tag{5.85}$$
2. For a planar perspective projection; i.e. $\lambda = Z$, we have
$$\boldsymbol{u} = (-\widehat{\boldsymbol{x}} + \boldsymbol{x}e_3^T\widehat{\boldsymbol{x}})\omega + \frac{1}{\lambda}(I - \boldsymbol{x}e_3^T)v, \tag{5.86}$$
or in coordinates,
$$\begin{bmatrix}\dot{x}\\ \dot{y}\end{bmatrix} = \begin{bmatrix}-xy & x^2 & -y\\ -(1+y^2) & xy & x\end{bmatrix}\omega + \frac{1}{\lambda}\begin{bmatrix}1 & 0 & -x\\ 0 & 1 & -y\end{bmatrix} v. \tag{5.87}$$
3. Show that in the planar perspective case, equation (5.76) is equivalent to
$$\boldsymbol{u} = (I - \boldsymbol{x}e_3^T)H\boldsymbol{x}. \tag{5.88}$$
From this equation, discuss under what conditions the motion field for a planar scene is an affine function of the image coordinates; i.e.
$$\boldsymbol{u} = A\boldsymbol{x}, \tag{5.89}$$
where A is a constant 3×3 affine matrix that does not depend on the image point $\boldsymbol{x}$.

Exercise 5.21 (Programming: implementation of (discrete) eight-point algorithm). Implement a version of the three-step pose estimation algorithm for two views. Your Matlab code should be responsible for

- Initialization: Generate a set of n (≥ 8) 3-D points; generate a rigid-body motion (R, T) between two camera frames and project (the coordinates of) the points (relative to the camera frame) onto the image plane correctly. Here you may assume that the focal length is 1. This step will give you corresponding images as input to the algorithm.
- Motion Recovery: using the corresponding images and the algorithm to compute the motion $(\tilde{R}, \tilde{T})$ and compare it to the ground truth (R, T).

After you get the correct answer from the above steps, here are a few suggestions for you to try with the algorithm (or improve it):

- A more realistic way to generate these 3-D points is to make sure that they are all indeed "in front of" the image plane before and after the camera moves.
- Systematically add some noise to the projected images and see how the algorithm responds. Try different camera motions and different layouts of the points in 3-D.
- Finally, to make the algorithm fail, take all the 3-D points from some plane in front of the camera. Run the program and see what you get (especially with some noise on the images).

Exercise 5.22 (Programming: implementation of the continuous eight-point algorithm). Implement a version of the four-step velocity estimation algorithm for optical flow.