Chapter 15

Models for transformations

In this chapter, we consider a pinhole camera viewing a plane in the world. In these circumstances, the camera equations simplify to reflect the fact that there is a one-to-one mapping between points on this plane and points in the image.

Mappings between the plane and the image can be described using a family of 2D geometric transformations. In this chapter, we characterize these transformations and show how to estimate their parameters from data. We revisit the three geometric problems from chapter 14 for the special case of a planar scene.

To motivate the ideas of this chapter, consider an augmented reality application in which we wish to superimpose 3D content onto a planar marker (figure 15.1). To do this, we must establish the rotation and translation of the plane relative to the camera. We will do this in two stages. First, we will estimate the 2D transformation between points on the marker and points in the image. Second, we will extract the rotation and translation from the transformation parameters.

15.1 2D transformation models

In this section, we consider a family of 2D transformations, starting with the simplest and working toward the most general. We will motivate each by considering viewing a planar scene under different viewing conditions.

15.1.1 Euclidean transformation model

Consider a calibrated camera viewing a fronto-parallel plane at known distance, D (i.e., a plane whose normal corresponds to the w-axis of the camera). This may seem like a contrived situation, but it is exactly what happens in machine inspection applications: an overhead camera views a conveyor belt and examines objects that contain little or no depth variation.

We assume that a position on the plane can be described by a 3D position $\mathbf{w} = [u, v, 0]^T$, measured in real-world units such as millimeters. The w-coordinate

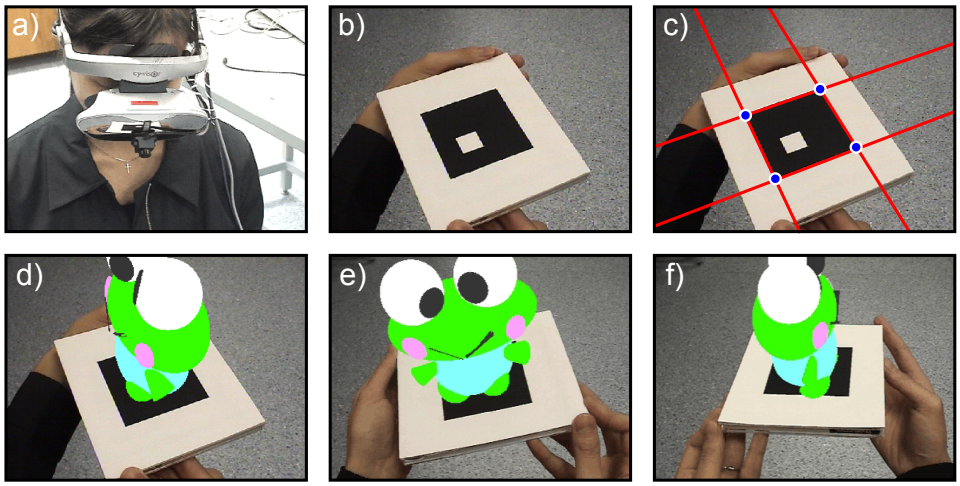

Figure 15.1 Video see-through augmented reality with a planar scene. a) The user views the world through a head mounted display with a camera attached to the front. The images from the camera are analyzed and augmented in near-real time and displayed to the user. b) Here, the world consists of a planar 2D marker. c) The marker corners are found by fitting edges to its sides and finding their intersections. d) The geometric transformation between the 2D positions of the corners on the marker surface and the corresponding positions in the image is computed. This transformation is analyzed to find the rotation and translation of the camera relative to the marker. This allows us to superimpose a 3D object as if it were rigidly attached to the surface of the image. e-f) As the marker is manipulated the superimposed object changes pose appropriately.

represents directions perpendicular to the plane, and is hence always zero. Consequently, we will sometimes treat $\mathbf{w} = [u, v]^T$ as a 2D coordinate.

Applying the pinhole camera model to this situation gives

$$\lambda\tilde{\mathbf{x}} = \mathbf{\Lambda}[\boldsymbol{\Omega}, \boldsymbol{\tau}]\tilde{\mathbf{w}}, \tag{15.1}$$

where $\tilde{\mathbf{x}}$ is the 2D observed image position represented as a homogeneous 3-vector and $\tilde{\mathbf{w}}$ is the 3D point in the world represented as a homogeneous 4-vector. Writing this out explicitly, we have

$$\begin{aligned}\lambda\begin{bmatrix}x\\y\\1\end{bmatrix} &= \begin{bmatrix}\phi_x & \gamma & \delta_x\\ 0 & \phi_y & \delta_y\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\omega_{11} & \omega_{12} & 0 & \tau_x\\ \omega_{21} & \omega_{22} & 0 & \tau_y\\ 0 & 0 & 1 & D\end{bmatrix}\begin{bmatrix}u\\v\\0\\1\end{bmatrix}\\ &= \begin{bmatrix}\phi_x & \gamma & \delta_x\\ 0 & \phi_y & \delta_y\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}\omega_{11} & \omega_{12} & \tau_x\\ \omega_{21} & \omega_{22} & \tau_y\\ 0 & 0 & D\end{bmatrix}\begin{bmatrix}u\\v\\1\end{bmatrix},\end{aligned} \tag{15.2}$$

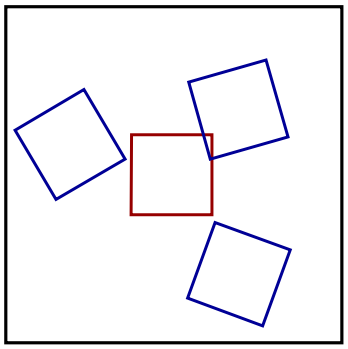

Figure 15.2 The 2D Euclidean transformation describes 2D rigid rotations and translations. The blue squares are all Euclidean transformations of the original red square. The transformation has three parameters: the rotation angle and the translations in the x-and y-directions. When a camera views a fronto-parallel plane at a known distance, the relation between the normalized camera coordinates and the 2D positions on the plane is a Euclidean transformation.

where the 3D rotation matrix $\boldsymbol{\Omega}$ takes a special form with only four unknowns, reflecting the fact that the plane is known to be fronto-parallel.

We can move the distance parameter D into the intrinsic matrix without changing the last of these three equations and equivalently write

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & D \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \tau_x \\ \omega_{21} & \omega_{22} & \tau_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \tag{15.3}$$

If we now eliminate the effect of this modified intrinsic matrix, by pre-multiplying both left and right by its inverse, we get

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} & \tau_x \\ \omega_{21} & \omega_{22} & \tau_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{15.4}$$

where x' and y' are camera coordinates that are normalized with respect to this modified intrinsic matrix.

The mapping in equation 15.4 is known as a *Euclidean transformation*. It can be equivalently written in Cartesian coordinates as

Problem 15.1

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \omega_{11} & \omega_{12} \\ \omega_{21} & \omega_{22} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} \tau_x \\ \tau_y \end{bmatrix}, \tag{15.5}$$

or for short, we may write

$$\mathbf{x}' = \mathbf{euc}[\mathbf{w}, \boldsymbol{\Omega}, \boldsymbol{\tau}], \tag{15.6}$$

where $\mathbf{x}' = [x', y']^T$ contains the normalized camera coordinates and $\mathbf{w} = [u, v]^T$ is the real-world position on the plane.

The Euclidean transformation describes rigid rotations and translations in the plane (figure 15.2). Although this transformation appears to take six separate

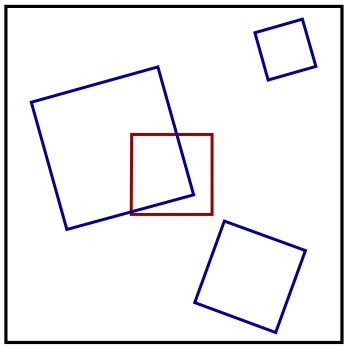

Figure 15.3 The similarity transformation describes rotations, translations and isotropic scalings. The blue quadrilaterals are all similarity transformations of the original red square. The transformation has four parameters: the rotation angle, the scaling and the translations in the x- and y-directions. When a camera views a fronto-parallel plane at unknown distance, the relation between the normalized camera coordinates and positions on the plane is a similarity.

parameters, the rotation matrix $\mathbf{\Omega}$ can be re-expressed in terms of the rotation angle θ,

$$\begin{bmatrix}\omega_{11} & \omega_{12}\\ \omega_{21} & \omega_{22}\end{bmatrix} = \begin{bmatrix}\cos[\theta] & \sin[\theta]\\ -\sin[\theta] & \cos[\theta]\end{bmatrix}, \tag{15.7}$$

and hence the actual number of parameters is three (the two offsets τ_x and τ_y, and the rotation, θ).

15.1.2 Similarity transformation model

Now consider a calibrated camera viewing a fronto-parallel plane at *unknown* distance D. The relationship between image points $\mathbf{x} = [x, y]^T$ and points $\mathbf{w} = [u, v, 0]^T$ on the plane is once more given by equation 15.2. Converting to normalized image co-ordinates by pre-multiplying both sides by the inverse of the intrinsic matrix gives

$$\lambda\begin{bmatrix}x'\\ y'\\ 1\end{bmatrix} = \begin{bmatrix}\omega_{11} & \omega_{12} & 0 & \tau_x\\ \omega_{21} & \omega_{22} & 0 & \tau_y\\ 0 & 0 & 1 & D\end{bmatrix}\begin{bmatrix}u\\ v\\ 0\\ 1\end{bmatrix} = \begin{bmatrix}\omega_{11} & \omega_{12} & \tau_x\\ \omega_{21} & \omega_{22} & \tau_y\\ 0 & 0 & D\end{bmatrix}\begin{bmatrix}u\\ v\\ 1\end{bmatrix}. \tag{15.8}$$

We now multiply each of these three equations by $\rho = 1/D$ to get

$$\rho\lambda\begin{bmatrix}x'\\ y'\\ 1\end{bmatrix} = \begin{bmatrix}\rho\omega_{11} & \rho\omega_{12} & \rho\tau_x\\ \rho\omega_{21} & \rho\omega_{22} & \rho\tau_y\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}u\\ v\\ 1\end{bmatrix}. \tag{15.9}$$

This is the homogeneous representation of the *similarity transformation*. However, it is usual to incorporate ρ into the constant λ on the left hand side so that $\lambda \leftarrow \rho\lambda$ and into the translation parameters so that $\tau_x \leftarrow \rho\tau_x$ and $\tau_y \leftarrow \rho\tau_y$ on the right hand side to yield

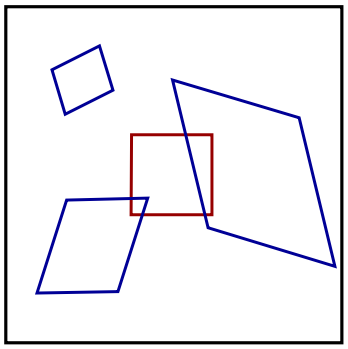

Figure 15.4 The affine transformation describes rotations, translations, scalings and shears. The blue quadrilaterals are all affine transformations of the original red square. The affine transformation has six parameters: the translations in the x- and y-directions and four parameters that determine the other effects. Notice that lines that were originally parallel remain parallel after the affine transformation is applied, so in each case the square becomes a parallelogram.

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \rho\omega_{11} & \rho\omega_{12} & \tau_x \\ \rho\omega_{21} & \rho\omega_{22} & \tau_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \tag{15.10}$$

Converting to Cartesian coordinates, we have

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \rho\omega_{11} & \rho\omega_{12} \\ \rho\omega_{21} & \rho\omega_{22} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} \tau_x \\ \tau_y \end{bmatrix}, \tag{15.11}$$

or for short,

$$\mathbf{x}' = \mathbf{sim}[\mathbf{w}, \boldsymbol{\Omega}, \boldsymbol{\tau}, \rho]. \tag{15.12}$$

The similarity transformation is a Euclidean transformation with a scaling (figure 15.3) and has four parameters: the rotation, the scaling, and two translations.

15.1.3 Affine transformation model

We motivated each of the previous transformations by considering a camera viewing a fronto-parallel plane. Ultimately, we wish to describe the relationship between image points and points on a plane in general position. As an intermediate step, let us generalize the transformations in equations 15.4 and 15.10 to

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \tau_x \\ \phi_{21} & \phi_{22} & \tau_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{15.13}$$

where $\phi_{11}, \phi_{12}, \phi_{21}$, and ϕ_{22} are now unconstrained, so can take arbitrary values. This is known as an *affine transformation*. In Cartesian coordinates, we have

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} \\ \phi_{21} & \phi_{22} \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} + \begin{bmatrix} \tau_x \\ \tau_y \end{bmatrix}, \tag{15.14}$$

Figure 15.5 Approximating projection of a plane. a) A planar object viewed with a camera with a narrow field of view (long focal length) from a large distance. The depth variation within the object is small compared to the distance from the camera to the plane. Here, perspective distortion is small, and the relationship between points in the image and points on the surface is well approximated by an affine transform. b) The same planar object viewed with a wide field of view (short focal length) from a short distance. The depth variation within the object is comparable to the average distance from the camera to the plane. An affine transformation cannot describe this situation well. c) However, a projective transformation (homography) captures the relationship between points on the surface and points in this image.

or for short, we might write

$$\mathbf{x}' = \mathbf{aff}[\mathbf{w}, \boldsymbol{\Phi}, \boldsymbol{\tau}]. \tag{15.15}$$

Note that the camera calibration matrix $\boldsymbol{\Lambda}$ also has the form of an affine transformation (i.e., a 3×3 matrix with two zeros in the bottom row). The product of two affine transformations is a third affine transformation, so if equation 15.15 is true, then there is also an affine transformation between points on the plane and the original (un-normalized) pixel positions.

Problem 15.7

The affine transformation encompasses both Euclidean and similarity transformations, but also includes *shears* (figure 15.4). However, it is far from general, and a notable restriction is that parallel lines are always mapped to other parallel lines. It has six unknown parameters, each of which can take any value.

The question remains as to whether the affine transformation really does provide a good mapping between points on a plane and their positions in the image. This is indeed the case when the depth variation of the plane as seen by the camera is small relative to the mean distance from the camera. In practice, this occurs when the viewing angle is not too oblique, the camera is distant, and the field of view is small (figure 15.5a). In more general situations, the affine transformation is not a good approximation. A simple counter-example is the convergence of parallel train tracks in an image as they become more distant. The affine transformation cannot describe this situation, as it can only map parallel lines on the object to parallel lines in the image.

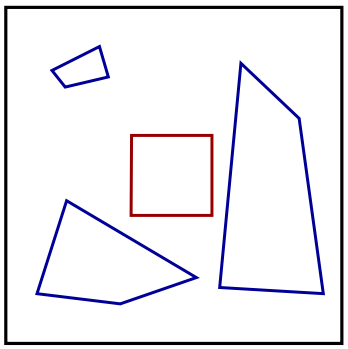

Figure 15.6 The projective transformation (also known as a collinearity or homography) can map any four points in the plane to any other four points. Rotations, translations, scalings, and shears and are all special cases. The blue quadrilaterals are all projective transformations of the original red square. The projective transformation has eight parameters. Lines that were parallel are not constrained to remain parallel after the projective transformation is applied.

15.1.4 Projective transformation model

Finally, we investigate what really happens when a pinhole camera views a plane from an arbitrary viewpoint. The relationship between a point $\mathbf{w} = [u, v, 0]^T$ on the plane and the position $\mathbf{x} = [x, y]^T$ to which it is projected is

$$\begin{aligned}\lambda\begin{bmatrix}x\\y\\1\end{bmatrix} &= \begin{bmatrix}\phi_x & \gamma & \delta_x\\0 & \phi_y & \delta_y\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}\omega_{11} & \omega_{12} & \omega_{13} & \tau_x\\\omega_{21} & \omega_{22} & \omega_{23} & \tau_y\\\omega_{31} & \omega_{32} & \omega_{33} & \tau_z\end{bmatrix}\begin{bmatrix}u\\v\\0\\1\end{bmatrix}\\ &= \begin{bmatrix}\phi_x & \gamma & \delta_x\\0 & \phi_y & \delta_y\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}\omega_{11} & \omega_{12} & \tau_x\\\omega_{21} & \omega_{22} & \tau_y\\\omega_{31} & \omega_{32} & \tau_z\end{bmatrix}\begin{bmatrix}u\\v\\1\end{bmatrix}. \end{aligned} \tag{15.16}$$

Combining the two 3×3 matrices by multiplying them together, the result is a transformation with the general form

$$\lambda\begin{bmatrix}x\\y\\1\end{bmatrix} = \begin{bmatrix}\phi_{11} & \phi_{12} & \phi_{13}\\\phi_{21} & \phi_{22} & \phi_{23}\\\phi_{31} & \phi_{32} & \phi_{33}\end{bmatrix}\begin{bmatrix}u\\v\\1\end{bmatrix}, \tag{15.17}$$

and is variously known as a *projective transformation*, a *collinearity*, or a *homography*. In Cartesian coordinates the homography is written as

$$\begin{aligned}x &= \frac{\phi_{11}u + \phi_{12}v + \phi_{13}}{\phi_{31}u + \phi_{32}v + \phi_{33}}\\ y &= \frac{\phi_{21}u + \phi_{22}v + \phi_{23}}{\phi_{31}u + \phi_{32}v + \phi_{33}},\end{aligned} \tag{15.18}$$

or for short,

$$\mathbf{x} = \mathbf{hom}[\mathbf{w}, \mathbf{\Phi}]. \tag{15.19}$$

The homography can map any four points in the plane to any other four

Problem 15.8

points (figure 15.6). It is a linear transformation in homogenous coordinates (equation 15.17) but is nonlinear in Cartesian coordinates (equation 15.18). It subsumes the Euclidean, similarity, and affine transformations as special cases. It exactly describes the mapping between the 2D coordinates of points on a plane in the real world and their positions in an image of that plane (figure 15.5c).

Although there are nine entries in the matrix $\boldsymbol{\Phi}$, the homography only contains eight degrees of freedom; the entries are redundant with respect to scale. It is easy to see that a constant re-scaling of all nine values produces the same transformation, as the scaling factor cancels out of the numerator and denominator in equation 15.18. The properties of the homography are discussed further in section 15.5.1.

15.1.5 Adding uncertainty

The four geometric models presented in the preceding sections are deterministic. However, in a real system, the measured positions of features in the image are subject to noise, and we need to incorporate this uncertainty into our models. In particular, we will assume that the positions $\mathbf{x}_i$ in the image are corrupted by normally distributed noise with spherical covariance so that, for example, the likelihood for the homography becomes

$$Pr(\mathbf{x}|\mathbf{w}) = \text{Norm}_{\mathbf{x}}\left[\mathbf{hom}[\mathbf{w}, \boldsymbol{\Phi}], \sigma^2\mathbf{I}\right]. \tag{15.20}$$

This is a generative model for the 2D image data $\mathbf{x}$. It can be thought of as a simplified version of the pinhole camera model that is specialized for viewing planar scenes; it is a recipe that tells us how to find the position in the image $\mathbf{x}$ corresponding to a point $\mathbf{w}$ on the surface of the planar object in the world.

The learning and inference problems in this model (figure 15.7) are:

- **Learning**: we are given pairs of points $\{\mathbf{x}_i, \mathbf{w}_i\}_{i=1}^I$ where $\mathbf{x}_i$ is a position in the image and $\mathbf{w}_i$ is the corresponding position on the plane in the world. The goal is to use these to establish the parameters $\boldsymbol{\theta}$ of the transformation. For example, in the case of the homography, the parameters $\boldsymbol{\theta}$ would comprise the nine entries of the matrix $\boldsymbol{\Phi}$.
- **Inference**: we are given a new point in the image $\mathbf{x}^*$, and our goal is to find the position on the plane $\mathbf{w}^*$ that projected to it.

We consider these two problems in the following sections.

15.2 Learning in transformation models

We are given a set of I 2D positions $\mathbf{w}_i = [u_i, v_i]^T$ on the surface of the plane and the I corresponding 2D image positions $\mathbf{x}_i = [x_i, y_i]^T$ in the image. We select a

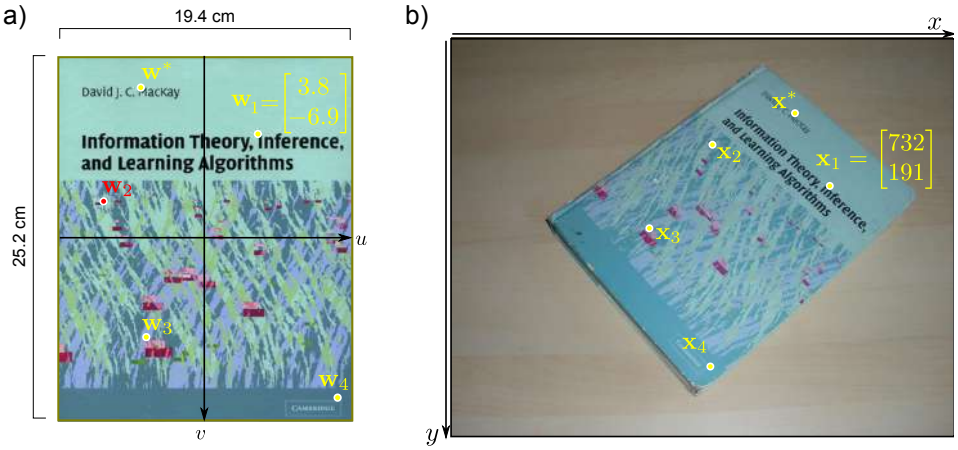

Figure 15.7 Learning and inference for transformation models. a) Planar object surface (position measured in cm). b) Image (position measured in pixels). In learning, we estimate the parameters of the mapping from points $\mathbf{w}$ on the object surface to image positions $\mathbf{x}_i$ based on pairs $\{\mathbf{x}_i, \mathbf{w}_i\}_{i=1}^{I}$ of known correspondences. We can use this mapping to find the position $\mathbf{x}^*$ in the image to which a point $\mathbf{w}^*$ on the object surface will project. In inference, we reverse this process: given a position $\mathbf{x}^*$ in the image, the goal is to establish the corresponding position $\mathbf{w}^*$ on the object surface.

transformation class of the form $\mathbf{trans}[\mathbf{w}_i, \boldsymbol{\theta}]$. The goal of the learning algorithm is then to estimate the parameters $\boldsymbol{\theta}$ that best map the points $\mathbf{w}_i$ to the image positions $\mathbf{x}_i$.

Adopting the maximum likelihood approach, we have

$$\begin{aligned} \hat{\boldsymbol{\theta}} &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\prod_{i=1}^{I} \mathrm{Norm}_{\mathbf{x}_i} \left[\mathbf{trans}[\mathbf{w}_i, \boldsymbol{\theta}], \sigma^2 \mathbf{I} \right] \right] \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \log \left[\mathrm{Norm}_{\mathbf{x}_i} \left[\mathbf{trans}[\mathbf{w}_i, \boldsymbol{\theta}], \sigma^2 \mathbf{I} \right] \right] \right], \end{aligned} \tag{15.21}$$

where, as usual, we have taken the logarithm which is a monotonic transformation and hence does not affect the position of the maximum. Substituting in the expression for the normal distribution and simplifying, we get the least squares problem

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} \left(\mathbf{x}_i - \mathbf{trans}[\mathbf{w}_i, \boldsymbol{\theta}] \right)^T \left(\mathbf{x}_i - \mathbf{trans}[\mathbf{w}_i, \boldsymbol{\theta}] \right) \right]. \tag{15.22}$$

The solutions to this least squares problem for each of the four transformation types are presented in sections 15.2.1-15.2.4. The details differ in each case, but

they have the common approach of reducing the problem into a standard form for which the solution is known. The algorithms are somewhat involved, and these sections can be skipped on first reading.

15.2.1 Learning Euclidean parameters

Algorithm 15.1

The Euclidean transformation is determined by a 2×2 rotation matrix $\boldsymbol{\Omega}$ and a 2×1 translation vector $\boldsymbol{\tau} = [\tau_x, \tau_y]^T$ (equations 15.4 and 15.5). Each pair of matching points $\{\mathbf{x}_i, \mathbf{w}_i\}$ contributes two constraints to the solution (deriving from the x- and y-coordinates). Since there are three underlying degrees of freedom, we will require at least $I = 2$ pairs of points to get a unique estimate.

Our goal is to solve the problem

$$\begin{aligned}\hat{\boldsymbol{\Omega}}, \hat{\boldsymbol{\tau}} &= \underset{\boldsymbol{\Omega},\boldsymbol{\tau}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} (\mathbf{x}_i - \mathbf{euc}[\mathbf{w}_i, \boldsymbol{\Omega}, \boldsymbol{\tau}])^T (\mathbf{x}_i - \mathbf{euc}[\mathbf{w}_i, \boldsymbol{\Omega}, \boldsymbol{\tau}])\right] \\ &= \underset{\boldsymbol{\Omega},\boldsymbol{\tau}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} (\mathbf{x}_i - \boldsymbol{\Omega}\mathbf{w}_i - \boldsymbol{\tau})^T (\mathbf{x}_i - \boldsymbol{\Omega}\mathbf{w}_i - \boldsymbol{\tau})\right], \end{aligned} \tag{15.23}$$

with the constraint that $\boldsymbol{\Omega}$ is a rotation matrix so that $\boldsymbol{\Omega}\boldsymbol{\Omega}^T = \mathbf{I}$ and $|\boldsymbol{\Omega}| = 1$.

An expression for the translation vector can be found by taking the derivative of the objective function with respect to $\boldsymbol{\tau}$, setting the result to zero and simplifying. The result is the mean difference vector between the two sets of points after the rotation has been applied

$$\hat{\boldsymbol{\tau}} = \frac{\sum_{i=1}^{I} \mathbf{x}_i - \boldsymbol{\Omega}\mathbf{w}_i}{I} = \boldsymbol{\mu}_x - \boldsymbol{\Omega}\boldsymbol{\mu}_w, \tag{15.24}$$

where $\boldsymbol{\mu}_x$ is the mean of the points $\{\mathbf{x}_i\}$ and $\boldsymbol{\mu}_w$ is the mean of the points $\{\mathbf{w}_i\}$. Substituting this result into the original criterion, we get

$$\hat{\boldsymbol{\Omega}} = \underset{\boldsymbol{\Omega}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} \left((\mathbf{x}_i - \boldsymbol{\mu}_x) - \boldsymbol{\Omega}(\mathbf{w}_i - \boldsymbol{\mu}_w)\right)^T \left((\mathbf{x}_i - \boldsymbol{\mu}_x) - \boldsymbol{\Omega}(\mathbf{w}_i - \boldsymbol{\mu}_w)\right)\right]. \tag{15.25}$$

Defining matrices $\mathbf{B} = [\mathbf{x}_1 - \boldsymbol{\mu}_x, \mathbf{x}_2 - \boldsymbol{\mu}_x, \ldots, \mathbf{x}_I - \boldsymbol{\mu}_x]$ and $\mathbf{A} = [\mathbf{w}_1 - \boldsymbol{\mu}_w, \mathbf{w}_2 - \boldsymbol{\mu}_w, \ldots, \mathbf{w}_I - \boldsymbol{\mu}_w]$, we can re-write the objective function for the best rotation $\boldsymbol{\Omega}$ as

$$\hat{\boldsymbol{\Omega}} = \underset{\boldsymbol{\Omega}}{\operatorname{argmin}} \left[|\mathbf{B} - \boldsymbol{\Omega}\mathbf{A}|_F\right] \qquad \text{subject to } \boldsymbol{\Omega}\boldsymbol{\Omega}^T = \mathbf{I}, |\boldsymbol{\Omega}| = 1, \tag{15.26}$$

where $|\bullet|_F$ denotes the Frobenius norm. This is an example of an *orthogonal Procrustes problem*. A closed form solution can be found by computing the SVD $\mathbf{U}\mathbf{L}\mathbf{V}^T = \mathbf{B}\mathbf{A}^T$, and then choosing $\hat{\boldsymbol{\Omega}} = \mathbf{V}\mathbf{U}^T$ (see appendix C.7.3).

15.2.2 Learning similarity parameters

The similarity transformation is determined by a 2×2 rotation matrix $\boldsymbol{\Omega}$, a 2×1 translation vector $\boldsymbol{\tau}$, and a scaling factor ρ (equations 15.10 and 15.11). There are four underlying degrees of freedom, so we will require at least $I = 2$ pairs of matching points $\{\mathbf{x}_i, \mathbf{w}_i\}$ to guarantee a unique solution. Algorithm 15.2

The objective function for maximum likelihood fitting of the parameters is

$$\begin{aligned}\hat{\boldsymbol{\Omega}}, \hat{\boldsymbol{\tau}}, \hat{\rho} &= \underset{\boldsymbol{\Omega},\boldsymbol{\tau},\rho}{\operatorname{argmin}} \left[\sum_{i=1}^{I} (\mathbf{x}_i - \mathbf{sim}[\mathbf{w}_i, \boldsymbol{\Omega}, \boldsymbol{\tau}, \rho])^T (\mathbf{x}_i - \mathbf{sim}[\mathbf{w}_i, \boldsymbol{\Omega}, \boldsymbol{\tau}, \rho])\right] \\ &= \underset{\boldsymbol{\Omega},\boldsymbol{\tau},\rho}{\operatorname{argmin}} \left[\sum_{i=1}^{I} (\mathbf{x}_i - \rho\boldsymbol{\Omega}\mathbf{w}_i - \boldsymbol{\tau})^T (\mathbf{x}_i - \rho\boldsymbol{\Omega}\mathbf{w}_i - \boldsymbol{\tau})\right], \qquad (15.27)\end{aligned}$$

with the constraint that $\boldsymbol{\Omega}$ is a rotation matrix so that $\boldsymbol{\Omega}\boldsymbol{\Omega}^T = \mathbf{I}$ and $|\boldsymbol{\Omega}| = 1$.

To optimize this criterion, we compute $\boldsymbol{\Omega}$ exactly as for the Euclidean transform. The maximum likelihood solution for the scaling factor is given by

$$\hat{\rho} = \frac{\sum_{i=1}^{I} (\mathbf{x}_i - \boldsymbol{\mu}_x)^T \hat{\boldsymbol{\Omega}} (\mathbf{w}_i - \boldsymbol{\mu}_w)}{\sum_{i=1}^{I} (\mathbf{w}_i - \boldsymbol{\mu}_w)^T (\mathbf{w}_i - \boldsymbol{\mu}_w)}, \qquad (15.28)$$

and the translation can be found using

$$\hat{\boldsymbol{\tau}} = \frac{\sum_{i=1}^{I} (\mathbf{x}_i - \hat{\rho}\hat{\boldsymbol{\Omega}}\mathbf{w}_i)}{I}. \qquad (15.29)$$

15.2.3 Learning affine parameters

The affine transformation is parameterized by an unconstrained 2×2 matrix $\boldsymbol{\Phi}$ and a 2×1 translation vector $\boldsymbol{\tau}$ (equations 15.13 and 15.14). There are six unknowns, and so we need a minimum of $I=3$ pairs of matching points $\{\mathbf{x}_i, \mathbf{w}_i\}$ to guarantee a unique solution. The learning problem can be stated as Algorithm 15.3

$$\begin{aligned}\hat{\boldsymbol{\Phi}}, \hat{\boldsymbol{\tau}} &= \underset{\boldsymbol{\Phi},\boldsymbol{\tau}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} (\mathbf{x}_i - \mathbf{aff}[\mathbf{w}_i, \boldsymbol{\Phi}, \boldsymbol{\tau}])^T (\mathbf{x}_i - \mathbf{aff}[\mathbf{w}_i, \boldsymbol{\Phi}, \boldsymbol{\tau}])\right] \\ &= \underset{\boldsymbol{\Phi},\boldsymbol{\tau}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} (\mathbf{x}_i - \boldsymbol{\Phi}\mathbf{w}_i - \boldsymbol{\tau})^T (\mathbf{x}_i - \boldsymbol{\Phi}\mathbf{w}_i - \boldsymbol{\tau})\right]. \qquad (15.30)\end{aligned}$$

To solve this problem, observe that we can re-express $\boldsymbol{\Phi}\mathbf{w}_i + \boldsymbol{\tau}$ as a linear function of the unknown elements of $\boldsymbol{\Phi}$ and $\boldsymbol{\tau}$

$$\mathbf{\Phi}\mathbf{w}_i + \boldsymbol{\tau} = \begin{bmatrix} u_i & v_i & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & u_i & v_i & 1 \end{bmatrix} \begin{bmatrix} \phi_{11} \\ \phi_{12} \\ \tau_x \\ \phi_{21} \\ \phi_{22} \\ \tau_y \end{bmatrix} = \mathbf{A}_i\mathbf{b}, \tag{15.31}$$

where $\mathbf{A}_i$ is a 2×6 matrix based on the point $\mathbf{w}_i$, and $\mathbf{b}$ contains the unknown parameters. The problem can now be written as

$$\hat{\mathbf{b}} = \underset{\mathbf{b}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} (\mathbf{x}_i - \mathbf{A}_i\mathbf{b})^T(\mathbf{x}_i - \mathbf{A}_i\mathbf{b}) \right], \tag{15.32}$$

which is a linear least squares problem and can be solved easily (appendix C.7.1).

15.2.4 Learning projective parameters

The projective transformation or homography is parameterized by a 3×3 matrix $\mathbf{\Phi}$ (equations 15.17 and 15.18) which is ambiguous up to scale, giving a total of eight degrees of freedom. Consequently, we need a minimum of $I = 4$ pairs of corresponding points for a unique solution. This neatly matches our expectations: a homography can map any four points in the plane to any other four points, and so it is reasonable that we should need at least four pairs of points to determine it.

Algorithm 15.4

The learning problem can be stated as

$$\begin{aligned} \hat{\mathbf{\Phi}} &= \underset{\mathbf{\Phi}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} (\mathbf{x}_i - \mathbf{hom}[\mathbf{w}_i, \mathbf{\Phi}])^T(\mathbf{x}_i - \mathbf{hom}[\mathbf{w}_i, \mathbf{\Phi}]) \right] \qquad (15.33) \\ &= \underset{\mathbf{\Phi}}{\operatorname{argmin}} \left[\sum_{i=1}^{I} \left(x_i - \frac{\phi_{11}u_i + \phi_{12}v_i + \phi_{13}}{\phi_{31}u_i + \phi_{32}v_i + \phi_{33}} \right)^2 + \left(y_i - \frac{\phi_{21}u_i + \phi_{22}v_i + \phi_{23}}{\phi_{31}u_i + \phi_{32}v_i + \phi_{33}} \right)^2 \right]. \end{aligned}$$

Unfortunately, there is no closed form solution to this nonlinear problem and to find the answer, we must rely on gradient-based optimization techniques. Since there is a scale ambiguity, this optimization would normally be carried out under the constraint that the sum of the squares of the elements of $\mathbf{\Phi}$ is one.

A successful optimization procedure depends on a good initial starting point, and for this we use the *direct linear transformation* or *DLT* algorithm. The DLT algorithm uses homogeneous coordinates where the homography is a linear transformation and finds a closed form solution for the algebraic error. This is not the same as optimizing the true objective function (equation 15.33), but provides a result that is usually very close to the true answer and can be used as a starting point for the nonlinear optimization of the true criterion. In homogeneous coordinates we have

$$\lambda \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix} \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix}. \tag{15.34}$$

Each homogeneous coordinate can be considered as a direction in 3D space (figure 14.11). So, this equation states that the left-hand side $\tilde{\mathbf{x}}_i$ represents the same direction in space as the right-hand side, $\boldsymbol{\Phi}\tilde{\mathbf{w}}_i$. If this is the case, their cross product must be zero, so that

$$\tilde{\mathbf{x}} \times \boldsymbol{\Phi}\tilde{\mathbf{w}} = \mathbf{0}. \tag{15.35}$$

Writing this constraint in full gives the relations

Problem 15.9

$$\begin{bmatrix} y(\phi_{31}u + \phi_{32}v + \phi_{33}) - (\phi_{21}u + \phi_{22}v + \phi_{23}) \\ (\phi_{11}u + \phi_{12}v + \phi_{13}) - x(\phi_{31}u + \phi_{32}v + \phi_{33}) \\ x(\phi_{21}u + \phi_{22}v + \phi_{23}) - y(\phi_{11}u + \phi_{12}v + \phi_{13}) \end{bmatrix} = \mathbf{0}. \tag{15.36}$$

This appears to provide three linear constraints on the elements of $\boldsymbol{\Phi}$. However, only two of these three equations are independent, so we discard the third. We now stack the first two constraints from each of the I pairs of points $\{\mathbf{x}_i, \mathbf{w}_i\}$ to form the system

$$\begin{bmatrix} 0 & 0 & 0 & -u_1 & -v_1 & -1 & y_1u_1 & y_1v_1 & y_1 \\ u_1 & v_1 & 1 & 0 & 0 & 0 & -x_1u_1 & -x_1v_1 & -x_1 \\ 0 & 0 & 0 & -u_2 & -v_2 & -1 & y_2u_2 & y_2v_2 & y_2 \\ u_2 & v_2 & 1 & 0 & 0 & 0 & -x_2u_2 & -x_2v_2 & -x_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & -u_I & -v_I & -1 & y_Iu_I & y_Iv_I & y_I \\ u_I & v_I & 1 & 0 & 0 & 0 & -x_Iu_I & -x_Iv_I & -x_I \end{bmatrix} \begin{bmatrix} \phi_{11} \\ \phi_{12} \\ \phi_{13} \\ \phi_{21} \\ \phi_{22} \\ \phi_{23} \\ \phi_{31} \\ \phi_{32} \\ \phi_{33} \end{bmatrix} = \mathbf{0}, \tag{15.37}$$

which has the form $\mathbf{A}\boldsymbol{\phi} = \mathbf{0}$.

We solve this system of equations in a least squares sense with the constraint $\boldsymbol{\phi}^T\boldsymbol{\phi} = 1$ to prevent the trivial solution $\boldsymbol{\phi} = \mathbf{0}$. This is a standard problem (see appendix C.7.2). To find the solution, we compute the SVD $\mathbf{A} = \mathbf{U}\mathbf{L}\mathbf{V}^T$ and choose $\boldsymbol{\phi}$ to be the last column of $\mathbf{V}$. This is reshaped into a 3×3 matrix $\boldsymbol{\Phi}$ and used as a starting point for the nonlinear optimization of the true criterion (equation 15.33).

15.3 Inference in transformation models

We have introduced four transformations (Euclidean, similarity, affine, projective) that relate positions $\mathbf{w}$ on a real-world plane to their projected positions $\mathbf{x}$ in the

Algorithm 15.5

image, and discussed how to learn their parameters. In each case, the transformation took the form of a generative model $Pr(\mathbf{x}|\mathbf{w})$. In this section, we consider how to infer the world position $\mathbf{w}$ from the image position $\mathbf{x}$.

For simplicity, we will take a maximum likelihood approach to this problem. For the generic transformation $\mathbf{trans}[\mathbf{w}_i, \boldsymbol{\theta}]$, we seek

$$\begin{aligned}\hat{\mathbf{w}} &= \underset{\mathbf{w}}{\text{argmax}} \left[\log\left[\text{Norm}_{\mathbf{x}}\left[\mathbf{trans}[\mathbf{w}, \boldsymbol{\theta}], \sigma^2\mathbf{I}\right]\right]\right] \\ &= \underset{\mathbf{w}}{\text{argmin}} \left[\left(\mathbf{x} - \mathbf{trans}[\mathbf{w}, \boldsymbol{\theta}]\right)^T \left(\mathbf{x} - \mathbf{trans}[\mathbf{w}, \boldsymbol{\theta}]\right)\right]. \end{aligned} \tag{15.38}$$

It is clear that this will be achieved when the image point and the predicted image point agree exactly so that

$$\mathbf{x} = \mathbf{trans}[\mathbf{w}, \boldsymbol{\theta}]. \tag{15.39}$$

We can find the $\mathbf{w} = [u, v]^T$ that makes this true by moving to homogeneous coordinates. Each of the four transformations can be written in the form

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \tag{15.40}$$

where the exact expressions are given by equations 15.4, 15.10, 15.13, and 15.17.

To find the position $\mathbf{w} = [u, v]^T$, we simply pre-multiply by the inverse of the transformation matrix to yield

$$\lambda' \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix}^{-1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \tag{15.41}$$

and then recover u and v by converting back to Cartesian co-ordinates.

15.4 Three geometric problems for planes

We have introduced transformation models that mapped from 2D coordinates on the plane to 2D coordinates in the image, the most general of which was the homography. Now we relate this model back to the full pinhole camera and revisit the three geometric problems from chapter 14 for the special case of a planar scene. We will show how to

- learn the extrinsic parameters (compute the geometric relationship between the plane and the camera),
- learn the intrinsic parameters (calibrate from a plane), and
- infer the 3D coordinates of a point in the plane relative to the camera, given its image position.

These three problems are illustrated in figures 15.8, 15.9 and 15.11, respectively.

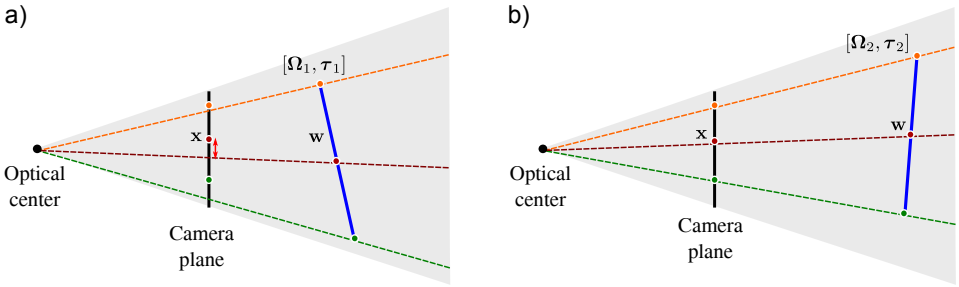

Figure 15.8 Problem 1 - Learning extrinsic parameters. Given points $\{\mathbf{w}_i\}_{i=1}^I$ on a plane, their positions $\{\mathbf{x}\}_{i=1}^I$ in the image and intrinsic parameters $\mathbf{\Lambda}$, find the rotation $\mathbf{\Omega}$ and translation $\boldsymbol{\tau}$ relating the camera and the plane. a) When the rotation and translation are wrong, the image points predicted by the model (where the rays strike the image plane) do not agree well with the observed points $\mathbf{x}$. b) When the rotation and translation are correct, they agree well and the likelihood $Pr(\mathbf{x}|\mathbf{w}, \mathbf{\Lambda}, \mathbf{\Omega}, \boldsymbol{\tau})$ will be high.

15.4.1 Problem 1: learning extrinsic parameters

Given I 3D points $\{\mathbf{w}_i\}_{i=1}^I$ *that lie on a plane* so that $w_i = 0$, their corresponding projections in the image $\{\mathbf{x}_i\}_{i=1}^I$ and known intrinsic matrix $\mathbf{\Lambda}$, estimate the extrinsic parameters

Algorithm 15.6

$$\begin{aligned}\hat{\mathbf{\Omega}}, \hat{\boldsymbol{\tau}} &= \underset{\mathbf{\Omega},\boldsymbol{\tau}}{\text{argmax}} \left[\sum_{i=1}^{I} \log\left[Pr(\mathbf{x}_i|\mathbf{w}_i, \mathbf{\Lambda}, \mathbf{\Omega}, \boldsymbol{\tau})\right]\right] \\ &= \underset{\mathbf{\Omega},\boldsymbol{\tau}}{\text{argmax}} \left[\sum_{i=1}^{I} \log\left[\text{Norm}_{\mathbf{x}_i}[\mathbf{pinhole}[\mathbf{w}_i, \mathbf{\Lambda}, \mathbf{\Omega}, \boldsymbol{\tau}], \sigma^2\mathbf{I}]\right]\right], \quad (15.42)\end{aligned}$$

where $\mathbf{\Omega}$ is a 3×3 rotation matrix and $\boldsymbol{\tau}$ is a 3×1 translation vector (figure 15.8). The extrinsic parameters transform points $\mathbf{w} = [u, v, 0]$ on the plane into the coordinate system of the camera. Unfortunately, this problem (still) cannot be solved in closed form and requires nonlinear optimization. As usual, it is possible to get a good initial estimate for the parameters using a closed form algebraic solution based on homogeneous coordinates.

From equation 15.16, the relation between a homogeneous point on the plane and its projection in the image is

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \lambda' \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & D \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \tau_x \\ \omega_{21} & \omega_{22} & \tau_y \\ \omega_{31} & \omega_{32} & \tau_z \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \quad (15.43)$$

Our approach is to (i) calculate the homography $\mathbf{\Phi}$ between the points $\mathbf{w} = [u, v]^T$ on the plane and the points $\mathbf{x} = [x, y]^T$ in the image using the method of

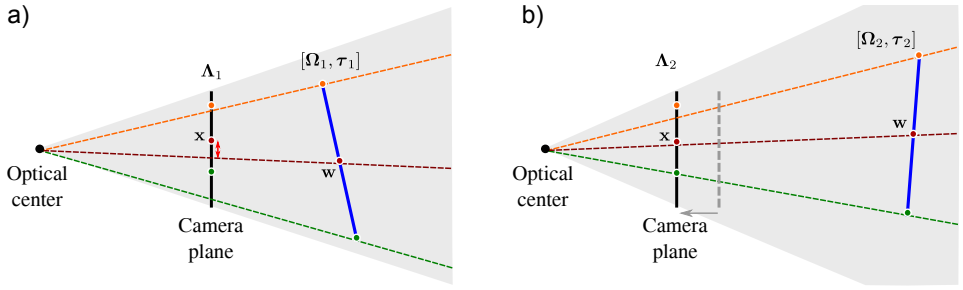

Figure 15.9 Problem 2 - Learning intrinsic parameters. Given a set of points $\{\mathbf{w}_i\}_{i=1}^I$ on a plane in the world (blue line) and the 2D positions $\{\mathbf{x}_i\}_{i=1}^I$ of these points in an image, find the intrinsic parameters $\mathbf{\Lambda}$. To do this, we must also simultaneously estimate the extrinsic parameters $\mathbf{\Omega}, \boldsymbol{\tau}$. a) When the intrinsic and extrinsic parameters are wrong, the prediction of the pinhole camera (where rays strike the image plane) deviates significantly from the known 2D points. b) When they are correct, the prediction of the model will agree with the observed image. To make the solution unique, the plane must be seen from several different angles.

section 15.2.4 and then (ii) decompose this homography to recover the rotation matrix $\mathbf{\Omega}$ and translation vector $\boldsymbol{\tau}$.

As a first step toward this decomposition, we eliminate the effect of the intrinsic parameters by pre-multiplying the estimated homography by the inverse of the intrinsic matrix $\mathbf{\Lambda}$. This gives a new homography $\mathbf{\Phi}' = \mathbf{\Lambda}^{-1}\mathbf{\Phi}$ such that

$$\begin{bmatrix} \phi'_{11} & \phi'_{12} & \phi'_{13} \\ \phi'_{21} & \phi'_{22} & \phi'_{23} \\ \phi'_{31} & \phi'_{32} & \phi'_{33} \end{bmatrix} = \lambda' \begin{bmatrix} \omega_{11} & \omega_{12} & \tau_x \\ \omega_{21} & \omega_{22} & \tau_y \\ \omega_{31} & \omega_{32} & \tau_z \end{bmatrix} \tag{15.44}$$

To estimate the first two columns of the rotation matrix $\mathbf{\Omega}$, we compute the SVD of the first two columns of $\mathbf{\Phi}'$

$$\begin{bmatrix} \phi'_{11} & \phi'_{12} \\ \phi'_{21} & \phi'_{22} \\ \phi'_{31} & \phi'_{32} \end{bmatrix} = \mathbf{U}\mathbf{L}\mathbf{V}^T, \tag{15.45}$$

and then set

$$\begin{bmatrix} \omega_{11} & \omega_{12} \\ \omega_{21} & \omega_{22} \\ \omega_{31} & \omega_{32} \end{bmatrix} = \mathbf{U} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix} \mathbf{V}^T. \tag{15.46}$$

These operations find the closest valid first two columns of a rotation matrix to the first two columns of $\mathbf{\Phi}'$ in a least squares sense. This approach is very closely related to the solution to the orthogonal Procrustes problem (appendix C.7.3).

We then compute the last column $[\omega_{13}, \omega_{23}, \omega_{33}]^T$ of the rotation matrix by taking the cross product of the first two columns: this guarantees a vector that is

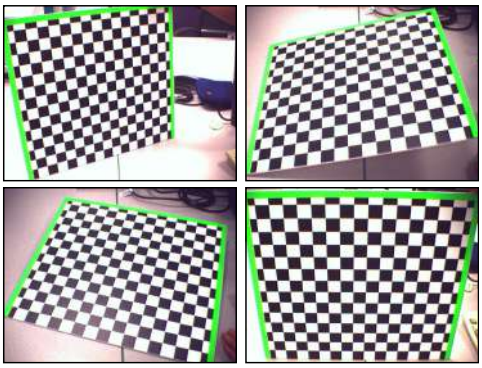

Figure 15.10 Calibration from a plane. It is considerably easier to make a 2D calibration target than to machine an accurate 3D object. Consequently, calibration is usually based on viewing planes such as this checkerboard. Unfortunately, a single view of a plane is not sufficient to uniquely determine the intrinsic parameters. Therefore cameras are usually calibrated using several images of the same plane, where the plane has a different pose relative to the camera in each image.

also length one and is perpendicular to the first two columns, but the sign may still be wrong: we test the determinant of the resulting rotation matrix $\boldsymbol{\Omega}$, and if it is -1, we multiply the last column by -1.

We can now estimate the scaling factor λ' by taking the average of the scaling factors between these six elements

$$\lambda' = \frac{\sum_{m=1}^{3}\sum_{n=1}^{2}\phi'_{mn}/\omega_{mn}}{6}, \tag{15.47}$$

and this allows us to estimate the translation vector as $\boldsymbol{\tau} = [\phi'_{13}, \phi'_{23}, \phi'_{33}]^T/\lambda'$. The result of this algorithm is a very good initial estimate of the extrinsic matrix $[\boldsymbol{\Omega}, \boldsymbol{\tau}]$, which can be improved by optimizing the correct objective function (equation 15.42).

15.4.2 Problem 2: learning intrinsic parameters

In this section, we revisit the problem of learning the intrinsic parameters of the camera. This is known as *camera calibration* (figure 15.9). In section 14.5, we developed a method based on viewing a special 3D calibration target. In practice, it is hard to manufacture a three-dimensional object with easy-to-find visual features at precisely known locations. However, it is easy to manufacture a 2D object of this kind. For example, it is possible to print out a checkerboard pattern and attach this to a flat surface (figure 15.10). Unfortunately, a single view of a 2D calibration object is not sufficient to uniquely identify the intrinsic parameters. However, observing the same pattern from several different viewpoints does suffice.

Algorithm 15.7

Hence, the calibration problem can be reformulated as follows: given a planar object, with I distinct 3D points $\{\mathbf{w}_i\}_{i=1}^{I}$ on the surface and the corresponding projections in J images $\{\mathbf{x}_{ij}\}_{i=1,j=1}^{I,J}$, establish the intrinsic parameters in the form of the intrinsic matrix $\boldsymbol{\Lambda}$:

$$\hat{\boldsymbol{\Lambda}} = \underset{\boldsymbol{\Lambda}}{\operatorname{argmax}}\left[\max_{\boldsymbol{\Omega}_{1\ldots J},\boldsymbol{\tau}_{1\ldots J}}\left[\sum_{i=1}^{I}\sum_{j=1}^{J}\log\left[Pr(\mathbf{x}_{ij}|\mathbf{w}_i,\boldsymbol{\Lambda},\boldsymbol{\Omega}_j,\boldsymbol{\tau}_j)\right]\right]\right]. \tag{15.48}$$

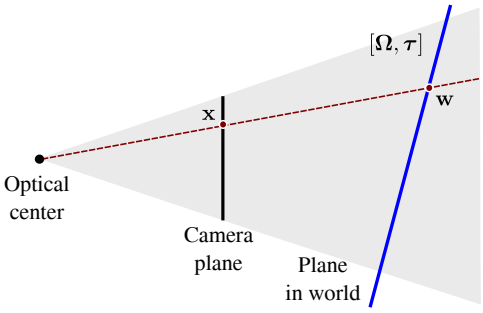

Figure 15.11 Inferring a 3D position relative to the camera. When the object is planar (blue line), there is a one to one mapping between points $\mathbf{x}$ in the image and points $\mathbf{w}$ on the plane. If we know the intrinsic matrix, and the rotation and translation of the plane relative to the camera, we can infer the 3D position $\mathbf{w}$ from the observed 2D image point $\mathbf{x}$.

A simple approach to this problem is to use a coordinate ascent technique in which we alternately

- estimate the J extrinsic matrices relating the object frame of reference to the camera frame of reference in each of the J images,

$$\hat{\boldsymbol{\Omega}}_j, \hat{\boldsymbol{\tau}}_j = \underset{\boldsymbol{\Omega}_j, \boldsymbol{\tau}_j}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \log\left[Pr(\mathbf{x}_{ij} | \mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j) \right] \right], \tag{15.49}$$

  using the method of the previous section and then,

- estimate the intrinsic parameters using a minor variation of the method described in section 14.5:

$$\hat{\boldsymbol{\Lambda}} = \underset{\boldsymbol{\Lambda}}{\operatorname{argmax}} \left[\sum_{i=1}^{I} \sum_{j=1}^{J} \log\left[Pr(\mathbf{x}_{ij} | \mathbf{w}_i, \boldsymbol{\Lambda}, \boldsymbol{\Omega}_j, \boldsymbol{\tau}_j) \right] \right]. \tag{15.50}$$

As for the original calibration method (section 14.5), a few iterations of this procedure will yield a useful initial estimate of the intrinsic parameters, and these can be improved by directly optimizing the true objective function (equation 15.48). It should be noted that this method is quite inefficient and is described for pedagogical reasons; it is easy both to understand and to implement. A modern implementation would use a more sophisticated technique to find the intrinsic parameters (see Hartley & Zisserman 2004).

15.4.3 Problem 3: inferring 3D position relative to camera

Given a calibrated camera (i.e., camera with known $\boldsymbol{\Lambda}$) that is known to be related to a planar scene by extrinsic parameters $\boldsymbol{\Omega}, \boldsymbol{\tau}$, find the 3D point $\mathbf{w}$ that is responsible for a given 2D position $\mathbf{x}$ in the image.

When the scene is planar, there is normally a one-to-one relationship between points in the 3D world and points in the image. To compute the 3D point corresponding to a point $\mathbf{x}$, we initially infer the position $\mathbf{w} = [u, v, 0]^T$ on the plane. We exploit our knowledge of the intrinsic and extrinsic parameters to compute the homography $\boldsymbol{\Phi}$ mapping from points in the world to points in the image,

$$\mathbf{T} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix} = \begin{bmatrix} \phi_x & \gamma & \delta_x \\ 0 & \phi_y & \delta_y \\ 0 & 0 & D \end{bmatrix} \begin{bmatrix} \omega_{11} & \omega_{12} & \tau_x \\ \omega_{21} & \omega_{22} & \tau_y \\ \omega_{31} & \omega_{32} & \tau_z \end{bmatrix}. \tag{15.51}$$

We then infer the coordinates $\mathbf{w} = [u, v, 0]^T$ on the plane by inverting this transformation

$$\tilde{\mathbf{w}} = \mathbf{T}^{-1}\tilde{\mathbf{x}}. \tag{15.52}$$

Finally, we transfer the coordinates back to the frame of reference of the camera to give

$$\mathbf{w}' = \boldsymbol{\Omega}\mathbf{w} + \boldsymbol{\tau}. \tag{15.53}$$

15.5 Transformations between images

So far, we have considered transformations between a plane in the world and its image in the camera. We now consider two cameras viewing the same planar scene. The one-to-one mapping from positions on the plane to the positions in the first camera can be described by a homography. Similarly, the one-to-one mapping from positions on the plane to the positions in the second camera can be described by a second homography. It follows that there is a one-to-one mapping from the position in the first camera to the position in the second camera. This can also be described by a homography. The same logic follows for the other transformation types. Consequently, it is very common to find pairs of real-world images that are geometrically related by one of these transformations. For example, the image of the photograph in figure 15.5a is related by a homography to that in 15.5b.

Let us denote the 3×3 matrix mapping points on the plane to points in the first image by $\mathbf{T}_1$. Similarly, we denote the 3×3 matrix mapping points on the plane to points in the second image by $\mathbf{T}_2$. To map from image 1 to image 2, we first apply the transformation from image 1 to the plane itself. By the argument of the previous section, this is $\mathbf{T}_1^{-1}$. Then we apply the transformation from the plane to image 2, which is $\mathbf{T}_2$. The mapping $\mathbf{T}_3$ from image 1 to image 2 is the concatenation of these operations and is hence $\mathbf{T}_3 = \mathbf{T}_2\mathbf{T}_1^{-1}$ (figure 15.12).

15.5.1 Geometric properties of the homography

We've seen that transformations between planes in the world and the image plane are described by homographies, and so are transformations between multiple images of a real-world plane. There is another important family where the images are related to one another by the homography Recall that the homography mapping point $\mathbf{x}_1$ to point $\mathbf{x}_2$ is linear in homogeneous coordinates:

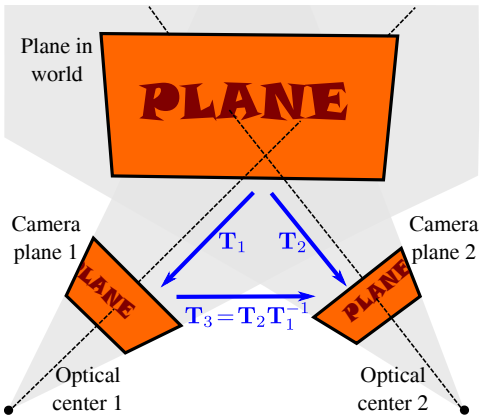

Figure 15.12 Transformations between images. Two cameras view the same planar scene. The relations between the 2D points on this plane and the two images are captured by the 3×3 transformation matrices $\mathbf{T}_1$ and $\mathbf{T}_2$, respectively. It follows that the transformation from the first image to the points on the plane is given by $\mathbf{T}^{-1}$. We can compute the transformation $\mathbf{T}_3$ from the first image to the second image by transforming from the first image to the plane and then transforming from the plane to the second image, giving the final result $\mathbf{T}_3 = \mathbf{T}_2\mathbf{T}_1^{-1}$.

$$\lambda \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \phi_{11} & \phi_{12} & \phi_{13} \\ \phi_{21} & \phi_{22} & \phi_{23} \\ \phi_{31} & \phi_{32} & \phi_{33} \end{bmatrix} \begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix}. \tag{15.54}$$

The homogeneous coordinates represent 2D points as directions or rays in a 3D space (figure 14.11). When we apply a homography to a set of 2D points, we can think of this as applying a linear transformation (rotation, scaling, and shearing) to a bundle of rays in 3D. The positions where the transformed rays strike the plane at $w = 1$ determine the final 2D positions.

We could yield the same results by keeping the rays fixed and applying the inverse transformation to the plane so that it cuts the rays in a different way. Since any plane can be mapped to any other plane by a linear transformation, it follows that the images created by cutting a ray bundle with different planes are all related to one another by homographies (figure 15.13). In other words, the images seen by different cameras *with the pinhole in the same place* are related by homographies. So, for example, if a camera zooms (the focal length increases), then the images before and after the zoom are related by a homography.

Problem 15.10
Problem 15.11

This relationship encompasses an important special case (figure 15.14). If the camera rotates *but does not translate*, then the image plane still intersects the same set of rays. It follows that the projected points $\mathbf{x}_1$ before the rotation and the projected points $\mathbf{x}_2$ after the rotation are related by a homography. It can be shown that the homography $\mathbf{\Phi}$ mapping from image 1 to image 2 is given by

$$\mathbf{\Phi} = \mathbf{\Lambda}\mathbf{\Omega}_2\mathbf{\Lambda}^{-1}, \tag{15.55}$$

where $\mathbf{\Lambda}$ is the intrinsic matrix and $\mathbf{\Omega}_2$ is the rotation matrix that maps the coordinate system of the second camera to the first. This relationship is exploited when we stitch together images to form panoramas (section 15.7.2).

In conclusion, the homography maps between:

- points on a plane in the world and their positions in an image,

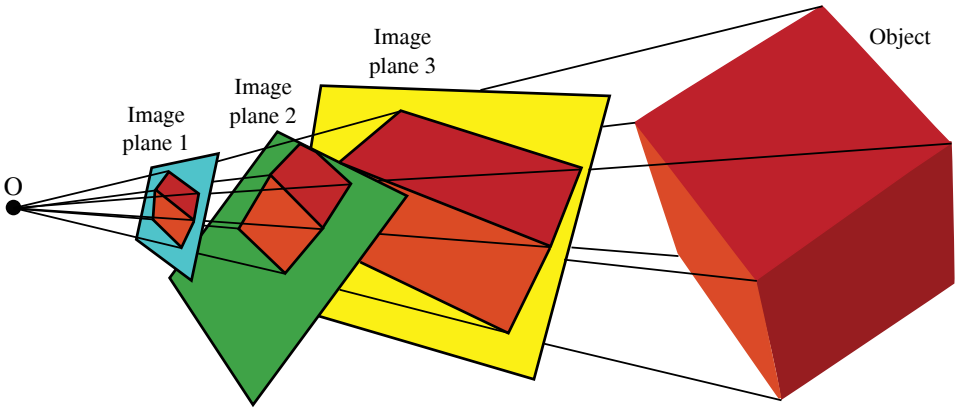

Figure 15.13 Geometric interpretation of homography. A ray bundle is formed by connecting rays from an optical center to points on a real-world object (the cube). A set of planes cut the ray bundle, forming a series of images of the cube. Each of these images is related to every other by a homography.

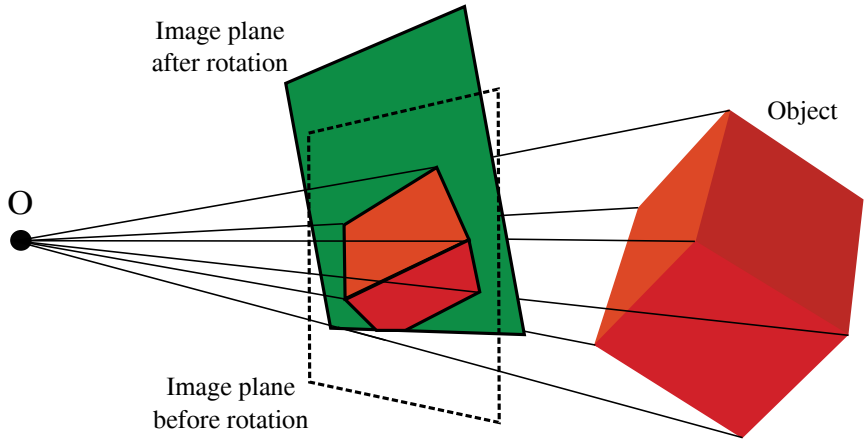

Figure 15.14 Images under pure camera rotation. When the camera rotates but does not translate, the bundle of rays remains the same, but is cut by a different plane. It follows that the two images are related by a homography.

- points in two different images of the same plane, and
- two images of a 3D object where the camera has rotated but not translated.

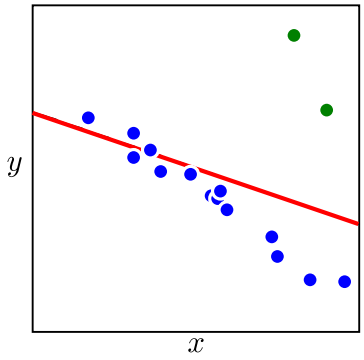

Figure 15.15 Motivation for random sample consensus (RANSAC). The majority of this data set (blue points) can be explained well by a linear regression model, but there are two outliers (green points). Unfortunately, if we fit the linear regression model to all of this data, the mean prediction (red line) is dragged toward the outliers and no longer describes the majority of the data well. The RANSAC algorithm circumvents this problem by establishing which data points are outliers and fitting the model to the remaining data.

15.5.2 Computing transformations between images

In the previous sections we have argued that it is common for two images to be related to one another by a homography. If we denote the points in image 1 by $\mathbf{x}_i$ and their corresponding positions in image 2 as $\mathbf{y}_i$, then we could for example describe the mapping as a homography

$$Pr(\mathbf{x}_i|\mathbf{y}_i) = \text{Norm}_{\mathbf{x}_i}\left[\mathbf{hom}[\mathbf{y}_i, \mathbf{\Phi}], \sigma^2\mathbf{I}\right]. \tag{15.56}$$

This method ascribes all of the noise to the first image and it should be noted that it is not quite correct: we should really build a model that explains both sets of image data with a set of hidden variables representing the original 3D points so that the estimated point positions in each image are subject to noise. Nonetheless, the model in equation 15.56 works well in practice. It is possible to learn the parameters using the technique of section 15.2.

15.6 Robust learning of transformations

We have discussed transformation models that can be used either to (i) map the positions of points on a real-world plane to their projections in the image or (ii) map the positions of points in one image to their corresponding positions in another. Until now, we have assumed that we know a set of corresponding points from which to learn the parameters of the transformation model. However, establishing these correspondences automatically is a challenging task in itself.

Let us consider the case where we wish to compute a transformation between two images. A simple method to establish correspondences would be to compute interest points in each image and to characterize the region around each point using a region descriptor such as the SIFT descriptor (section 13.3.2). We could then greedily associate points based on the similarity of the region descriptors (as in figure 15.17c-d). Depending on the scene, this is likely to produce a set of matches that are $70-90\%$ correct. Unfortunately the remaining erroneous

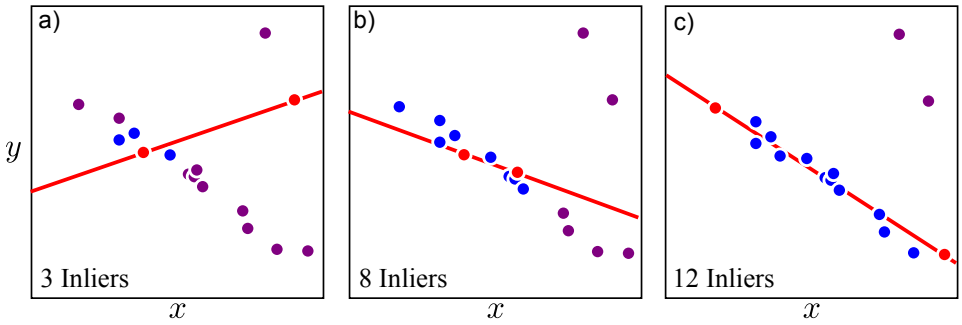

Figure 15.16 RANSAC procedure. a) We select a random minimal subset of points to fit the line (red points). We fit the line to these points and count how many of the other points agree with this solution (blue points). These are termed inliers. Here there are only three inliers. b,c) This procedure is repeated with different minimal subsets of points. After a number of iterations, we choose the fit that had the most inliers. We refit the line using only the inliers from this fit.

correspondences (which we will call *outliers*) can severely hamper our ability to compute the transformation between the images. To cope with this problem, we need robust learning methods.

15.6.1 RANSAC

Random sample consensus or *RANSAC* is a general method for fitting models to data where the data are corrupted by outliers. These outliers violate the assumptions of the underlying probability model (usually a normal distribution) and can cause the estimated parameters to deviate significantly from their correct values.

For pedagogical reasons, we will describe RANSAC using the example of linear regression. We will subsequently return to the problem of learning parameters in transformation models. The linear regression model is

$$Pr(y|x) = \text{Norm}_y[ax + b, \sigma^2] \tag{15.57}$$

and was discussed at length in section 8.1. Figure 15.15 shows an example of the predicted mean for y when we fit the model to data with two outliers. The fit has been unduly influenced by the outliers and no longer describes the data well.

The goal of RANSAC is to identify which points are outliers and to eliminate them from the final fit. This is a chicken and egg problem: if we had the final fit, then it would be easy to identify the outliers (they are not well described by the model), and if we knew which points were outliers, it would be easy to compute the final fit.

RANSAC works by repeatedly fitting models based on random subsets of the data. The hope is that sooner or later, there will be no outliers in the chosen subset,

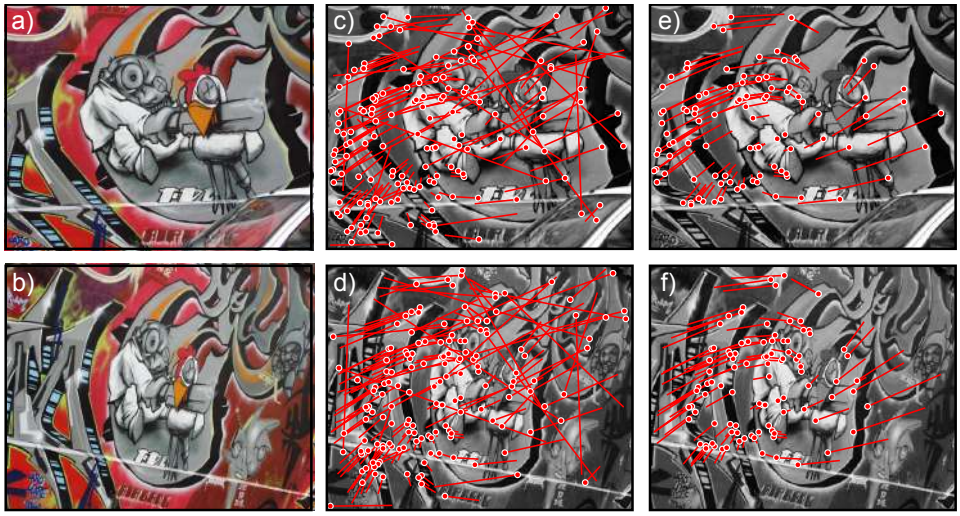

Figure 15.17 Fitting a homography with RANSAC. a,b) Original images of a textured plane. c,d) 162 strongest matches selected by greedy method. In each case the associated line travels from the interest point to the position of its match in the other image. These matches are clearly polluted by outliers. e,f) Model fitted from 102 inliers identified by applying 100 iterations of RANSAC. These matches form a coherent pattern as they are all described by a homography.

and so we will fit a good model. To enhance the probability of this happening, RANSAC chooses subsets of the minimal size required to uniquely fit the model. For example, in the case of the line, it would choose subsets of size two.

Having chosen a minimal subset of data points and fitted the model, RANSAC assesses its quality. It does this by classifying points as inliers or outliers. This requires some knowledge about the expected amount of variation around the true model. For linear regression this means we need some prior knowledge of the variance parameter σ^2. If a data point exceeds the expected variation (perhaps two standard deviations from the mean), then it is classified as an outlier. Otherwise, it is an inlier. For each minimal subset of data we count the number of inliers.

We repeat this procedure a number of times: on each iteration we choose a random minimal subset of points, fit a model, and count the number of data points that agree (the inliers). After a predetermined number of iterations, we then choose the model that had the most inliers and re-fit the model from these alone.

The complete RANSAC algorithm hence proceeds as follows (figure 15.16)

1. Randomly choose a minimal subset of data.
2. Use this subset to estimate the parameters.
3. Compute the number of inliers for this model.
4. Repeat steps 1-3 a fixed number of times.

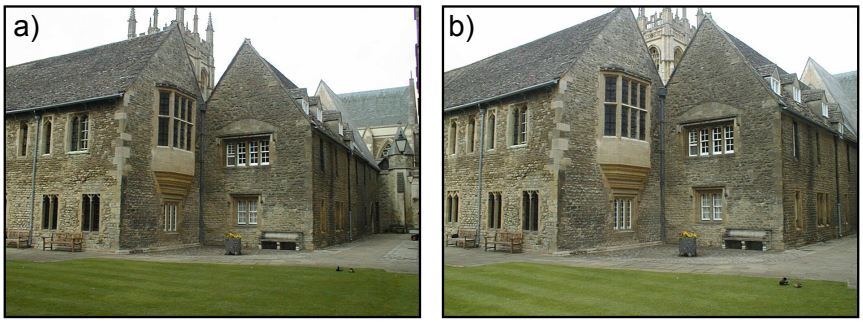

Figure 15.18 Piecewise planarity. a) Although this scene is clearly not well described by a plane, it is well described by a set of planes. b) Consequently, the mapping to this second image of the same scene can be described by a set of homographies. Images from Oxford Colleges dataset.

5. Re-estimate model using inliers from the best fit.

If we know the degree to which our data are polluted with outliers, it is possible to estimate the number of iterations that provide any given chance of finding the correct answer.

Problem 15.12

Now let us return to the question of how to apply the model to fitting geometric transformations. We will use the example of a homography (equation 15.56). Here we repeatedly choose subsets of hypothesized matches between the two images. The size of the subset is chosen to be four as this is the minimum number of pairs of points that are needed to uniquely identify the eight degrees of freedom of the homography.

Algorithm 15.8

For each subset, we count the number of inliers by evaluating equation 15.56 and selecting those where the likelihood exceeds some pre-determined value. In practice, this means measuring the distance between the points $\mathbf{x}_i$ in the first image and the mapped position $\mathbf{hom}[\mathbf{y}, \boldsymbol{\Phi}]$ of the points from the second image. After repeating this many times, we identify the trial with the most inliers (figure 15.17) and recompute the model from these inliers alone.

Problem 15.13

15.6.2 Sequential RANSAC

Now let us look at a more challenging task. So far, we assumed that one set of points could be mapped to another using a single transformation model. However, many scenes containing man-made objects are piecewise planar (figure 15.18). It follows that images of such scenes are related by piecewise homographies. In this section, we will consider methods for fitting this type of model.

Algorithm 15.9

The first approach is *sequential RANSAC*. The idea is simple: we fit a single homography to the scene using the RANSAC method. In principle this will correctly fit a model to one of the planes in the image and reject all the other points

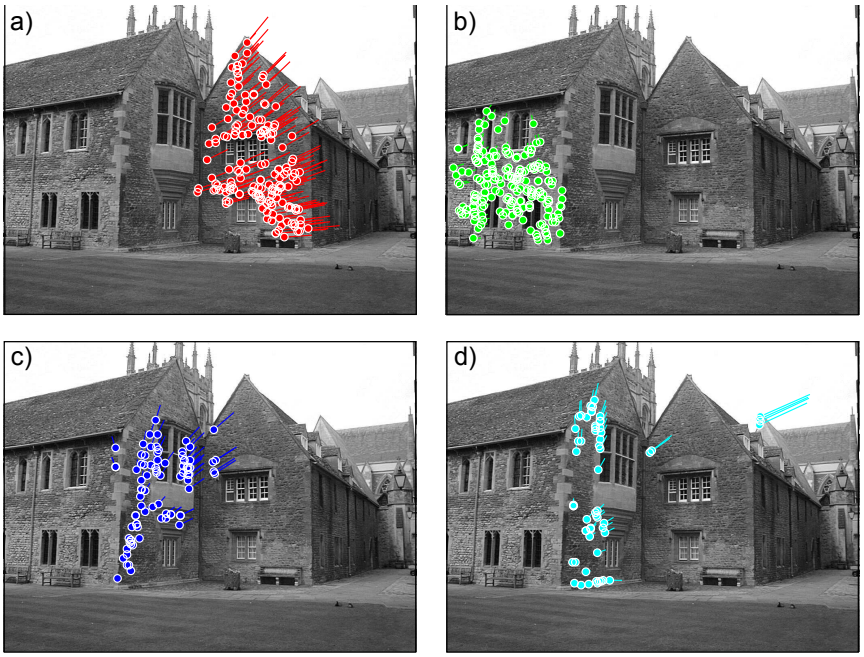

Figure 15.19 Sequential fitting of homographies mapping between images in figure 15.18 using RANSAC. a) Running the RANSAC fitting procedure once identifies a set of points that all lie on the same plane in the image. b) Running the RANSAC procedure again on the points that were not explained by the first plane identifies a second set of points that lie on a different plane. c) Third iteration. the plane discovered does not correspond to a real plane in the scene. d) Fourth iteration. The model erroneously associates parts of the image which are quite separate. This sequential approach fails for two reasons. First, it is greedy and cannot recover from earlier errors. Second, it does not encourage spatial smoothness (nearby points should belong to the same model).

as outliers. We then remove the points that belong to this plane (i.e., the inliers to the final homography) and repeat the procedure on the remaining points. Ideally, each iteration will identify a new plane in the image.

Unfortunately, this method does not work well in practice (figure 15.19) for two reasons. First, the algorithm is greedy: any matches that are erroneously incorporated into, or missed by one of the earlier models cannot be correctly assigned later. Second, the model has no notion of spatial coherence, ignoring the intuition that nearby points are more likely to belong to the same plane.

15.6.3 PEaRL

The *Propose, Expand and Re-Learn* or *PEaRL* algorithm solves both of these problems. In the 'propose' stage, K hypothetical models are generated where K is usually of the order of several thousand. This can be achieved by using a RANSAC type procedure in which minimal subsets of points are chosen, a model is fitted, and the inliers are counted. This is done repeatedly until we have several thousand models $\{\boldsymbol{\Phi}_k\}_{k=1}^{K}$ that each have a reasonable degree of support. Algorithm 15.10

In the 'expand' stage, we model the assignment of matched pairs to the proposed models as a multi-label Markov random field. We associate a label $\mathbf{l}_i \in [1, 2, \ldots, K]$ to each match $\{\mathbf{x}_i, \mathbf{y}_i\}$ where the value of the label determines which one of the K models is present at this point. The likelihood of each data pair $\{\mathbf{x}_i, \mathbf{y}_i\}$ under the k^{th} model is given by

$$Pr(\mathbf{x}_i|\mathbf{y}_i, l_i = k) = \text{Norm}_{\mathbf{x}_i}\left[\mathbf{hom}[\mathbf{y}_i, \boldsymbol{\Phi}_k], \sigma^2\mathbf{I}\right]. \tag{15.58}$$

To incorporate spatial coherence, we choose a prior over the labels that encourages neighbors to take similar values. In particular, we choose an MRF with a Potts model potential

$$Pr(\mathbf{l}) = \frac{1}{Z}\exp\left[-\sum_{i,j\in\mathcal{N}_p} w_{ij}\delta[l_i - l_j]\right], \tag{15.59}$$

where Z is the constant partition function, w_{ij} is a weight associated with the pair of matches i and j, and $\delta[\bullet]$ is a function that returns 1 when the argument is zero and returns 0 otherwise. The neighborhood $\mathcal{N}_i$ of each point must be chosen in advance. One simple technique is to compute the K-nearest neighbors for each point in the first image and declare two points to be neighbors if either is in the other's set of closest neighbors. The weights w_{ij} are chosen so that points that are close in the image are more tightly coupled than points that are distant.

The goal of the 'expand' stage is then to infer the labels l_i and hence the association between models and data points. This can be accomplished using the alpha expansion algorithm (section 12.4.1). Finally, having associated each data point with one of the models, we move to the 'relearn' stage. Here, we re-estimate the parameters of each model based on the data that was associated with it. The 'expand' and 'relearn' stages are iterated until no further progress is made. At the end of this process, we throw away any models that do not have sufficient support in the final solution (figure 15.20).

15.7 Applications

In this section we present two examples of the techniques in this chapter. First, we discuss augmented reality in which we attempt to render an object onto a plane in the scene. This application exploits the fact that there is a homography between the image of the plane and the original object surface, and uses the method of

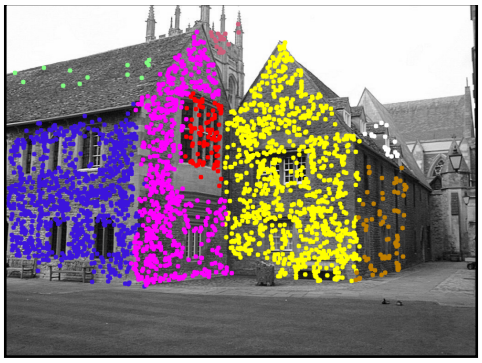

Figure 15.20 Results of the PEaRL algorithm. It formulates the problem as inference in a multi-label MRF. The MRF label associated with each matching pair denotes the index of one of a set of possible models. Inferring these labels is alternated with refining the parameters of the proposed models. The colored points denote different labels in the final solution. The algorithm has successfully identified many of the surfaces in the scene. Adapted from Isack & Boykov (2012). ©2012 Springer.

section 15.4.1 to decompose this homography to find the relative position of the camera and the plane. Second, we discuss creating visual panoramas. The method exploits the fact that multiple images taken from a camera that has rotated but not translated are all related by homographies.

15.7.1 Augmented reality tracking

Figure 15.1 shows an example of augmented reality tracking. To accomplish this, the following steps were taken. First, the four corners of the marker are found as in figure 15.1c; the marker was designed so that the corners can be easily identified using a set of image processing operations. In brief, the image is thresholded, and then connected dark regions are found. The pixels around the edge of each region are identified. Then four 2D lines are fit to the edge pixels using sequential RANSAC. Regions that are not well explained by four lines are discarded. The only remaining region in this case is the square marker.

The positions where the fitted lines intersect provides four points $\{\mathbf{x}_i\}_{i=1}^4$ in the image, and we know the corresponding positions $\{\mathbf{w}_i\}_{i=1}^4$ in cm that occur on the surface of the planar marker. It is assumed that the camera is calibrated, and so we know the intrinsic matrix $\boldsymbol{\Lambda}$. We can now compute the extrinsic parameters $\boldsymbol{\Omega}, \boldsymbol{\tau}$ using the algorithm from section 15.4.1.

To render the graphical object, we first set up a viewing frustum (the graphics equivalent of the 'camera') so that it has the same field of view as specified by the intrinsic parameters. Then we render the model from the appropriate perspective using the extrinsic parameters as the *modelview matrix*. The result is that the object appears to be attached rigidly to the scene.

In this system, the points on the marker were found using a sequence of image-processing operations. However, this method is rather outdated. It is now possible to reliably identify natural features on an object so there is no need to use a marker with any special characteristics. One way to do this is to match SIFT features between a reference image of the object and the current scene. These interest point descriptors are invariant to image scaling and rotation, and for textured objects it is usually possible to generate a high percentage of correct matches. RANSAC is

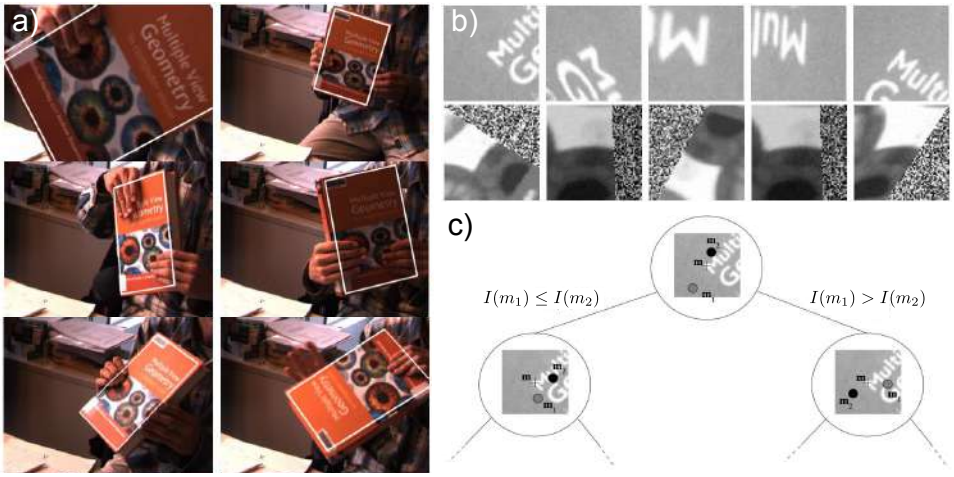

Figure 15.21 Robust tracking using keypoints. a) Lepetit *et al.* (2005) presented a system that automatically tracked objects such as this book. b) In the learning stage, the regions around the keypoints were subjected to a number of random affine transformations. c) Keypoints in the image were classified as belonging to a known keypoint on the object, using a tree-based classifier that compared the intensity at nearby points. Adapted from Lepetit *et al.* (2005). ©2005 IEEE.

used to eliminate any mis-matched pairs.

However, SIFT features are themselves relatively slow to compute. Lepetit *et al.* (2005) described a system that applies machine learning techniques to identify objects at interactive speeds (figure 15.21). They first identify K stable keypoints (e.g., Harris corners) on the object to be tracked. They then create a training set for each keypoint by subjecting the region around it to a large number of random affine transformations. Finally, they train a multi-class classification tree that takes a new keypoint and assigns it to match one of the K original points. At each branch of the tree, the decision is made by a pairwise intensity comparison. This system works very quickly and reliably matches most of the feature points. Once more, RANSAC can be used to eliminate any erroneous matches.

15.7.2 Visual panoramas

A second application of the ideas of this chapter is the computation of visual panoramas. Recall that a set of pictures that are taken by rotating a camera about the optical center are all related by homographies. Hence, if we take pictures of this kind that are partially overlapping, it is possible to map them all into one large image. This process is known as *image mosaicing*.

Problem 15.14

An example is shown in figure 15.22. This was constructed by placing the second

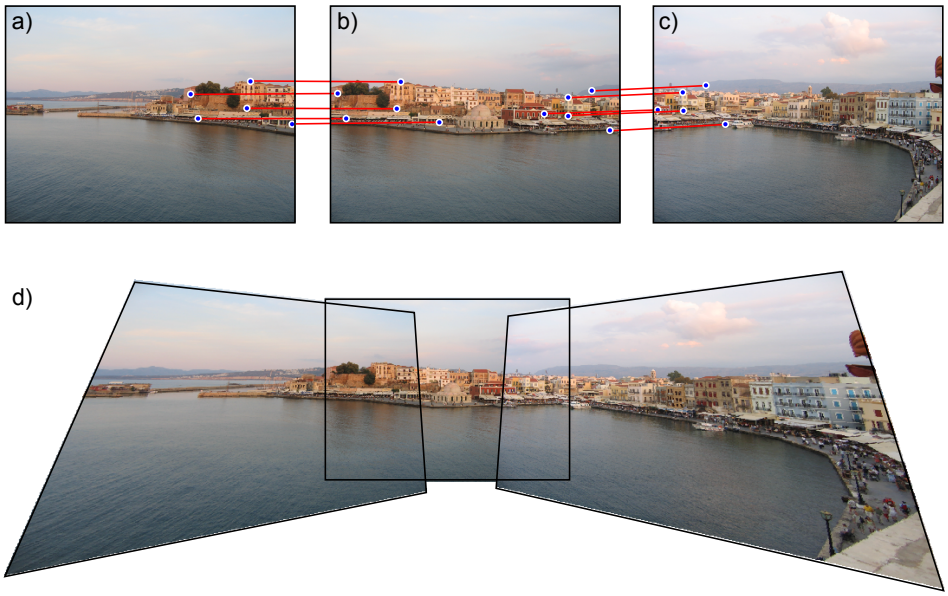

Figure 15.22 Computing visual panoramas. a-c) Three images of the same scene where the camera has rotated but not translated. Five matching points have been identified by hand between each pair. d) A panorama can be created by mapping the first and third images into the frame of reference of the second image.

(middle) photo into the center of a much larger empty image. Then 5 matches were identified by hand between this expanded second image and the first image and the homography from the expanded second image to the first image was computed. For each empty pixel in the expanded second image we compute the position in the first image using this homography. If this falls within the image boundaries, we copy the pixel value.

This process can be completely automated by finding features and fitting a homography to each pair of images using a robust technique such as RANSAC. In a real system, the final result is typically projected onto a cylinder and unrolled to give a more visually pleasing result.

Discussion

This chapter has presented a number of important ideas. First, we discussed a family of transformations and how each can be related to a camera viewing the scene under special conditions. These transformations are used widely within machine vision and we will see them exploited in chapters 17, 18 and 19. Second, we

discussed a more practical method for camera calibration based on viewing a plane at a number of different orientations. Finally, we have also presented RANSAC, which is a robust method to fit models, even in the presence of noisy observations.

Notes

Transformations: For more information concerning the hierarchy of 2D transformations, consult Hartley & Zisserman (2004). The closed form solutions for the rotation and similarity transformations are special cases of *Procrustes problems*. Many more details about this type of problem can be found in Gower & Dijksterhuis (2004). The direct linear transformation algorithm for estimating the homography dates back to at least Sutherland (1963). Hartley & Zisserman (2004) present a detailed discussion of different objective functions for estimating homographies. In this chapter, we have discussed the estimation of transformations from point matches. However, it is also possible to compute transformations from other geometric primitives. For example, methods exist to compute the homography from matched lines (see problem 15.3), a combination of points and lines (Murino *et al.* 2002), or conics (Sugimoto 2000).

Robust estimation: The RANSAC algorithm is due to Fischler & Bolles (1981). It has spawned many variants, the most notable of which is MLESAC (Torr & Zisserman 2000) which puts this fitting method on a sound probabilistic footing (see problem 15.13 for a related model). Chum *et al.* (2005) and Frahm & Pollefeys (2006) present variants of RANSAC that can cope with degenerate data (where the model is non-unique). Torr (1998) and Vincent & Laganiere (2001) used RANSAC to sequentially estimate multiple geometric entities. Both Raguram *et al.* (2008) and Choi *et al.* (2009) present recent quantitative comparisons of variations of the RANSAC algorithm. The PEaRL algorithm is due to Isack & Boykov (2012). In the original paper, they also include an extra cost which encourages parsimony (to describe the data with as few models as possible) Other approaches to robust estimation include (i) the use of long-tailed distributions such as the t-distribution (section 7.5), (ii) M-estimators (Huber 2009) which replace the least-squares criterion with another function that penalizes large deviations less stringently, and (iii) the self explanatory least median of squares regression (Rousseeuw 1984).

Augmented reality: Pose estimation methods for augmented reality initially relied on detecting special patterns in the scene known as fiducial markers. Early examples used circular patterns (e.g. Cho *et al.* 1998; State *et al.* 1996) but these were largely supplanted by square markers (e.g., Rekimoto 1998; Kato *et al.* 2000; Kato & Billinghurst 1999; Koller *et al.* 1997). The system described in the text is ARToolkit (Kato & Billinghurst 1999; Kato *et al.* 2000) and can be downloaded from http://www.hitl.washington.edu/artoolkit/.

Other systems have used 'natural image features.' For example, Harris (1992) estimated the pose of an object using line segments. Simon *et al.* (2000) and Simon & Berger (2002) estimated the pose of planes in the image using the results of a corner detector. More information about computing the pose of a plane can be found in Sturm (2000).

More recent systems have used interest point detectors such as SIFT features which are robust to changes in illumination and pose (e.g., Skrypnyk & Lowe 2004). To increase the speed of systems, features are matched using machine learning techniques (Lepetit & Fua 2006; Özuysal *et al.* 2010) and current systems can now operate at interactive speeds on mobile hardware (Wagner *et al.* 2008). A review of methods to estimate and track the pose of rigid objects can be found in Lepetit & Fua (2005).

Calibration from a plane: Algorithms for calibration from several views of a plane can be found in Sturm & Maybank (1999) and Zhang (2000). They are now used much more frequently than calibration based on 3D objects, for the simple reason that accurate 3D objects are harder to manufacture.

Image mosaics: The presented method for computing a panorama by creating a mosaic of images is naïve in a number of ways. First, it is more sensible to explicitly estimate

the rotation matrix and calibration parameters rather than the homography (Szeliski & Shum 1997; Shum & Szeliski 2000; Brown & Lowe 2007). Second, the method that we describe projects all of the images onto a single plane, but this does not work well when the panorama is too wide as the images become increasingly distorted. A more sensible approach is to project images onto a cylinder (Szeliski 1996; Chen 1995) which is then unrolled and displayed as an image. Third, the method to blend together images is not discussed. This is particularly important if there are moving objects in the image. A good review of these and other issues can be found in Szeliski (2006) and chapter 9 of Szeliski (2010).

Problems

Problem 15.1 The 2D point $\mathbf{x}_2$ is created by a rotating point $\mathbf{x}_1$ using the rotation matrix $\boldsymbol{\Omega}_1$ and then translating it by the translation vector $\boldsymbol{\tau}_1$ so that

$$\mathbf{x}_2 = \boldsymbol{\Omega}_1\mathbf{x}_1 + \boldsymbol{\tau}_1.$$

Find the parameters $\boldsymbol{\Omega}_2$ and $\boldsymbol{\tau}_2$ of the inverse transformation

$$\mathbf{x}_1 = \boldsymbol{\Omega}_2\mathbf{x}_2 + \boldsymbol{\tau}_2$$

in terms of the original parameters $\boldsymbol{\Omega}_1$ and $\boldsymbol{\tau}_1$.

Problem 15.2 A 2D line can be as expressed as $ax + by + c = 0$ or in homogeneous terms

$$\mathbf{l}\tilde{\mathbf{x}} = 0,$$

where $\mathbf{l} = [a, b, c]$. If points are transformed so that

$$\tilde{\mathbf{x}}' = \mathbf{T}\tilde{\mathbf{x}},$$

what is the equation of the transformed line?

Problem 15.3 Using your solution from problem 15.2, develop a linear algorithm for estimating a homography based on a number of matched lines between the two images (i.e., the analogue of the DLT algorithm for matched lines).

Problem 15.4 A conic (see problem 14.6) is defined by

$$\tilde{\mathbf{x}}^T\mathbf{C}\tilde{\mathbf{x}} = 0,$$

where $\mathbf{C}$ is a 3×3 matrix. If the points in the image undergo the transformation

$$\tilde{\mathbf{x}}' = \mathbf{T}\tilde{\mathbf{x}},$$

then what is the equation of the transformed conic?

Problem 15.5 All of the 2D transformations in this chapter (Euclidean, similarity, affine, projective) have 3D equivalents. For each class write out the 4×4 matrix that describes the 3D transformation in homogeneous coordinates. How many independent parameters does each model have?

Problem 15.6 Devise an algorithm to estimate a 3D affine transformation based on two sets of matching 3D points. What is the minimum number of points required to get a unique estimate of the parameters of this model?

Problem 15.7 A 1D affine transformation acts on 1D points x as $x' = ax + b$. Show that the ratio of two distances is *invariant* to a 1D affine transformation so that

$$I = \frac{x_1 - x_2}{x_2 - x_3} = \frac{x'_1 - x'_2}{x'_2 - x'_3}.$$

Problem 15.8 A 1D projective transform acts on 1D points x as $x' = (ax + b)/(cx + d)$. Show that the *cross-ratio* of distances is *invariant* to a 1D projective transformation so that

$$I = \frac{(x_3 - x_1)(x_4 - x_2)}{(x_3 - x_2)(x_4 - x_1)} = \frac{(x'_3 - x'_1)(x'_4 - x'_2)}{(x'_3 - x'_2)(x'_4 - x'_1)}.$$

It was proposed at one point to exploit this type of invariance to recognize planar objects under different transformations (Rothwell *et al.* 1995). However, this is rather impractical as it assumes that we can identify a number of the points on the object in a first place.

Problem 15.9 Show that equation 15.36 follows from equation 15.35.

Problem 15.10 A camera with intrinsic matrix $\boldsymbol{\lambda}$ and extrinsic parameters $\boldsymbol{\Omega} = \mathbf{I}, \boldsymbol{\tau} = \mathbf{0}$ takes an image and then rotates to a new position $\boldsymbol{\Omega} = \boldsymbol{\Omega}_1, \boldsymbol{\tau} = \mathbf{0}$ and takes a second image. Show that the homography relating these two images is given by

$$\boldsymbol{\Phi} = \boldsymbol{\Lambda}\boldsymbol{\Omega}_1\boldsymbol{\Lambda}^{-1}.$$

Problem 15.11 Consider two images of the same scene taken from a camera that rotates, but does not translate between taking the images. What is the minimum number of point matches required to recover a 3D rotation between two images taken using a camera where the intrinsic matrix is known?

Problem 15.12 Consider the problem of computing a homography from point matches that include outliers. If 50% of the initial matches are correct, how many iterations of the RANSAC algorithm would we expect to have to run in order to have a 95% chance of computing the correct homography?

Problem 15.13 A different approach to fitting transformations in the presence of outliers is to model the uncertainty as a mixture of two Gaussians. The first Gaussian models the image noise, and the second Gaussian, which has a very large variance, accounts for the outliers. For example, for the affine transformation we would have

$$Pr(\mathbf{x}|\mathbf{w}) = \lambda \text{Norm}_{\mathbf{x}}\left[\mathbf{aff}[\mathbf{w}, \boldsymbol{\Phi}, \boldsymbol{\tau}], \sigma^2\mathbf{I}\right] + (1-\lambda)\text{Norm}_{\mathbf{x}}\left[\mathbf{aff}[\mathbf{w}, \boldsymbol{\Phi}, \boldsymbol{\tau}], \sigma_0^2\mathbf{I}\right] +$$

where λ is the probability of being an inlier, σ^2 is the image noise and σ_0^2 is the large variance that accounts for the outliers. Sketch an approach to learning the parameters $\sigma^2, \boldsymbol{\Phi}, \boldsymbol{\tau}$, and λ of this model. You may assume that σ_0^2 is fixed. Identify a possible weakness of this model.

Problem 15.14 In the description of how to compute the panorama (section 15.7.2), it is suggested that we take each pixel in the central image and transform it into the other images and then copy the color. What is wrong with the alternate strategy of taking each pixel from the other images and transforming them into the central image?