Chapter 16

Multiple cameras

This chapter extends the discussion of the pinhole camera model. In chapter 14 we showed how to find the 3D position of a point based on its projections into multiple cameras. However, this approach was contingent on knowing the intrinsic and extrinsic parameters of these cameras, and this information is often unknown. In this chapter, we discuss methods for reconstruction in the absence of such information. Before reading this chapter, readers should ensure that they are familiar with the mathematical formulation of the pinhole camera (section 14.1).

To motivate these methods, consider a single camera moving around a static object. The goal is to build a 3D model from the images taken by the camera. To do this, we will also need to simultaneously establish the properties of the camera and its position in each frame. This problem is widely known as *structure from motion*, although this is something of a misnomer as both 'structure' and 'motion' are recovered simultaneously.

The structure from motion problem can be stated formally as follows. We are given J images of a rigid object that is characterized by I distinct 3D points $\{\mathbf{w}_i\}_{i=1}^{I}$. The images are taken with the same camera at a series of unknown positions. Given the projections $\{\mathbf{x}_{ij}\}_{i=1,j=1}^{I,J}$ of the I points in the J images, establish the 3D positions $\{\mathbf{w}_i\}_{i=1}^{I}$ of the points in the world, the fixed intrinsic parameters $\boldsymbol{\Lambda}$, and the extrinsic parameters $\{\boldsymbol{\Omega}_j, \boldsymbol{\tau}_j\}_{j=1}^{J}$ for each image:

$$\begin{aligned}\{\hat{\mathbf{w}}_i\}_{i=1}^{I}, \{\hat{\boldsymbol{\Omega}}_j, \hat{\boldsymbol{\tau}}_j\}_{j=1}^{J}, \hat{\boldsymbol{\Lambda}} \qquad (16.1)\\ = \underset{\mathbf{w},\boldsymbol{\Omega},\boldsymbol{\tau},\boldsymbol{\Lambda}}{\operatorname{argmax}} \left[\sum_{i=1}^{I}\sum_{j=1}^{J} \log[Pr(\mathbf{x}_{ij}|\mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j)]\right]\\ = \underset{\mathbf{w},\boldsymbol{\Omega},\boldsymbol{\tau},\boldsymbol{\Lambda}}{\operatorname{argmax}} \left[\sum_{i=1}^{I}\sum_{j=1}^{J} \log\left[\text{Norm}_{\mathbf{x}_{ij}}\left[\mathbf{pinhole}[\mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j], \sigma^2\mathbf{I}\right]\right]\right].\end{aligned}$$

Since this objective function is based on the normal distribution, we can reformulate it as a least-squares problem of the form

$$\{\hat{\mathbf{w}}_i\}_{i=1}^{I}, \{\hat{\boldsymbol{\Omega}}_j, \hat{\boldsymbol{\tau}}_j\}_{j=1}^{J}, \hat{\boldsymbol{\Lambda}} = \tag{16.2}$$
$$\underset{\mathbf{w},\boldsymbol{\Omega},\boldsymbol{\tau},\boldsymbol{\Lambda}}{\operatorname{argmin}} \left[\sum_{i=1}^{I}\sum_{j=1}^{J} (\mathbf{x}_{ij} - \mathbf{pinhole}[\mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j])^T (\mathbf{x}_{ij} - \mathbf{pinhole}[\mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j]) \right],$$

in which the goal is to minimize the total squared distance between the observed image points and those predicted by the model. This is known as the *squared reprojection error*. Unfortunately, there is no simple closed form solution for this problem, and we must ultimately rely on nonlinear optimization. However, to ensure that the optimization converges, we need good initial estimates of the unknown parameters.

To simplify our discussion, we will concentrate first on the case where we have $J=2$ views, and the intrinsic matrix $\boldsymbol{\Lambda}$ is known. We already saw how to estimate 3D points given known camera positions in section 14.6, so the unresolved problem is to get good initial estimates of the extrinsic parameters. Surprisingly, it is possible to do this based on just examining the positions of corresponding points without having to reconstruct their 3D world positions. To understand why, we must first learn more about the geometry of two views.

16.1 Two-view geometry

In this section, we show that there is a geometric relationship between corresponding points in two images of the same scene. This relationship depends only on the intrinsic parameters of the two cameras and their relative translation and rotation.

16.1.1 The epipolar constraint

Consider a single camera viewing a 3D point $\mathbf{w}$ in the world. We know that $\mathbf{w}$ must lie somewhere on the ray that passes through the optical center and position $\mathbf{x}_1$ on the image plane (figure 16.1). However, from one camera alone, we cannot know how far along this ray the point is.

Now consider a second camera viewing the same 3D world point. We know from the first camera that this point must lie along a particular ray in 3D space. It follows that the projected position $\mathbf{x}_2$ of this point in the second image must lie somewhere along the projection of this ray in the second image. The ray in 3D projects to a 2D line which is known as an *epipolar line*.

This geometric relationship tells us something important: for any point in the first image, the corresponding point in the second image is constrained to lie on a line. This is known as the *epipolar constraint*. The particular line that it is constrained to lie on depends on the intrinsic parameters of the cameras and the relative translation and rotation of the two cameras (determined by the extrinsic parameters).

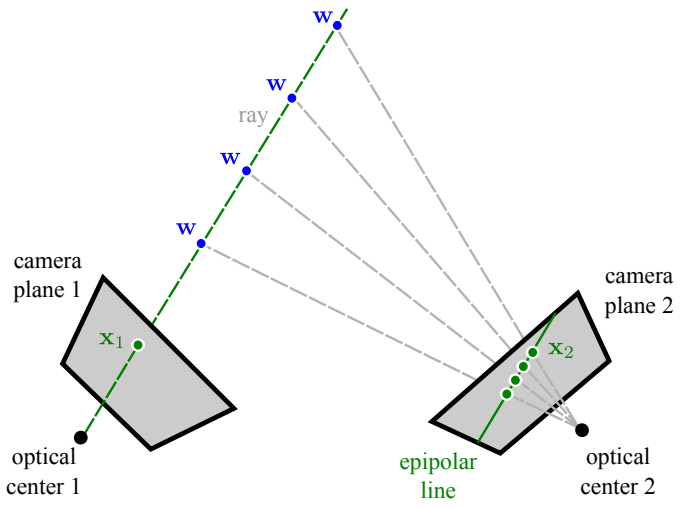

Figure 16.1 Epipolar line. Consider point $\mathbf{x}_1$ in the first image. The 3D point $\mathbf{w}$ that projected to $\mathbf{x}_1$ must lie somewhere along the ray that passes from the optical center of camera 1 through the position $\mathbf{x}_1$ in the image plane (dashed green line). However, we don't know where along that ray it lies (4 possibilities shown). It follows that $\mathbf{x}_2$, the projected position in camera 2 must lie somewhere on the projection of this ray. The projection of this ray is a line in image 2 and is referred to as an epipolar line.

The epipolar constraint has two important practical implications.

1. Given the intrinsic and extrinsic parameters, we can find point correspondences relatively easily: for a given point in the first image, we only need to perform a 1D search along the epipolar line in the second image for the corresponding position.
2. The constraint on corresponding points is a function of the intrinsic and extrinsic parameters; given the intrinsic parameters, we can use the observed pattern of point correspondences to determine the extrinsic parameters and hence establish the geometric relationship between the two cameras.

16.1.2 Epipoles

Now consider a number of points in the first image. Each is associated with a ray in 3D space. Each ray projects to form an epipolar line in the second image. Since all the rays converge at the optical center of the first camera, the epipolar lines must converge at a single point in the second image plane; this is the image in the second camera of the optical center of the first camera and is known as the *epipole* (figure 16.2). Similarly, points in image 2 induce epipolar lines in image 1, and

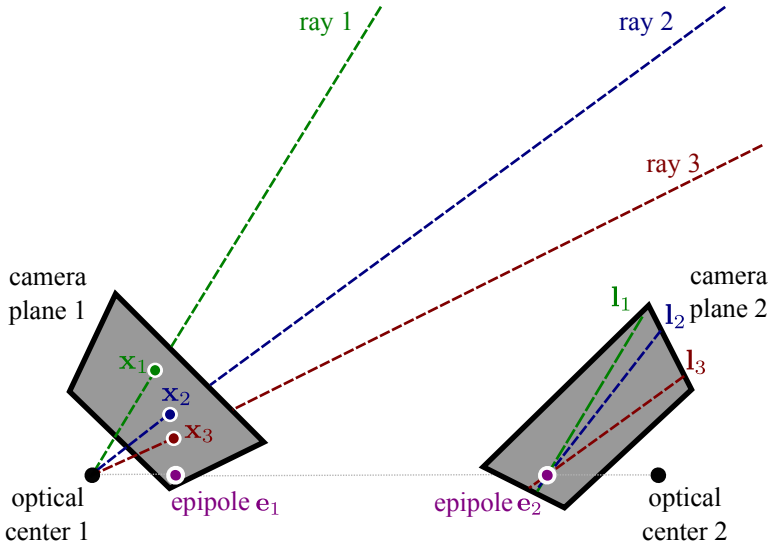

Figure 16.2 Epipoles. Consider several observed points $\{\mathbf{x}_i\}_{i=1}^{I}$ in image 1. For each point, the corresponding 3D world position $\mathbf{w}_i$ lies on a different ray. Each ray projects to an epipolar line $\mathbf{l}_i$ in image 2. Since the rays converge in 3D space at the optical center of camera 1, the epipolar lines must also converge. The point where they converge is known as the epipole $\mathbf{e}_2$. It is the projection of the optical center of camera 1 into camera 2. Similarly, the epipole $\mathbf{e}_1$ is the projection of the optical center of camera 2 into camera 1.

these epipolar lines converge at the epipole in image 1. This is the image in camera 1 of the optical center of camera 2.

Problem 16.1

The epipoles are not necessarily within the observed images: the epipolar lines may converge to a point outside the visible area. Two common cases are illustrated in figure 16.3. When the cameras are oriented in the same direction (i.e., no relative rotation) and the translation is perpendicular to their optical axes (figure 16.3a) then the epipolar lines are parallel and the epipoles (where they converge) are hence at infinity. When the cameras are oriented in the same direction and the translationt is parallel to their optical axes (figure 16.3b), then the epipoles are in the middle of the images and the epipolar lines form a radial pattern. These examples illustrate that the pattern of epipolar lines provides information about the relative position and orientation of the cameras.

16.2 The essential matrix

Now we will capture these geometric intuitions in the form of a mathematical model. For simplicity, we will assume that the world coordinate system is centered on the first camera so that the extrinsic parameters (rotation and translation) of the first camera are $\{\mathbf{I}, \mathbf{0}\}$. The second camera may be in any general position $\{\boldsymbol{\Omega}, \boldsymbol{\tau}\}$. We will further assume that the cameras are normalized so that $\boldsymbol{\Lambda}_1 = \boldsymbol{\Lambda}_2 = \mathbf{I}$. In homogeneous coordinates, a 3D point $\mathbf{w}$ is projected into the two cameras as

$$\begin{aligned} \lambda_1\tilde{\mathbf{x}}_1 &= [\mathbf{I}, \mathbf{0}]\tilde{\mathbf{w}} \\ \lambda_2\tilde{\mathbf{x}}_2 &= [\boldsymbol{\Omega}, \boldsymbol{\tau}]\tilde{\mathbf{w}}. \end{aligned} \tag{16.3}$$

where $\tilde{\mathbf{x}}_1$ is the observed position in the first camera, $\tilde{\mathbf{x}}_2$ is the observed position in the second camera, and both are expressed in homogeneous coordinates.

Expanding the first of these relations we get

$$\lambda_1 \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ w \end{bmatrix}, \tag{16.4}$$

This simplifies to

$$\lambda_1\tilde{\mathbf{x}}_1 = \mathbf{w}. \tag{16.5}$$

By a similar process, the projection in the second camera can be written as

$$\lambda_2\tilde{\mathbf{x}}_2 = \boldsymbol{\Omega}\mathbf{w} + \boldsymbol{\tau}. \tag{16.6}$$

Finally, substituting equation 16.5 into equation 16.6 yields

$$\lambda_2\tilde{\mathbf{x}}_2 = \lambda_1\boldsymbol{\Omega}\tilde{\mathbf{x}}_1 + \boldsymbol{\tau}. \tag{16.7}$$

This relationship represents a constraint between the possible positions of corresponding points $\mathbf{x}_1$ and $\mathbf{x}_2$ in the two images. The constraint is parameterized by the rotation and translation $\{\boldsymbol{\Omega}, \boldsymbol{\tau}\}$ of the second camera relative to the first.

We will now manipulate the relationship in equation 16.7 into a form that can be more easily related to the epipolar lines and the epipoles. We first take the cross product of both sides with the translation vector $\boldsymbol{\tau}$. This removes the last term as the cross product of any vector with itself is zero. Now we have

$$\lambda_2\boldsymbol{\tau} \times \tilde{\mathbf{x}}_2 = \lambda_1\boldsymbol{\tau} \times \boldsymbol{\Omega}\tilde{\mathbf{x}}_1. \tag{16.8}$$

Then we take the inner product of both sides with $\tilde{\mathbf{x}}_2$. The left hand side disappears since $\boldsymbol{\tau} \times \tilde{\mathbf{x}}_2$ must be perpendicular to $\tilde{\mathbf{x}}_2$, and so we have

$$\tilde{\mathbf{x}}_2^T\boldsymbol{\tau} \times \boldsymbol{\Omega}\tilde{\mathbf{x}}_1 = 0, \tag{16.9}$$

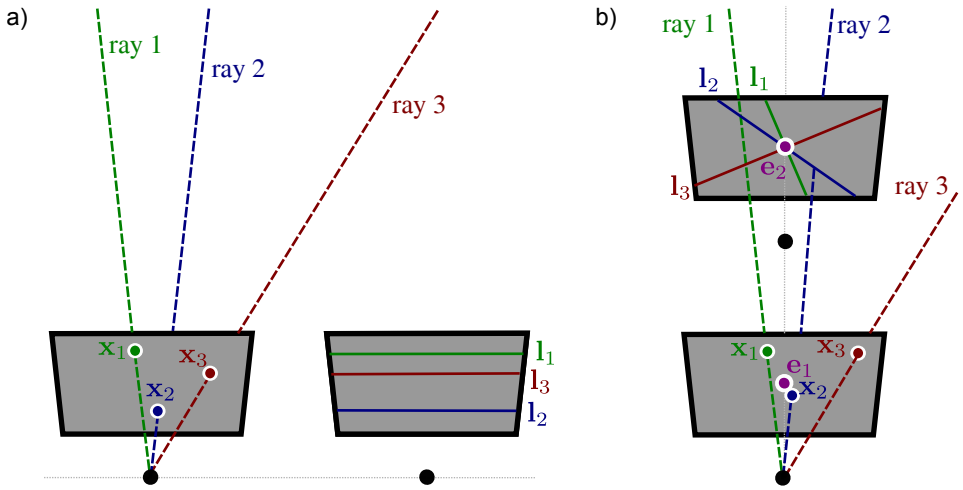

Figure 16.3 Epipolar lines and epipoles. a) When the camera movement is a pure translation perpendicular to the optical axis (parallel to the image plane) the epipolar lines are parallel and the epipole is at infinity. b) When the camera movement is a pure translation along the optical axis the epipoles are in the center of the image and the epipolar lines form a radial pattern.

where we have also eliminated the scaling factors λ_1 and λ_2 by dividing by them. Finally, we note that the cross product operation $\boldsymbol{\tau}\times$ can be expressed as multiplication by the rank 2 skew-symmetric 3×3 matrix $\boldsymbol{\tau}_\times$:

Problem 16.2

$$\boldsymbol{\tau}_\times = \begin{bmatrix} 0 & -\tau_z & \tau_y \\ \tau_z & 0 & -\tau_x \\ -\tau_y & \tau_x & 0 \end{bmatrix}. \tag{16.10}$$

Hence equation 16.9 has the form

$$\tilde{\mathbf{x}}_2^T \mathbf{E} \tilde{\mathbf{x}}_1 = 0, \tag{16.11}$$

Problem 16.3

where $\mathbf{E} = \boldsymbol{\tau}_\times \boldsymbol{\Omega}$ is known as the *essential matrix*. Equation 16.11 is an elegant formulation of the mathematical constraint between the positions of corresponding points $\mathbf{x}_1$ and $\mathbf{x}_2$ in two normalized cameras.

16.2.1 Properties of the essential matrix

Problem 16.4

The 3×3 essential matrix captures the geometric relationship between the two cameras and has rank 2 so that $\det[\mathbf{E}] = 0$. The first two singular values of the essential matrix are always identical and the third is zero. It depends only on the rotation and translation between the cameras, each of which has 3 parameters, and so one might think it would have 6 degrees of freedom. However, it operates on

homogeneous variables $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$ and is hence ambiguous up to scale: multiplying all of the entries of the essential matrix by any constant does not change its properties. For this reason, it is usually considered as having 5 degrees of freedom.

Since there are fewer degrees of freedom than there are unknowns, the nine entries of the matrix must obey a set of algebraic constraints. These can be expressed compactly as

$$2\mathbf{E}\mathbf{E}^T\mathbf{E} - \text{trace}[\mathbf{E}\mathbf{E}^T]\mathbf{E} = \mathbf{0}. \tag{16.12}$$

These constraints are sometimes exploited in the computation of the essential matrix, although in this volume we use a simpler method (section 16.4).

The epipolar lines are easily retrieved from the essential matrix. The condition for a point being on a line is $ax + by + c = 0$ or

Problem 16.5

$$\begin{bmatrix} a & b & c \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = 0. \tag{16.13}$$

In homogeneous co-ordinates, this can be written as $\mathbf{l}\tilde{\mathbf{x}} = 0$ where $\mathbf{l} = [a, b, c]$ is a 1×3 vector representing the line.

Now consider the essential matrix relation

$$\tilde{\mathbf{x}}_2^T\mathbf{E}\tilde{\mathbf{x}}_1 = 0. \tag{16.14}$$

Since $\tilde{\mathbf{x}}_2^T\mathbf{E}$ is a 1×3 vector, this relationship has the form $\mathbf{l}_1\tilde{\mathbf{x}}_1 = 0$ The line $\mathbf{l}_1 = \tilde{\mathbf{x}}_2^T\mathbf{E}$ is the epipolar line in image 1 due to the point $\mathbf{x}_2$ in image 2. By a similar argument, we can find the epipolar line $\mathbf{l}_2$ in the second camera due to the point $\mathbf{x}_1$ in the first camera. The final relations are

$$\begin{aligned} \mathbf{l}_1 &= \tilde{\mathbf{x}}_2^T\mathbf{E} \\ \mathbf{l}_2 &= \tilde{\mathbf{x}}_1^T\mathbf{E}^T. \end{aligned} \tag{16.15}$$

The epipoles can also be extracted from the essential matrix. Every epipolar line in image 1 passes through the epipole $\tilde{\mathbf{e}}_1$, so at the epipole $\tilde{\mathbf{e}}_1$ we have $\tilde{\mathbf{x}}_2^T\mathbf{E}\tilde{\mathbf{e}}_1 = 0$ for all $\tilde{\mathbf{x}}_2$. This implies that $\tilde{\mathbf{e}}_1$ must lie in the right null-space of $\mathbf{E}$ (see appendix C.2.7). By a similar argument, the epipole $\tilde{\mathbf{e}}_2$ in the second image must lie in the left null space of $\mathbf{E}$. Hence, we have the relations

$$\begin{aligned} \tilde{\mathbf{e}}_1 &= \textbf{null}[\mathbf{E}] \\ \tilde{\mathbf{e}}_2 &= \textbf{null}[\mathbf{E}^T]. \end{aligned} \tag{16.16}$$

In practice, the epipoles can be retrieved by computing the singular value decomposition $\mathbf{E} = \mathbf{U}\mathbf{L}\mathbf{V}^T$ of the essential matrix, and setting $\tilde{\mathbf{e}}_1$ to the last column of $\mathbf{V}$ and $\tilde{\mathbf{e}}_2$ to the last row of $\mathbf{U}$.

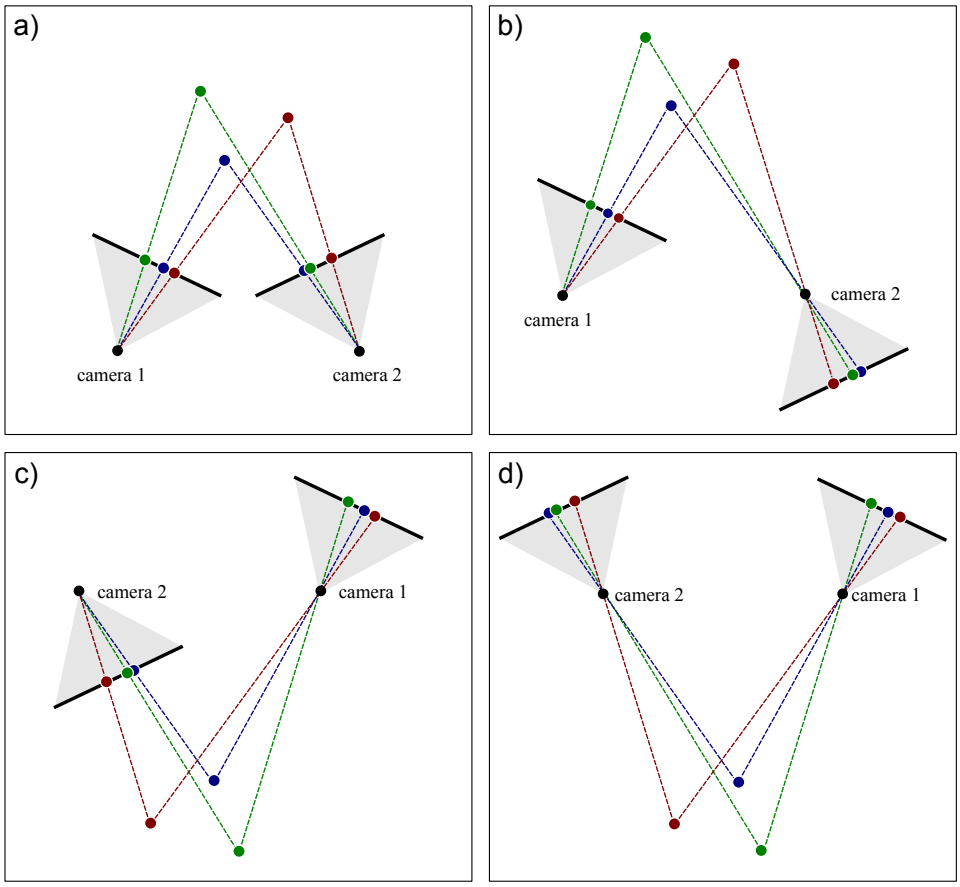

Figure 16.4 Four-fold ambiguity of reconstruction from two pinhole cameras. The mathematical model for the pinhole camera does not distinguish between points that are in front of and points that are behind the camera. This leads to a four-fold ambiguity when we extract the rotation $\boldsymbol{\Omega}$ and translation $\boldsymbol{\tau}$ relating the cameras from the essential matrix. a) Correct solution. Points are in front of both cameras. b) Incorrect solution. The images are identical, but with this interpretation, the points are behind camera 2. c) Incorrect solution with points behind camera 1. d) Incorrect solution with points behind both cameras.

16.2.2 Decomposition of essential matrix

We saw previously that the essential matrix is defined as

$$\mathbf{E} = \boldsymbol{\tau}_{\times}\boldsymbol{\Omega}, \tag{16.17}$$

Algorithm 16.1

where $\boldsymbol{\Omega}$ and $\boldsymbol{\tau}$ are the rotation matrix and translation vector that map points in the coordinate system of camera 2 to the coordinate system of camera 1, and $\boldsymbol{\tau}_{\times}$

is a 3×3 matrix derived from the translation vector.

We will defer the question of how to compute the essential matrix from a set of corresponding points until section 16.3. For now, we will concentrate on how to decompose a given essential matrix $\mathbf{E}$ to recover this rotation $\boldsymbol{\Omega}$ and translation $\boldsymbol{\tau}$. This is known as the *relative orientation* problem.

In due course, we shall see that we can compute the rotation exactly, whereas it is only possible to compute the translation up to an unknown scaling factor. This remaining uncertainty reflects the geometric ambiguity of the system; from the images alone, we cannot tell if these cameras are far apart and looking at a large distant object or close together and looking at a small nearby object.

To decompose $\mathbf{E}$, we define the matrix

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{16.18}$$

and then take the singular value decomposition $\mathbf{E} = \mathbf{U}\mathbf{L}\mathbf{V}^T$. We now choose

$$\begin{aligned} \boldsymbol{\tau}_\times &= \mathbf{U}\mathbf{L}\mathbf{W}\mathbf{U}^T \\ \boldsymbol{\Omega} &= \mathbf{U}\mathbf{W}^{-1}\mathbf{V}^T. \end{aligned} \tag{16.19}$$

It is convention to set the magnitude of the translation vector $\boldsymbol{\tau}$ that is recovered from the matrix $\boldsymbol{\tau}_\times$ to unity. The above decomposition is not obvious, but it is easily checked that multplying the derived expressions for $\boldsymbol{\tau}_\times$ and $\boldsymbol{\Omega}$ yields $\mathbf{E} = \mathbf{U}\mathbf{L}\mathbf{V}^T$. This method assumes that we started with a valid essential matrix where the first two singular values are identical and the third is zero. If this is not the case (due to noise) we can substitute $\mathbf{L}' = \mathbf{diag}[1, 1, 0]$ for $\mathbf{L}$ in the solution for $\tau_\times$. For a detailed proof of this decomposition, consult Hartley & Zisserman (2004).

Problem 16.6

This solution is only one of four possible combinations of $\boldsymbol{\Omega}$ and $\boldsymbol{\tau}$ that are compatible with $\mathbf{E}$ (figure 16.4). This four-fold ambiguity is due to the fact that the pinhole model cannot distinguish between objects that are behind the camera (and are not imaged in real cameras) and those that are in front of the camera.

Part of the uncertainty is captured mathematically by our lack of knowledge of the sign of the essential matrix (recall it is ambiguous up to scale) and hence the sign of the recovered translation. Hence, we can generate a second solution by multiplying the translation vector by -1. The other component of the uncertainty results from an ambiguity in the decomposition of the essential matrix; we can equivalently replace $\mathbf{W}$ for $\mathbf{W}^{-1}$ in the decomposition procedure, and this leads to two more solutions.

Fortunately, we can resolve this ambiguity using a corresponding pair of points from the two images. For each putative solution, we reconstruct the 3D position associated with this pair (section 14.6). For one of the four possible combinations of $\boldsymbol{\Omega}, \boldsymbol{\tau}$, the point will be in front of both cameras, and this is the correct solution. In each of the other three cases, the point will be reconstructed behind one or both of the cameras (figure 16.4). For a robust estimate, we would repeat this procedure with a number of corresponding points and base our decision on the total number of votes for each of the four interpretations.

16.3 The fundamental matrix

The derivation of the essential matrix in section 16.2 used normalized cameras (where $\boldsymbol{\Lambda}_1 = \boldsymbol{\Lambda}_2 = \mathbf{I}$). The fundamental matrix plays the role of the essential matrix for cameras with arbitrary intrinsic matrices $\boldsymbol{\Lambda}_1$ and $\boldsymbol{\Lambda}_2$. The general projection equations for the two cameras are

$$\begin{aligned} \lambda_1\tilde{\mathbf{x}}_1 &= \boldsymbol{\Lambda}_1[\mathbf{I},\mathbf{0}]\tilde{\mathbf{w}} \\ \lambda_2\tilde{\mathbf{x}}_2 &= \boldsymbol{\Lambda}_2[\boldsymbol{\Omega},\boldsymbol{\tau}]\tilde{\mathbf{w}}, \end{aligned} \tag{16.20}$$

Problem 16.7

and we can use similar manipulations to those presented in section 16.2 to derive the constraint

$$\tilde{\mathbf{x}}_2^T\boldsymbol{\Lambda}_2^{-T}\mathbf{E}\boldsymbol{\Lambda}_1^{-1}\tilde{\mathbf{x}}_1 = 0, \tag{16.21}$$

or

$$\tilde{\mathbf{x}}_2^T\mathbf{F}\tilde{\mathbf{x}}_1 = 0, \tag{16.22}$$

where the 3×3 matrix $\mathbf{F} = \boldsymbol{\Lambda}_2^{-T}\mathbf{E}\boldsymbol{\Lambda}_1^{-1} = \boldsymbol{\Lambda}_2^{-T}\boldsymbol{\tau}_\times\boldsymbol{\Omega}\boldsymbol{\Lambda}_1^{-1}$ is termed the *fundamental matrix*. Like the essential matrix, it also has rank two, but unlike the essential matrix it has seven degrees of freedom.

If we know the fundamental matrix $\mathbf{F}$ and the intrinsic matrices $\boldsymbol{\Lambda}_1$ and $\boldsymbol{\Lambda}_2$, it is possible to recover the essential matrix $\mathbf{E}$ using the relation

$$\mathbf{E} = \boldsymbol{\Lambda}_2^T\mathbf{F}\boldsymbol{\Lambda}_1, \tag{16.23}$$

and this can further be decomposed to find the rotation and translation between the cameras using the method of section 16.2.2. It follows that for calibrated cameras, if we can estimate the fundamental matrix, then we can find the rotation and translation between the cameras. Hence, we now turn our attention on how to compute the fundamental matrix.

16.3.1 Estimation of the fundamental matrix

The fundamental matrix relation (equation 16.22) is a constraint on the possible positions of corresponding points in the first and second images. This constraint is parameterized by the nine entries of $\mathbf{F}$. It follows that if we analyze a set of corresponding points, we can observe *how* they are constrained, and from this we can deduce the entries of the fundamental matrix $\mathbf{F}$.

A suitable cost function for the fundamental matrix can be found by considering the epipolar lines. Consider a pair of matching points $\{\mathbf{x}_{i1}, \mathbf{x}_{i2}\}$ in images 1 and 2, respectively. Each point induces an epipolar line in the other image: the point $\mathbf{x}_{i1}$ induces line $\mathbf{l}_{i2}$ in image 2 and the point $\mathbf{x}_{i2}$ induces the line $\mathbf{l}_{i1}$ in image 1. When the fundamental matrix is correct, each point should lie exactly on the epipolar line induced by the corresponding point in the other image (figure 16.5). We hence

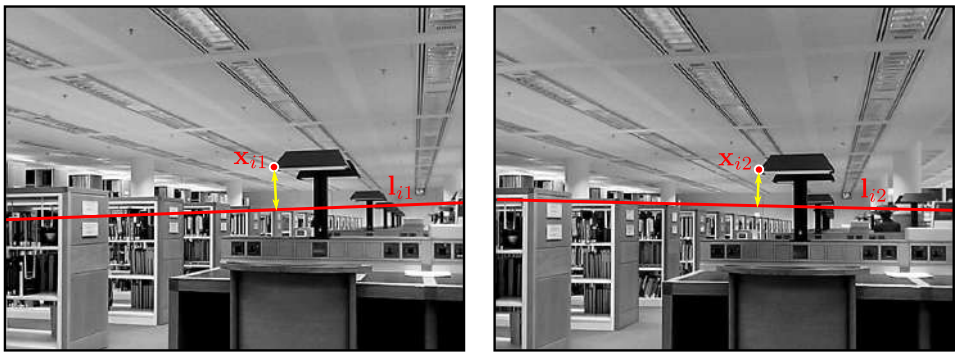

Figure 16.5 Cost function for estimating fundamental matrix. The point $\mathbf{x}_{i1}$ in image 1 induces the epipolar line $\mathbf{l}_{i2}$ in image 2. When the fundamental matrix is correct, the matching point $\mathbf{x}_{i2}$ will be on this line. Similarly the point $\mathbf{x}_{i2}$ in image 2 induces the epipolar line $\mathbf{l}_{i1}$ in image 1. When the fundamental matrix is correct, the point $\mathbf{x}_{i1}$ will be on this line. The cost function is the sum of the squares of the distances between these epipolar lines and the points (yellow arrows). This is termed *symmetric epipolar distance*.

minimize the squared distance between every point and the epipolar line predicted by its match in the other image so that

$$\hat{\mathbf{F}} = \underset{\mathbf{F}}{\text{argmin}} \left[\sum_{i=1}^{I} \left(\left(\text{dist}[\mathbf{x}_{i1}, \mathbf{l}_{i1}]\right)^2 + \left(\text{dist}[\mathbf{x}_{i2}, \mathbf{l}_{i2}]\right)^2 \right) \right], \tag{16.24}$$

where the distance between a 2D point $\mathbf{x} = [x, y]^T$ and a line $\mathbf{l} = [a, b, c]$ is

$$\text{dist}[\mathbf{x}, \mathbf{l}] = \frac{ax + by + c}{\sqrt{a^2 + b^2}}. \tag{16.25}$$

Here too, it is not possible to find the minimum of equation 16.24 in closed form, and we must rely on nonlinear optimization methods. It is possible to get a good starting point for this optimization using the *eight-point algorithm*.

16.3.2 The eight-point algorithm

The eight-point algorithm converts the corresponding 2D points to homogeneous coordinates and then solves for the fundamental matrix in closed form. It does not directly optimize the cost function in equation 16.24, but instead minimizes an algebraic error. However, the solution to this problem is usually very close to the values that optimize the desired cost function.

Algorithm 16.2
Algorithm 16.3

In homogeneous coordinates, the relationship between the i^{th} point $\mathbf{x}_{i1} = [x_{i1}, y_{i1}]^T$ in image 1 and the i^{th} point $\mathbf{x}_{i2} = [x_{i2}, y_{i2}]^T$ in image 2 is

$$\begin{bmatrix} x_{i2} & y_{i2} & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} x_{i1} \\ y_{i1} \\ 1 \end{bmatrix} = 0, \tag{16.26}$$

where f_{pq} represents one of the entries in the fundamental matrix. When we write this constraint out in full, we get

$$x_{i2}x_{i1}f_{11}+x_{i2}y_{i1}f_{12}+x_{i2}f_{13}+y_{i2}x_{i1}f_{21}+y_{i2}y_{i1}f_{22}+y_{i2}f_{23}+x_{i1}f_{31}+y_{i1}f_{32}+f_{33} = 0. \tag{16.27}$$

This can be expressed as an inner product

$$[x_{i2}x_{i1}, x_{i2}y_{i1}, x_{i2}, y_{i2}x_{i1}, y_{i2}y_{i1}, y_{i2}, x_{i1}, y_{i1}, 1]\mathbf{f} = 0, \tag{16.28}$$

where $\mathbf{f} = [f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33}]^T$ is a vectorized version of the fundamental matrix, $\mathbf{F}$.

This provides one linear constraint on the elements of $\mathbf{F}$. Consequently, given I matching points, we can stack these constraints to form the system

$$\mathbf{Af} = \begin{bmatrix} x_{12}x_{11} & x_{12}y_{11} & x_{12} & y_{12}x_{11} & y_{12}y_{11} & y_{12} & x_{11} & y_{11} & 1 \\ x_{22}x_{21} & x_{22}y_{21} & x_{22} & y_{22}x_{21} & y_{22}y_{21} & y_{22} & x_{21} & y_{21} & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{I2}x_{I1} & x_{I2}y_{I1} & x_{I2} & y_{I2}x_{I1} & y_{I2}y_{I1} & y_{I2} & x_{I1} & y_{I1} & 1 \end{bmatrix} \mathbf{f} = \mathbf{0}. \tag{16.29}$$

Since the elements of $\mathbf{f}$ are ambiguous up to scale, we solve this system with the constraint that $|\mathbf{f}| = 1$. This also avoids the trivial solution $\mathbf{f} = \mathbf{0}$. This is a minimum direction problem (see appendix C.7.2). The solution can be found by taking the singular value decomposition, $\mathbf{A} = \mathbf{ULV}^T$ and setting $\mathbf{f}$ to be the last column of $\mathbf{V}$. The matrix $\mathbf{F}$ is then formed by reshaping $\mathbf{f}$ to form a 3×3 matrix.

Problem 16.8

There are 8 degrees of freedom in the fundamental matrix (it is ambiguous with respect to scale) and so we require a minimum of $I = 8$ pairs of points. For this reason, this algorithm is called the *eight-point algorithm*.

In practice, there are several further concerns in implementing this algorithm:

- Since the data are noisy, the singularity constraint of the resulting fundamental matrix will not be obeyed in general (i.e., the estimated matrix will be full rank, not rank two). We re-introduce this constraint by taking the singular decomposition of $\mathbf{F}$, setting the last singular value to zero, and multiplying the terms back out. This provides the closest singular matrix under a Frobenius norm.
- Equation 16.29 is badly scaled since some terms are on the order of pixels squared (~ 10000) and some are of the order ~ 1. To improve the quality of the solution, it is wise to pre-normalize the data (see Hartley 1997). We transform the points in image 1 as $\tilde{\mathbf{x}}'_{i1} = \mathbf{T}_1\tilde{\mathbf{x}}_{i1}$ and the points in image 2 as $\tilde{\mathbf{x}}'_{i2} = \mathbf{T}_2\tilde{\mathbf{x}}_{i2}$. The transformations $\mathbf{T}_1$ and $\mathbf{T}_2$ are chosen to map the mean of the points in their respective image to zero, and to ensure that the variance in the x- and y-dimensions is one. We then compute the matrix $\mathbf{F}'$ from the

transformed data using the eight-point algorithm, and recover the original fundamental matrix as $\mathbf{F} = \mathbf{T}_2^T \mathbf{F}' \mathbf{T}_1$.

- The algorithm will only work if the three-dimensional positions $\mathbf{w}_i$ corresponding to the eight pairs of points $\mathbf{x}_{i1}, \mathbf{x}_{i2}$ are in general position. For example, if they all fall on a plane, then the equations become degenerate and we cannot get a unique solution; here, the relation between the points in the two images is given by a homography (see chapter 15). Similarly, in the case where there is no translation (i.e., $\boldsymbol{\tau} = \mathbf{0}$), the relation between the two images is a homography and there is no unique solution for the fundamental matrix.
- In the subsequent nonlinear optimization, we must also ensure that the rank of $\mathbf{F}$ is two. In order to do this, it is usual to re-parameterize the fundamental matrix to ensure that this will be the case.

16.4 Two-view reconstruction pipeline

We now put all of these ideas together and present a rudimentary pipeline for the reconstruction of a static 3D scene based on two images taken from unknown positions, but with cameras where we know the intrinsic parameters. We apply the following steps (figures 16.6 and 16.7).

1. **Compute image features**. We find salient points in each image using an interest point detector such as the SIFT detector (section 13.2.3).
2. **Compute feature descriptors**. We characterize the region around each feature in each image with a low dimensional vector. One possibility would be to use the SIFT descriptor (section 13.3.2).
3. **Find initial matches**. We greedily match features between the two images. For example, we might base this on the squared distance between their region descriptors and stop this procedure when the squared distance exceeds a predefined threshold to minimize false matches. We might also reject points where the ratio between the quality of the best and second best match in the other image is too close to one (suggesting that alternative matches are plausible).
4. **Compute fundamental matrix.** We compute the fundamental matrix using the eight-point algorithm. Since some matches are likely to be incorrect, we use a robust estimation procedure such as RANSAC (section 15.6).
5. **Refine matches.** We again greedily match features, but this time we exploit our knowledge of the epipolar geometry: if a putative match is not close to the induced epipolar line, it is rejected. We recompute the fundamental matrix based on all of the remaining point matches.
6. **Estimate essential matrix.** We estimate the essential matrix from the fundamental matrix using equation 16.23.

7. **Decompose essential matrix.** We extract estimates of the rotation and translation between the cameras (i.e., the extrinsic parameters) by decomposing the essential matrix (section 16.2.2). This provides four possible solutions.

8. **Estimate 3D points.** For each solution, we reconstruct the 3D position of the points using the linear solution from section 14.6. We retain the extrinsic parameters where most of the reconstructed points are in front of both cameras.

After this procedure, we have a set of I points $\{\mathbf{x}_{i1}\}_{i=1}^{I}$ in the first image, a set of I corresponding points $\{\mathbf{x}_{i2}\}_{i=1}^{I}$ in the second image, and a good initial estimate of the 3D world positions $\{\mathbf{w}_i\}_{i=1}^{I}$ that were responsible for them. We also have initial estimates of the extrinsic parameters $\{\boldsymbol{\Omega}, \boldsymbol{\tau}\}$. We now optimize the true cost function

$$\begin{aligned}\hat{\mathbf{w}}_{1\ldots I}, \hat{\boldsymbol{\Omega}}, \hat{\boldsymbol{\tau}} &= \underset{\mathbf{w},\boldsymbol{\Omega},\boldsymbol{\tau}}{\operatorname{argmax}} \left[\sum_{i=1}^{I}\sum_{j=1}^{2} \log[Pr(\mathbf{x}_{ij}|\mathbf{w}_i, \boldsymbol{\Lambda}_j, \boldsymbol{\Omega}, \boldsymbol{\tau})]\right] \qquad (16.30)\\ &= \underset{\mathbf{w},\boldsymbol{\Omega},\boldsymbol{\tau}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \log\left[\text{Norm}_{\mathbf{x}_{i1}}[\mathbf{pinhole}[\mathbf{w}_i, \boldsymbol{\Lambda}_1, \mathbf{I}, \mathbf{0}], \sigma^2\mathbf{I}]\right]\right.\\ &\qquad\qquad \left. + \sum_{i=1}^{I} \log\left[\text{Norm}_{\mathbf{x}_{i2}}[\mathbf{pinhole}[\mathbf{w}_i, \boldsymbol{\Lambda}_2, \boldsymbol{\Omega}, \boldsymbol{\tau}], \sigma^2\mathbf{I}]\right]\right],\end{aligned}$$

to refine these estimates. In doing so, we must ensure to enforce the constraints that $|\boldsymbol{\tau}| = 1$ and $\boldsymbol{\Omega}$ is a valid rotation matrix (see appendix B).

16.4.1 Minimal solutions

The pipeline described previously is rather naïve; in practice the fundamental and essential matrices can be estimated considerably more efficiently.

For example, the fundamental matrix contains seven degrees of freedom and so it is actually possible to solve for it using only seven pairs of points. Unsurprisingly, this is known as the *seven point algorithm*. It is is more complex as it relies on the seven linear constraints and the nonlinear constraint $\det[\mathbf{F}] = 0$. However, a robust solution can be computed much more efficiently using RANSAC if only seven points are required.

Even if we use this seven-point algorithm, the method is still inefficient. When we know the intrinsic parameters of the cameras, it is possible to compute the essential matrix (and hence the relative orientation of the cameras) using a minimal five-point solution. This is based on five linear constraints from observed corresponding points and the nonlinear constraints relating the nine parameters of the essential matrix (equation 16.12). This method has the advantages of being much quicker to estimate in the context of a RANSAC algorithm and being robust to non-general configurations of the scene points.

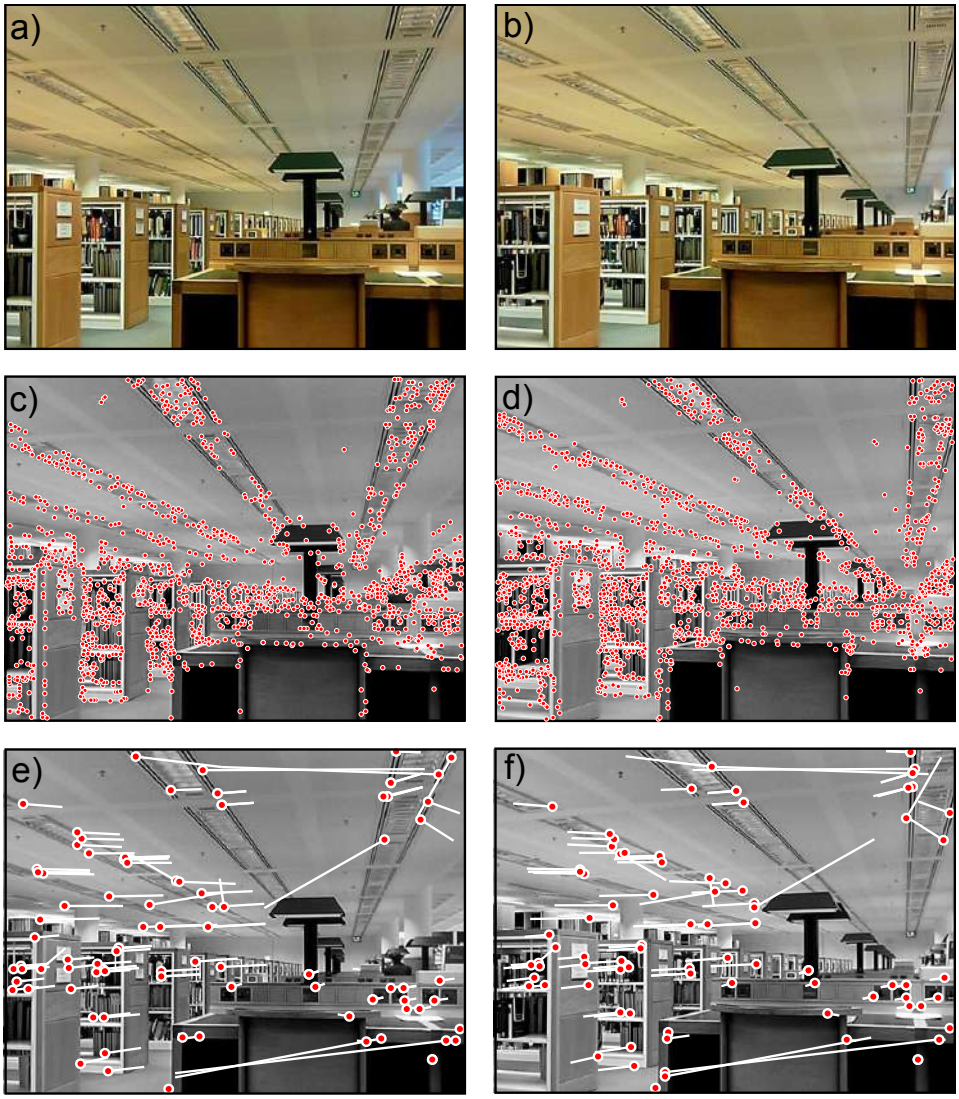

Figure 16.6 Two view reconstruction pipeline (steps 1-3). a-b) A pair of images of a static scene, captured from two slightly different positions. c-d) SIFT features are computed in each image. A region descriptor is calculated for each feature point that provides a low dimensional characterization of the region around the point. Points in the left and right images are matched using a greedy procedure; the pair of points for which the region descriptors are most similar in a least squares sense are chosen first. Then the pair of remaining points for which the descriptors are most similar are chosen, and so on. This continues until the minimum squared distance between the descriptors exceeds a threshold. e-f) Results of greedy matching procedure: the lines represent the offset to the matching point. Most matches are correct, but there are clearly also some outliers.

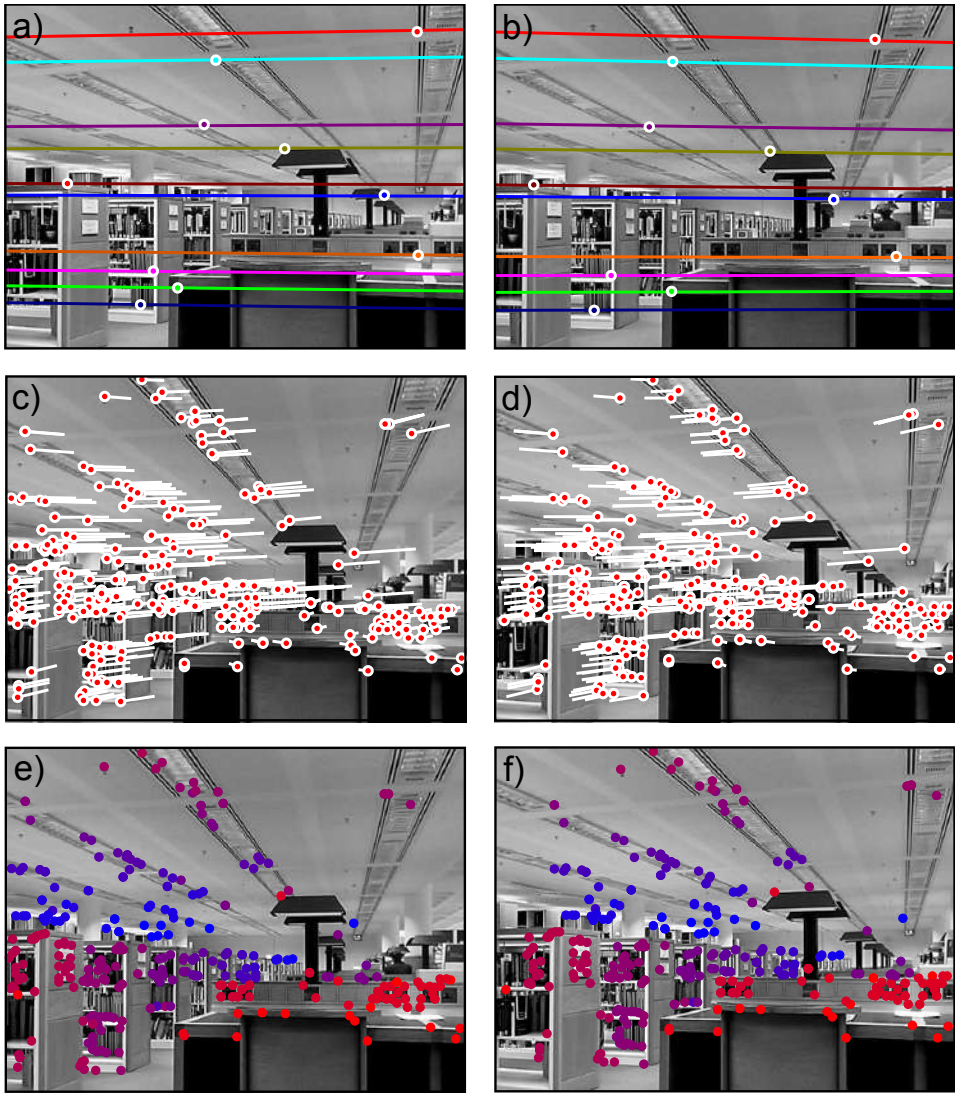

Figure 16.7 Two view reconstruction pipeline (steps 4-8). A fundamental matrix is fitted using a robust estimation procedure such as RANSAC. a-b) Eight matches with maximum agreement from the rest of the data. For each feature, the epipolar line is plotted in the other image. In each case, the matching point lies on or very near the epipolar line. The resulting fundamental matrix can be decomposed to get estimates for the relative rotation and translation between the cameras. c-d) Result of greedily matching original feature points taking into account the epipolar geometry. Matches where the symmetric epipolar distance exceeds a threshold are rejected. e-f) Computed w coordinate (depth) relative to first camera for each feature. Red features are closer, and blue features are further away. Almost all of the distances agree with our perceived understanding of the scene.

16.5 Rectification

The preceding procedure provides a set of sparse matches between the two images. These may suffice for some tasks such as navigation, but if we wish to build an accurate model of the scene, we need to estimate the depth at every point in the image. This is known as *dense reconstruction*.

Dense stereo reconstruction algorithms (see sections 11.8.2 and 12.8.3) generally assume that the corresponding point lies on the same horizontal scanline in the other image. The goal of *rectification* is to preprocess the image pair so that this is true. In other words, we will transform the images so that each epipolar line is horizontal and so the epipolar lines associated with a point fall on the same scanline to that point in the other image. We will describe two different approaches to this problem.

16.5.1 Planar rectification

We note that the epipolar lines are naturally horizontal and aligned when the camera motion is purely horizontal and both image planes are perpendicular to the w-axis (figure 16.3a). The key idea of planar rectification is to manipulate the two images to recreate these viewing conditions. We apply homographies $\mathbf{\Phi}_1$ and $\mathbf{\Phi}_2$ to the two images so that they cut their respective ray bundles in the desired way (figure 16.8).

Algorithm 16.3

In fact there is an entire family of homographies that accomplish this goal. One possible way to select a suitable pair is to first work with image 2. We apply a series of transformations $\mathbf{\Phi}_2 = \mathbf{T}_3\mathbf{T}_2\mathbf{T}_1$, which collectively move the epipole $\mathbf{e}_2$ to a position at infinity $[1, 0, 0]^T$.

We first center the co-ordinate system on the principal point,

$$\mathbf{T}_1 = \begin{bmatrix} 1 & 0 & -\delta_x \\ 0 & 1 & -\delta_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{16.31}$$

Then we rotate the image about this center until the epipole lies on the x-axis,

$$\mathbf{T}_2 = \begin{bmatrix} \cos[-\theta] & -\sin[-\theta] & 0 \\ \sin[-\theta] & \cos[-\theta] & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{16.32}$$

where $\theta = \text{atan2}[e_y, e_x]$ is the angle of the translated epipole $\mathbf{e} = [e_x, e_y]$. Finally, we translate the epipole to infinity, using the transformation

$$\mathbf{T}_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1/e_x & 0 & 1 \end{bmatrix}, \tag{16.33}$$

where e_x is the x coordinate of the epipole after the previous two transformations.

After these transformations, the epipole in the second image is at infinity in the horizontal direction. The epipolar lines in this image must converge at the epipole, and are consequently parallel and horizontal as desired.

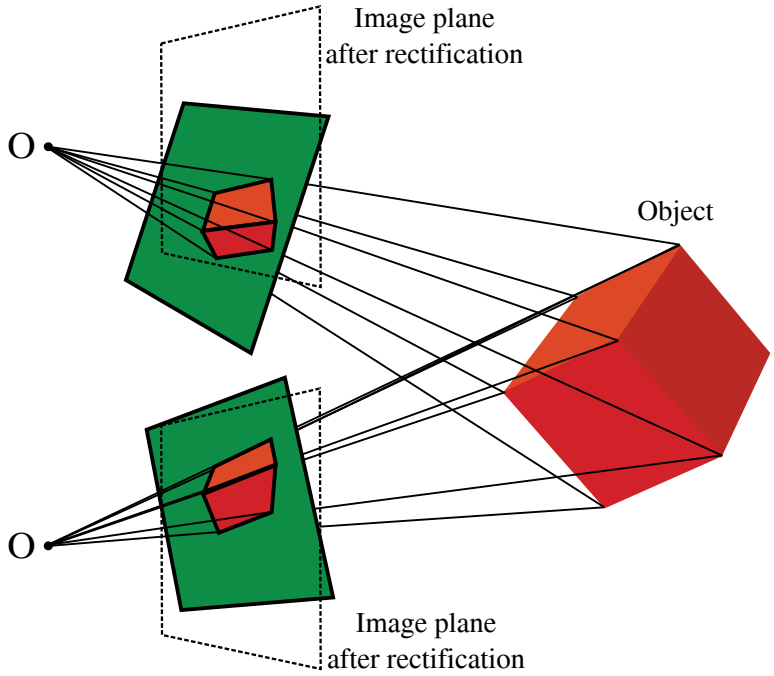

Figure 16.8 Planar rectification. Green quadrilaterals represent image planes of two cameras viewing a 3D object (cube). The goal of planar rectification is to transform each of these planes so that the final configuration replicates figure 16.3a. After this transformation (dotted lines), the image planes are coplanar, and the translation between the cameras is parallel to this plane. Now the epipolar lines are horizontal and aligned. Since the transformed planes are just different cuts through the respective ray bundles, each transformation can be accomplished using a homography.

Now we consider the first image. We cannot simply apply the same procedure as this will not guarantee that the epipolar lines in the first image will be aligned with those in the second. It transpires however, that there is a family of possible transformations that do make the epipolar lines of this image horizontal and aligned with those in the second image. This family can (not obviously) be parameterized as

$$\boldsymbol{\Phi}_1[\boldsymbol{\alpha}] = (\mathbf{I} + \mathbf{e}_2\boldsymbol{\alpha}^T)\boldsymbol{\Phi}_2\mathbf{M}, \tag{16.34}$$

where $\mathbf{e}_2 = [1,0,0]^T$ is the transformed epipole in the second image, and $\boldsymbol{\alpha} = [\alpha_1, \alpha_2, \alpha_3]^T$ is a 3D vector that selects the particular transformation from the family. The matrix $\mathbf{M}$ comes from the decomposition of the fundamental matrix into $\mathbf{F} = \mathbf{SM}$, where $\mathbf{S}$ is skew symmetric (see below). A proof of the relation in equation 16.34 can be found in Hartley & Zisserman (2004).

A sensible criterion is to choose $\boldsymbol{\alpha}$ so that it minimizes the disparity,

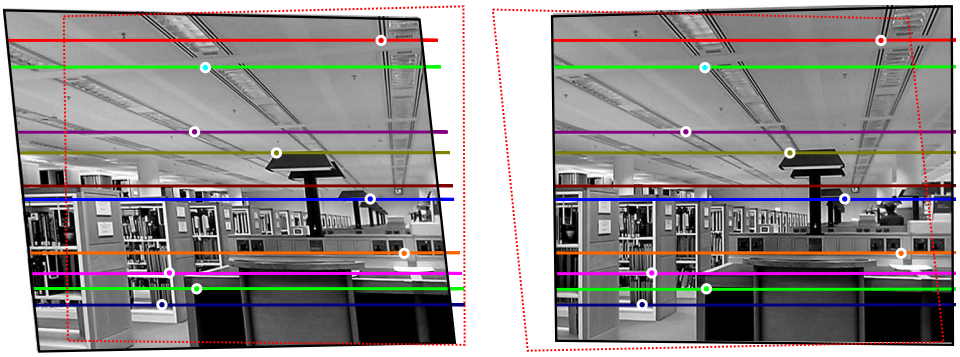

Figure 16.9 Planar rectification. The images from figures 16.6 and 16.7 have been rectified by applying homographies. After rectification, each point induces an epipolar line in the other image that is horizontal and on the same scanline (compare to figure 16.7a). This means that the match is guaranteed to be on the same scanline. In this figure, the red dotted line is the superimposed outline of the other image.

$$\hat{\boldsymbol{\alpha}} = \tag{16.35}$$
$$\underset{\boldsymbol{\alpha}}{\text{argmax}}\left[\sum_{i=1}^{I}(\mathbf{hom}[\mathbf{x}_{i1},\boldsymbol{\Phi}_1[\boldsymbol{\alpha}]]-\mathbf{hom}[\mathbf{x}_{i2},\boldsymbol{\Phi}_2])^T(\mathbf{hom}[\mathbf{x}_{i1},\boldsymbol{\Phi}_1[\boldsymbol{\alpha}]]-\mathbf{hom}[\mathbf{x}_{i2},\boldsymbol{\Phi}_2])\right].$$

This criterion simplifies to solving the least squares problem $|\mathbf{A}\boldsymbol{\alpha}-\mathbf{b}|^2$ where

$$\mathbf{A} = \begin{bmatrix} x'_{11} & y'_{11} & 1 \\ x'_{21} & y'_{21} & 1 \\ \vdots & \vdots & \vdots \\ x'_{I1} & y'_{I1} & 1 \end{bmatrix} \text{ and } \mathbf{b} = \begin{bmatrix} x'_{12} \\ x'_{22} \\ \vdots \\ x'_{I2} \end{bmatrix}, \tag{16.36}$$

where the vectors $\mathbf{x}'_{ij} = [x'_{ij}, y'_{ij}]^T$ are defined by

$$\begin{aligned} \mathbf{x}'_{i1} &= \mathbf{hom}[\mathbf{x}_{i1}, \boldsymbol{\Phi}_2\mathbf{M}] \\ \mathbf{x}'_{i2} &= \mathbf{hom}[\mathbf{x}_{i2}, \boldsymbol{\Phi}_2]. \end{aligned} \tag{16.37}$$

This least squares problem can be solved using the standard approach (appendix C.7.1). Figure 16.9 shows example rectified images. After these transformations, the corresponding points are guaranteed to be on the same horizontal scanline, and dense stereo reconstruction can proceed.

Decomposition of the fundamental matrix

The preceding algorithm requires the matrix $\mathbf{M}$ from the decomposition of the fundamental matrix as $\mathbf{F} = \mathbf{SM}$, where $\mathbf{S}$ is skew symmetric. A suitable way to

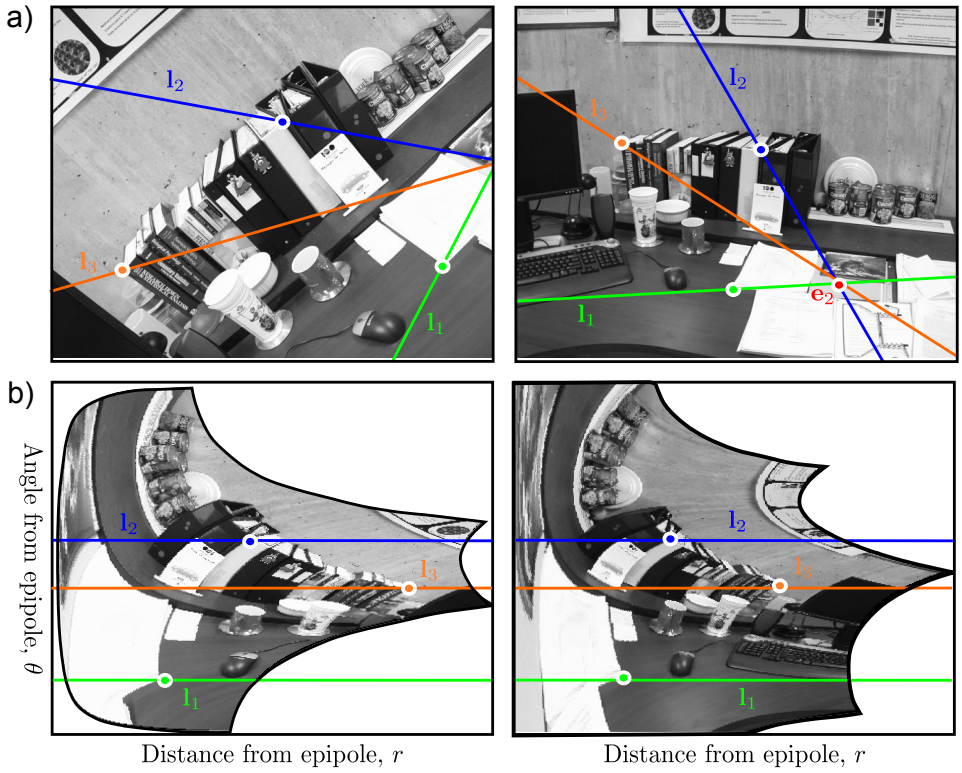

Figure 16.10 Polar rectification. When the epipole is inside one of the images, the planar rectification method is no longer suitable. An alternative in this situation is to perform a nonlinear warp of each image, in which the two new dimension corresponds to the distance and angle from the epipole, respectively. This is known as *polar rectification*.

do this is to compute the singular value decomposition of the fundamental matrix $\mathbf{F} = \mathbf{U}\mathbf{L}\mathbf{V}^T$. We then define the matrices

$$\mathbf{L}' = \begin{bmatrix} l_{11} & 0 & 0 \\ 0 & l_{22} & 0 \\ 0 & 0 & \frac{l_{11}+l_{22}}{2} \end{bmatrix} \text{ and } \mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{16.38}$$

where l_{ii} denotes the i^{th} element from the diagonal of $\mathbf{L}$. Finally, we choose

$$\mathbf{M} = \mathbf{U}\mathbf{W}\mathbf{L}'\mathbf{V}^T. \tag{16.39}$$

16.5.2 Polar rectification

The planar rectification method described in section 16.5.1 is suitable when the epipole is sufficiently far outside the image. Since the basis of this method is to map the epipoles to infinity it cannot work when the epipole is inside the image, and distorts the image a great deal if it is close to the image. Under these circumstances, a *polar rectification* is preferred.

Polar rectification applies a nonlinear warp to each image so that corresponding points are mapped to the same scanline. Each new image is formed by re-sampling the original images so that the first new axis is the distance from the epipole and the second new axis is the angle from the epipole (figure 16.10). This approach can distort the image significantly but works for all camera configurations.

This method is conceptually simple, but should be implemented with caution; when the epipole lies within the camera image, it is important to ensure that the correct half of the epipolar line is aligned with the appropriate part of the other image. The reader is encouraged to consult the original description (Pollefeys *et al.* 1999b) before implementing this algorithm.

16.5.3 After rectification

After rectification, the horizontal offset between every point in the first image and its corresponding point in the second image can be computed using a dense stereo algorithm (see sections 11.8.2 and 12.8.3). Typical results are shown in figure 16.11. Each point and its match are then warped back to their original positions (i.e., their image positions are 'un-rectified'). For each pair of 2D points, the depth can then be computed using the algorithm of section 14.6.

Finally, we may wish to view the model from a novel direction. For the two-view case, a simple way to do this is to form a 3D triangular mesh and to texture this mesh using the information from one or the other image. If we have computed dense matches using a stereo matching algorithm, then the mesh can be computed from the perspective of one camera with two triangles per pixel and cut where there are sharp depth discontinuities. If we have only a sparse correspondence between the two images, then it is usual to triangulate the projections of the 3D points in one image using a technique such as Delaunay triangulation to form the mesh. The textured mesh can now be viewed from a novel direction using the standard computer graphics pipeline.

16.6 Multi-view reconstruction

So far, we have considered reconstruction based on two views of a scene. Of course, it is common to have more than two views. For example, we might want to build 3D models using a video sequence taken by a single moving camera, or equivalently from a video sequence from a static camera of a rigidly moving object (figure 16.12).

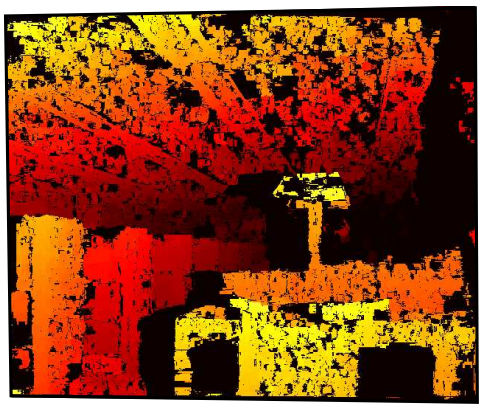

Figure 16.11 Disparity. After rectification, the horizontal offset at each point can be computed using a dense stereo algorithm. Here we used the method of Sizintsev *et al.* (2010). Color indicates the horizontal shift (disparity) between images. Black regions indicate places where the matching was ambiguous or where the corresponding part of the scene was occluded. Given these horizontal correspondences, we undo the rectification to get a dense set of matching points and their offsets (now displaced in 2D). The 3D position can now be computed using the method from section 14.6.

This problem is often referred to as *structure from motion* or *multi-view reconstruction*.

The problem is conceptually very similar to the two camera case. Once again, the solution ultimately relies on a nonlinear optimization in which we manipulate the camera position and the three-dimensional points to minimize the squared reprojection error (equation 16.2), and hence maximize the likelihood of the model. However, the multi-view case does bring several new aspects to the problem.

First, if we have a number of frames all taken with the same camera, there are now sufficient constraints to estimate the intrinsic parameters as well. We initialize the camera matrix with sensible values and add this to the final optimization. This process is known as *auto-calibration*. Second, matching points in video footage is easier, because the changes between adjacent frames tend to be small, and so the features can be explicitly tracked in two dimensions. However, it is usual for some points to be occluded in any given frame, and so we must keep track of which points are present at which time (figure 16.12f).

Problems 16.9

Third, there are now additional constraints on feature matching, which make it easier to eliminate outliers in the matching set of points. Consider a point that is matched between three frames. The point in the third frame will be constrained to lie on an epipolar line due to the first frame and another epipolar line due to the second frame: its position has to be at the intersection of the two lines and so is determined exactly. Unfortunately, this method will not work when the two predicted epipolar lines are the same as there is not a unique intersection. Consequently, the position in the third view is computed in a different way in practice. Just as we derived the fundamental matrix relation (equation 16.22) constraining the positions of matching points between two views, it is possible to derive a closed-form relation that constrains the positions across three images. The three-view analogue of the fundamental matrix is called the *tri-focal tensor*. It can be used to predict the position of the point in a third image given its position in the first and second images even when the epipolar lines are parallel. There is also a relation between four images, which is captured by the quadri-focal tensor, but there are no further relations between points in $J > 5$ images.

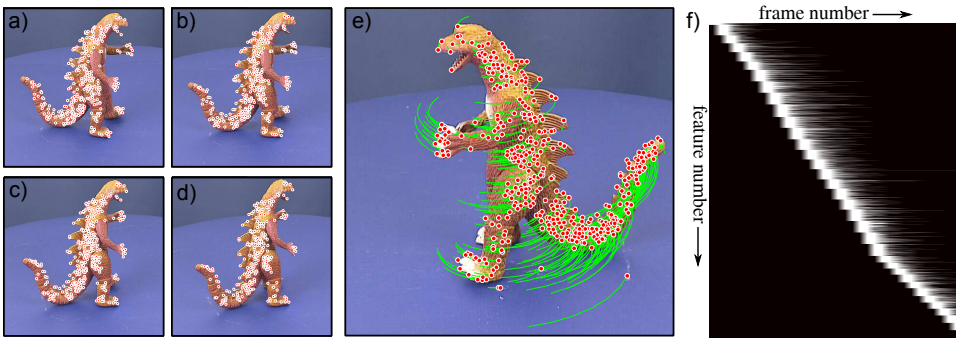

Figure 16.12 Multi-frame structure from motion. The goal is to construct a 3D model from a continuous video stream of a moving camera viewing a static object, or a static camera viewing a moving rigid object. a-d) Features are computed in each frame and tracked through the sequence. e) Features in current frame and their history. f) In each new frame, a number of new features is identified, and these are tracked until they are occluded or the correspondence is lost. Here, the white pixels indicate that a feature was present in a frame, and black pixels indicate that it was absent.

Finally, there are new ways to get initial estimates of the unknown quantities. It may not be practical to get the initial estimates of camera position by computing the transformation between adjacent frames and chaining these through the sequence. The translation between adjacent frames may be too small to reliably estimate the motion, and errors accrue as we move through the sequence. Moreover, it is difficult to maintain a consistent estimate of the (ambiguous) scale throughout. To this end, methods have been developed that simultaneously provide an initial estimate of all the camera positions and 3D points, some of which are based on factorization of a matrix containing all the (x, y) positions of every point tracked throughout the sequence.

Problem 16.10

16.6.1 Bundle adjustment

After finding the initial estimates of the 3D positions (structure) and the camera positions (motion), we must again resort to a large nonlinear optimization problem to fine-tune these parameters. With I tracked points over J frames, the problem is formulated as

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\left[\sum_{i=1}^{I}\sum_{j=1}^{J}\log[Pr(\mathbf{x}_{ij}|\mathbf{w}_i,\boldsymbol{\Lambda},\boldsymbol{\Omega}_j,\boldsymbol{\tau}_j)]\right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}}\left[\sum_{i=1}^{I}\sum_{j=1}^{J}\log\left[\text{Norm}_{\mathbf{x}_{ij}}[\mathbf{pinhole}[\mathbf{w}_i,\boldsymbol{\Lambda},\boldsymbol{\Omega}_j,\boldsymbol{\tau}_j],\sigma^2\mathbf{I}]\right]\right],\end{aligned} \tag{16.40}$$

where $\boldsymbol{\theta}$ contains the unknown world points $\{\mathbf{w}_i\}_{i=1}^{I}$, the intrinsic matrix $\boldsymbol{\Lambda}$, and the extrinsic parameters $\{\boldsymbol{\Omega}_j,\boldsymbol{\tau}_j\}_{j=1}^{J}$. This optimization problem is known as *Euclidean bundle adjustment*. As for the two view case, it is necessary to constrain the overall scale of the solution in some way.

One way to solve this optimization problem is to use an alternating approach. We first improve the log likelihood with respect to each of the extrinsic sets of parameters $\{\boldsymbol{\Omega}_j,\boldsymbol{\tau}_j\}$ (and possibly the intrinsic matrix $\boldsymbol{\Lambda}$ if unknown), and then update each 3D position $\mathbf{w}_i$. This is known as *resection-intersection*. It seems attractive as it only involves optimizing over a small subset of parameters at any one time. However, this type of *coordinate ascent* is inefficient: it cannot take advantage of the large gains that come from varying all of the parameters at once.

To make progress, we note that the cost function is based on the normal distribution and so can be re-written in the least squares form

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\left[\mathbf{z}^T\mathbf{z}\right], \tag{16.41}$$

where the vector $\mathbf{z}$ contains the squared differences between the observed feature positions $\mathbf{x}_{ij}$ and the positions $\mathbf{pinhole}[\mathbf{w}_i,\boldsymbol{\Lambda},\boldsymbol{\Omega}_j,\boldsymbol{\tau}_j]$ predicted by the model with the current parameters:

$$\mathbf{z} = \begin{bmatrix}\mathbf{x}_{11}-\mathbf{pinhole}[\mathbf{w}_1,\boldsymbol{\Lambda},\boldsymbol{\Omega}_1,\boldsymbol{\tau}_1]\\ \mathbf{x}_{12}-\mathbf{pinhole}[\mathbf{w}_1,\boldsymbol{\Lambda},\boldsymbol{\Omega}_2,\boldsymbol{\tau}_2]\\ \vdots \\ \mathbf{x}_{IJ}-\mathbf{pinhole}[\mathbf{w}_I,\boldsymbol{\Lambda},\boldsymbol{\Omega}_J,\boldsymbol{\tau}_J]\end{bmatrix}. \tag{16.42}$$

The Gauss-Newton method (appendix B.2.3) is specialized to this type of problem and updates the current estimate $\boldsymbol{\theta}^{[t]}$ of the parameters using:

$$\boldsymbol{\theta}^{[t]} = \boldsymbol{\theta}^{[t-1]} + \lambda(\mathbf{J}^T\mathbf{J})^{-1}\frac{\partial f}{\partial\boldsymbol{\theta}}, \tag{16.43}$$

where $\mathbf{J}$ is the Jacobian matrix. The entry in the m^{th} row and n^{th} column of $\mathbf{J}$ consists of the derivative of the m^{th} element of $\mathbf{z}$ with respect to the n^{th} element of the parameter vector $\boldsymbol{\theta}$:

$$J_{mn} = \frac{\partial z_m}{\partial\theta_n}. \tag{16.44}$$

In a real structure from motion problem, there might be thousands of scene points, each with three unknowns, and also thousands of camera positions, each with six unknowns. At each stage of the optimization, we must invert $\mathbf{J}^T\mathbf{J}$, which

is a square matrix whose dimension is the same as the number of unknowns. When the number of unknowns is large, inverting this matrix becomes expensive.

However, it is possible to build a practical system by exploiting the the sparse structure of $\mathbf{J}^T\mathbf{J}$. This sparsity results from the fact that every squared error term does not depend on every unknown. There is one contributing error term per observed 2D point and this depends only on the associated 3D point, the intrinsic parameters, and the camera position in that frame. Problem 16.11

To exploit this structure, we order the elements of the Jacobian matrix as $\mathbf{J} = [\mathbf{J}_{\mathbf{w}}, \mathbf{J}_{\boldsymbol{\Omega}}]$ where $\mathbf{J}_{\mathbf{w}}$ contains the terms that relate to the unknown world points $\{\mathbf{w}_i\}_{i=1}^{I}$ in turn, and $\mathbf{J}_{\boldsymbol{\Omega}}$ contains the terms that relate to the unknown camera positions $\{\boldsymbol{\Omega}_j, \boldsymbol{\tau}_j\}_{j=1}^{J}$. For pedagogical reasons, we will assume that the intrinsic matrix $\boldsymbol{\Lambda}$ is known here and hence has no entries in the Jacobian. We see that the matrix to be inverted becomes

$$\mathbf{J}^T\mathbf{J} = \begin{bmatrix} \mathbf{J}_{\mathbf{w}}^T\mathbf{J}_{\mathbf{w}} & \mathbf{J}_{\mathbf{w}}^T\mathbf{J}_{\boldsymbol{\Omega}} \\ \mathbf{J}_{\boldsymbol{\Omega}}^T\mathbf{J}_{\mathbf{w}} & \mathbf{J}_{\boldsymbol{\Omega}}^T\mathbf{J}_{\boldsymbol{\Omega}} \end{bmatrix}. \tag{16.45}$$

We now note that the matrices in the top-left and bottom-right of this matrix are block-diagonal (different world points do not interact with one another, and neither do the parameters from different cameras). Hence, these two sub-matrices can be inverted very efficiently. The Schur complement relation (appendix C.8.2), allows us to exploit this fact to reduce the complexity of the larger matrix inversion.

The preceding description is only a sketch of a real bundle adjustment algorithm; in a real system, additional sparseness in $\mathbf{J}^T\mathbf{J}$ would be exploited, a more sophisticated optimization method such as Levenberg-Marquardt would be employed, and a robust cost function would be used to reduce the effect of outliers.

16.7 Applications

We first describe a typical pipeline for recovering a 3D mesh model from a sequence of video frames. We then discuss a system that can extract 3D information about a scene from images gathered by a search engine on the internet, and use this information to help navigate through the set of photos. Finally, we discuss a multi-camera system for capturing 3D objects that uses a volumetric representation and exploits a Markov random field prior to get a smooth reconstruction.

16.7.1 3D reconstruction pipeline

Pollefeys & Van Gool (2002) present a complete pipeline for constructing 3D models from a sequence of images taken from an uncalibrated hand-held camera (figure 16.13). In the first stage, they compute a set of interest points (corners) in each image. When the image data consist of individual still photos, these points are matched between images. When the image data consist of continuous video, they are tracked between frames. In either case, a sparse set of potential cor-

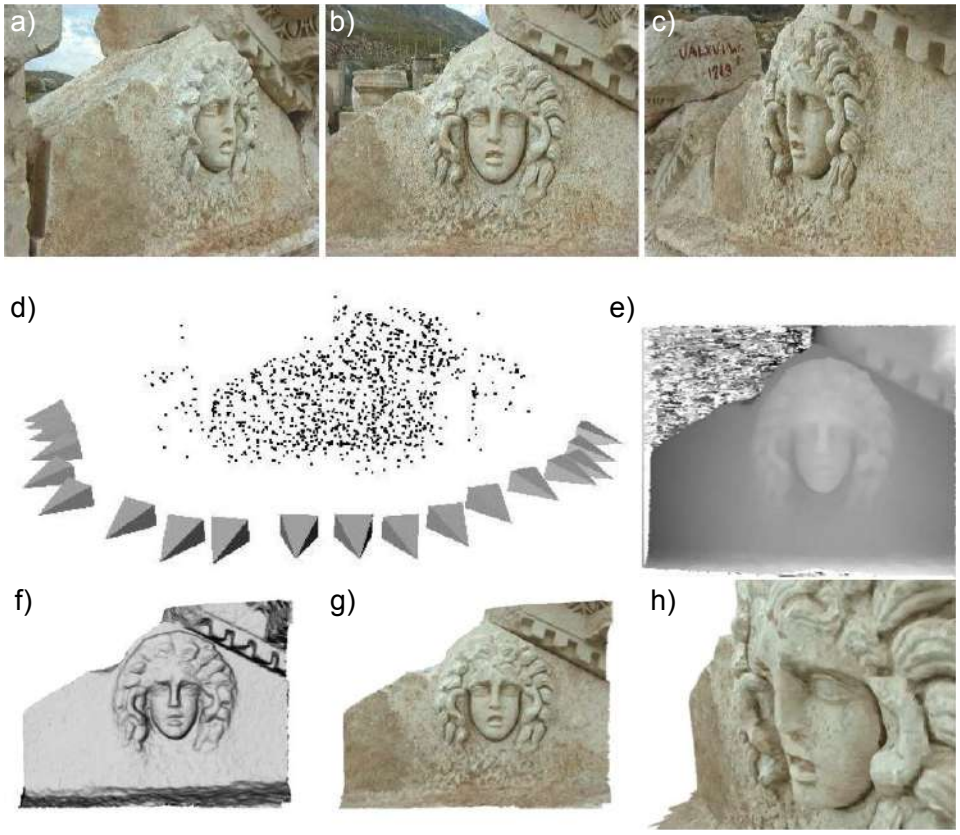

Figure 16.13 3D reconstruction pipeline. a-c) A 20 second video sequence of the camera panning around the medusa carving was captured. Every 20^{th} frame was used for the reconstruction, three of which are shown here. d) Sparse reconstruction (points) and estimated camera positions (pyramids) after bundle adjustment procedure. e) Depth map after dense stereo matching. f) Shaded 3D mesh model. g-h) Two views of textured 3D mesh model. Adapted from Pollefeys & Van Gool (2002). ©2002 Wiley.

respondences is obtained. The multi-view relations are estimated using a robust procedure, and these are then used to eliminate outliers from the correspondence set.

To estimate the motion of the cameras, two images are chosen and a projective reconstruction is computed (i.e., a reconstruction that is ambiguous up to a 3D projective transformation because the intrinsic parameters are not known). For each of the other images in turn, the pose for the camera is determined relative to this reconstruction and the reconstruction is refined. In this way it is possible to incorporate views that have no common features with the original two frames.

A subsequent bundle adjustment procedure minimizes the re-projection errors

Figure 16.14 Photo-tourism. a) A sparse 3D model of an object is computed from a set of photographs retrieved from the internet and the relative positions of the cameras (pyramids) are estimated. b) This 3D model is used as the basis of an interface that provides novel ways to explore the photo-collection by moving from image to image in 3D space. Adapted from Snavely *et al.* (2006). ©2006 ACM.

to get more accurate estimates of the camera positions and the points in the 3D world. At this stage, the reconstruction is still ambiguous up to a projective ambiguity and only now are initial estimates of the intrinsic parameters computed using a specialized procedure (see Pollefeys *et al.* 1999a). Finally, a full bundle adjustment method is applied; it simultaneously refines the estimates of the intrinsic parameters, camera positions, and 3D structure of the scene.

Successive pairs of images are rectified, and a dense set of disparities are computed using a multi-resolution dynamic programming technique. Given these dense correspondences, it is possible to compute an estimate of the 3D scene from the point of view of both cameras. A final estimate of the 3D structure relative to a reference frame is computed by fusing all of these independent estimates using a Kalman filter (see chapter 19).

For relatively simple scenes, a 3D mesh is computed by placing the vertices of the triangles in 3D space according to the values found in the depth map of the reference frame. The associated texture map can be retrieved from one or more of the original images. For more complex scenes, a single reference frame may not suffice and so several meshes are computed from different reference frames and fused together. Figures 16.13g-h show examples of the resulting textured 3D model of a Medusa head at the ancient site of Sagalassos in Turkey. More details concerning this pipeline can be found in Pollefeys *et al.* (2004).

16.7.2 Photo-tourism

Snavely *et al.* (2006) present a system for browsing a collection of images of an object that were gathered from the internet. A sparse 3D model of the object is

created by locating SIFT features in each image and finding a set of correspondences between pairs of images by computing the fundamental matrix using the eight-point algorithm with RANSAC.

A bundle adjustment procedure is then applied to estimate the camera positions and a sparse 3D model of the scene (figure 16.14a). This optimization procedure starts with only a single pair of images and gradually includes images based on their overlap with the current reconstruction, 're-bundling' at each stage. The intrinsic matrix of each camera is also estimated in this step, but this is simplified by assuming that the center of projection is co-incident with the image center, that the skew is zero, and that the pixels are square, leaving a single focal length parameter. This is initialized in the optimization using information from the EXIF tags of the image when they are present. The bundle adjustment procedure was lengthy; for the model of Notre-Dame, it took two weeks to compute a model from 2635 photos of which 597 images were ultimately included. However, more recent approaches to reconstruction from internet photos such as that of Frahm *et al.* (2010) are considerably faster.

This sparse 3D model of the scene is exploited to create a set of tools for navigating around the set of photographs. For example, it is possible to

- select a particular view based on a 3D rendering (as in figure 16.14a),
- find images of the object that are similar to the current view,
- retrieve images of the object taken from the left or the right of the current position (effectively pan around the object),
- find images that are from a similar viewpoint but closer or further from the object (zoom into / away from the object),
- and annotate objects and have these annotations transferred to other images.

This system was extended by Snavely *et al.* (2008) to allow more natural interaction with the space of images. For example, in this system it is possible to pan smoothly around objects by warping the original photos, so that they appear to define a smooth path through space.

16.7.3 Volumetric graph cuts

The reconstruction pipeline described in section 16.7.1 has the potential disadvantage that it requires the merging of multiple meshes of the object computed from different viewpoints. Vogiatzis *et al.* (2007) presented a system that uses a volumetric representation of depth to avoid this problem. In other words the 3D space that we wish to reconstruct is divided into a 3D grid, and each constituent element (voxel) is simply labeled as being inside or outside the object. Hence, reconstruction can be viewed as a binary segmentation of the 3D space.

The relative positions of the cameras are computed using a standard bundle adjustment approach. However, the reconstruction problem is now formulated in terms of an energy function consisting of two terms and optimized using graph cuts. First, there is an occupation cost for labeling each voxel as either foreground

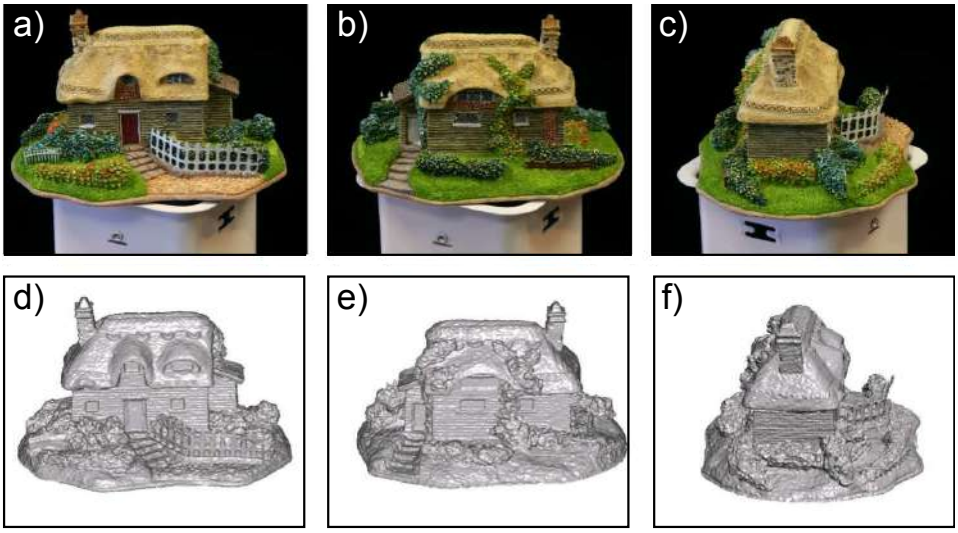

Figure 16.15 Volumetric graph cuts. a-c) Three of the original photos used to build the 3D model. d-f) Renderings of the resulting model from similar viewpoints. Adapted from Vogiatzis *et al.* (2007). ©2007 IEEE.

or background. Second, there is a discontinuity cost for lying at the boundary between the two partitions. We will now examine each of these terms in more detail.

The cost of labeling a voxel as being within the object is set to a very high value if this voxel does not project into the silhouette of the object in each image (i.e., it is not within the visual hull). Conversely, it is assumed that concavities in the object do not extend beyond a fixed distance from this visual hull, so a very high cost is paid if voxels close to the center of the visual hull are labeled as being outside the object. For the remaining voxels, a data-independent cost is set that favors the voxel being part of the object and produces a ballooning tendency which counters the shrinking bias of the graph cut solution, which pays a cost at transitions between the object and the space around it.

The discontinuity cost for lying on the boundary of the object depends on the *photo-consistency* of the voxel; a voxel is deemed photo-consistent if it projects to positions with similar RGB values in all of the cameras from which it is visible. Of course, to evaluate this, we must estimate the set of cameras in which this point is visible. One approach to this is to approximate the shape of the object using the visual hull. However, Vogiatzis *et al.* (2007) propose a more sophisticated method in which each camera votes for the photo-consistency of the voxel based on its pattern of correlation with the other images.

The final optimization problem now takes the form of a sum of unary occupation costs and pairwise terms that encourage the final voxel label field to be smooth. These pairwise terms are modified by the discontinuity cost (an example

of using geodesic distance in graph cuts) so that the transition from foreground to background is more likely in regions where the photo-consistency is high. Figure 16.15 shows an example of a volumetric 3D model computed in this way.

Discussion

This chapter has not introduced any truly new models; rather we have explored the ramifications of using multiple projective pinhole camera models simultaneously. It is now possible to use these ideas to reconstruct 3D models from camera sequences of rigid objects with well-behaved optical properties. However, 3D reconstruction in more general cases remains an open research problem.

Notes

Multi-view geometry: For more information about general issues in multiview geometry consult the books by Faugeras *et al.* (2001), Hartley & Zisserman (2004) and Ma *et al.* (2004) and the online tutorial by Pollefeys (2002). A summary of multi-view relations was presented by Moons (1998).

Essential and fundamental matrices: The essential matrix was described by Longuet-Higgins (1981) and its properties were explored by Huang & Faugeras (1989), Horn (1990) and Maybank (1998) among others. The fundamental matrix was discussed in Faugeras (1992), Faugeras *et al.* (1992), Hartley (1992), and Hartley (1994). The eight-point algorithm for computing the essential matrix is due to Longuet-Higgins (1981). Hartley (1997) described a method for rescaling in the eight-point algorithm that improved its accuracy. Details of the seven-point algorithm for computing the fundamental matrix can be found in Hartley & Zisserman (2004). Nistér (2004) and Stewénius *et al.* (2006) describe methods for the relative orientation problem that work directly with five-point correspondences between the cameras.

Rectification: The planar rectification algorithm described in the text is adapted from the description in Hartley & Zisserman (2004). Other variations on planar rectification can be found in Fusiello *et al.* (2000), Loop & Zhang (1999) and Ma *et al.* (2004). The polar rectification procedure is due to Pollefeys *et al.* (1999b).

Features and feature tracking: The algorithms in this chapter rely on the computation of distinctive points in the image. Typically, these are found using the Harris corner detector (Harris & Stephens 1988) or the SIFT detector (Lowe 2004). A more detailed discussion of how these points are computed can be found in section 13.2. Methods for tracking points in smooth video sequences (as opposed to matching them across views with a wide baseline) are discussed in Lucas & Kanade (1981), Tomasi & Kanade (1991) and Shi & Tomasi (1994).

Reconstruction pipelines: Several authors have described pipelines for computing 3D structure based on a set of images of a rigid object including Fitzgibbon & Zisserman (1998), Pollefeys *et al.* (2004), Brown & Lowe (2005) and Agarwal *et al.* (2009). Newcombe & Davison (2010) present a recent system that runs at interactive speeds. A summary of this area can be found in Moons *et al.* (2009).

Factorization: Tomasi & Kanade (1992) developed an exact ML solution for the projection matrices and 3D points in a set of images based on factorization. This solution assumes that the projection process is affine (a simplification of the full pinhole model) and that every point is visible in every image. Sturm & Triggs (1996) developed a similar method that could be used for the full projective camera. Buchanan & Fitzgibbon (2005) discuss approaches to this problem when the complete set of data are not available.

Bundle adjustment: Bundle adjustment is a complex topic which is reviewed in Triggs *et al.* (1999). More recently Engels *et al.* (2006) discuss a real-time bundle adjustment approach that works with temporal sub-windows from a video sequence. Recent approaches to bundle adjustment have adopted a conjugate gradient optimization strategy (Byröd & Åström 2010; Agarwal *et al.* 2010). A public implementation of bundle adjustment has been made available by Lourakis & Argyros (2009). A system which is cutting edge at the time of writing is described in Jeong *et al.* (2010), and recent methods that use multicore processing have also been developed (Wu *et al.* 2011).

Multi-view reconstruction: The stereo algorithms in chapters 11 and 12 compute an estimate of depth at each pixel of one or both of the input images. An alternative strategy

is to use an image-independent representation of shape. Examples of such representations include voxel occupancy grids (Vogiatzis *et al.* 2007; Kutulakos & Seitz 2000), level sets (Faugeras & Keriven 1998; Pons *et al.* 2007) and polygonal meshes (Fua & Leclerc 1995, Hernández & Schmitt 2004). Many multi-view reconstruction techniques also enforce the constraints imposed by the silhouettes (see section 14.7.2) on the final solution (e.g., Sinha & Pollefeys 2005; Sinha *et al.* 2007; Kolev & Cremers 2008). A review of multi-view reconstruction techniques can be found in Seitz *et al.* (2006).

Problems

Problem 16.1 Sketch the pattern of epipolar lines on the images in figure 16.17a.

Problem 16.2 Show that the cross product relation can be written in terms of a matrix multiplication so that

$$\mathbf{a}\times\mathbf{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix}\begin{bmatrix} b_1 \\ b_2 \\ b_3 \end{bmatrix}.$$

Problem 16.3 Consider figure 16.16. Write the direction of the three 3D vectors $\mathbf{O}_1\mathbf{O}_2$, $\mathbf{O}_1\mathbf{w}$, and $\mathbf{O}_2\mathbf{w}$ in terms of the observed image positions $\mathbf{x}_1$,$\mathbf{x}_2$ and the rotation $\boldsymbol{\Omega}$ and translation $\boldsymbol{\tau}$ between the cameras. The scale of the vectors is unimportant.

The three vectors that you have found must be coplanar. The criterion for three 3D vectors $\mathbf{a},\mathbf{b},\mathbf{c}$ being coplanar can be written as $\mathbf{a}.(\mathbf{b}\times\mathbf{c}) = 0$. Use this criterion to derive the essential matrix.

Problem 16.4 A clueless computer vision professor writes:

"The essential matrix is a 3×3 matrix that relates image coordinates between two images of the same scene. It contains 8 independent degrees of freedom (it is ambiguous up to scale). It has rank 2. If we know the intrinsic matrices of the two cameras, we can use the essential matrix to recover the rotation and translation between the cameras exactly."

Edit this statement to make it factually correct.

Problem 16.5 The essential matrix relates points in two cameras so that

$$\mathbf{x}_2^T\mathbf{E}\mathbf{x}_1 = 0$$

is given by

$$\mathbf{E} = \begin{bmatrix} 0 & 0 & 10 \\ 0 & 0 & 0 \\ -10 & 0 & 0 \end{bmatrix}.$$

What is the epipolar line in image 2 corresponding to the point $x_1 = [1,-1,1]$? What is the epipolar line in image 2 corresponding to the points $x_1 = [-5,-2,1]$? Determine the position of the epipole in image 2. What can you say about the motion of the cameras?

Problem 16.6 Show that we can retrieve the essential matrix by multiplying together the expressions from the decomposition (equations 16.19) as $\mathbf{E} = \boldsymbol{\tau}_\times\boldsymbol{\Omega}$.

Problem 16.7 Derive the fundamental matrix relation:

$$\tilde{\mathbf{x}}_2^T \boldsymbol{\Lambda}_2^{-T} \mathbf{E} \boldsymbol{\Lambda}_1^{-1} \tilde{\mathbf{x}}_1 = 0.$$

Problem 16.8 I intend to compute the fundamental matrix using the eight-point algorithm. Unfortunately, my data set is polluted by 30% outliers. How many iterations of the RANSAC algorithm will I need to run to have a 99% probability of success (i.e., computing the fundamental matrix from eight inliers at least once)? How many iterations will I need if I use an algorithm based on seven points?

Problem 16.9 We are given the fundamental matrix $\mathbf{F}_{13}$ relating images 1 and 3 and the fundamental matrix $\mathbf{F}_{23}$ relating images 2 and 3. I am now given corresponding points $\mathbf{x}_1$ and $\mathbf{x}_2$ in images 1 and 2, respectively. Derive a formula for the position of the corresponding point in image 3.

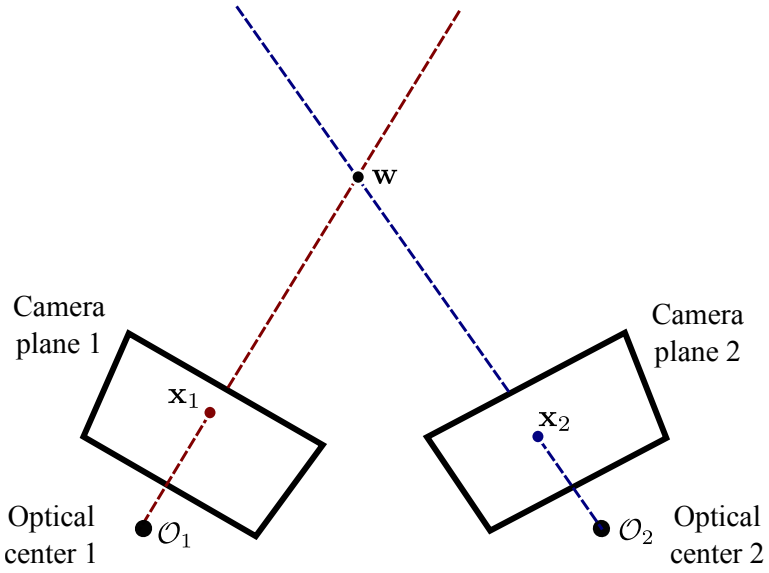

Figure 16.16 Figure for problem 16.3.

Problem 16.10 Tomasi-Kanade factorization. In the orthographic camera (figure 14.19), the projection process can be described as

$$\begin{aligned}\begin{bmatrix} x \\ y \end{bmatrix} &= \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} & \tau_x \\ \omega_{21} & \omega_{22} & \omega_{23} & \tau_y \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \pi_{11} & \pi_{12} & \pi_{13} \\ \pi_{21} & \pi_{22} & \pi_{23} \end{bmatrix} \begin{bmatrix} u \\ v \\ w \end{bmatrix} + \begin{bmatrix} \tau_x' \\ \tau_y' \end{bmatrix},\end{aligned}$$

or in matrix form

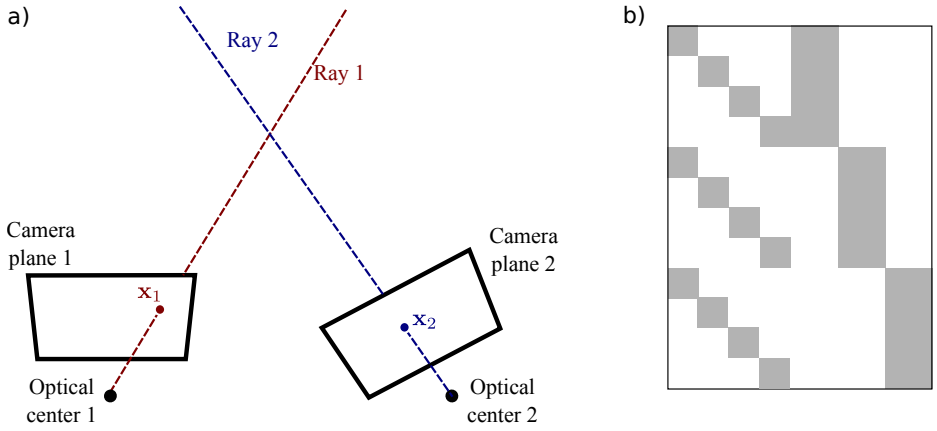

Figure 16.17 a) Figure for problem 16.1. c) Figure for problem 16.11. Gray regions represent non-zero entries in this portrait Jacobian matrix.

$$\mathbf{x} = \mathbf{\Pi}\mathbf{w} + \boldsymbol{\tau}'.$$

Now consider a data matrix $\mathbf{X}$ containing the positions $\{\mathbf{x}_{ij}\}_{i,j=1}^{IJ}$ of J points as seen in I images so that

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_{11} & \mathbf{x}_{12} & \ldots & \mathbf{x}_{iJ} \\ \mathbf{x}_{21} & \mathbf{x}_{22} & \ldots & \mathbf{x}_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{I1} & \mathbf{x}_{I2} & \ldots & \mathbf{x}_{IJ}, \end{bmatrix}$$

and where $\mathbf{x}_{ij} = [x_{ij}, y_{ij}]^T$.

(i) Show that the matrix $\mathbf{X}$ can be written in the form

$$\mathbf{X} = \mathbf{P}\mathbf{W} + \mathbf{T}$$

where $\mathbf{P}$ contains all of the I 3×2 projection matrices $\{\mathbf{\Pi}_i\}_{i=1}^{I}$, $\mathbf{W}$ contains all of the J 3D world positions $\{\mathbf{w}_j\}_{j=1}^{J}$ and $\mathbf{T}$ contains the translation vectors $\{\boldsymbol{\tau}_i'\}_{i=1}^{I}$.

(ii) Devise an algorithm to recover the matrices $\mathbf{P}, \mathbf{W}$ and $\mathbf{T}$ from the measurements $\mathbf{X}$. Is your solution unique?

Problem 16.11 Consider a Jacobian that has a structure of non-zero entries as shown in figure 16.17b. Draw an equivalent image that shows the structure of the non-zero entries in the matrix $\mathbf{J}^T\mathbf{J}$. Describe how you would use the Schur complement relation to invert this matrix efficiently.

Part VI

Models for vision

Part VI: Models for vision

In the final part of this book, we discuss four families of models. There is very little new theoretical material; these models are straight applications of the learning and inference techniques introduced in the first nine chapters. Nonetheless, this material addresses some of the most important machine vision applications: shape modeling, face recognition, tracking and object recognition.

In chapter 17, we discuss models that characterize the shape of objects. This is a useful goal in itself as knowledge of shape can help localize or segment an object. Furthermore, shape models can be used in combination with models for the RGB values to provide a more accurate generative account of the observed data.

In chapter 18, we investigate models that distinguish between the identities of objects and the style in which they are observed; a prototypical example of this type of application would be face recognition. Here the goal is to build a generative model of the data that can separate critical information about identity from the irrelevant image changes due to pose, expression and lighting.

In chapter 19, we discuss a family of models for tracking visual objects through time sequences. These are essentially graphical models based on chains such as those discussed in chapter 11. However, there are two main differences. First, we focus here on the case where the unknown variable is continuous rather than discrete. Second, we do not usually have the benefit of observing the full sequence; we must make a decision at each time based on information from only the past.

Finally, in chapter 20, we consider models for object and scene recognition. An important recent discovery is that good object recognition performance can be achieved using a discrete representation where the image is characterized as an unstructured histogram of *visual words*. Hence, this chapter considers models where the observed data are discrete.

It is notable that all of these families of models are generative; it has proven difficult to integrate complex knowledge about the structure of visual problem into discriminative models.