Chapter 7

Structure from motion

R. Szeliski, *Computer Vision: Algorithms and Applications*, Texts in Computer Science,
DOI 10.1007/978-1-84882-935-0_7, © Springer-Verlag London Limited 2011

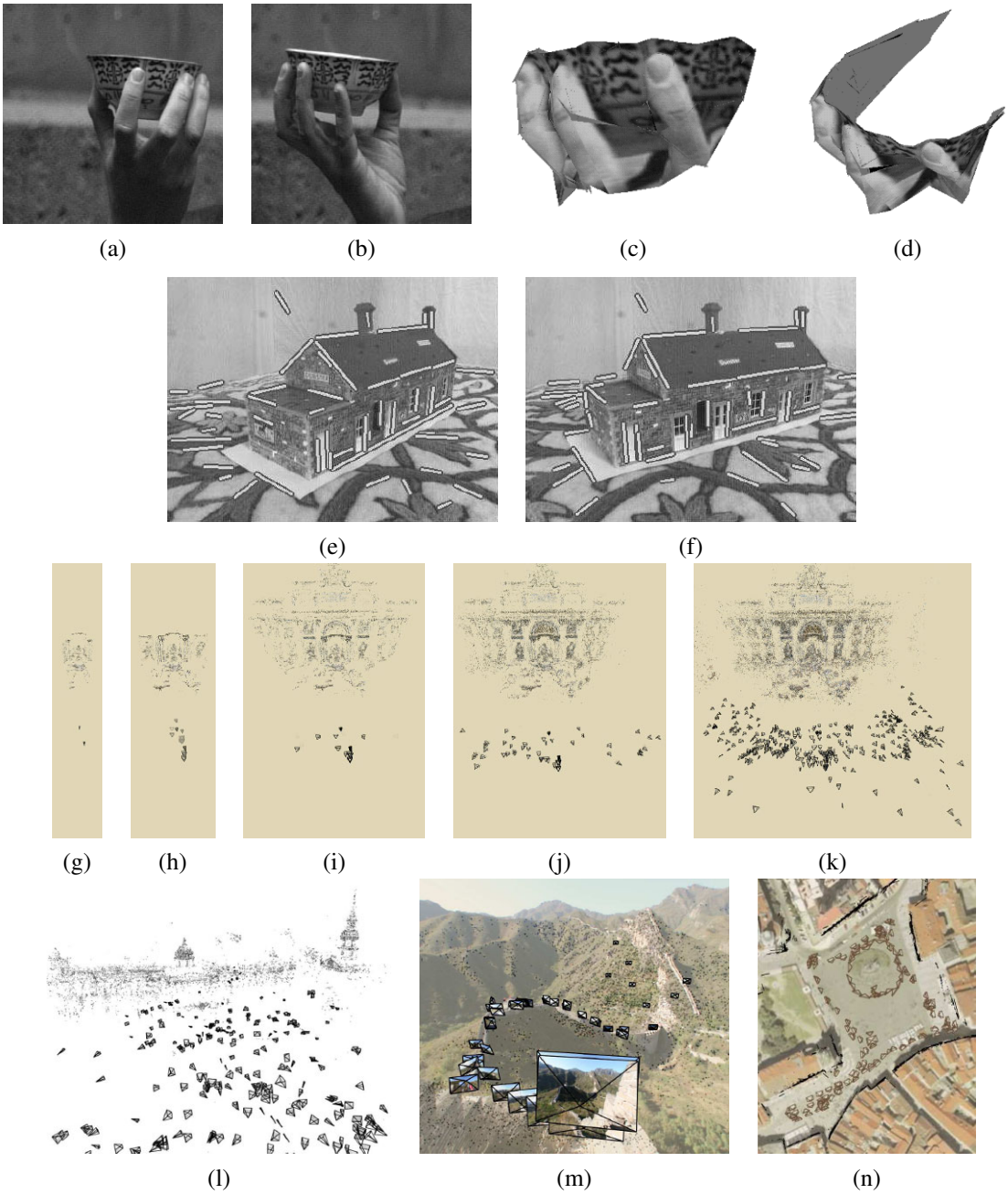


Figure 7.1 Structure from motion systems: (a–d) orthographic factorization (Tomasi and Kanade 1992) © 1992 Springer; (e–f) line matching (Schmid and Zisserman 1997) © 1997 IEEE; (g–k) incremental structure from motion (Snavely, Seitz, and Szeliski 2006); (l) 3D reconstruction of Trafalgar Square (Snavely, Seitz, and Szeliski 2006); (m) 3D reconstruction of the Great Wall of China (Snavely, Seitz, and Szeliski 2006); (n) 3D reconstruction of the Old Town Square, Prague (Snavely, Seitz, and Szeliski 2006) © 2006 ACM.

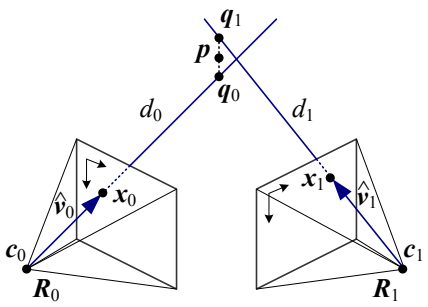


Figure 7.2 3D point triangulation by finding the point $\boldsymbol{p}$ that lies nearest to all of the optical rays $\boldsymbol{c}_j + d_j\hat{\boldsymbol{v}}_j$.

In the previous chapter, we saw how 2D and 3D point sets could be aligned and how such alignments could be used to estimate both a camera's pose and its internal calibration parameters. In this chapter, we look at the converse problem of estimating the locations of 3D points from multiple images given only a sparse set of correspondences between image features. While this process often involves simultaneously estimating both 3D geometry (structure) and camera pose (motion), it is commonly known as *structure from motion* (Ullman 1979).

The topics of projective geometry and structure from motion are extremely rich and some excellent textbooks and surveys have been written on them (Faugeras and Luong 2001; Hartley and Zisserman 2004; Moons, Van Gool, and Vergauwen 2010). This chapter skips over a lot of the richer material available in these books, such as the trifocal tensor and algebraic techniques for full self-calibration, and concentrates instead on the basics that we have found useful in large-scale, image-based reconstruction problems (Snavely, Seitz, and Szeliski 2006).

We begin with a brief discussion of *triangulation* (Section 7.1), which is the problem of estimating a point's 3D location when it is seen from multiple cameras. Next, we look at the two-frame structure from motion problem (Section 7.2), which involves the determination of the *epipolar geometry* between two cameras and which can also be used to recover certain information about the camera intrinsics using self-calibration (Section 7.2.2). Section 7.3 looks at *factorization* approaches to simultaneously estimating structure and motion from large numbers of point tracks using orthographic approximations to the projection model. We then develop a more general and useful approach to structure from motion, namely the simultaneous *bundle adjustment* of all the camera and 3D structure parameters (Section 7.4). We also look at special cases that arise when there are higher-level structures, such as lines and planes, in the scene (Section 7.5).

7.1 Triangulation

The problem of determining a point's 3D position from a set of corresponding image locations and known camera positions is known as *triangulation*. This problem is the converse of the pose estimation problem we studied in Section 6.2.

One of the simplest ways to solve this problem is to find the 3D point $\boldsymbol{p}$ that lies closest to all of the 3D rays corresponding to the 2D matching feature locations $\{\boldsymbol{x}_j\}$ observed by cam-

eras $\{\boldsymbol{P}_j = \boldsymbol{K}_j[\boldsymbol{R}_j|\boldsymbol{t}_j]\}$, where $\boldsymbol{t}_j = -\boldsymbol{R}_j\boldsymbol{c}_j$ and $\boldsymbol{c}_j$ is the jth camera center (2.55–2.56). As you can see in Figure 7.2, these rays originate at $\boldsymbol{c}_j$ in a direction $\hat{\boldsymbol{v}}_j = \mathcal{N}(\boldsymbol{R}_j^{-1}\boldsymbol{K}_j^{-1}\boldsymbol{x}_j)$. The nearest point to $\boldsymbol{p}$ on this ray, which we denote as $\boldsymbol{q}_j$, minimizes the distance

$$\|\boldsymbol{c}_j + d_j\hat{\boldsymbol{v}}_j - \boldsymbol{p}\|^2, \tag{7.1}$$

which has a minimum at $d_j = \hat{\boldsymbol{v}}_j \cdot (\boldsymbol{p} - \boldsymbol{c}_j)$. Hence,

$$\boldsymbol{q}_j = \boldsymbol{c}_j + (\hat{\boldsymbol{v}}_j\hat{\boldsymbol{v}}_j^T)(\boldsymbol{p} - \boldsymbol{c}_j) = \boldsymbol{c}_j + (\boldsymbol{p} - \boldsymbol{c}_j)_{\|}, \tag{7.2}$$

in the notation of Equation (2.29), and the squared distance between $\boldsymbol{p}$ and $\boldsymbol{q}_j$ is

$$r_j^2 = \|(\boldsymbol{I} - \hat{\boldsymbol{v}}_j\hat{\boldsymbol{v}}_j^T)(\boldsymbol{p} - \boldsymbol{c}_j)\|^2 = \|(\boldsymbol{p} - \boldsymbol{c}_j)_{\perp}\|^2. \tag{7.3}$$

The optimal value for $\boldsymbol{p}$, which lies closest to all of the rays, can be computed as a regular least squares problem by summing over all the r_j^2 and finding the optimal value of $\boldsymbol{p}$,

$$\boldsymbol{p} = \left[\sum_j (\boldsymbol{I} - \hat{\boldsymbol{v}}_j\hat{\boldsymbol{v}}_j^T)\right]^{-1}\left[\sum_j (\boldsymbol{I} - \hat{\boldsymbol{v}}_j\hat{\boldsymbol{v}}_j^T)\boldsymbol{c}_j\right]. \tag{7.4}$$

An alternative formulation, which is more statistically optimal and which can produce significantly better estimates if some of the cameras are closer to the 3D point than others, is to minimize the residual in the measurement equations

$$x_j = \frac{p_{00}^{(j)}X + p_{01}^{(j)}Y + p_{02}^{(j)}Z + p_{03}^{(j)}W}{p_{20}^{(j)}X + p_{21}^{(j)}Y + p_{22}^{(j)}Z + p_{23}^{(j)}W} \tag{7.5}$$

$$y_j = \frac{p_{10}^{(j)}X + p_{11}^{(j)}Y + p_{12}^{(j)}Z + p_{13}^{(j)}W}{p_{20}^{(j)}X + p_{21}^{(j)}Y + p_{22}^{(j)}Z + p_{23}^{(j)}W}, \tag{7.6}$$

where (x_j, y_j) are the measured 2D feature locations and $\{p_{00}^{(j)} \ldots p_{23}^{(j)}\}$ are the known entries in camera matrix $\boldsymbol{P}_j$ (Sutherland 1974).

As with Equations (6.21, 6.33, and 6.34), this set of non-linear equations can be converted into a linear least squares problem by multiplying both sides of the denominator. Note that if we use homogeneous coordinates $\boldsymbol{p} = (X, Y, Z, W)$, the resulting set of equations is homogeneous and is best solved as a singular value decomposition (SVD) or eigenvalue problem (looking for the smallest singular vector or eigenvector). If we set $W = 1$, we can use regular linear least squares, but the resulting system may be singular or poorly conditioned, i.e., if all of the viewing rays are parallel, as occurs for points far away from the camera.

For this reason, it is generally preferable to parameterize 3D points using homogeneous coordinates, especially if we know that there are likely to be points at greatly varying distances from the cameras. Of course, minimizing the set of observations (7.5–7.6) using non-linear least squares, as described in (6.14 and 6.23), is preferable to using linear least squares, regardless of the representation chosen.

For the case of two observations, it turns out that the location of the point $\boldsymbol{p}$ that exactly minimizes the true reprojection error (7.5–7.6) can be computed using the solution of degree six equations (Hartley and Sturm 1997). Another problem to watch out for with triangulation is the issue of *chirality*, i.e., ensuring that the reconstructed points lie in front of all the

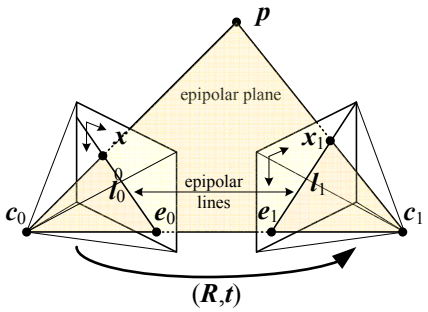


Figure 7.3 Epipolar geometry: The vectors $\boldsymbol{t} = \boldsymbol{c}_1 - \boldsymbol{c}_0$, $\boldsymbol{p} - \boldsymbol{c}_0$ and $\boldsymbol{p} - \boldsymbol{c}_1$ are co-planar and define the basic epipolar constraint expressed in terms of the pixel measurements $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$.

cameras (Hartley 1998). While this cannot always be guaranteed, a useful heuristic is to take the points that lie behind the cameras because their rays are diverging (imagine Figure 7.2 where the rays were pointing *away* from each other) and to place them on the plane at infinity by setting their W values to 0.

7.2 Two-frame structure from motion

So far in our study of 3D reconstruction, we have always assumed that either the 3D point positions or the 3D camera poses are known in advance. In this section, we take our first look at *structure from motion*, which is the simultaneous recovery of 3D structure and pose from image correspondences.

Consider Figure 7.3, which shows a 3D point $\boldsymbol{p}$ being viewed from two cameras whose relative position can be encoded by a rotation $\boldsymbol{R}$ and a translation $\boldsymbol{t}$. Since we do not know anything about the camera positions, without loss of generality, we can set the first camera at the origin $\boldsymbol{c}_0 = \boldsymbol{0}$ and at a canonical orientation $\boldsymbol{R}_0 = \boldsymbol{I}$.

Now notice that the observed location of point $\boldsymbol{p}$ in the first image, $\boldsymbol{p}_0 = d_0\hat{\boldsymbol{x}}_0$ is mapped into the second image by the transformation

$$d_1\hat{\boldsymbol{x}}_1 = \boldsymbol{p}_1 = \boldsymbol{R}\boldsymbol{p}_0 + \boldsymbol{t} = \boldsymbol{R}(d_0\hat{\boldsymbol{x}}_0) + \boldsymbol{t}, \tag{7.7}$$

where $\hat{\boldsymbol{x}}_j = \boldsymbol{K}_j^{-1}\boldsymbol{x}_j$ are the (local) ray direction vectors. Taking the cross product of both sides with $\boldsymbol{t}$ in order to annihilate it on the right hand side yields[1]

$$d_1[\boldsymbol{t}]_\times\hat{\boldsymbol{x}}_1 = d_0[\boldsymbol{t}]_\times\boldsymbol{R}\hat{\boldsymbol{x}}_0. \tag{7.8}$$

Taking the dot product of both sides with $\hat{\boldsymbol{x}}_1$ yields

$$d_0\hat{\boldsymbol{x}}_1^T([\boldsymbol{t}]_\times\boldsymbol{R})\hat{\boldsymbol{x}}_0 = d_1\hat{\boldsymbol{x}}_1^T[\boldsymbol{t}]_\times\hat{\boldsymbol{x}}_1 = 0, \tag{7.9}$$

since the right hand side is a triple product with two identical entries. (Another way to say this is that the cross product matrix $[\boldsymbol{t}]_\times$ is skew symmetric and returns 0 when pre- and post-multiplied by the same vector.)

[1] The cross-product operator $[\,]_\times$ was introduced in (2.32).

We therefore arrive at the basic *epipolar constraint*

$$\hat{\boldsymbol{x}}_1^T \boldsymbol{E} \hat{\boldsymbol{x}}_0 = 0, \tag{7.10}$$

where

$$\boldsymbol{E} = [\boldsymbol{t}]_\times \boldsymbol{R} \tag{7.11}$$

is called the *essential matrix* (Longuet-Higgins 1981).

An alternative way to derive the epipolar constraint is to notice that in order for the cameras to be oriented so that the rays $\hat{\boldsymbol{x}}_0$ and $\hat{\boldsymbol{x}}_1$ intersect in 3D at point $\boldsymbol{p}$, the vectors connecting the two camera centers $\boldsymbol{c}_1 - \boldsymbol{c}_0 = -\boldsymbol{R}_1^{-1}\boldsymbol{t}$ and the rays corresponding to pixels $\boldsymbol{x}_0$ and $\boldsymbol{x}_1$, namely $\boldsymbol{R}_j^{-1}\hat{\boldsymbol{x}}_j$, must be co-planar. This requires that the triple product

$$(\hat{\boldsymbol{x}}_0, \boldsymbol{R}^{-1}\hat{\boldsymbol{x}}_1, -\boldsymbol{R}^{-1}\boldsymbol{t}) = (\boldsymbol{R}\hat{\boldsymbol{x}}_0, \hat{\boldsymbol{x}}_1, -\boldsymbol{t}) = \hat{\boldsymbol{x}}_1 \cdot (\boldsymbol{t} \times \boldsymbol{R}\hat{\boldsymbol{x}}_0) = \hat{\boldsymbol{x}}_1^T([\boldsymbol{t}]_\times \boldsymbol{R})\hat{\boldsymbol{x}}_0 = 0. \tag{7.12}$$

Notice that the essential matrix $\boldsymbol{E}$ maps a point $\hat{\boldsymbol{x}}_0$ in image 0 into a line $\boldsymbol{l}_1 = \boldsymbol{E}\hat{\boldsymbol{x}}_0$ in image 1, since $\hat{\boldsymbol{x}}_1^T\boldsymbol{l}_1 = 0$ (Figure 7.3). All such lines must pass through the second *epipole* $\boldsymbol{e}_1$, which is therefore defined as the left singular vector of $\boldsymbol{E}$ with a 0 singular value, or, equivalently, the projection of the vector $\boldsymbol{t}$ into image 1. The dual (transpose) of these relationships gives us the epipolar line in the first image as $\boldsymbol{l}_0 = \boldsymbol{E}^T\hat{\boldsymbol{x}}_1$ and $\boldsymbol{e}_0$ as the zero-value right singular vector of $\boldsymbol{E}$.

Given this fundamental relationship (7.10), how can we use it to recover the camera motion encoded in the essential matrix $\boldsymbol{E}$? If we have N corresponding measurements $\{(\boldsymbol{x}_{i0}, \boldsymbol{x}_{i1})\}$, we can form N homogeneous equations in the nine elements of $\boldsymbol{E} = \{e_{00} \dots e_{22}\}$,

$$\begin{aligned} x_{i0}x_{i1}e_{00} &+ y_{i0}x_{i1}e_{01} &+ x_{i1}e_{02} &+ \\ x_{i0}y_{i1}e_{00} &+ y_{i0}y_{i1}e_{11} &+ y_{i1}e_{12} &+ \\ x_{i0}e_{20} &+ y_{i0}e_{21} &+ e_{22} &= 0 \end{aligned} \tag{7.13}$$

where $\boldsymbol{x}_{ij} = (x_{ij}, y_{ij}, 1)$. This can be written more compactly as

$$[\boldsymbol{x}_{i1}\, \boldsymbol{x}_{i0}^T] \otimes \boldsymbol{E} = \boldsymbol{Z}_i \otimes \boldsymbol{E} = \boldsymbol{z}_i \cdot \boldsymbol{f} = 0, \tag{7.14}$$

where $\otimes$ indicates an element-wise multiplication and summation of matrix elements, and $\boldsymbol{z}_i$ and $\boldsymbol{f}$ are the rasterized (vector) forms of the $\boldsymbol{Z}_i = \hat{\boldsymbol{x}}_{i1}\hat{\boldsymbol{x}}_{i0}^T$ and $\boldsymbol{E}$ matrices.[2] Given $N \geq 8$ such equations, we can compute an estimate (up to scale) for the entries in $\boldsymbol{E}$ using an SVD.

In the presence of noisy measurements, how close is this estimate to being statistically optimal? If you look at the entries in (7.13), you can see that some entries are the products of image measurements such as $x_{i0}y_{i1}$ and others are direct image measurements (or even the identity). If the measurements have comparable noise, the terms that are products of measurements have their noise amplified by the other element in the product, which can lead to very poor scaling, e.g., an inordinately large influence of points with large coordinates (far away from the image center).

In order to counteract this trend, Hartley (1997a) suggests that the point coordinates should be translated and scaled so that their centroid lies at the origin and their variance is unity, i.e.,

$$\tilde{x}_i = s(x_i - \mu_x) \tag{7.15}$$

$$\tilde{y}_i = s(x_i - \mu_y) \tag{7.16}$$

[2] We use $\boldsymbol{f}$ instead of $\boldsymbol{e}$ to denote the rasterized form of $\boldsymbol{E}$ to avoid confusion with the epipoles $\boldsymbol{e}_j$.

such that $\sum_i \tilde{x}_i = \sum_i \tilde{y}_i = 0$ and $\sum_i \tilde{x}_i^2 + \sum_i \tilde{y}_i^2 = 2n$, where n is the number of points.[3]

Once the essential matrix $\tilde{\boldsymbol{E}}$ has been computed from the transformed coordinates $\{(\tilde{\boldsymbol{x}}_{i0}, \tilde{\boldsymbol{x}}_{i1})\}$, where $\tilde{\boldsymbol{x}}_{ij} = \boldsymbol{T}_j \hat{\boldsymbol{x}}_{ij}$, the original essential matrix $\boldsymbol{E}$ can be recovered as

$$\boldsymbol{E} = \boldsymbol{T}_1 \tilde{\boldsymbol{E}} \boldsymbol{T}_0. \tag{7.17}$$

In his paper, Hartley (1997a) compares the improvement due to his re-normalization strategy to alternative distance measures proposed by others such as Zhang (1998a,b) and concludes that his simple re-normalization in most cases is as effective as (or better than) alternative techniques. Torr and Fitzgibbon (2004) recommend a variant on this algorithm where the norm of the upper 2×2 sub-matrix of $\boldsymbol{E}$ is set to 1 and show that it has even better stability with respect to 2D coordinate transformations.

Once an estimate for the essential matrix $\boldsymbol{E}$ has been recovered, the direction of the translation vector $\boldsymbol{t}$ can be estimated. Note that the absolute distance between the two cameras can never be recovered from pure image measurements alone, regardless of how many cameras or points are used. Knowledge about absolute camera and point positions or distances, often called *ground control points* in photogrammetry, is always required to establish the final scale, position, and orientation.

To estimate this direction $\hat{\boldsymbol{t}}$, observe that under ideal noise-free conditions, the essential matrix $\boldsymbol{E}$ is singular, i.e., $\hat{\boldsymbol{t}}^T \boldsymbol{E} = 0$. This singularity shows up as a singular value of 0 when an SVD of $\boldsymbol{E}$ is performed,

$$\boldsymbol{E} = [\hat{\boldsymbol{t}}]_\times \boldsymbol{R} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \begin{bmatrix} \boldsymbol{u}_0 & \boldsymbol{u}_1 & \hat{\boldsymbol{t}} \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_0^T \\ \boldsymbol{v}_1^T \\ \boldsymbol{v}_2^T \end{bmatrix} \tag{7.18}$$

When $\boldsymbol{E}$ is computed from noisy measurements, the singular vector associated with the smallest singular value gives us $\hat{\boldsymbol{t}}$. (The other two singular values should be similar but are not, in general, equal to 1 because $\boldsymbol{E}$ is only computed up to an unknown scale.)

Because $\boldsymbol{E}$ is rank-deficient, it turns out that we actually only need seven correspondences of the form of Equation (7.14) instead of eight to estimate this matrix (Hartley 1994a; Torr and Murray 1997; Hartley and Zisserman 2004). (The advantage of using fewer correspondences inside a RANSAC robust fitting stage is that fewer random samples need to be generated.) From this set of seven homogeneous equations (which we can stack into a 7×9 matrix for SVD analysis), we can find two independent vectors, say $\boldsymbol{f}_0$ and $\boldsymbol{f}_1$ such that $\boldsymbol{z}_i \cdot \boldsymbol{f}_j = 0$. These two vectors can be converted back into 3×3 matrices $\boldsymbol{E}_0$ and $\boldsymbol{E}_1$, which span the solution space for

$$\boldsymbol{E} = \alpha \boldsymbol{E}_0 + (1 - \alpha) \boldsymbol{E}_1. \tag{7.19}$$

To find the correct value of α, we observe that $\boldsymbol{E}$ has a zero determinant, since it is rank deficient, and hence

$$\det |\alpha \boldsymbol{E}_0 + (1 - \alpha) \boldsymbol{E}_1| = 0. \tag{7.20}$$

This gives us a cubic equation in α, which has either one or three solutions (roots). Substituting these values into (7.19) to obtain $\boldsymbol{E}$, we can test this essential matrix against other unused feature correspondences to select the correct one.

[3] More precisely, Hartley (1997a) suggests scaling the points "so that the average distance from the origin is equal to $\sqrt{2}$" but the heuristic of unit variance is faster to compute (does not require per-point square roots) and should yield comparable improvements.

Once $\hat{\boldsymbol{t}}$ has been recovered, how can we estimate the corresponding rotation matrix $\boldsymbol{R}$? Recall that the cross-product operator $[\hat{\boldsymbol{t}}]_\times$ (2.32) projects a vector onto a set of orthogonal basis vectors that include $\hat{\boldsymbol{t}}$, zeros out the $\hat{\boldsymbol{t}}$ component, and rotates the other two by 90°,

$$[\hat{\boldsymbol{t}}]_\times = \boldsymbol{S}\boldsymbol{Z}\boldsymbol{R}_{90^\circ}\boldsymbol{S}^T = \begin{bmatrix} \boldsymbol{s}_0 & \boldsymbol{s}_1 & \hat{\boldsymbol{t}} \end{bmatrix} \begin{bmatrix} 1 & & \\ & 1 & \\ & & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & \\ 1 & 0 & \\ & & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{s}_0^T \\ \boldsymbol{s}_1^T \\ \hat{\boldsymbol{t}}^T \end{bmatrix}, \tag{7.21}$$

where $\hat{\boldsymbol{t}} = \boldsymbol{s}_0 \times \boldsymbol{s}_1$. From Equations (7.18 and 7.21), we get

$$\boldsymbol{E} = [\hat{\boldsymbol{t}}]_\times \boldsymbol{R} = \boldsymbol{S}\boldsymbol{Z}\boldsymbol{R}_{90^\circ}\boldsymbol{S}^T\boldsymbol{R} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T, \tag{7.22}$$

from which we can conclude that $\boldsymbol{S} = \boldsymbol{U}$. Recall that for a noise-free essential matrix, ($\boldsymbol{\Sigma} = \boldsymbol{Z}$), and hence

$$\boldsymbol{R}_{90^\circ}\boldsymbol{U}^T\boldsymbol{R} = \boldsymbol{V}^T \tag{7.23}$$

and

$$\boldsymbol{R} = \boldsymbol{U}\boldsymbol{R}_{90^\circ}^T\boldsymbol{V}^T. \tag{7.24}$$

Unfortunately, we only know both $\boldsymbol{E}$ and $\hat{\boldsymbol{t}}$ up to a sign. Furthermore, the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are not guaranteed to be rotations (you can flip both their signs and still get a valid SVD). For this reason, we have to generate all four possible rotation matrices

$$\boldsymbol{R} = \pm\boldsymbol{U}\boldsymbol{R}_{\pm 90^\circ}^T\boldsymbol{V}^T \tag{7.25}$$

and keep the two whose determinant $|\boldsymbol{R}| = 1$. To disambiguate between the remaining pair of potential rotations, which form a *twisted pair* (Hartley and Zisserman 2004, p. 240), we need to pair them with both possible signs of the translation direction $\pm\hat{\boldsymbol{t}}$ and select the combination for which the largest number of points is seen in front of both cameras.[4]

The property that points must lie in front of the camera, i.e., at a positive distance along the viewing rays emanating from the camera, is known as *chirality* (Hartley 1998). In addition to determining the signs of the rotation and translation, as described above, the chirality (sign of the distances) of the points in a reconstruction can be used inside a RANSAC procedure (along with the reprojection errors) to distinguish between likely and unlikely configurations.[5] Chirality can also be used to transform projective reconstructions (Sections 7.2.1 and 7.2.2) into *quasi-affine* reconstructions (Hartley 1998).

The normalized "eight-point algorithm" (Hartley 1997a) described above is not the only way to estimate the camera motion from correspondences. Variants include using seven points while enforcing the rank two constraint in $\boldsymbol{E}$ (7.19–7.20) and a five-point algorithm that requires finding the roots of a 10th degree polynomial (Nistér 2004). Since such algorithms use fewer points to compute their estimates, they are less sensitive to outliers when used as part of a random sampling (RANSAC) strategy.

[4] In the noise-free case, a single point suffices. It is safer, however, to test all or a sufficient subset of points, downweighting the ones that lie close to the plane at infinity, for which it is easy to get depth reversals.

[5] Note that as points get further away from a camera, i.e., closer toward the plane at infinity, errors in chirality become more likely.

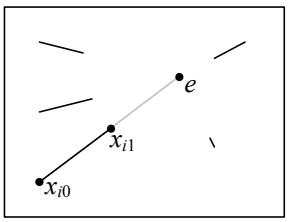


Figure 7.4 Pure translational camera motion results in visual motion where all the points move towards (or away from) a common *focus of expansion* (FOE) $\boldsymbol{e}$. They therefore satisfy the triple product condition $(\boldsymbol{x}_0, \boldsymbol{x}_1, \boldsymbol{e}) = \boldsymbol{e} \cdot (\boldsymbol{x}_0 \times \boldsymbol{x}_1) = 0$.

Pure translation (known rotation)

In the case where we know the rotation, we can pre-rotate the points in the second image to match the viewing direction of the first. The resulting set of 3D points all move towards (or away from) the *focus of expansion* (FOE), as shown in Figure 7.4.[6] The resulting essential matrix $\boldsymbol{E}$ is (in the noise-free case) skew symmetric and so can be estimated more directly by setting $e_{ij} = -e_{ji}$ and $e_{ii} = 0$ in (7.13). Two points with non-zero parallax now suffice to estimate the FOE.

A more direct derivation of the FOE estimate can be obtained by minimizing the triple product

$$\sum_i (\boldsymbol{x}_{i0}, \boldsymbol{x}_{i1}, \boldsymbol{e})^2 = \sum_i ((\boldsymbol{x}_{i0} \times \boldsymbol{x}_{i1}) \cdot \boldsymbol{e})^2, \tag{7.26}$$

which is equivalent to finding the null space for the set of equations

$$(y_{i0} - y_{i1})e_0 + (x_{i1} - x_{i0})e_1 + (x_{i0}y_{i1} - y_{i0}x_{i1})e_2 = 0. \tag{7.27}$$

Note that, as in the eight-point algorithm, it is advisable to normalize the 2D points to have unit variance before computing this estimate.

In situations where a large number of points at infinity are available, e.g., when shooting outdoor scenes or when the camera motion is small compared to distant objects, this suggests an alternative RANSAC strategy for estimating the camera motion. First, pick a pair of points to estimate a rotation, hoping that both of the points lie at infinity (very far from the camera). Then, compute the FOE and check whether the residual error is small (indicating agreement with this rotation hypothesis) and whether the motions towards or away from the epipole (FOE) are all in the same direction (ignoring very small motions, which may be noise-contaminated).

Pure rotation

The case of pure rotation results in a degenerate estimate of the essential matrix $\boldsymbol{E}$ and of the translation direction $\hat{\boldsymbol{t}}$. Consider first the case of the rotation matrix being known. The estimates for the FOE will be degenerate, since $\boldsymbol{x}_{i0} \approx \boldsymbol{x}_{i1}$, and hence (7.27), is degenerate. A similar argument shows that the equations for the essential matrix (7.13) are also rank-deficient.

[6] Fans of *Star Trek* and *Star Wars* will recognize this as the "jump to hyperdrive" visual effect.

This suggests that it might be prudent before computing a full essential matrix to first compute a rotation estimate $\boldsymbol{R}$ using (6.32), potentially with just a small number of points, and then compute the residuals after rotating the points before proceeding with a full $\boldsymbol{E}$ computation.

7.2.1 Projective (uncalibrated) reconstruction

In many cases, such as when trying to build a 3D model from Internet or legacy photos taken by unknown cameras without any EXIF tags, we do not know ahead of time the intrinsic calibration parameters associated with the input images. In such situations, we can still estimate a two-frame reconstruction, although the true metric structure may not be available, e.g., orthogonal lines or planes in the world may not end up being reconstructed as orthogonal.

Consider the derivations we used to estimate the essential matrix $\boldsymbol{E}$ (7.10–7.12). In the uncalibrated case, we do not know the calibration matrices $\boldsymbol{K}_j$, so we cannot use the normalized ray directions $\hat{\boldsymbol{x}}_j = \boldsymbol{K}_j^{-1}\boldsymbol{x}_j$. Instead, we have access only to the image coordinates $\boldsymbol{x}_j$, and so the essential matrix (7.10) becomes

$$\hat{\boldsymbol{x}}_1^T \boldsymbol{E} \hat{\boldsymbol{x}}_1 = \boldsymbol{x}_1^T \boldsymbol{K}_1^{-T} \boldsymbol{E} \boldsymbol{K}_0^{-1} \boldsymbol{x}_0 = \boldsymbol{x}_1^T \boldsymbol{F} \boldsymbol{x}_0 = 0, \tag{7.28}$$

where

$$\boldsymbol{F} = \boldsymbol{K}_1^{-T} \boldsymbol{E} \boldsymbol{K}_0^{-1} = [\boldsymbol{e}]_\times \tilde{\boldsymbol{H}} \tag{7.29}$$

is called the *fundamental matrix* (Faugeras 1992; Hartley, Gupta, and Chang 1992; Hartley and Zisserman 2004).

Like the essential matrix, the fundamental matrix is (in principle) rank two,

$$\boldsymbol{F} = [\boldsymbol{e}]_\times \tilde{\boldsymbol{H}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = \begin{bmatrix} \boldsymbol{u}_0 & \boldsymbol{u}_1 & \boldsymbol{e}_1 \end{bmatrix} \begin{bmatrix} \sigma_0 & & \\ & \sigma_1 & \\ & & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{v}_0^T \\ \boldsymbol{v}_1^T \\ \boldsymbol{e}_0^T \end{bmatrix}. \tag{7.30}$$

Its smallest left singular vector indicates the epipole $\boldsymbol{e}_1$ in the image 1 and its smallest right singular vector is $\boldsymbol{e}_0$ (Figure 7.3). The homography $\tilde{\boldsymbol{H}}$ in (7.29), which in principle should equal

$$\tilde{\boldsymbol{H}} = \boldsymbol{K}_1^{-T} \boldsymbol{R} \boldsymbol{K}_0^{-1}, \tag{7.31}$$

cannot be uniquely recovered from $\boldsymbol{F}$, since any homography of the form $\tilde{\boldsymbol{H}}' = \tilde{\boldsymbol{H}} + \boldsymbol{e}\boldsymbol{v}^T$ results in the same $\boldsymbol{F}$ matrix. (Note that $[\boldsymbol{e}]_\times$ annihilates any multiple of $\boldsymbol{e}$.)

Any one of these valid homographies $\tilde{\boldsymbol{H}}$ maps some plane in the scene from one image to the other. It is not possible to tell in advance which one it is without either selecting four or more co-planar correspondences to compute $\tilde{\boldsymbol{H}}$ as part of the $\boldsymbol{F}$ estimation process (in a manner analogous to guessing a rotation for $\boldsymbol{E}$) or mapping all points in one image through $\tilde{\boldsymbol{H}}$ and seeing which ones line up with their corresponding locations in the other.[7]

In order to create a *projective* reconstruction of the scene, we can pick any valid homography $\tilde{\boldsymbol{H}}$ that satisfies Equation (7.29). For example, following a technique analogous to Equations (7.18–7.24), we get

$$\boldsymbol{F} = [\boldsymbol{e}]_\times \tilde{\boldsymbol{H}} = \boldsymbol{S}\boldsymbol{Z}\boldsymbol{R}_{90^\circ}\boldsymbol{S}^T\tilde{\boldsymbol{H}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T \tag{7.32}$$

[7] This process is sometimes referred to as *plane plus parallax* (Section 2.1.5) (Kumar, Anandan, and Hanna 1994; Sawhney 1994).

and hence

$$\tilde{\boldsymbol{H}} = \boldsymbol{U}\boldsymbol{R}_{90^\circ}^T \hat{\boldsymbol{\Sigma}} \boldsymbol{V}^T, \tag{7.33}$$

where $\hat{\boldsymbol{\Sigma}}$ is the singular value matrix with the smallest value replaced by a reasonable alternative (say, the middle value).[8] We can then form a pair of camera matrices

$$\boldsymbol{P}_0 = [\boldsymbol{I}|\boldsymbol{0}] \quad \text{and} \quad \boldsymbol{P}_0 = [\tilde{\boldsymbol{H}}|\boldsymbol{e}], \tag{7.34}$$

from which a projective reconstruction of the scene can be computed using triangulation (Section 7.1).

While the projective reconstruction may not be useful in practice, it can often be *upgraded* to an affine or metric reconstruction, as detailed below. Even without this step, however, the fundamental matrix $\boldsymbol{F}$ can be very useful in finding additional correspondences, as they must all lie on corresponding epipolar lines, i.e., any feature $\boldsymbol{x}_0$ in image 0 must have its correspondence lying on the associated epipolar line $\boldsymbol{l}_1 = \boldsymbol{F}\boldsymbol{x}_0$ in image 1, assuming that the point motions are due to a rigid transformation.

7.2.2 Self-calibration

The results of structure from motion computation are much more useful (and intelligible) if a *metric* reconstruction is obtained, i.e., one in which parallel lines are parallel, orthogonal walls are at right angles, and the reconstructed model is a scaled version of reality. Over the years, a large number of *self-calibration* (or *auto-calibration*) techniques have been developed for converting a projective reconstruction into a metric one, which is equivalent to recovering the unknown calibration matrices $\boldsymbol{K}_j$ associated with each image (Hartley and Zisserman 2004; Moons, Van Gool, and Vergauwen 2010).

In situations where certain additional information is known about the scene, different methods may be employed. For example, if there are parallel lines in the scene (usually, having several lines converge on the same vanishing point is good evidence), three or more vanishing points, which are the images of points at infinity, can be used to establish the homography for the plane at infinity, from which focal lengths and rotations can be recovered. If two or more finite *orthogonal* vanishing points have been observed, the single-image calibration method based on vanishing points (Section 6.3.2) can be used instead.

In the absence of such external information, it is not possible to recover a fully parameterized independent calibration matrix $\boldsymbol{K}_j$ for each image from correspondences alone. To see this, consider the set of all camera matrices $\boldsymbol{P}_j = \boldsymbol{K}_j[\boldsymbol{R}_j|\boldsymbol{t}_j]$ projecting world coordinates $\boldsymbol{p}_i = (X_i, Y_i, Z_i, W_i)$ into screen coordinates $\boldsymbol{x}_{ij} \sim \boldsymbol{P}_j\boldsymbol{p}_i$. Now consider transforming the 3D scene $\{\boldsymbol{p}_i\}$ through an arbitrary 4×4 projective transformation $\tilde{\boldsymbol{H}}$, yielding a new model consisting of points $\boldsymbol{p}'_i = \tilde{\boldsymbol{H}}\boldsymbol{p}_i$. Post-multiplying each $\boldsymbol{P}_j$ matrix by $\tilde{\boldsymbol{H}}^{-1}$ still produces the same screen coordinates and a new set calibration matrices can be computed by applying RQ decomposition to the new camera matrix $\boldsymbol{P}'_j = \boldsymbol{P}_j\tilde{\boldsymbol{H}}^{-1}$.

For this reason, all self-calibration methods assume some restricted form of the calibration matrix, either by setting or equating some of their elements or by assuming that they do not vary over time. While most of the techniques discussed by Hartley and Zisserman (2004);

[8] Hartley and Zisserman (2004, p. 237) recommend using $\tilde{\boldsymbol{H}} = [\boldsymbol{e}]_\times \boldsymbol{F}$ (Luong and Viéville 1996), which places the camera on the plane at infinity.

Moons, Van Gool, and Vergauwen (2010) require three or more frames, in this section we present a simple technique that can recover the focal lengths (f_0, f_1) of both images from the fundamental matrix $\boldsymbol{F}$ in a two-frame reconstruction (Hartley and Zisserman 2004, p. 456).

To accomplish this, we assume that the camera has zero skew, a known aspect ratio (usually set to 1), and a known optical center, as in Equation (2.59). How reasonable is this assumption in practice? The answer, as with many questions, is "it depends".

If absolute metric accuracy is required, as in photogrammetry applications, it is imperative to pre-calibrate the cameras using one of the techniques from Section 6.3 and to use ground control points to pin down the reconstruction. If instead, we simply wish to reconstruct the world for visualization or image-based rendering applications, as in the Photo Tourism system of Snavely, Seitz, and Szeliski (2006), this assumption is quite reasonable in practice.

Most cameras today have square pixels and an optical center near the middle of the image, and are much more likely to deviate from a simple camera model due to radial distortion (Section 6.3.5), which should be compensated for whenever possible. The biggest problems occur when images have been cropped off-center, in which case the optical center will no longer be in the middle, or when perspective pictures have been taken of a different picture, in which case a general camera matrix becomes necessary.[9]

Given these caveats, the two-frame focal length estimation algorithm based on the Kruppa equations developed by Hartley and Zisserman (2004, p. 456) proceeds as follows. Take the left and right singular vectors $\{\boldsymbol{u}_0, \boldsymbol{u}_1, \boldsymbol{v}_0, \boldsymbol{v}_1\}$ of the fundamental matrix $\boldsymbol{F}$ (7.30) and their associated singular values $\{\sigma_0, \sigma_1\}$ and form the following set of equations:

$$\frac{\boldsymbol{u}_1^T \boldsymbol{D}_0 \boldsymbol{u}_1}{\sigma_0^2 \boldsymbol{v}_0^T \boldsymbol{D}_1 \boldsymbol{v}_0} = -\frac{\boldsymbol{u}_0^T \boldsymbol{D}_0 \boldsymbol{u}_1}{\sigma_0 \sigma_1 \boldsymbol{v}_0^T \boldsymbol{D}_1 \boldsymbol{v}_1} = \frac{\boldsymbol{u}_0^T \boldsymbol{D}_0 \boldsymbol{u}_0}{\sigma_1^2 \boldsymbol{v}_1^T \boldsymbol{D}_1 \boldsymbol{v}_1}, \tag{7.35}$$

where the two matrices

$$\boldsymbol{D}_j = \boldsymbol{K}_j \boldsymbol{K}_j^T = \mathrm{diag}(f_j^2, f_j^2, 1) = \begin{bmatrix} f_j^2 & & \\ & f_j^2 & \\ & & 1 \end{bmatrix} \tag{7.36}$$

encode the unknown focal lengths. For simplicity, let us rewrite each of the numerators and denominators in (7.35) as

$$e_{ij0}(f_0^2) = \boldsymbol{u}_i^T \boldsymbol{D}_0 \boldsymbol{u}_j = a_{ij} + b_{ij} f_0^2, \tag{7.37}$$

$$e_{ij1}(f_1^2) = \sigma_i \sigma_j \boldsymbol{v}_i^T \boldsymbol{D}_1 \boldsymbol{v}_j = c_{ij} + d_{ij} f_1^2. \tag{7.38}$$

Notice that each of these is affine (linear plus constant) in either f_0^2 or f_1^2. Hence, we can cross-multiply these equations to obtain quadratic equations in f_j^2, which can readily be solved. (See also the work by Bougnoux (1998) for some alternative formulations.)

An alternative solution technique is to observe that we have a set of three equations related by an unknown scalar λ, i.e.,

$$e_{ij0}(f_0^2) = \lambda e_{ij1}(f_1^2) \tag{7.39}$$

(Richard Hartley, personal communication, July 2009). These can readily be solved to yield $(f_0^2, \lambda f_1^2, \lambda)$ and hence (f_0, f_1).

[9] In Photo Tourism, our system registered photographs of an information sign outside Notre Dame with real pictures of the cathedral.

How well does this approach work in practice? There are certain degenerate configurations, such as when there is no rotation or when the optical axes intersect, when it does not work at all. (In such a situation, you can vary the focal lengths of the cameras and obtain a deeper or shallower reconstruction, which is an example of a *bas-relief ambiguity* (Section 7.4.3).) Hartley and Zisserman (2004) recommend using techniques based on three or more frames. However, if you find two images for which the estimates of $(f_0^2, \lambda f_1^2, \lambda)$ are well conditioned, they can be used to initialize a more complete bundle adjustment of all the parameters (Section 7.4). An alternative, which is often used in systems such as Photo Tourism, is to use camera EXIF tags or generic default values to initialize focal length estimates and refine them as part of bundle adjustment.

7.2.3 *Application*: View morphing

An interesting application of basic two-frame structure from motion is *view morphing* (also known as *view interpolation*, see Section 13.1), which can be used to generate a smooth 3D animation from one view of a 3D scene to another (Chen and Williams 1993; Seitz and Dyer 1996).

To create such a transition, you must first smoothly interpolate the camera matrices, i.e., the camera positions, orientations, and focal lengths. While simple linear interpolation can be used (representing rotations as quaternions (Section 2.1.4)), a more pleasing effect is obtained by *easing in* and *easing out* the camera parameters, e.g., using a raised cosine, as well as moving the camera along a more circular trajectory (Snavely, Seitz, and Szeliski 2006).

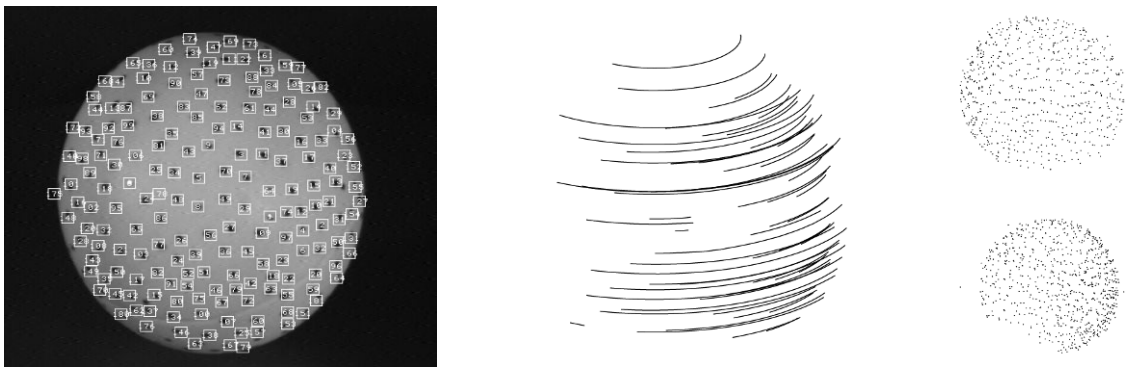
To generate in-between frames, either a full set of 3D correspondences needs to be established (Section 11.3) or 3D models (proxies) must be created for each reference view. Section 13.1 describes several widely used approaches to this problem. One of the simplest is to just triangulate the set of matched feature points in each image, e.g., using Delaunay triangulation. As the 3D points are re-projected into their intermediate views, pixels can be mapped from their original source images to their new views using affine or projective mapping (Szeliski and Shum 1997). The final image is then composited using a linear blend of the two reference images, as with usual morphing (Section 3.6.3).

7.3 Factorization

When processing video sequences, we often get extended *feature tracks* (Section 4.1.4) from which it is possible to recover the structure and motion using a process called *factorization*. Consider the tracks generated by a rotating ping pong ball, which has been marked with dots to make its shape and motion more discernable (Figure 7.5). We can readily see from the shape of the tracks that the moving object must be a sphere, but how can we infer this mathematically?

It turns out that, under orthography or related models we discuss below, the shape and motion can be recovered simultaneously using a singular value decomposition (Tomasi and Kanade 1992). Consider the orthographic and weak perspective projection models introduced in Equations (2.47–2.49). Since the last row is always $[0\ 0\ 0\ 1]$, there is no perspective division and we can write

$$\boldsymbol{x}_{ji} = \tilde{\boldsymbol{P}}_j \bar{\boldsymbol{p}}_i, \tag{7.40}$$



(a) (b) (c)

Figure 7.5 3D reconstruction of a rotating ping pong ball using factorization (Tomasi and Kanade 1992) © 1992 Springer: (a) sample image with tracked features overlaid; (b) subsampled feature motion stream; (c) two views of the reconstructed 3D model.

where $\boldsymbol{x}_{ji}$ is the location of the ith point in the jth frame, $\tilde{\boldsymbol{P}}_j$ is the upper 2×4 portion of the projection matrix $\boldsymbol{P}_j$, and $\bar{\boldsymbol{p}}_i = (X_i, Y_i, Z_i, 1)$ is the augmented 3D point position.[10]

Let us assume (for now) that every point i is visible in every frame j. We can take the *centroid* (average) of the projected point locations $\boldsymbol{x}_{ji}$ in frame j,

$$\bar{\boldsymbol{x}}_j = \frac{1}{N}\sum_i \boldsymbol{x}_{ji} = \tilde{\boldsymbol{P}}_j \frac{1}{N}\sum_i \bar{\boldsymbol{p}}_i = \tilde{\boldsymbol{P}}_j \bar{\boldsymbol{c}}, \tag{7.41}$$

where $\bar{\boldsymbol{c}} = (\bar{X}, \bar{Y}, \bar{Z}, 1)$ is the augmented 3D centroid of the point cloud.

Since world coordinate frames in structure from motion are always arbitrary, i.e., we cannot recover true 3D locations without ground control points (known measurements), we can place the origin of the world at the centroid of the points, i.e, $\bar{X} = \bar{Y} = \bar{Z} = 0$, so that $\bar{\boldsymbol{c}} = (0, 0, 0, 1)$. We see from this that the centroid of the 2D points in each frame $\bar{\boldsymbol{x}}_j$ directly gives us the last element of $\tilde{\boldsymbol{P}}_j$.

Let $\tilde{\boldsymbol{x}}_{ji} = \boldsymbol{x}_{ji} - \bar{\boldsymbol{x}}_j$ be the 2D point locations after their image centroid has been subtracted. We can now write

$$\tilde{\boldsymbol{x}}_{ji} = \boldsymbol{M}_j \boldsymbol{p}_i, \tag{7.42}$$

where $\boldsymbol{M}_j$ is the upper 2×3 portion of the projection matrix $\boldsymbol{P}_j$ and $\boldsymbol{p}_i = (X_i, Y_i, Z_i)$. We can concatenate all of these measurement equations into one large matrix

$$\hat{\boldsymbol{X}} = \begin{bmatrix} \tilde{\boldsymbol{x}}_{11} & \cdots & \tilde{\boldsymbol{x}}_{1i} & \cdots & \tilde{\boldsymbol{x}}_{1N} \\ \vdots & & \vdots & & \vdots \\ \tilde{\boldsymbol{x}}_{j1} & \cdots & \tilde{\boldsymbol{x}}_{ji} & \cdots & \tilde{\boldsymbol{x}}_{jN} \\ \vdots & & \vdots & & \vdots \\ \tilde{\boldsymbol{x}}_{M1} & \cdots & \tilde{\boldsymbol{x}}_{Mi} & \cdots & \tilde{\boldsymbol{x}}_{MN} \end{bmatrix} = \begin{bmatrix} \boldsymbol{M}_1 \\ \vdots \\ \boldsymbol{M}_j \\ \vdots \\ \boldsymbol{M}_M \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_1 & \cdots & \boldsymbol{p}_i & \cdots & \boldsymbol{p}_N \end{bmatrix} = \hat{\boldsymbol{M}}\hat{\boldsymbol{S}}. \tag{7.43}$$

$\hat{\boldsymbol{X}}$ is called the *measurement* matrix and $\hat{\boldsymbol{M}}$ and ($\hat{\boldsymbol{S}}$ are the *motion*) and *structure* matrices, respectively (Tomasi and Kanade 1992).

[10] In this section, we index the 2D point positions as $\boldsymbol{x}_{ji}$ instead of $\boldsymbol{x}_{ij}$, since this is the convention adopted by factorization papers (Tomasi and Kanade 1992) and is consistent with the factorization given in (7.43).

Because the motion matrix $\hat{\boldsymbol{M}}$ is $2M \times 3$ and the structure matrix $\hat{\boldsymbol{S}}$ is $3 \times N$, an SVD applied to $\hat{\boldsymbol{X}}$ has only three non-zero singular values. In the case where the measurements in $\hat{\boldsymbol{X}}$ are noisy, SVD returns the rank-three factorization of $\hat{\boldsymbol{X}}$ that is the closest to $\hat{\boldsymbol{X}}$ in a least squares sense (Tomasi and Kanade 1992; Golub and Van Loan 1996; Hartley and Zisserman 2004).

It would be nice if the SVD of $\hat{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T$ directly returned the matrices $\hat{\boldsymbol{M}}$ and $\hat{\boldsymbol{S}}$, but it does not. Instead, we can write the relationship

$$\hat{\boldsymbol{X}} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^T = [\boldsymbol{U}\boldsymbol{Q}][\boldsymbol{Q}^{-1}\boldsymbol{\Sigma}\boldsymbol{V}^T] \tag{7.44}$$

and set $\hat{\boldsymbol{M}} = \boldsymbol{U}\boldsymbol{Q}$ and $\hat{\boldsymbol{S}} = \boldsymbol{Q}^{-1}\boldsymbol{\Sigma}\boldsymbol{V}^T$.[11]

How can we recover the values of the 3×3 matrix $\boldsymbol{Q}$? This depends on the motion model being used. In the case of orthographic projection (2.47), the entries in $\boldsymbol{M}_j$ are the first two rows of rotation matrices $\boldsymbol{R}_j$, so we have

$$\begin{aligned} \boldsymbol{m}_{j0} \cdot \boldsymbol{m}_{j0} &= \boldsymbol{u}_{2j}\boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{u}_{2j}^T &&= 1, \\ \boldsymbol{m}_{j0} \cdot \boldsymbol{m}_{j1} &= \boldsymbol{u}_{2j}\boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{u}_{2j+1}^T &&= 0, \\ \boldsymbol{m}_{j1} \cdot \boldsymbol{m}_{j1} &= \boldsymbol{u}_{2j+1}\boldsymbol{Q}\boldsymbol{Q}^T\boldsymbol{u}_{2j+1}^T &&= 1, \end{aligned} \tag{7.45}$$

where $\boldsymbol{u}_k$ are the 3×1 *rows* of the matrix $\boldsymbol{U}$. This gives us a large set of equations for the entries in the matrix $\boldsymbol{Q}\boldsymbol{Q}^T$, from which the matrix $\boldsymbol{Q}$ can be recovered using a matrix square root (Appendix A.1.4). If we have scaled orthography (2.48), i.e., $\boldsymbol{M}_j = s_j\boldsymbol{R}_j$, the first and third equations are equal to s_j and can be set equal to each other.

Note that even once $\boldsymbol{Q}$ has been recovered, there still exists a bas-relief ambiguity, i.e., we can never be sure if the object is rotating left to right or if its depth reversed version is moving the other way. (This can be seen in the classic rotating Necker Cube visual illusion.) Additional cues, such as the appearance and disappearance of points, or perspective effects, both of which are discussed below, can be used to remove this ambiguity.

For motion models other than pure orthography, e.g., for scaled orthography or para-perspective, the approach above must be extended in the appropriate manner. Such techniques are relatively straightforward to derive from first principles; more details can be found in papers that extend the basic factorization approach to these more flexible models (Poelman and Kanade 1997). Additional extensions of the original factorization algorithm include multi-body rigid motion (Costeira and Kanade 1995), sequential updates to the factorization (Morita and Kanade 1997), the addition of lines and planes (Morris and Kanade 1998), and re-scaling the measurements to incorporate individual location uncertainties (Anandan and Irani 2002).

A disadvantage of factorization approaches is that they require a complete set of tracks, i.e., each point must be visible in each frame, in order for the factorization approach to work. Tomasi and Kanade (1992) deal with this problem by first applying factorization to smaller denser subsets and then using known camera (motion) or point (structure) estimates to *hallucinate* additional missing values, which allows them to incrementally incorporate more features and cameras. Huynh, Hartley, and Heyden (2003) extend this approach to view missing data as special cases of outliers. Buchanan and Fitzgibbon (2005) develop fast iterative algorithms for performing large matrix factorizations with missing data. The general topic of

[11] Tomasi and Kanade (1992) first take the square root of $\boldsymbol{\Sigma}$ and distribute this to $\boldsymbol{U}$ and $\boldsymbol{V}$, but there is no particular reason to do this.

principal component analysis (PCA) with missing data also appears in other computer vision problems (Shum, Ikeuchi, and Reddy 1995; De la Torre and Black 2003; Gross, Matthews, and Baker 2006; Torresani, Hertzmann, and Bregler 2008; Vidal, Ma, and Sastry 2010).

7.3.1 Perspective and projective factorization

Another disadvantage of regular factorization is that it cannot deal with perspective cameras. One way to get around this problem is to perform an initial affine (e.g., orthographic) reconstruction and to then correct for the perspective effects in an iterative manner (Christy and Horaud 1996).

Observe that the object-centered projection model (2.76)

$$x_{ji} = s_j \frac{\boldsymbol{r}_{xj} \cdot \boldsymbol{p}_i + t_{xj}}{1 + \eta_j \boldsymbol{r}_{zj} \cdot \boldsymbol{p}_i} \tag{7.46}$$

$$y_{ji} = s_j \frac{\boldsymbol{r}_{yj} \cdot \boldsymbol{p}_i + t_{yj}}{1 + \eta_j \boldsymbol{r}_{zj} \cdot \boldsymbol{p}_i} \tag{7.47}$$

differs from the scaled orthographic projection model (7.40) by the inclusion of the denominator terms $(1 + \eta_j \boldsymbol{r}_{zj} \cdot \boldsymbol{p}_i)$.[12]

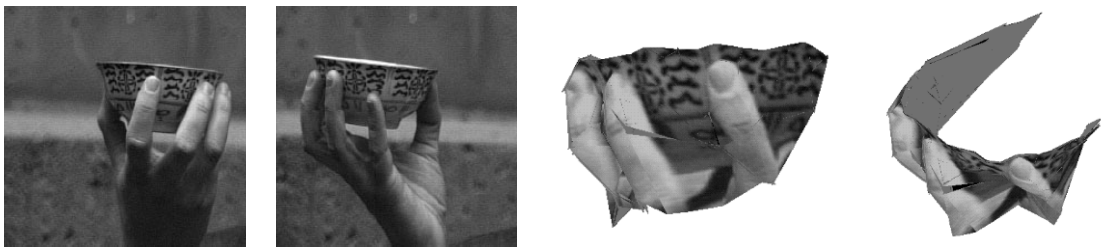
If we knew the correct values of $\eta_j = t_{zj}^{-1}$ and the structure and motion parameters $\boldsymbol{R}_j$ and $\boldsymbol{p}_i$, we could cross-multiply the left hand side (visible point measurements x_{ji} and y_{ji}) by the denominator and get corrected values, for which the bilinear projection model (7.40) is exact. In practice, after an initial reconstruction, the values of η_j can be estimated independently for each frame by comparing reconstructed and sensed point positions. (The third row of the rotation matrix $\boldsymbol{r}_{zj}$ is always available as the cross-product of the first two rows.) Note that since the η_j are determined from the image measurements, the cameras do not have to be pre-calibrated, i.e., their focal lengths can be recovered from $f_j = s_j/\eta_j$.

Once the η_j have been estimated, the feature locations can then be corrected before applying another round of factorization. Note that because of the initial depth reversal ambiguity, both reconstructions have to be tried while calculating η_j. (The incorrect reconstruction will result in a negative η_j, which is not physically meaningful.) Christy and Horaud (1996) report that their algorithm usually converges in three to five iterations, with the majority of the time spent in the SVD computation.

An alternative approach, which does not assume partially calibrated cameras (known optical center, square pixels, and zero skew) is to perform a fully *projective* factorization (Sturm and Triggs 1996; Triggs 1996). In this case, the inclusion of the third row of the camera matrix in (7.40) is equivalent to multiplying each reconstructed measurement $\boldsymbol{x}_{ji} = \boldsymbol{M}_j \boldsymbol{p}_i$ by its inverse (projective) depth $\eta_{ji} = d_{ji}^{-1} = 1/(\boldsymbol{P}_{j2}\boldsymbol{p}_i)$ or, equivalently, multiplying each measured position by its projective depth d_{ji},

$$\hat{\boldsymbol{X}} = \begin{bmatrix} d_{11}\tilde{\boldsymbol{x}}_{11} & \cdots & d_{1i}\tilde{\boldsymbol{x}}_{1i} & \cdots & d_{1N}\tilde{\boldsymbol{x}}_{1N} \\ \vdots & & \vdots & & \vdots \\ d_{j1}\tilde{\boldsymbol{x}}_{j1} & \cdots & d_{ji}\tilde{\boldsymbol{x}}_{ji} & \cdots & d_{jN}\tilde{\boldsymbol{x}}_{jN} \\ \vdots & & \vdots & & \vdots \\ d_{M1}\tilde{\boldsymbol{x}}_{M1} & \cdots & d_{Mi}\tilde{\boldsymbol{x}}_{Mi} & \cdots & d_{MN}\tilde{\boldsymbol{x}}_{MN} \end{bmatrix} = \hat{\boldsymbol{M}}\hat{\boldsymbol{S}}. \tag{7.48}$$

[12] Assuming that the optical center (c_x, c_y) lies at $(0, 0)$ and that pixels are square.



(a) (b) (c) (d)

Figure 7.6 3D teacup model reconstructed from a 240-frame video sequence (Tomasi and Kanade 1992) © 1992 Springer: (a) first frame of video; (b) last frame of video; (c) side view of 3D model; (d) top view of 3D model.

In the original paper by Sturm and Triggs (1996), the projective depths d_{ji} are obtained from two-frame reconstructions, while in later work (Triggs 1996; Oliensis and Hartley 2007), they are initialized to $d_{ji} = 1$ and updated after each iteration. Oliensis and Hartley (2007) present an update formula that is guaranteed to converge to a fixed point. None of these authors suggest actually estimating the third row of $\boldsymbol{P}_j$ as part of the projective depth computations. In any case, it is unclear when a fully projective reconstruction would be preferable to a partially calibrated one, especially if they are being used to initialize a full bundle adjustment of all the parameters.

One of the attractions of factorization methods is that they provide a "closed form" (sometimes called a "linear") method to initialize iterative techniques such as bundle adjustment. An alternative initialization technique is to estimate the homographies corresponding to some common plane seen by all the cameras (Rother and Carlsson 2002). In a calibrated camera setting, this can correspond to estimating consistent rotations for all of the cameras, for example, using matched vanishing points (Antone and Teller 2002). Once these have been recovered, the camera positions can then be obtained by solving a linear system (Antone and Teller 2002; Rother and Carlsson 2002; Rother 2003).

7.3.2 *Application*: Sparse 3D model extraction

Once a multi-view 3D reconstruction of the scene has been estimated, it then becomes possible to create a texture-mapped 3D model of the object and to look at it from new directions.

The first step is to create a denser 3D model than the sparse point cloud that structure from motion produces. One alternative is to run dense multi-view stereo (Sections 11.3–11.6). Alternatively, a simpler technique such as 3D triangulation can be used, as shown in Figure 7.6, in which 207 reconstructed 3D points are triangulated to produce a surface mesh.

In order to create a more realistic model, a *texture map* can be extracted for each triangle face. The equations to map points on the surface of a 3D triangle to a 2D image are straightforward: just pass the local 2D coordinates on the triangle through the 3×4 camera projection matrix to obtain a 3×3 homography (planar perspective projection). When multiple source images are available, as is usually the case in multi-view reconstruction, either the closest and most fronto-parallel image can be used or multiple images can be blended in to deal with view-dependent foreshortening (Wang, Kang, Szeliski *et al.* 2001) or to obtain super-resolved results (Goldluecke and Cremers 2009) Another alternative is to create a sep-

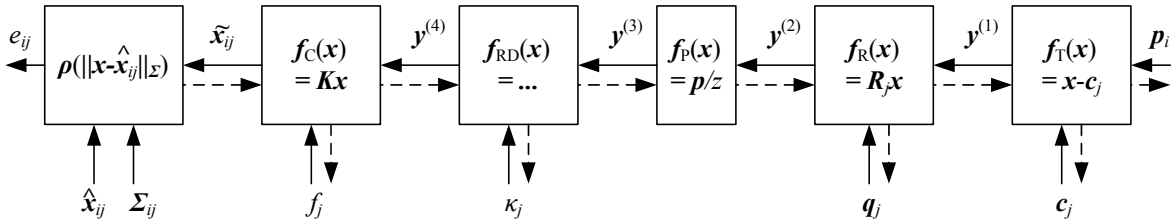


Figure 7.7 A set of chained transforms for projecting a 3D point $\boldsymbol{p}_i$ into a 2D measurement $\boldsymbol{x}_{ij}$ through a series of transformations $\boldsymbol{f}^{(k)}$, each of which is controlled by its own set of parameters. The dashed lines indicate the flow of information as partial derivatives are computed during a backward pass. The formula for the radial distortion function is $\boldsymbol{f}_{\mathrm{RD}}(\boldsymbol{x}) = (1 + \kappa_1 r^2 + \kappa_2 r^4)\boldsymbol{x}$.

arate texture map from each reference camera and to blend between them during rendering, which is known as *view-dependent texture mapping* (Section 13.1.1) (Debevec, Taylor, and Malik 1996; Debevec, Yu, and Borshukov 1998).

7.4 Bundle adjustment

As we have mentioned several times before, the most accurate way to recover structure and motion is to perform robust non-linear minimization of the measurement (re-projection) errors, which is commonly known in the photogrammetry (and now computer vision) communities as *bundle adjustment*.[13] Triggs, McLauchlan, Hartley *et al.* (1999) provide an excellent overview of this topic, including its historical development, pointers to the photogrammetry literature (Slama 1980; Atkinson 1996; Kraus 1997), and subtle issues with gauge ambiguities. The topic is also treated in depth in textbooks and surveys on multi-view geometry (Faugeras and Luong 2001; Hartley and Zisserman 2004; Moons, Van Gool, and Vergauwen 2010).

We have already introduced the elements of bundle adjustment in our discussion on iterative pose estimation (Section 6.2.2), i.e., Equations (6.42–6.48) and Figure 6.5. The biggest difference between these formulas and full bundle adjustment is that our feature location measurements $\boldsymbol{x}_{ij}$ now depend not only on the point (track index) i but also on the camera pose index j,

$$\boldsymbol{x}_{ij} = \boldsymbol{f}(\boldsymbol{p}_i, \boldsymbol{R}_j, \boldsymbol{c}_j, \boldsymbol{K}_j), \tag{7.49}$$

and that the 3D point positions $\boldsymbol{p}_i$ are also being simultaneously updated. In addition, it is common to add a stage for radial distortion parameter estimation (2.78),

$$\boldsymbol{f}_{\mathrm{RD}}(\boldsymbol{x}) = (1 + \kappa_1 r^2 + \kappa_2 r^4)\boldsymbol{x}, \tag{7.50}$$

if the cameras being used have not been pre-calibrated, as shown in Figure 7.7.

While most of the boxes (transforms) in Figure 7.7 have previously been explained (6.47), the leftmost box has not. This box performs a robust comparison of the predicted and mea-

[13] The term "bundle" refers to the bundles of rays connecting camera centers to 3D points and the term "adjustment" refers to the iterative minimization of re-projection error. Alternative terms for this in the vision community include *optimal motion estimation* (Weng, Ahuja, and Huang 1993) and *non-linear least squares* (Appendix A.3) (Taylor, Kriegman, and Anandan 1991; Szeliski and Kang 1994).

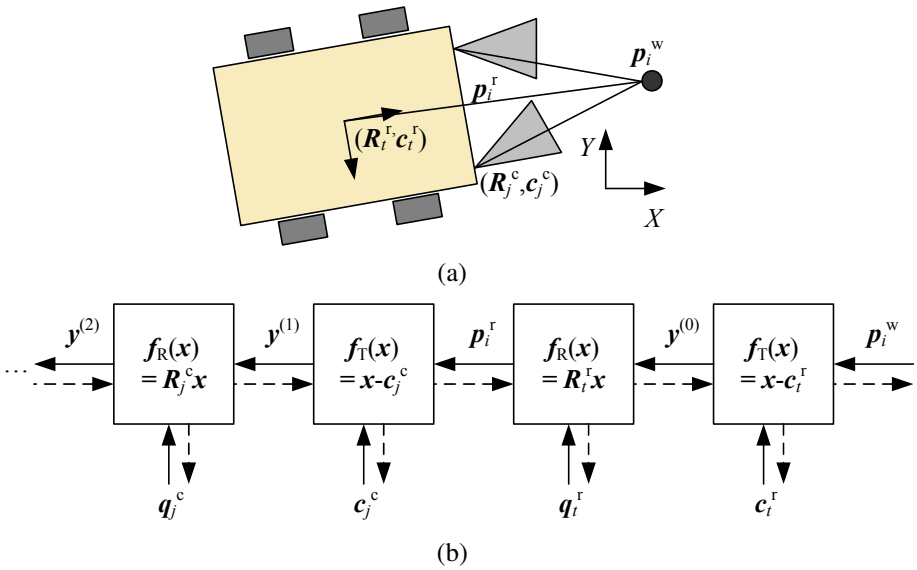


Figure 7.8 A camera rig and its associated transform chain. (a) As the mobile rig (robot) moves around in the world, its pose with respect to the world at time t is captured by $(\boldsymbol{R}_t^{\mathrm{r}}, \boldsymbol{c}_t^{\mathrm{r}})$. Each camera's pose with respect to the rig is captured by $(\boldsymbol{R}_j^{\mathrm{c}}, \boldsymbol{c}_j^{\mathrm{c}})$. (b) A 3D point with world coordinates $\boldsymbol{p}_i^{\mathrm{w}}$ is first transformed into rig coordinates $\boldsymbol{p}_i^{\mathrm{r}}$, and then through the rest of the camera-specific chain, as shown in Figure 7.7.

sured 2D locations $\hat{\boldsymbol{x}}_{ij}$ and $\tilde{\boldsymbol{x}}_{ij}$ after re-scaling by the measurement noise covariance Σ_{ij}. In more detail, this operation can be written as

$$\begin{aligned} \boldsymbol{r}_{ij} &= \tilde{\boldsymbol{x}}_{ij} - \hat{\boldsymbol{x}}_{ij}, && (7.51) \\ s_{ij}^2 &= \boldsymbol{r}_{ij}^T \Sigma_{ij}^{-1} \boldsymbol{r}_{ij}, && (7.52) \\ e_{ij} &= \hat{\rho}(s_{ij}^2), && (7.53) \end{aligned}$$

where $\hat{\rho}(r^2) = \rho(r)$. The corresponding Jacobians (partial derivatives) can be written as

$$\begin{aligned} \frac{\partial e_{ij}}{\partial s_{ij}^2} &= \hat{\rho}'(s_{ij}^2), && (7.54) \\ \frac{\partial s_{ij}^2}{\partial \tilde{\boldsymbol{x}}_{ij}} &= \Sigma_{ij}^{-1} \boldsymbol{r}_{ij}. && (7.55) \end{aligned}$$

The advantage of the chained representation introduced above is that it not only makes the computations of the partial derivatives and Jacobians simpler but it can also be adapted to any camera configuration. Consider for example a pair of cameras mounted on a robot that is moving around in the world, as shown in Figure 7.8a. By replacing the rightmost two transformations in Figure 7.7 with the transformations shown in Figure 7.8b, we can simultaneously recover the position of the robot at each time and the calibration of each camera with respect to the rig, in addition to the 3D structure of the world.

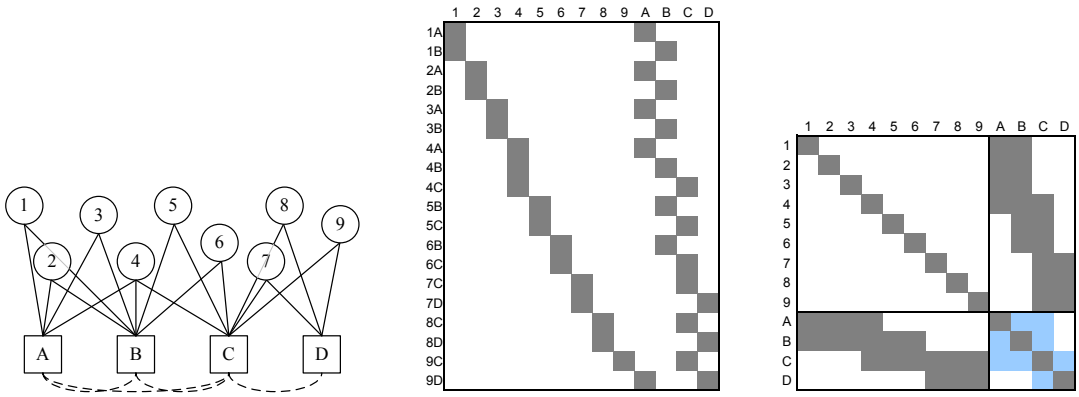


Figure 7.9 (a) Bipartite graph for a toy structure from motion problem and (b) its associated Jacobian $\boldsymbol{J}$ and (c) Hessian $\boldsymbol{A}$. Numbers indicate 3D points and letters indicate cameras. The dashed arcs and light blue squares indicate the fill-in that occurs when the structure (point) variables are eliminated.

7.4.1 Exploiting sparsity

Large bundle adjustment problems, such as those involving reconstructing 3D scenes from thousands of Internet photographs (Snavely, Seitz, and Szeliski 2008b; Agarwal, Snavely, Simon *et al.* 2009; Agarwal, Furukawa, Snavely *et al.* 2010; Snavely, Simon, Goesele *et al.* 2010), can require solving non-linear least squares problems with millions of measurements (feature matches) and tens of thousands of unknown parameters (3D point positions and camera poses). Unless some care is taken, these kinds of problem can become intractable, since the (direct) solution of dense least squares problems is cubic in the number of unknowns.

Fortunately, structure from motion is a *bipartite* problem in structure and motion. Each feature point $\boldsymbol{x}_{ij}$ in a given image depends on one 3D point position $\boldsymbol{p}_i$ and one 3D camera pose $(\boldsymbol{R}_j, \boldsymbol{c}_j)$. This is illustrated in Figure 7.9a, where each circle (1–9) indicates a 3D point, each square (A–D) indicates a camera, and lines (edges) indicate which points are visible in which cameras (2D features). If the values for all the points are known or fixed, the equations for all the cameras become independent, and vice versa.

If we order the structure variables before the motion variables in the Hessian matrix $\boldsymbol{A}$ (and hence also the right hand side vector $\boldsymbol{b}$), we obtain a structure for the Hessian shown in Figure 7.9c.[14] When such a system is solved using sparse Cholesky factorization (see Appendix A.4) (Björck 1996; Golub and Van Loan 1996), the *fill-in* occurs in the smaller motion Hessian A_{cc} (Szeliski and Kang 1994; Triggs, McLauchlan, Hartley *et al.* 1999; Hartley and Zisserman 2004; Lourakis and Argyros 2009; Engels, Stewénius, and Nistér 2006). Some recent papers by (Byröd and øAström 2009), Jeong, Nistér, Steedly *et al.* (2010) and (Agarwal, Snavely, Seitz *et al.* 2010) explore the use of iterative (conjugate gradient) techniques for the solution of bundle adjustment problems.

[14] This ordering is preferable when there are fewer cameras than 3D points, which is the usual case. The exception is when we are tracking a small number of points through many video frames, in which case this ordering should be reversed.

In more detail, the *reduced* motion Hessian is computed using the *Schur complement*,

$$\boldsymbol{A}'_{cc} = \boldsymbol{A}_{cc} - \boldsymbol{A}_{pc}^T \boldsymbol{A}_{pp}^{-1} \boldsymbol{A}_{pc}, \tag{7.56}$$

where $\boldsymbol{A}_{pp}$ is the point (structure) Hessian (the top left block of Figure 7.9c), $\boldsymbol{A}_{pc}$ is the point-camera Hessian (the top right block), and $\boldsymbol{A}_{cc}$ and $\boldsymbol{A}'_{cc}$ are the motion Hessians before and after the point variable elimination (the bottom right block of Figure 7.9c). Notice that $\boldsymbol{A}'_{cc}$ has a non-zero entry between two cameras if they see any 3D point in common. This is indicated with dashed arcs in Figure 7.9a and light blue squares in Figure 7.9c.

Whenever there are global parameters present in the reconstruction algorithm, such as camera intrinsics that are common to all of the cameras, or camera rig calibration parameters such as those shown in Figure 7.8, they should be ordered last (placed along the right and bottom edges of $\boldsymbol{A}$) in order to reduce fill-in.

Engels, Stewénius, and Nistér (2006) provide a nice recipe for sparse bundle adjustment, including all the steps needed to initialize the iterations, as well as typical computation times for a system that uses a fixed number of backward-looking frames in a real-time setting. They also recommend using homogeneous coordinates for the structure parameters $\boldsymbol{p}_i$, which is a good idea, since it avoids numerical instabilities for points near infinity.

Bundle adjustment is now the standard method of choice for most structure-from-motion problems and is commonly applied to problems with hundreds of weakly calibrated images and tens of thousands of points, e.g., in systems such as Photosynth. (Much larger problems are commonly solved in photogrammetry and aerial imagery, but these are usually carefully calibrated and make use of surveyed ground control points.) However, as the problems become larger, it becomes impractical to re-solve full bundle adjustment problems at each iteration.

One approach to dealing with this problem is to use an incremental algorithm, where new cameras are added over time. (This makes particular sense if the data is being acquired from a video camera or moving vehicle (Nistér, Naroditsky, and Bergen 2006; Pollefeys, Nistér, Frahm *et al.* 2008).) A Kalman filter can be used to incrementally update estimates as new information is acquired. Unfortunately, such sequential updating is only statistically optimal for linear least squares problems.

For non-linear problems such as structure from motion, an extended Kalman filter, which linearizes measurement and update equations around the current estimate, needs to be used (Gelb 1974; Viéville and Faugeras 1990). To overcome this limitation, several passes can be made through the data (Azarbayejani and Pentland 1995). Because points disappear from view (and old cameras become irrelevant), a *variable state dimension filter* (VSDF) can be used to adjust the set of state variables over time, for example, by keeping only cameras and point tracks seen in the last k frames (McLauchlan 2000). A more flexible approach to using a fixed number of frames is to propagate corrections backwards through points and cameras until the changes on parameters are below a threshold (Steedly and Essa 2001). Variants of these techniques, including methods that use a fixed window for bundle adjustment (Engels, Stewénius, and Nistér 2006) or select keyframes for doing full bundle adjustment (Klein and Murray 2008) are now commonly used in real-time tracking and augmented-reality applications, as discussed in Section 7.4.2.

When maximum accuracy is required, it is still preferable to perform a full bundle adjustment over all the frames. In order to control the resulting computational complexity, one

approach is to lock together subsets of frames into locally rigid configurations and to optimize the relative positions of these cluster (Steedly, Essa, and Dellaert 2003). A different approach is to select a smaller number of frames to form a *skeletal set* that still spans the whole dataset and produces reconstructions of comparable accuracy (Snavely, Seitz, and Szeliski 2008b). We describe this latter technique in more detail in Section 7.4.4, where we discuss applications of structure from motion to large image sets.

While bundle adjustment and other robust non-linear least squares techniques are the methods of choice for most structure-from-motion problems, they suffer from initialization problems, i.e., they can get stuck in local energy minima if not started sufficiently close to the global optimum. Many systems try to mitigate this by being conservative in what reconstruction they perform early on and which cameras and points they add to the solution (Section 7.4.4). An alternative, however, is to re-formulate the problem using a norm that supports the computation of global optima.

Kahl and Hartley (2008) describe techniques for using L_∞ norms in geometric reconstruction problems. The advantage of such norms is that globally optimal solutions can be efficiently computed using second-order cone programming (SOCP). The disadvantage is that L_∞ norms are particularly sensitive to outliers and so must be combined with good outlier rejection techniques before they can be used.

7.4.2 *Application*: Match move and augmented reality

One of the neatest applications of structure from motion is to estimate the 3D motion of a video or film camera, along with the geometry of a 3D scene, in order to superimpose 3D graphics or computer-generated images (CGI) on the scene. In the visual effects industry, this is known as the *match move* problem (Roble 1999), since the motion of the synthetic 3D camera used to render the graphics must be *matched* to that of the real-world camera. For very small motions, or motions involving pure camera rotations, one or two tracked points can suffice to compute the necessary visual motion. For planar surfaces moving in 3D, four points are needed to compute the homography, which can then be used to insert planar overlays, e.g., to replace the contents of advertising billboards during sporting events.

The general version of this problem requires the estimation of the full 3D camera pose along with the focal length (zoom) of the lens and potentially its radial distortion parameters (Roble 1999). When the 3D structure of the scene is known ahead of time, pose estimation techniques such as *view correlation* (Bogart 1991) or *through-the-lens camera control* (Gleicher and Witkin 1992) can be used, as described in Section 6.2.3.

For more complex scenes, it is usually preferable to recover the 3D structure simultaneously with the camera motion using structure-from-motion techniques. The trick with using such techniques is that in order to prevent any visible jitter between the synthetic graphics and the actual scene, features must be tracked to very high accuracy and ample feature tracks must be available in the vicinity of the insertion location. Some of today's best known match move software packages, such as the *boujou* package from 2d3,[15] which won an Emmy award in 2002, originated in structure-from-motion research in the computer vision community (Fitzgibbon and Zisserman 1998).

[15] http://www.2d3.com/.

(a) (b)

Figure 7.10 3D augmented reality: (a) Darth Vader and a horde of Ewoks battle it out on a table-top recovered using real-time, keyframe-based structure from motion (Klein and Murray 2007) © 2007 IEEE; (b) a virtual teapot is fixed to the top of a real-world coffee cup, whose pose is re-recognized at each time frame (Gordon and Lowe 2006) © 2007 Springer.

Closely related to the match move problem is robotics navigation, where a robot must estimate its location relative to its environment, while simultaneously avoiding any dangerous obstacles. This problem is often known as *simultaneous localization and mapping* (SLAM) (Thrun, Burgard, and Fox 2005) or *visual odometry* (Levin and Szeliski 2004; Nistér, Naroditsky, and Bergen 2006; Maimone, Cheng, and Matthies 2007). Early versions of such algorithms used range-sensing techniques, such as ultrasound, laser range finders, or stereo matching, to estimate local 3D geometry, which could then be fused into a 3D model. Newer techniques can perform the same task based purely on visual feature tracking, sometimes not even requiring a stereo camera rig (Davison, Reid, Molton *et al.* 2007).

Another closely related application is *augmented reality*, where 3D objects are inserted into a video feed in real time, often to annotate or help users understand a scene (Azuma, Baillot, Behringer *et al.* 2001). While traditional systems require prior knowledge about the scene or object being visually tracked (Rosten and Drummond 2005), newer systems can simultaneously build up a model of the 3D environment and then track it, so that graphics can be superimposed.

Klein and Murray (2007) describe a *parallel tracking and mapping* (PTAM) system, which simultaneously applies full bundle adjustment to keyframes selected from a video stream, while performing robust real-time pose estimation on intermediate frames. Figure 7.10a shows an example of their system in use. Once an initial 3D scene has been reconstructed, a dominant plane is estimated (in this case, the table-top) and 3D animated characters are virtually inserted. Klein and Murray (2008) extend their previous system to handle even faster camera motion by adding edge features, which can still be detected even when interest points become too blurred. They also use a direct (intensity-based) rotation estimation algorithm for even faster motions.

Instead of modeling the whole scene as one rigid reference frame, Gordon and Lowe (2006) first build a 3D model of an individual object using feature matching and structure from motion. Once the system has been initialized, for every new frame, they find the object and its pose using a 3D instance recognition algorithm, and then superimpose a graphical

object onto that model, as shown in Figure 7.10b.

While reliably tracking such objects and environments is now a well-solved problem, determining which pixels should be occluded by foreground scene elements still remains an open problem (Chuang, Agarwala, Curless *et al.* 2002; Wang and Cohen 2007a).

7.4.3 Uncertainty and ambiguities

Because structure from motion involves the estimation of so many highly coupled parameters, often with no known "ground truth" components, the estimates produced by structure from motion algorithms can often exhibit large amounts of uncertainty (Szeliski and Kang 1997). An example of this is the classic *bas-relief ambiguity*, which makes it hard to simultaneously estimate the 3D depth of a scene and the amount of camera motion (Oliensis 2005).[16]

As mentioned before, a unique coordinate frame and scale for a reconstructed scene cannot be recovered from monocular visual measurements alone. (When a stereo rig is used, the scale can be recovered if we know the distance (baseline) between the cameras.) This seven-degree-of-freedom *gauge ambiguity* makes it tricky to compute the covariance matrix associated with a 3D reconstruction (Triggs, McLauchlan, Hartley *et al.* 1999; Kanatani and Morris 2001). A simple way to compute a covariance matrix that ignores the gauge freedom (indeterminacy) is to throw away the seven smallest eigenvalues of the information matrix (inverse covariance), whose values are equivalent to the problem Hessian $\boldsymbol{A}$ up to noise scaling (see Section 6.1.4 and Appendix B.6). After we do this, the resulting matrix can be inverted to obtain an estimate of the parameter covariance.

Szeliski and Kang (1997) use this approach to visualize the largest directions of variation in typical structure from motion problems. Not surprisingly, they find that (ignoring the gauge freedoms), the greatest uncertainties for problems such as observing an object from a small number of nearby viewpoints are in the depths of the 3D structure relative to the extent of the camera motion.[17]

It is also possible to estimate *local* or *marginal* uncertainties for individual parameters, which corresponds simply to taking block sub-matrices from the full covariance matrix. Under certain conditions, such as when the camera poses are relatively certain compared to 3D point locations, such uncertainty estimates can be meaningful. However, in many cases, individual uncertainty measures can mask the extent to which reconstruction errors are correlated, which is why looking at the first few modes of greatest joint variation can be helpful.

The other way in which gauge ambiguities affect structure from motion and, in particular, bundle adjustment is that they make the system Hessian matrix $\boldsymbol{A}$ rank-deficient and hence impossible to invert. A number of techniques have been proposed to mitigate this problem (Triggs, McLauchlan, Hartley *et al.* 1999; Bartoli 2003). In practice, however, it appears that simply adding a small amount of the Hessian diagonal $\lambda \text{diag}(\boldsymbol{A})$ to the Hessian $\boldsymbol{A}$ itself, as is done in the Levenberg–Marquardt non-linear least squares algorithm (Appendix A.3), usually works well.

[16] Bas-relief refers to a kind of sculpture in which objects, often on ornamental friezes, are sculpted with less depth than they actually occupy. When lit from above by sunlight, they appear to have true 3D depth because of the ambiguity between relative depth and the angle of the illuminant (Section 12.1.1).

[17] A good way to minimize the amount of such ambiguities is to use wide field of view cameras (Antone and Teller 2002; Levin and Szeliski 2006).

Figure 7.11 Incremental structure from motion (Snavely, Seitz, and Szeliski 2006) © 2006 ACM: Starting with an initial two-frame reconstruction of Trevi Fountain, batches of images are added using pose estimation, and their positions (along with the 3D model) are refined using bundle adjustment.

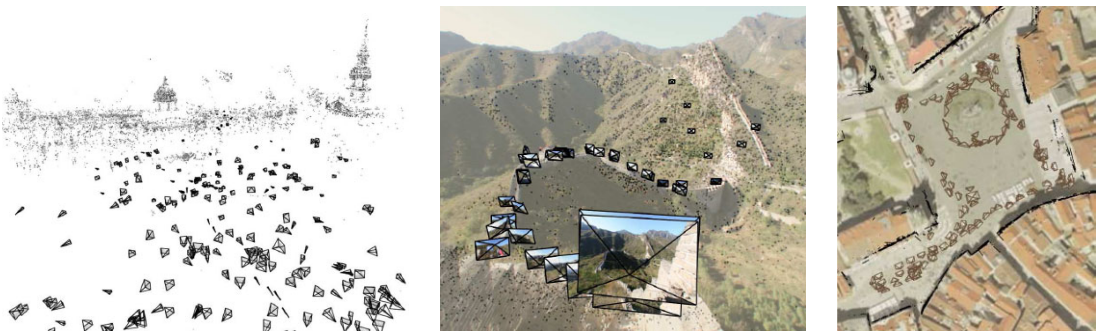
7.4.4 *Application*: Reconstruction from Internet photos

The most widely used application of structure from motion is in the reconstruction of 3D objects and scenes from video sequences and collections of images (Pollefeys and Van Gool 2002). The last decade has seen an explosion of techniques for performing this task automatically without the need for any manual correspondence or pre-surveyed ground control points. A lot of these techniques assume that the scene is taken with the same camera and hence the images all have the same intrinsics (Fitzgibbon and Zisserman 1998; Koch, Pollefeys, and Van Gool 2000; Schaffalitzky and Zisserman 2002; Tuytelaars and Van Gool 2004; Pollefeys, Nistér, Frahm *et al.* 2008; Moons, Van Gool, and Vergauwen 2010). Many of these techniques take the results of the sparse feature matching and structure from motion computation and then compute dense 3D surface models using multi-view stereo techniques (Section 11.6) (Koch, Pollefeys, and Van Gool 2000; Pollefeys and Van Gool 2002; Pollefeys, Nistér, Frahm *et al.* 2008; Moons, Van Gool, and Vergauwen 2010).

The latest innovation in this space has been the application of structure from motion and multi-view stereo techniques to thousands of images taken from the Internet, where very little is known about the cameras taking the photographs (Snavely, Seitz, and Szeliski 2008a). Before the structure from motion computation can begin, it is first necessary to establish sparse correspondences between different pairs of images and to then link such correspondences into *feature tracks*, which associate individual 2D image features with global 3D points. Because the $O(N^2)$ comparison of all pairs of images can be very slow, a number of techniques have been developed in the recognition community to make this process faster (Section 14.3.2) (Nistér and Stewénius 2006; Philbin, Chum, Sivic *et al.* 2008; Li, Wu, Zach *et al.* 2008; Chum, Philbin, and Zisserman 2008; Chum and Matas 2010).

To begin the reconstruction process, it is important to to select a good pair of images, where there are both a large number of consistent matches (to lower the likelihood of incorrect correspondences) and a significant amount of out-of-plane parallax,[18] to ensure that a stable reconstruction can be obtained (Snavely, Seitz, and Szeliski 2006). The EXIF tags associated with the photographs can be used to get good initial estimates for camera focal lengths, although this is not always strictly necessary, since these parameters are re-adjusted

[18] A simple way to compute this is to robustly fit a homography to the correspondences and measure reprojection errors.



(a) (b) (c)

Figure 7.12 3D reconstructions produced by the incremental structure from motion algorithm developed by Snavely, Seitz, and Szeliski (2006) © 2006 ACM: (a) cameras and point cloud from Trafalgar Square; (b) cameras and points overlaid on an image from the Great Wall of China; (c) overhead view of a reconstruction of the Old Town Square in Prague registered to an aerial photograph.

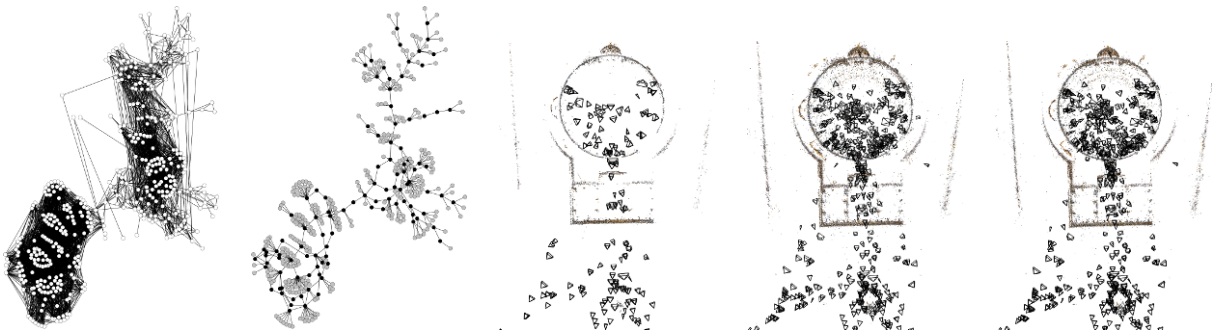
as part of the bundle adjustment process.

Once an initial pair has been reconstructed, the pose of cameras that see a sufficient number of the resulting 3D points can be estimated (Section 6.2) and the complete set of cameras and feature correspondences can be used to perform another round of bundle adjustment. Figure 7.11 shows the progression of the incremental bundle adjustment algorithm, where sets of cameras are added after each successive round of bundle adjustment, while Figure 7.12 shows some additional results. An alternative to this kind of *seed and grow* approach is to first reconstruct triplets of images and then hierarchically merge triplets into larger collections, as described by Fitzgibbon and Zisserman (1998).

Unfortunately, as the incremental structure from motion algorithm continues to add more cameras and points, it can become extremely slow. The direct solution of a dense system of $O(N)$ equations for the camera pose updates can take $O(N^3)$ time; while structure from motion problems are rarely dense, scenes such as city squares have a high percentage of cameras that see points in common. Re-running the bundle adjustment algorithm after every few camera additions results in a quartic scaling of the run time with the number of images in the dataset. One approach to solving this problem is to select a smaller number of images for the original scene reconstruction and to fold in the remaining images at the very end.

Snavely, Seitz, and Szeliski (2008b) develop an algorithm for computing such a *skeletal set* of images, which is guaranteed to produce a reconstruction whose error is within a bounded factor of the optimal reconstruction accuracy. Their algorithm first evaluates all pairwise uncertainties (position covariances) between overlapping images and then chains them together to estimate a lower bound for the relative uncertainty of any distant pair. The skeletal set is constructed so that the maximal uncertainty between any pair grows by no more than a constant factor. Figure 7.13 shows an example of the skeletal set computed for 784 images of the Pantheon in Rome. As you can see, even though the skeletal set contains just a fraction of the original images, the shapes of the skeletal set and full bundle adjusted reconstructions are virtually indistinguishable.

The ability to automatically reconstruct 3D models from large, unstructured image collections has opened a wide variety of additional applications, including the ability to automat-



(a) (b) (c) (d) (e)

Figure 7.13 Large scale structure from motion using skeletal sets (Snavely, Seitz, and Szeliski 2008b) © 2008 IEEE: (a) original match graph for 784 images; (b) skeletal set containing 101 images; (c) top-down view of scene (Pantheon) reconstructed from the skeletal set; (d) reconstruction after adding in the remaining images using pose estimation; (e) final bundle adjusted reconstruction, which is almost identical.

ically find and label locations and regions of interest (Simon, Snavely, and Seitz 2007; Simon and Seitz 2008; Gammeter, Bossard, Quack *et al.* 2009) and to cluster large image collections so that they can be automatically labeled (Li, Wu, Zach *et al.* 2008; Quack, Leibe, and Van Gool 2008). Some of these application are discussed in more detail in Section 13.1.2.

7.5 Constrained structure and motion

The most general algorithms for structure from motion make no prior assumptions about the objects or scenes that they are reconstructing. In many cases, however, the scene contains higher-level geometric primitives, such as lines and planes. These can provide information complementary to interest points and also serve as useful building blocks for 3D modeling and visualization. Furthermore, these primitives are often arranged in particular relationships, i.e., many lines and planes are either parallel or orthogonal to each other. This is particularly true of architectural scenes and models, which we study in more detail in Section 12.6.1.

Sometimes, instead of exploiting regularity in the scene structure, it is possible to take advantage of a constrained motion model. For example, if the object of interest is rotating on a turntable (Szeliski 1991b), i.e., around a fixed but unknown axis, specialized techniques can be used to recover this motion (Fitzgibbon, Cross, and Zisserman 1998). In other situations, the camera itself may be moving in a fixed arc around some center of rotation (Shum and He 1999). Specialized capture setups, such as mobile stereo camera rigs or moving vehicles equipped with multiple fixed cameras, can also take advantage of the knowledge that individual cameras are (mostly) fixed with respect to the capture rig, as shown in Figure 7.8.[19]

[19] Because of mechanical compliance and jitter, it may be prudent to allow for a small amount of individual camera rotation around a nominal position.

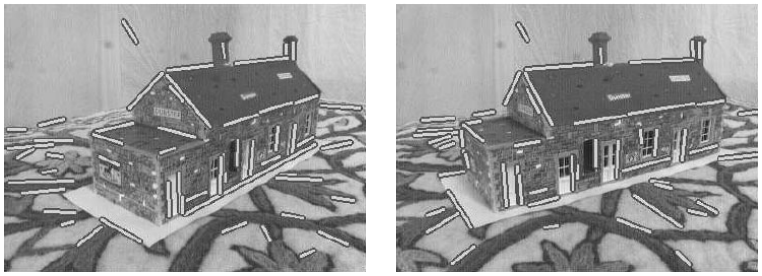


Figure 7.14 Two images of a toy house along with their matched 3D line segments (Schmid and Zisserman 1997) © 1997 Springer.

7.5.1 Line-based techniques

It is well known that pairwise epipolar geometry cannot be recovered from line matches alone, even if the cameras are calibrated. To see this, think of projecting the set of lines in each image into a set of 3D planes in space. You can move the two cameras around into any configuration you like and still obtain a valid reconstruction for 3D lines.

When lines are visible in three or more views, the trifocal tensor can be used to transfer lines from one pair of images to another (Hartley and Zisserman 2004). The trifocal tensor can also be computed on the basis of line matches alone.

Schmid and Zisserman (1997) describe a widely used technique for matching 2D lines based on the average of 15×15 pixel correlation scores evaluated at all pixels along their common line segment intersection.[20] In their system, the epipolar geometry is assumed to be known, e.g., computed from point matches. For wide baselines, all possible homographies corresponding to planes passing through the 3D line are used to warp pixels and the maximum correlation score is used. For triplets of images, the trifocal tensor is used to verify that the lines are in geometric correspondence before evaluating the correlations between line segments. Figure 7.14 shows the results of using their system.

Bartoli and Sturm (2003) describe a complete system for extending three view relations (trifocal tensors) computed from manual line correspondences to a full bundle adjustment of all the line and camera parameters. The key to their approach is to use the Plücker coordinates (2.12) to parameterize lines and to directly minimize reprojection errors. It is also possible to represent 3D line segments by their endpoints and to measure either the reprojection error perpendicular to the detected 2D line segments in each image or the 2D errors using an elongated uncertainty ellipse aligned with the line segment direction (Szeliski and Kang 1994).

Instead of reconstructing 3D lines, Bay, Ferrari, and Van Gool (2005) use RANSAC to group lines into likely coplanar subsets. Four lines are chosen at random to compute a homography, which is then verified for these and other plausible line segment matches by evaluating color histogram-based correlation scores. The 2D intersection points of lines belonging to the same plane are then used as virtual measurements to estimate the epipolar geometry, which is more accurate than using the homographies directly.

[20] Because lines often occur at depth or orientation discontinuities, it may be preferable to compute correlation scores (or to match color histograms (Bay, Ferrari, and Van Gool 2005)) separately on each side of the line.

An alternative to grouping lines into coplanar subsets is to group lines by parallelism. Whenever three or more 2D lines share a common vanishing point, there is a good likelihood that they are parallel in 3D. By finding multiple vanishing points in an image (Section 4.3.3) and establishing correspondences between such vanishing points in different images, the relative rotations between the various images (and often the camera intrinsics) can be directly estimated (Section 6.3.2).

Shum, Han, and Szeliski (1998) describe a 3D modeling system which first constructs calibrated panoramas from multiple images (Section 7.4) and then has the user draw vertical and horizontal lines in the image to demarcate the boundaries of planar regions. The lines are initially used to establish an absolute rotation for each panorama and are later used (along with the inferred vertices and planes) to infer a 3D structure, which can be recovered up to scale from one or more images (Figure 12.15).

A fully automated approach to line-based structure from motion is presented vy Werner and Zisserman (2002). In their system, they first find lines and group them by common vanishing points in each image (Section 4.3.3). The vanishing points are then used to calibrate the camera, i.e., to performa a "metric upgrade" (Section 6.3.2). Lines corresponding to common vanishing points are then matched using both appearance (Schmid and Zisserman 1997) and trifocal tensors. The resulting set of 3D lines, color coded by common vanishing directions (3D orientations) is shown in Figure 12.16a. These lines are then used to infer planes and a block-structured model for the scene, as described in more detail in Section 12.6.1.

7.5.2 Plane-based techniques

In scenes that are rich in planar structures, e.g., in architecture and certain kinds of manufactured objects such as furniture, it is possible to directly estimate homographies between different planes, using either feature-based or intensity-based methods. In principle, this information can be used to simultaneously infer the camera poses and the plane equations, i.e., to compute plane-based structure from motion.

Luong and Faugeras (1996) show how a fundamental matrix can be directly computed from two or more homographies using algebraic manipulations and least squares. Unfortunately, this approach often performs poorly, since the algebraic errors do not correspond to meaningful reprojection errors (Szeliski and Torr 1998).

A better approach is to *hallucinate* virtual point correspondences within the areas from which each homography was computed and to feed them into a standard structure from motion algorithm (Szeliski and Torr 1998). An even better approach is to use full bundle adjustment with explicit plane equations, as well as additional constraints to force reconstructed co-planar features to lie exactly on their corresponding planes. (A principled way to do this is to establish a coordinate frame for each plane, e.g., at one of the feature points, and to use 2D in-plane parameterizations for the other points.) The system developed by Shum, Han, and Szeliski (1998) shows an example of such an approach, where the directions of lines and normals for planes in the scene are pre-specified by the user.

7.6 Additional reading

The topic of structure from motion is extensively covered in books and review articles on multi-view geometry (Faugeras and Luong 2001; Hartley and Zisserman 2004; Moons, Van Gool, and Vergauwen 2010). For two-frame reconstruction, Hartley (1997a) wrote a highly cited paper on the "eight-point algorithm" for computing an essential or fundamental matrix with reasonable point normalization. When the cameras are calibrated, the five-point algorithm of Nistér (2004) can be used in conjunction with RANSAC to obtain initial reconstructions from the minimum number of points. When the cameras are uncalibrated, various self-calibration techniques can be found in work by Hartley and Zisserman (2004); Moons, Van Gool, and Vergauwen (2010)—I only briefly mention one of the simplest techniques, the Kruppa equations (7.35).

In applications where points are being tracked from frame to frame, factorization techniques, based on either orthographic camera models (Tomasi and Kanade 1992; Poelman and Kanade 1997; Costeira and Kanade 1995; Morita and Kanade 1997; Morris and Kanade 1998; Anandan and Irani 2002) or projective extensions (Christy and Horaud 1996; Sturm and Triggs 1996; Triggs 1996; Oliensis and Hartley 2007), can be used.

Triggs, McLauchlan, Hartley *et al.* (1999) provide a good tutorial and survey on bundle adjustment, while Lourakis and Argyros (2009) and Engels, Stewénius, and Nistér (2006) provide tips on implementation and effective practices. Bundle adjustment is also covered in textbooks and surveys on multi-view geometry (Faugeras and Luong 2001; Hartley and Zisserman 2004; Moons, Van Gool, and Vergauwen 2010). Techniques for handling larger problems are described by Snavely, Seitz, and Szeliski (2008b); Agarwal, Snavely, Simon *et al.* (2009); Jeong, Nistér, Steedly *et al.* (2010); Agarwal, Snavely, Seitz *et al.* (2010). While bundle adjustment is often called as an inner loop inside incremental reconstruction algorithms (Snavely, Seitz, and Szeliski 2006), hierarchical (Fitzgibbon and Zisserman 1998; Farenzena, Fusiello, and Gherardi 2009) and global (Rother and Carlsson 2002; Martinec and Pajdla 2007) approaches for initialization are also possible and perhaps even preferable.

As structure from motion starts being applied to dynamic scenes, the topic of non-rigid structure from motion (Torresani, Hertzmann, and Bregler 2008), which we do not cover in this book, will become more important.

7.7 Exercises

Ex 7.1: Triangulation Use the calibration pattern you built and tested in Exercise 6.7 to test your triangulation accuracy. As an alternative, generate synthetic 3D points and cameras and add noise to the 2D point measurements.

1. Assume that you know the camera pose, i.e., the camera matrices. Use the 3D distance to rays (7.4) or linearized versions of Equations (7.5–7.6) to compute an initial set of 3D locations. Compare these to your known ground truth locations.
2. Use iterative non-linear minimization to improve your initial estimates and report on the improvement in accuracy.
3. (Optional) Use the technique described by Hartley and Sturm (1997) to perform two-frame triangulation.

4. See if any of the failure modes reported by Hartley and Sturm (1997) or Hartley (1998) occur in practice.

Ex 7.2: Essential and fundamental matrix Implement the two-frame $\boldsymbol{E}$ and $\boldsymbol{F}$ matrix estimation techniques presented in Section 7.2, with suitable re-scaling for better noise immunity.

1. Use the data from Exercise 7.1 to validate your algorithms and to report on their accuracy.
2. (Optional) Implement one of the improved $\boldsymbol{F}$ or $\boldsymbol{E}$ estimation algorithms, e.g., using renormalization (Zhang 1998b; Torr and Fitzgibbon 2004; Hartley and Zisserman 2004), RANSAC (Torr and Murray 1997), least media squares (LMS), or the five-point algorithm developed by Nistér (2004).

Ex 7.3: View morphing and interpolation Implement automatic view morphing, i.e., compute two-frame structure from motion and then use these results to generate a smooth animation from one image to the next (Section 7.2.3).

1. Decide how to represent your 3D scene, e.g., compute a Delaunay triangulation of the matched point and decide what to do with the triangles near the border. (Hint: try fitting a plane to the scene, e.g., behind most of the points.)
2. Compute your in-between camera positions and orientations.
3. Warp each triangle to its new location, preferably using the correct perspective projection (Szeliski and Shum 1997).
4. (Optional) If you have a denser 3D model (e.g., from stereo), decide what to do at the "cracks".
5. (Optional) For a non-rigid scene, e.g., two pictures of a face with different expressions, not all of your matched points will obey the epipolar geometry. Decide how to handle them to achieve the best effect.

Ex 7.4: Factorization Implement the factorization algorithm described in Section 7.3 using point tracks you computed in Exercise 4.5.

1. (Optional) Implement uncertainty rescaling (Anandan and Irani 2002) and comment on whether this improves your results.
2. (Optional) Implement one of the perspective improvements to factorization discussed in Section 7.3.1 (Christy and Horaud 1996; Sturm and Triggs 1996; Triggs 1996). Does this produce significantly lower reprojection errors? Can you upgrade this reconstruction to a metric one?

Ex 7.5: Bundle adjuster Implement a full bundle adjuster. This may sound daunting, but it really is not.

1. Devise the internal data structures and external file representations to hold your camera parameters (position, orientation, and focal length), 3D point locations (Euclidean or homogeneous), and 2D point tracks (frame and point identifier as well as 2D locations).

2. Use some other technique, such as factorization, to initialize the 3D point and camera locations from your 2D tracks (e.g., a subset of points that appears in all frames).

3. Implement the code corresponding to the forward transformations in Figure 7.7, i.e., for each 2D point measurement, take the corresponding 3D point, map it through the camera transformations (including perspective projection and focal length scaling), and compare it to the 2D point measurement to get a residual error.

4. Take the residual error and compute its derivatives with respect to all the unknown motion and structure parameters, using backward chaining, as shown, e.g., in Figure 7.7 and Equation (6.47). This gives you the sparse Jacobian $\boldsymbol{J}$ used in Equations (6.13–6.17) and Equation (6.43).

5. Use a sparse least squares or linear system solver, e.g., MATLAB, SparseSuite, or SPARSKIT (see Appendix A.4 and A.5), to solve the corresponding linearized system, adding a small amount of diagonal preconditioning, as in Levenberg–Marquardt.

6. Update your parameters, make sure your rotation matrices are still orthonormal (e.g., by re-computing them from your quaternions), and continue iterating while monitoring your residual error.

7. (Optional) Use the "Schur complement trick" (7.56) to reduce the size of the system being solved (Triggs, McLauchlan, Hartley *et al.* 1999; Hartley and Zisserman 2004; Lourakis and Argyros 2009; Engels, Stewénius, and Nistér 2006).

8. (Optional) Implement your own iterative sparse solver, e.g., conjugate gradient, and compare its performance to a direct method.

9. (Optional) Make your bundle adjuster robust to outliers, or try adding some of the other improvements discussed in (Engels, Stewénius, and Nistér 2006). Can you think of any other ways to make your algorithm even faster or more robust?

Ex 7.6: Match move and augmented reality Use the results of the previous exercise to superimpose a rendered 3D model on top of video. See Section 7.4.2 for more details and ideas. Check for how "locked down" the objects are.

Ex 7.7: Line-based reconstruction Augment the previously developed bundle adjuster to include lines, possibly with known 3D orientations.

Optionally, use co-planar sets of points and lines to hypothesize planes and to enforce co-planarity (Schaffalitzky and Zisserman 2002; Robertson and Cipolla 2002)

Ex 7.8: Flexible bundle adjuster Design a bundle adjuster that allows for arbitrary chains of transformations and prior knowledge about the unknowns, as suggested in Figures 7.7–7.8.

Ex 7.9: Unordered image matching Compute the camera pose and 3D structure of a scene from an arbitrary collection of photographs (Brown and Lowe 2003; Snavely, Seitz, and Szeliski 2006).