CHAPTER 31

Using Camera Models

31.1 CAMERA CALIBRATION FROM A 3D REFERENCE

Camera calibration involves estimating the intrinsic parameters of the camera, and perhaps lens parameters if needed, from one or more images. There are numerous strategies, all using versions of the following recipe: build a *calibration object*, where the positions of some points (*calibration points*) are known; view that object from one or more viewpoints; obtain the image locations of the calibration points; and solve an optimization problem to recover camera intrinsics and perhaps lens parameters. As one would expect, much depends on the choice of calibration object. If all the calibration points sit on an object, the extrinsics will yield the *pose* (for position and orientation) of the object with respect to the camera. We use a two step procedure: formulate the optimization problem, then find a good starting point.

31.1.1 Formulating the Optimization Problem

The optimization problem is relatively straightforward to formulate. Notation is the main issue. We have N reference points $\mathbf{s}_i = [s_{1,i}, s_{2,i}, s_{3,i}]$ with known position in some reference coordinate system in 3D. The measured location in the image for the i'th such point is $\hat{\mathbf{t}}_i = \left[\hat{t}_{1,i}, \hat{t}_{2,i}\right]$. There may be measurement errors, so the $\hat{\mathbf{t}}_i = \mathbf{t}_i + \xi_i$, where ξ_i is an error vector and $\mathbf{t}_i$ is the unknown true position of the image point. We will assume the magnitude of error does not depend on direction in the image plane (it is *isotropic*), so it is natural to minimize the squared magnitude of the error

$$\sum_i \xi_i^T \xi_i. \tag{31.1}$$

This error is *reprojection error*. The main issue here is writing out expressions for ξ_i in the appropriate coordinates. Write $\mathcal{K}$ for the intrinsic matrix whose u, v'th component will be k_{uv}, and recall this matrix is upper triangular and the bottom right element is 1. Write $\mathcal{T}_e$ for the extrinsic transformation, whose u, v'th component will be e_{uv}. Write

$$\begin{aligned} g_{1,i} &= e_{11}s_{1,i} + e_{12}s_{2,i} + e_{13}s_{3,i} + e_{14} \\ g_{2,i} &= e_{21}s_{1,i} + e_{22}s_{2,i} + e_{23}s_{3,i} + e_{24} \\ g_{3,i} &= e_{31}s_{1,i} + e_{32}s_{2,i} + e_{33}s_{3,i} + e_{34} \end{aligned}$$

for the 3D points transformed by the and

$$\begin{aligned} p_{1,i} &= \frac{k_{11}g_{1,i} + k_{12}g_{2,i} + k_{13}g_{3,i}}{g_{3,3}} \\ p_{2,i} &= \frac{k_{22}g_{1,i} + k_{23}g_{3,i}}{g_{3,i}}. \end{aligned}$$

The reprojection error is then

$$\sum_i \xi_i^T \xi_i = \sum_i (t_{1,i} - p_{1,i})^2 + (t_{2,i} - p_{2,i})^2 \tag{31.2}$$

This is a constrained optimization problem, because $\mathcal{T}_e$ is a Euclidean transformation. The constraints here are

$$\begin{aligned} 1 - \sum_v e_{1v}^2 = 0 \text{ and } 1 - \sum_v e_{2v}^2 = 0 \text{ and } 1 - \sum_v e_{3v}^2 = 0 \\ \sum_v e_{1v} e_{2v} = 0 \text{ and } \sum_v e_{1v} e_{3v} = 0 \text{ and } \sum_v e_{2v} e_{3v} = 0 \end{aligned} .$$

You might just throw this into a constrained optimizer (review Section 21.3), but experience shows good behavior requires a good start point. This can be obtained by a little manipulation, which I work through in the next section. Some readers may prefer to skip this at first (or even higher) reading because it's somewhat specialized, but it shows how the practical application of some tricks worth knowing.

31.1.2 Setting up a Start Point for Calibration

Write $\mathbf{C}_j^T$ for the j'th row of the camera matrix, and $\mathbf{S}_i = [s_{1,i}, s_{2,i}, s_{3,i}, 1]^T$ for homogeneous coordinates representing the i'th point in 3D. Then, assuming no errors in measurement

$$\hat{t}_{1,i} = \frac{\mathbf{C}_1^T \mathbf{S}_i}{\mathbf{C}_3^T \mathbf{S}_i} \text{ and } \hat{t}_{2,i} = \frac{\mathbf{C}_2^T \mathbf{S}_i}{\mathbf{C}_3^T \mathbf{S}_i}, \tag{31.3}$$

which you can rewrite as

$$\mathbf{C}_3^T \mathbf{S}_i \hat{t}_{1,i} - \mathbf{C}_1^T \mathbf{S}_i = 0 \text{ and } \mathbf{C}_3^T \mathbf{S}_i \hat{t}_{2,i} - \mathbf{C}_2^T \mathbf{S}_i = 0. \tag{31.4}$$

You now have two homogenous linear equations in the camera matrix elements for each pair (3D point, image point). There are a total of 12 degrees of freedom in the camera matrix, meaning we can recover a least squares solution from six point pairs. The solution will have the form $\lambda \mathcal{P}$ where λ is an unknown scale and $\mathcal{P}$ is a known matrix. This is a natural consequence of working with homogeneous equations, but also a natural consequence of working with homogeneous coordinates. You should check that if $\mathcal{P}$ is a projection from projective 3D to the projective plane, $\lambda \mathcal{P}$ will yield the same projection as long as $\lambda \neq 0$. Choose some scale. You can now factor $\lambda \mathcal{P}$ following Section 30.2.5.

Procedure: 31.1 *Calibrating a Camera using 3D Reference Points*

For N reference points $\mathbf{s}_i = [s_{1,i}, s_{2,i}, s_{3,i}]$ with known position in some reference coordinate system in 3D, write the measured location in the image for the i'th such point $\hat{\mathbf{t}}_i = \left[\hat{t}_{1,i}, \hat{t}_{2,i}\right]$. Now minimize

$$\sum_i \xi_i^T \xi_i = \sum_i (\hat{t}_{1,i} - p_{1,i})^2 + (\hat{t}_{2,i} - p_{2,i})^2 \tag{31.5}$$

where

$$\begin{aligned} p_{1,i} &= \frac{i_{11}g_{1,i} + i_{12}g_{2,i} + i_{13}g_{i,3}}{g_{i,3}} \\ p_{2,i} &= \frac{i_{22}g_{1,i} + i_{23}g_{i,3}}{g_{i,3}} \end{aligned}$$

and

$$\begin{aligned} g_{1,i} &= e_{11}s_{1,i} + e_{12}s_{2,i} + e_{13}s_{3,i} + e_{14} \\ g_{2,i} &= e_{21}s_{1,i} + e_{22}s_{2,i} + e_{23}s_{3,i} + e_{24} \\ g_{3,i} &= e_{31}s_{1,i} + e_{32}s_{2,i} + e_{33}s_{3,i} + e_{34} \end{aligned}$$

subject to:

$$1 - \sum_v e_{j,1v}^2 = 0 \text{ and } 1 - \sum_v e_{j,2v}^2 = 0 \text{ and } 1 - \sum_v e_{j,3v}^2 = 0$$

$$\sum_v e_{j,1v}e_{j,2v} = 0 \text{ and } \sum_v e_{j,1v}e_{j,3v} = 0 \text{ and } 1 - \sum_v e_{j,2v}e_{j,3v} = 0 \quad .$$

Use the start point of procedure 31.2

Procedure: 31.2 *Calibrating a Camera using 3D Reference Points: Start Point*

Estimate the rows of the camera matrix $\mathbf{C}_i$ using at least six points and

$$\mathbf{C}_3^T\mathbf{S}_i\hat{t}_{1,i} - \mathbf{C}_1^T\mathbf{S}_i = 0 \text{ and } \mathbf{C}_3^T\mathbf{S}_i\hat{t}_{2,i} - \mathbf{C}_2^T\mathbf{S}_i = 0. \tag{31.6}$$

Factor the result into intrinsic and extrinsic parameters using procedure 30.1.

31.1.3 Recovering Extrinsics from Known Correspondences

Now assume you *know* the camera intrinsics, and wish to recover extrinsics alone. Doing so is useful. For example, you could use this procedure to recover how a cam-

era moved around a 3D scene. In turn, you can render computer graphics objects into the scene, and have the viewpoint for these objects be the same as the viewpoint for the scene. This helps create consistent special effects for entertainment, but is also useful in other fields. For example, imagine a surgeon with a camera on their head is looking at a patient on an operating table. If you can tell where the camera is, you can superimpose various medical images and reconstructions on the image in the camera. You could show the result to the surgeon in, say, a virtual reality headset. The problem is best approached as an optimization problem, as below.

Procedure: 31.3 *Recovering extrinsics from known correspondences*

For N reference points $\mathbf{s}_i = [s_{1,i}, s_{2,i}, s_{3,i}]$ with known position in some reference coordinate system in 3D, write the measured location in the image for the i'th such point $\hat{\mathbf{t}}_i = \left[\hat{t}_{1,i}, \hat{t}_{2,i}\right]$. You know the intrinsic parameters $\mathcal{K}$ of the camera. To solve for the extrinsic parameters, assume you do not know the intrinsic parameters and obtain a start point using procedure 31.2. Now substitute your known intrinsics for the recovered intrinsics, and fix the intrinsic variables at that value. Finally minimize reprojection error as a function of extrinsics alone, following procedure 31.1

31.1.4 Reestimating Extrinsics after a Small Movement

Assume a camera moves rigidly through space, with rotation $\mathcal{R}(t)$ and translation $\mathbf{t}(t)$. You know $\mathcal{R}(t_0)$ and $\mathbf{t}(t_0)$, and wish to estimate $\mathcal{R}(t_0 + \delta t)$ and $\mathbf{t}(t_0 + \delta t)$. Notice that $\mathcal{R}(t_0 + \delta t) = \mathcal{R}(t_0)(\mathcal{I} + (\delta t)\Omega) + O(\delta t^2)$, where Ω is an anti-symmetric matrix (i.e. $\mathcal{O}\Updownarrow\rceil\}\dashv + \mathcal{O}\Updownarrow\rceil\}\dashv^T = \prime$). In turn, this means that if you know the movement is small, you can simplify the optimization problem of procedure 31.1.

Procedure: 31.4 *Updating a Calibration using 3D Reference Points*

You have N reference points $\mathbf{s}_i = [s_{1,i}, s_{2,i}, s_{3,i}]$ with known position in some reference coordinate system in 3D, and they stay in place. At time t_o, you computed $\mathcal{K}$ and

$$\mathcal{T}_e = \begin{bmatrix} \mathcal{R} & \mathbf{t} \\ \mathbf{0}^T & 1 \end{bmatrix}.$$

The camera makes a small movement, and you wish to update the extrinsic estimate. Write the measured location in the image for the i'th such point $\hat{\mathbf{t}}_i = \left[\hat{t}_{1,i}, \hat{t}_{2,i}\right]$. Write

$$\Omega = \begin{bmatrix} 0 & -a & -b \\ a & 0-c & \\ b & c0 & \end{bmatrix}$$

for an antisymmetric matrix, and $\mathbf{v}$ for the movement between frames. Now minimize

$$\sum_i \xi_i^T \xi_i = \sum_i (\hat{t}_{1,i} - p_{1,i})^2 + (\hat{t}_{2,i} - p_{2,i})^2 \tag{31.7}$$

where

$$\begin{aligned} p_{1,i} &= \frac{i_{11} g'_{1,i} + i_{12} g'_{2,i} + i_{13} g'_{i,3}}{g'_{i,3}} \\ p_{2,i} &= \frac{i_{22} g'_{1,i} + i_{23} g'_{i,3}}{g'_{i,3}} \end{aligned}$$

and

$$\mathbf{g}'_i = \mathcal{R}\mathbf{s}_i + \mathcal{R}\Omega + \mathbf{v}$$

As a function of a, b, c and $\mathbf{v}$. You could warm-start this optimization from $a = 0$, $b = 0$, $c = 0$, $\mathbf{v} = 0$.

You could repeatedly update using procedure 31.4, but eventually the rotation that you estimate will be wrong **exercises** . The usual – and effective – solution is to interleave this updating with a full optimization following procedure 31.3.

31.2 CALIBRATING THE EFFECTS OF LENS DISTORTION

Now assume the lens applies some form of geometric distortion, as in Section 21.3. There are now strong standard models of the major lens distortions (Section 21.3). We will now estimate lens parameters, camera intrinsics and camera extrinsics from a view of a calibration object (as in Section 21.3; note the methods of Section 21.3 apply to this problem too). As in those sections, we use a two step procedure: formulate the optimization problem (Section 21.3), then find a good starting point

(Section 21.3).

31.2.1 Modelling Geometric Lens Distortion

Geometric distortions caused by lenses are relatively easily modelled by assuming the lens causes (x, y) in the image plane to map to $(x+\delta x, y+\delta y)$ in the image plane. We seek a model for $\delta x, \delta y$ that has few parameters and that captures the main effects. A natural model of barrel distortion is that points are "pulled" toward the camera center, with points that are further from the center being "pulled" more. Similarly, pincushion distortion results from points being "pushed" away from the camera center, with distant points being pushed further (Figure **??**).

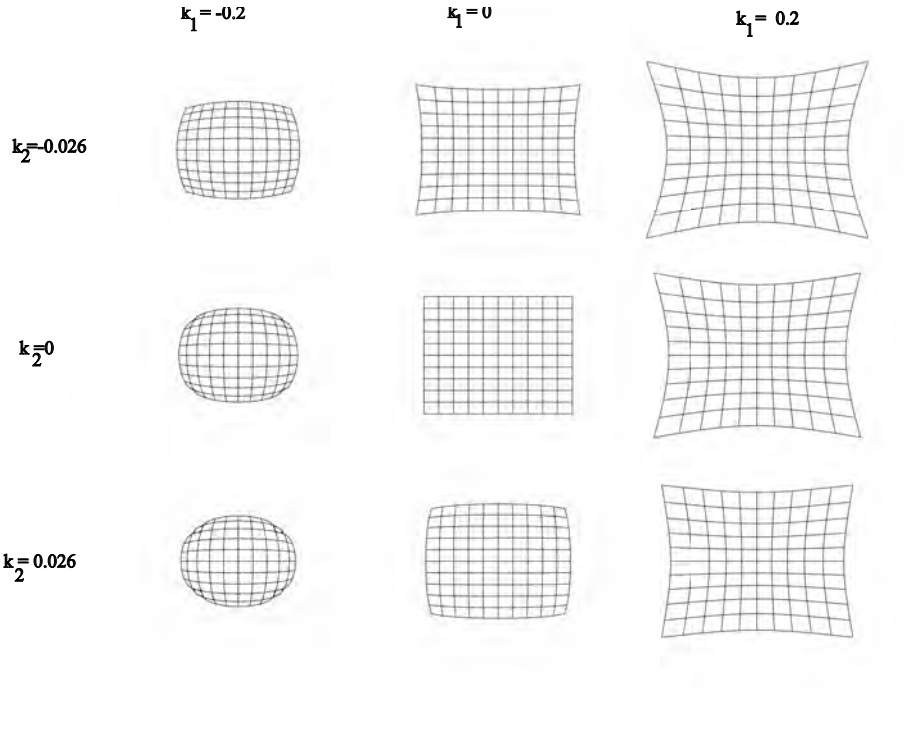

FIGURE 31.1: *The effects of k_1 and k_2 on a neutral grid (***center***), showing how the parameters implement various barrel or pincushion distortions. Notice how k_2 slightly changes the shape of the curves that k_1 produces from straight lines in the grid.*

Set up a polar coordinate system (r, θ) in the image plane using the image center as the origin. The figure and description suggest that barrel and pincushion distortion can be described by a map $(r, \theta) \rightarrow (r + \delta r, \theta)$. We model δr as a polynomial in r. Brown and Conrady [] established the model $\delta r = k_1 r^3 + k_2 r^5$ as sufficient for a wide range of distortions, and we use $(r, \theta) \rightarrow (r + k_1 r^3 + k_2 r^5, \theta)$ for unknown k_1, k_2. We must map this model to image coordinates to obtain a map $(x, y) \rightarrow (x + \delta x, y + \delta y)$. Since $\cos\theta = x/r$, $\sin\theta = y/r$, we have $(x, y) \rightarrow$

$(x + x(k_1(x^2 + y^2) + k_2(x^2 + y^2)^2), y + y(k_1(x^2 + y^2) + k_2(x^2 + y^2)^2))$. Figure 31.1 shows distortions resulting from different choices of k_1 and k_2. This model is known as a *radial distortion model*.

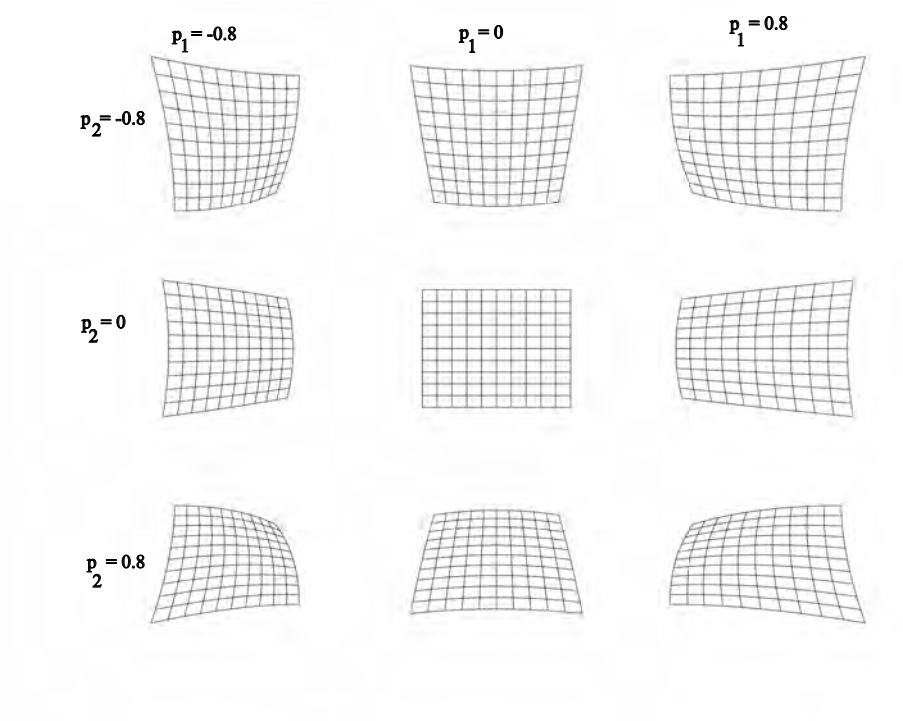

FIGURE 31.2: *The effects of p_1 and p_2 on a neutral grid (***center***), showing how the parameters implement various distortions. These parameters model effects that occur because the lens is off-center; note the grid "turning away" from the lens.*

More sophisticated lens distortion models account for the lens being off-center. This causes *tangential distortion* (Figure 31.2). The most commonly used model of tangential distortion is a map $(x, y) \to (x + p_1(x^2 + y^2 + 2x^2) + 2p_2xy, y + p_2(x^2 + y^2 + 2y^2) + 2p_1xy)$ (derived from []; more detail in, for example []).

Remember this: *A full lens distortion model is*

$$\begin{pmatrix} x \\ y \end{pmatrix} \to \begin{pmatrix} x + x(k_1(x^2 + y^2) + k_2(x^2 + y^2)^2) + p_1(x^2 + y^2 + 2x^2) + 2p_2xy \\ y + y(k_1(x^2 + y^2) + k_2(x^2 + y^2)^2) + p_2(x^2 + y^2 + 2y^2) + 2p_1xy \end{pmatrix}. \tag{31.8}$$

for k_1, k_2, p_1, p_2 parameters. It is common to ignore tangential distortion and focus on radial distortion by setting $p_1 = p_2 = 0$.

31.2.2 Lens Calibration: Formulating the Optimization Problem

Again, the optimization problem is relatively straightforward to formulate. Write $\hat{\mathbf{t}}_i = [t_{1,i}, t_{2,i}]$ for the measured x, y position in the image plane of the i'th reference point. We have that $\hat{\mathbf{t}}_i = \mathbf{t}_i + \xi_i$, where ξ_i is an error vector and $\mathbf{t}_i$ is the true (unknown) position of the i'th point. Again, assume the error is isotropic, so it is natural to minimize

$$\sum_i \xi_i^T \xi_i. \tag{31.9}$$

Expressions follow for $\xi_{i,j}$ in the appropriate coordinates (as in Section 31.1.1), though now accounting for the effects of the lens. Write:

$$\begin{aligned} g_{1,i} &= e_{11}s_{1,i} + e_{12}s_{2,i} + e_{13}s_{3,i} + e_{14} \\ g_{2,i} &= e_{21}s_{1,i} + e_{22}s_{2,i} + e_{23}s_{3,i} + e_{24} \\ g_{3,i} &= e_{31}s_{1,i} + e_{32}s_{2,i} + e_{33}s_{3,i} + e_{34}. \end{aligned}$$

and

$$\begin{aligned} p_{1,i} &= \frac{k_{11}g_{1,i} + k_{12}g_{2,i} + k_{13}g_{3,i}}{g_{3,i}} \\ p_{2,i} &= \frac{k_{22}g_{1,i} + k_{23}g_{3,i}}{g_{3,i}} \end{aligned}$$

and

$$\begin{aligned} l_{1,i} &= p_{1,i} + p_{1,i}(k_1(p_{1,i}^2 + p_{2,i}^2) + k_2(p_{1,i}^2 + p_{2,i}^2)^2) + p_1(p_{1,i}^2 + p_{2,i}^2 + 2p_{1,i}^2) + 2p_2p_{x,i}p_{2,i} \\ l_{2,i} &= p_{2,i} + p_{2,i}(k_1(p_{1,i}^2 + p_{2,i}^2) + k_2(p_{1,i}^2 + p_{2,i}^2)^2) + p_2(p_{1,i}^2 + p_{2,i}^2 + 2p_{2,i}^2) + 2p_1p_{1,i}p_{2,i}. \end{aligned}$$

Then you must minimize

$$\sum_i \xi_i^T \xi_i = \sum_i (t_{1,i} - l_{1,i})^2 + (t_{2,i} - l_{2,i})^2 \tag{31.10}$$

by choice of k_{ij}, e_{ij}, k_1, k_2, p_1 and p_2. As before, this is a constrained optimization problem, because $\mathcal{T}_e$ is a Euclidean transformation. The constraints here are

$$\begin{aligned} 1 - \sum_v e_{j,1v}^2 = 0 \text{ and } 1 - \sum_v e_{j,2v}^2 = 0 \text{ and } 1 - \sum_v e_{j,3v}^2 = 0 \\ \sum_v e_{j,1v}e_{j,2v} = 0 \text{ and } \sum_v e_{j,1v}e_{j,3v} = 0 \text{ and } \sum_v e_{j,2v}e_{j,3v} = 0 \end{aligned} .$$

As in Section 31.1.1, simply dropping this problem into a constrained optimizer is not a particularly good approach. If we assume the lens distortion is minor, we can obtain a start point for the intrinsics and the extrinsics using Section 31.1.2. We then use those parameters, together with $k_1 = 0$, $k_2 = 0$, $p_1 = 0$ and $p_2 = 0$, as a start point.

31.3 "FOUND" CALIBRATIONS

A useful set of camera intrinsics can be recovered from assumptions about the 3D world seen in a picture, assuming that vanishing points can be recovered in an image. One standard assumption is that we are viewing a *Manhattan world*. In this case, there are three cardinal directions. Each is normal to the other two (so you could make them be the x, y, and z directions in an appropriately chosen world coordinate system). Every line in the world is parallel to one of the three cardinal directions.

In a Manhattan world geometry, every image contains at most three families of line, one for each cardinal direction. Each family has a single vanishing point. Assume that there are lines from all families present in an image, and that the vanishing points can be found (Section 21.3). Then the vanishing points reveal camera intrinsics.

This is easily demonstrated in homogeneous coordinates. The three cardinal directions can be defined by their vanishing points *in 3D*. For simplicity, we assume that the cardinal directions are the x, y and z directions in the world coordinate system. In homogeneous coordinates, the collection of all lines parallel to the x axis can be written as (S, Ta, Tb, T) **exercises** , where S, T are parameters along the line *in homogeneous coordinates* and a, b select the particular line. All such lines contain the point $\mathbf{E}_x = (1, 0, 0, 0)$ – this is the point at infinity where all of these lines intersect. Similarly, the point at infinity where all lines parallel to the y (resp. z) axis intersect is $\mathbf{E}_y = (0, 1, 0, 0)$ (resp. $\mathbf{E}_z = (0, 0, 1, 0)$.

There will be three vanishing points in the image, $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$. The coordinates of these points are known, because we have detected them, and we represent these points in homogeneous coordinates. Each is the image of one of the 3D points at infinity. We *choose* $\mathbf{e}_1$ as the image of $\mathbf{E}_x$, etc. Note that there are six possible choices. Three of them will result in rotations of the world coordinates, and the other three will result in improper rotations (a rotation, followed by flipping the z-axis direction). For the moment, we ignore the consequences of choosing.

Now recall the parametrization of a camera as $\mathcal{K}[\mathcal{R} \mid \mathbf{t}]$. Furthermore, notice that each of the $\mathbf{E}$'s has a zero in the fourth component. We have that

$$\mathbf{E}_x = \begin{pmatrix} (\mathcal{K}\mathcal{R})^{-1}\mathbf{e}_1 \\ 0 \end{pmatrix} \text{ and } \mathbf{E}_y = \begin{pmatrix} (\mathcal{K}\mathcal{R})^{-1}\mathbf{e}_2 \\ 0 \end{pmatrix} \text{ and } \mathbf{E}_z = \begin{pmatrix} (\mathcal{K}\mathcal{R})^{-1}\mathbf{e}_3 \\ 0 \end{pmatrix}.$$

The cardinal directions are at right angles, so we have three constraints

$$\mathbf{E}_x^T\mathbf{E}_y = 0 \text{ and } \mathbf{E}_x^T\mathbf{E}_z = 0 \text{ and } \mathbf{E}_y^T\mathbf{E}_z = 0$$

and it follows that

$$\mathbf{e}_1\mathcal{K}^{-T}\mathcal{K}\mathbf{e}_2 = 0 \text{ and } \mathbf{e}_1\mathcal{K}^{-T}\mathcal{K}\mathbf{e}_3 = 0 \text{ and } \mathbf{e}_2\mathcal{K}^{-T}\mathcal{K}\mathbf{e}_3 = 0$$

(the rotations cancel because $\mathcal{R}^T = \mathcal{R}^{-1}$). Now assume that the aspect ratio a is 1, and the skew k is 0, so that

$$\mathcal{K} = \begin{pmatrix} s & 0 & c_x \\ 0 & s & c_y \\ 0 & 0 & 1 \end{pmatrix}.$$

In the first case, assume all three vanishing points are finite. Write the known coordinates of the vanishing points $\mathbf{e}_1 = (e_{1,1}, e_{2,1}, 1)^T$ and so on. Some algebra yields the constraints

$$\begin{aligned}
[e_{1,1}e_{1,2} + e_{2,1}e_{2,2}] + c_x[1 - e_{1,1} - e_{1,2}] + c_y[1 - e_{2,1} - e_{2,2}] + s^2 + c_x^2 + c_y^2 &= 0 \\
[e_{1,1}e_{1,3} + e_{2,1}e_{2,3}] + c_x[1 - e_{1,1} - e_{1,3}] + c_y[1 - e_{2,1} - e_{2,3}] + s^2 + c_x^2 + c_y^2 &= 0 \\
[e_{1,2}e_{1,3} + e_{2,2}e_{2,3}] + c_x[1 - e_{1,2} - e_{1,3}] + c_y[1 - e_{2,2} - e_{2,3}] + s^2 + c_x^2 + c_y^2 &= 0
\end{aligned}$$

It is straightforward to extract c_x, c_y and s from these equations **exercises** .

If one image vanishing point is at infinity (choose $\mathbf{e}_1$, so that $\mathbf{e}_1 = (e_{1,1}, e_{1,2}, 0)^T$), the equations become

$$\begin{aligned}
[e_{1,1}e_{1,2} + e_{2,1}e_{2,2}] + c_x[-e_{1,1} - e_{1,2}] + c_y[-e_{2,1} - e_{2,2}] &= 0 \\
[e_{1,1}e_{1,3} + e_{2,1}e_{2,3}] + c_x[-e_{1,1} - e_{1,3}] + c_y[-e_{2,1} - e_{2,3}] &= 0 \\
[e_{1,2}e_{1,3} + e_{2,2}e_{2,3}] + c_x[1 - e_{1,2} - e_{1,3}] + c_y[1 - e_{2,2} - e_{2,3}] + s^2 + c_x^2 + c_y^2 &= 0
\end{aligned}$$

and you can still solve for c_x, c_y and s **exercises** .

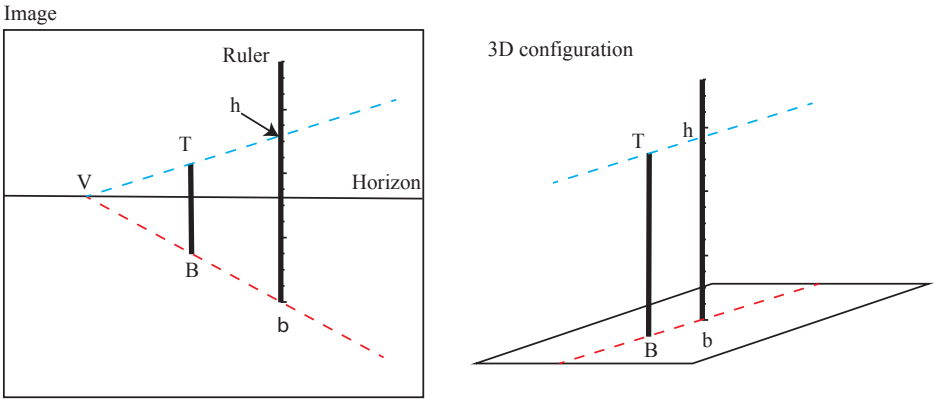

FIGURE 31.3: **Left**, *an image of a ruler and an object, which just both happen to be standing perpendicular to a ground plane. In an* uncalibrated *image like this, you can measure the height of the object. Construct the line bB (***red***), and intersect that with the horizon to get the point V. The line from the top of the object T to the true height of the object on the ruler (h) is parallel in 3D to bB. In turn, the line Th (***blue***) must intersect the horizon at V. So if you construct VT, it will intersect the ruler at h yielding the height of the object.* **Right** *shows a 3D view; the line Th must be parallel to bB, and so in the image these two lines intersect at the horizon.*

31.4 MEASURING LENGTHS IN A SINGLE VIEW

In many very useful cases, we can measure lengths in a single *uncalibrated* image.

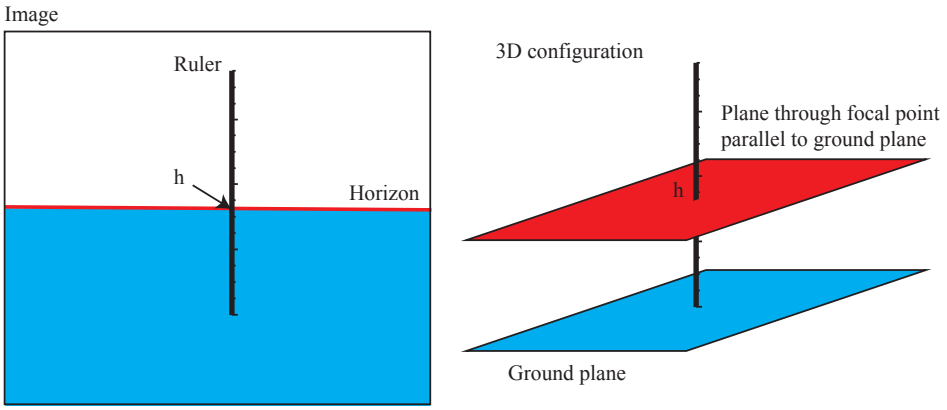

FIGURE 31.4: **Left**, *an image of a ruler which just happens to be standing perpendicular to a ground plane. In an* uncalibrated *image like this, we can measure the height of the camera focal point above the ground plane. The plane through the focal point parallel to the ground plan (and so the same height above the ground plane as the focal point) must form the horizon, so the intersection between horizon and ruler yields the height of the focal point.* **Right** *shows a 3D view; the bottom plane is the ground plane, and the top plane is the plane through the focal point parallel to the ground plane.*

31.4.1 Exploiting a Ruler

In the simplest, the image shows an object on a ground plane, there is a ruler conveniently standing normal to the ground plane, and we wish to measure the height of the object. This is shown in Figure 31.3, and I use the notation of that figure. Construct the line bB (from the base of the ruler to the base of the object) and intersect that line with the horizon to get vanishing point V. Now construct the line VT from that vanishing point to the top of the object, and intersect that line with the ruler. The intersection point with the ruler shows the height of the object.

A ruler perpendicular to a ground plane can reveal the height of the camera above the ground plane, too. The horizon in the image is formed by the plane through the camera focal point and parallel to the ground plane. But this plane must intersect the ruler at the height of the focal point above the ground plane. This means that the horizon intersects the ruler at the height of the focal point above the ground plane.

31.4.2 Measuring with a Reference Object

A ruler has the special property that it has lengths marked on it. Imagine we now have a reference line segment that has known height, but doesn't have other heights marked on it. Using this reference object takes care, because (say) the midpoint of the reference object *in the image* may not lie halfway up the reference object in 3D (Figure 31.5).

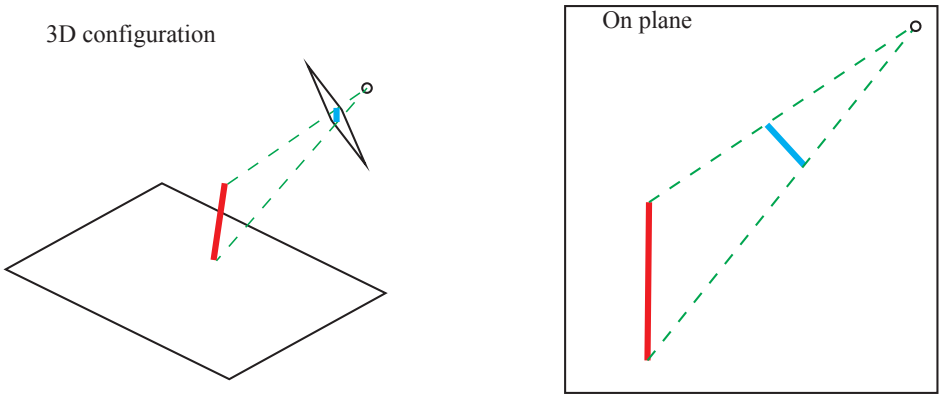

FIGURE 31.5: **Left**, *a perspective camera views a reference object (***red***) perpendicular to a ground plane. This produces a line segment (***blue***) in the image plane.* **Right** *shows the reference object and the line segment in the plane spanned by the object and the focal point. The red line is not parallel to the blue line.*

This is because the transformation from the reference line segment in 3D to the image is not an affine transformation – it is a projective transformation. Figure 31.5 sketches the geometry. Parametrize the reference line segment in 3D using affine coordinates to get $\mathbf{p}+t\mathbf{d}$, where $\mathbf{d}$ is a unit vector (so a step of 1 in t is a step of length 1 along the reference segment). Write c_{ij} for the i, j'th component of the 3×4 camera matrix. Then the homogeneous coordinates for the image line will be

$$\begin{pmatrix} (c_{11}p_1 + c_{12}p_2 + c_{13}p_3 + c_{14}) + t(c_{11}d_1 + c_{12}d_2 + c_{13}d_3 + c_{14}) \\ (c_{21}p_1 + c_{22}p_2 + c_{23}p_3 + c_{24}) + t(c_{21}d_1 + c_{22}d_2 + c_{23}d_3 + c_{24}) \\ (c_{31}p_1 + c_{32}p_2 + c_{33}p_3 + c_{34}) + t(c_{31}d_1 + c_{32}d_2 + c_{33}d_3 + c_{34}) \end{pmatrix} = \begin{pmatrix} a + bt \\ c + dt \\ e + ft \end{pmatrix}.$$

SInce we know the image is a line, we can ignore one of these three homogeneous coordinates, so the transformation is a projective transformation. Now on the 3D reference line segment, the points $t = 0$ and $t = 1$ are the same distance apart as the points $t = 1$ and $t = 2$. But in the image line, using affine coordinates, these points are

$$\frac{a}{c}, \frac{a+b}{c+d}, \frac{a+2b}{c+2d}$$

which are not, in general, evenly spaced (check this with, for example, $a = 0$, $b = 1$, $c = 1$, $d = 1$).

A clever trick from projective geometry allows us to use a reference object to measure heights. Write $\mathbf{P}_1, \ldots, \mathbf{P}_4$ for the coordinates of four points on a projective line, written in homogeneous coordinates. Write $\mathcal{M}$ for a projective transformation of the line to itself (so a 2×2 matrix with non-zero determinant. Finally, write

$$d(\mathbf{P}_i, \mathbf{P}_j) = \det\left([\mathbf{P}_i \mathbf{P}_j]\right).$$

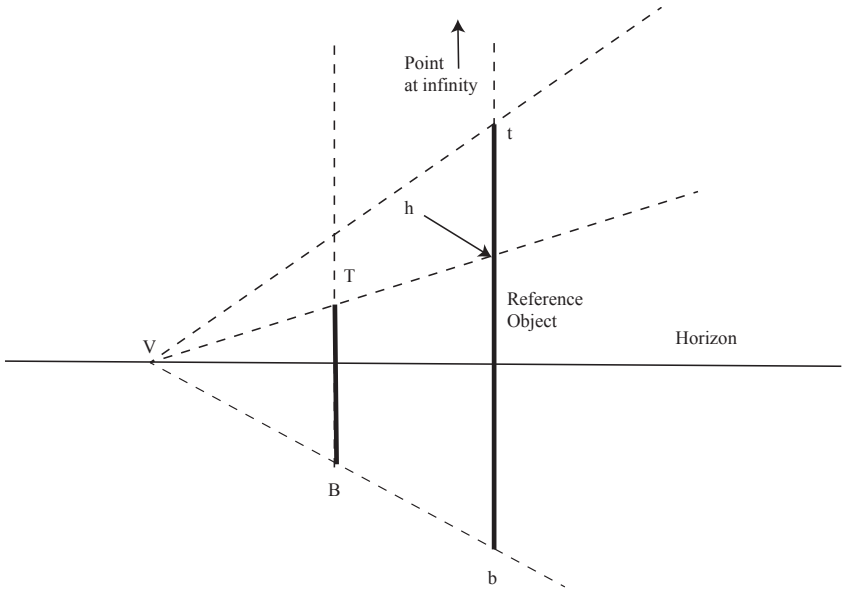

FIGURE 31.6: *A perspective camera views a reference object and another object perpendicular to a ground plane. This produces a line segment in the image plane. Constructing appropriate lines in the figure and taking a cross ratio yields the height of the object.*

Notice that

$$\det\left([\mathcal{M}\mathbf{P}_i\mathcal{M}\mathbf{P}_j)]\right) = \det\left(\mathcal{M}[\mathbf{P}_i\mathbf{P}_j]\right) = \det\left(\mathcal{M}\right)\det\left([\mathbf{P}_i\mathbf{P}_j]\right)$$

which means that

$$\frac{d(\mathbf{P}_1,\mathbf{P}_2)d(\mathbf{P}_3,\mathbf{P}_4)}{d(\mathbf{P}_1,\mathbf{P}_3)d(\mathbf{P}_2,\mathbf{P}_4)}$$

is a *projective invariant* — computing the value of this *cross ratio* using $\mathbf{P}_1,\ldots,\mathbf{P}_4$ or using $\mathcal{M}\mathbf{P}_1,\ldots,\mathcal{M}\mathbf{P}_4$ will yield the same number, as long as $\mathcal{M}$ is a projective transformation.

Now check that the cross-ratio of the four points $(0,1)$, $(a,1)$, $(b,1)$ and $(1,0)$ is a/b (notice the last point is the point at infinity). We can use this observation to measure height relative to a reference object. Using the notation of Figure 31.6, we construct the line Bb from the base of the object to the base of the reference object. Produce this line to intersect the horizon at V. Now construct VT, which intersects the reference object at h. In 3D, the line VT is parallel to the ground plane, so that the point h in 3D is the same height above the ground plane as the point T in 3D. The vanishing point for the vertical lines (the object and the reference object) is at infinity in this image, *so we know where it lies on line bt.* Write P for this vanishing point, r for the height of the reference object and o for the height of the object. Then we have

$$\frac{d(b,h)d(t,P)}{d(b,t),b(h,P)} = \frac{r}{o}$$

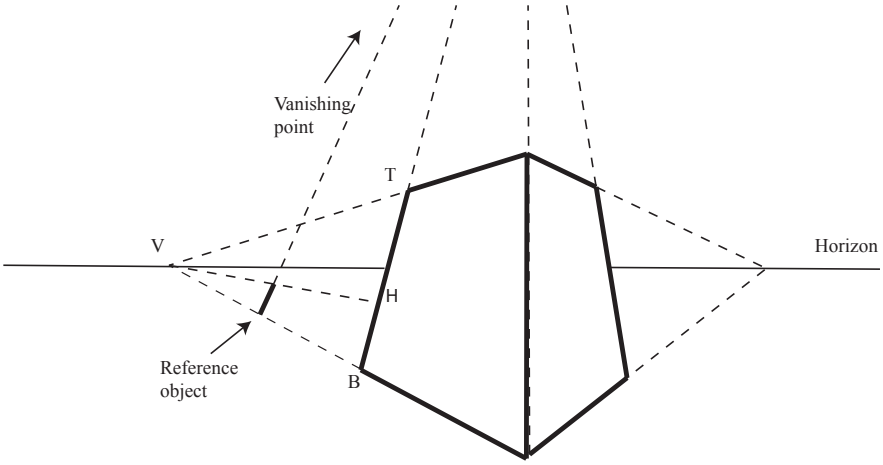

FIGURE 31.7: *A building and a person viewed in a more extreme perspective view than that of 31.6. The person has known height, and can act as reference object. The same construction as in that figure yields the height of the building relative to the person.*

but we *know* the height of the reference object and we can measure the cross ratio, so we can recover o.

31.5 YOU SHOULD

31.5.1 remember these facts:

Cameras: Models of lens distortions 542

31.5.2 be able to:

- Calibrate a camera using a set of reference points.
- Compute extrinsics for a camera of known intrinsics using reference points.
- Update extrinsics for a camera of known intrinsics using reference points.
- Calibrate lens distortion using reference points.
- Estimate the height of an item in a perspective image using a ruler as a reference.
- Estimate the height of the camera focal point in a perspective image using a ruler as a reference.

EXERCISES

QUICK CHECKS

31.1. Section 31.1.1 has constraints on an extrinsic matrix given by:

$$1 - \sum_v e_{1v}^2 = 0 \text{ and } 1 - \sum_v e_{2v}^2 = 0 \text{ and } 1 - \sum_v e_{3v}^2 = 0$$
$$\sum_v e_{1v}e_{2v} = 0 \text{ and } \sum_v e_{1v}e_{3v} = 0 \text{ and } \sum_v e_{2v}e_{3v} = 0 \quad .$$

Where do these constraints come from?

31.2. The procedure of Section 31.1.2 yields an estimate of intrinsic and extrinsic coordinates. The reason not to use this estimate, but instead to use reprojection error, is that this estimate minimizes the wrong error. Explain.

31.3. Section 31.1.3 gives a procedure to recover extrinsics when you *know* the intrinsics. This procedure assumes the intrinsics you know are right – when does this assumption make sense? when does it not?

31.4. Assume $\mathcal{R}$ is a rotation matrix, so that $\mathcal{R}^T\mathcal{R} = \mathcal{I}$ (the identity). Show that $\mathcal{R}(\mathcal{I}+\epsilon\Omega)$ is a rotation matrix up to order ϵ if and only if Ω is an antisymmetric matrix.

31.5. Procedure 31.4 has: "You could warm-start this optimization from $a = 0$, $b = 0$, $c = 0$, $\mathbf{v} = 0$." Explain.

31.6. Section 31.4.2 has: "Using this reference object takes care, because (say) the midpoint of the reference object *in the image* may not lie halfway up the reference object in 3D" Explain, with an example that is *not* Figure 31.5.

31.7. You want to measure the height of an object using the method of Section 31.4.2. Assume the ruler is perpendicular to the ground plane, but the object is at angle ϕ to the ground plane. Here ϕ is close to, but not exactly, 90^o. Can you estimate the height of the object if you don't know ϕ? If you do know ϕ?

LONGER PROBLEMS

31.8. Obtain a calibration figure (you could download and print one from `https://calib.io/`; look for "pattern generator" on the web page, and notice how an extensive range of calibration products is sold – it is really useful to solve camera calibration).

- **(a)** Use this figure to estimate camera intrinsics for 5 different views of the figure obtained using the same camera. How stable are the intrinsics?
- **(b)** What happens if you estimate intrinsics for views obtained on different days?
- **(c)** Put your camera on a tripod or some fairly stable location. Obtain 5 different views of the calibration figure taken with a reasonable time span (hours) separating them. How stable are the extrinsics? Do you think this is an estimation issue or an issue of how stable the location is? Why?

31.9. Obtain a calibration figure (you could also download and print one from `https://markhedleyjones.com/projects/calibration-checkerboard-collection`). Put your camera on a tripod or some fairly stable location. Obtain 10 different views of the calibration figure. Mark the location of the calibration figure in the first view, so that you can return it to that location. Between each

view, move the calibration figure slightly, ensuring that it returns to the initial location in the final view.

(a) Estimate the extrinsics for each view.

(b) From the extrinsics, you can compute the movement of the calibration figure from frame i to frame j. Is the transformation from the first frame to the last the identity?

(c) How trustworthy is the estimate of the extrinsics? Explore this point by computing the transformation from frame i to frame j and comparing it with the transformation from frame i to k composed with that from k to j.

(d) Try procedure 31.4; does this speed up the calibration? Does warm-starting help? How many updates can you do before the rotation matrix estimate becomes untrustworthy?

(e) Assume $\mathcal{R}$ is a rotation matrix, so that $\mathcal{R}^T\mathcal{R} = \mathcal{I}$ (the identity). Show that $\mathcal{R}(\mathcal{I} + \epsilon\Omega)$ is a rotation matrix up to order ϵ if and only if Ω is an antisymmetric matrix.

(f) Show that repeating procedure 31.4 means you may obtain a solution for $\mathcal{R}$ that is no longer a rotation matrix.

(g) For your dataset, how often can you repeat procedure 31.4 before you need to solve properly for $\mathcal{R}$?

(h) Imagine you wish to compute extrinsics for a moving camera. It is natural to use an adaptive method to choose whether you will use procedure 31.4 or procedure 31.3. How would you choose?