18

N-View Computational Methods

This chapter describes computational methods for estimating a projective or affine reconstruction from a set of images – in particular where the number of views is large.

We start with the most general case which is that of bundle adjustment for a projective reconstruction. This is then specialized to affine cameras and the important factorization algorithm introduced. A generalization of this algorithm to non-rigid scenes is given. A second specialization of bundle adjustment is then described for the case of scenes containing planes. Finally we discuss methods for obtaining point correspondences throughout an image sequence and a projective reconstruction from these correspondences.

18.1 Projective reconstruction – bundle adjustment

Consider a situation in which a set of 3D points $\mathbf{X}_j$ is viewed by a set of cameras with matrices $\mathtt{P}^i$. Denote by $\mathbf{x}_j^i$ the coordinates of the j-th point as seen by the i-th camera. We wish to solve the following reconstruction problem: given the set of image coordinates $\mathbf{x}_j^i$ find the set of camera matrices, $\mathtt{P}^i$, and the points $\mathbf{X}_j$ such that $\mathtt{P}^i\mathbf{X}_j = \mathbf{x}_j^i$. Without further restriction on the $\mathtt{P}^i$ or $\mathbf{X}_j$, such a reconstruction is a projective reconstruction, because the points $\mathbf{X}_j$ may differ by an arbitrary 3D projective transformation from the true reconstruction.

Bundle adjustment. If the image measurements are noisy then the equations $\mathbf{x}_j^i = \mathtt{P}^i\mathbf{X}_j$ will not be satisfied exactly. In this case we seek the Maximum Likelihood (ML) solution assuming that the measurement noise is Gaussian: we wish to estimate projection matrices $\hat{\mathtt{P}}^i$ and 3D points $\widehat{\mathbf{X}}_j$ which project exactly to image points $\hat{\mathbf{x}}_j^i$ as $\hat{\mathbf{x}}_j^i = \hat{\mathtt{P}}^i\widehat{\mathbf{X}}_j$, and also minimize the image distance between the reprojected point and detected (measured) image points $\mathbf{x}_j^i$ for every view in which the 3D point appears, i.e.

$$\min_{\hat{\mathtt{P}}^i, \widehat{\mathbf{X}}_j} \sum_{ij} d(\hat{\mathtt{P}}^i\widehat{\mathbf{X}}_j, \mathbf{x}_j^i)^2 \tag{18.1}$$

where $d(\mathbf{x}, \mathbf{y})$ is the geometric image distance between the homogeneous points $\mathbf{x}$ and $\mathbf{y}$. This estimation involving minimizing the reprojection error is known as *bundle adjustment* – it involves adjusting the bundle of rays between each camera centre and the set of 3D points (and equivalently between each 3D point and the set of camera centres).

Bundle adjustment should generally be used as a final step of any reconstruction algorithm. This method has the advantages of being tolerant of missing data while providing a true ML estimate. At the same time it allows assignment of individual covariances (or more general PDFs) to each measurement and may also be extended to include estimates of priors and constraints on camera parameters or point positions. In short, it would seem to be an ideal algorithm, except for the fact that: (i) it requires a good initialization to be provided, and (ii) it can become an extremely large minimization problem because of the number of parameters involved. We will discuss briefly these two points.

Iterative minimization. Since each camera has 11 degrees of freedom and each 3-space point 3 degrees of freedom, a reconstruction involving n points over m views requires minimization over $3n + 11m$ parameters. In fact, since entities are often over-parametrized (e.g. using 12 parameters for the homogeneous $\mathtt{P}$ matrix) this may be a lower bound. If the Levenberg–Marquardt algorithm is used to minimize (18.1) then matrices of dimension $(3n + 11m) \times (3n + 11m)$ must be factored (or sometimes inverted). As m and n increase this becomes extremely costly, and eventually impossible. There are several solutions to this problem:

(i) **Reduce n and/or m.** Do not include all the views or all the points, and fill these in later by resectioning or triangulation respectively; or, partition the data into several sets, bundle adjust each set separately and then merge. Such strategies are discussed further in section 18.6.

(ii) **Interleave.** Alternate minimizing reprojection error by varying the cameras with minimizing reprojection error by varying the points. Since each point is estimated independently given fixed cameras, and similarly each camera is estimated independently from fixed points, the largest matrix that must be inverted is the 11×11 matrix used to estimate one camera. Interleaving minimizes the same cost function as bundle adjustment, so the same solution should be obtained (provided there is a unique minimum), but it may take longer to converge. Interleaving is compared with bundle adjustment in [Triggs-00a].

(iii) **Sparse methods.** These are described in appendix 6(*p*597).

Initial solution. Several methods for initialization are described in the following sections. If the problem is restricted to affine cameras then factorization (section 18.2) gives a closed form optimal solution provided points are imaged in every view. Even with projective cameras there is an (iterative) factorization method (section 18.4) available provided points are imaged in every view. If there is more information available on the data, for example that it is partly coplanar, then again a closed form solution is possible (section 18.5). Finally, hierarchical methods can be used as described in section 18.6 for the case where points are not visible in every view.

18.2 Affine reconstruction – the factorization algorithm

In this section we describe reconstruction from a set of image point correspondences for images acquired by affine cameras. As described in section 17.5.1 the reconstruction in this case is affine.

The factorization algorithm of Tomasi and Kanade [Tomasi-92] to be presented below and summarized in algorithm 18.1 has the following property:

- *Under an assumption of isotropic mean-zero Gaussian noise independent and equal for each measured point, factorization achieves a Maximum Likelihood affine reconstruction.*

This fact was first pointed out by Reid and Murray [Reid-96]. However, the method requires a measurement of each point in all views. This is a limitation in practice, since matched points may be absent in some views.

An affine camera may be characterized by having its last row equal to $(0, 0, 0, 1)$. In this section, however, we will denote it somewhat differently, separating out the translation and the pure linear transformation part of the camera map. Thus we write

$$\begin{pmatrix} x \\ y \end{pmatrix} = \mathtt{M} \begin{pmatrix} \mathrm{X} \\ \mathrm{Y} \\ \mathrm{Z} \end{pmatrix} + \mathbf{t}$$

where $\mathtt{M}$ is a 2×3 matrix and $\mathbf{t}$ a 2-vector. From here on for ease of readability $\mathbf{x}$ represents an *in*homogeneous image point $\mathbf{x} = (x, y)^{\mathsf{T}}$, and $\mathbf{X}$ an *in*homogeneous world point $\mathbf{X} = (\mathrm{X}, \mathrm{Y}, \mathrm{Z})^{\mathsf{T}}$.

Our goal is to find a reconstruction to minimize geometric error in image coordinate measurements. That is, we wish to estimate cameras $\{\mathtt{M}^i, \mathbf{t}^i\}$ and 3D points $\{\mathbf{X}_j\}$ such that the distance between the estimated image points $\hat{\mathbf{x}}_j^i = \mathtt{M}^i \mathbf{X}_j + \mathbf{t}^i$ and measured image points $\mathbf{x}_j^i$ is minimized

$$\min_{\mathtt{M}^i, \mathbf{t}^i, \mathbf{X}_j} \sum_{ij} \left|\left| \mathbf{x}_j^i - \hat{\mathbf{x}}_j^i \right|\right|^2 = \min_{\mathtt{M}^i, \mathbf{t}^i, \mathbf{X}_j} \sum_{ij} \left|\left| \mathbf{x}_j^i - (\mathtt{M}^i \mathbf{X}_j + \mathbf{t}^i) \right|\right|^2 . \tag{18.2}$$

As is common in such minimization problems the translation vector $\mathbf{t}^i$ can be eliminated in advance by choosing the centroid of the points as the origin of the coordinate system. This is a consequence of the geometric fact that an affine camera maps the centroid of a set of 3D points to the centroid of their projections. Thus, if the coordinate origin is chosen as the centroid of the 3D points and of each set of image points then it follows that $\mathbf{t}^i = \mathbf{0}$. This step requires that the same n points be imaged in all views, i.e. that there are no views in which the image coordinates of any point are unknown. An analytical derivation of this result goes like this. The minimization with respect to $\mathbf{t}^i$ requires that

$$\frac{\partial}{\partial \mathbf{t}^i} \sum_{kj} \left|\left| \mathbf{x}_j^k - (\mathtt{M}^k \mathbf{X}_j + \mathbf{t}^k) \right|\right|^2 = \mathbf{0}$$

which after a brief calculation reduces to $\mathbf{t}^i = \langle \mathbf{x}^i \rangle - \mathtt{M}^i \langle \mathbf{X} \rangle$, where the centroids are

Objective

Given $n \geq 4$ image point correspondences over m views $\mathbf{x}_j^i$, $j = 1, \ldots, n$; $i = 1, \ldots, m$, determine affine camera matrices $\{\mathtt{M}^i, \mathbf{t}^i\}$ and 3D points $\{\mathbf{X}_j\}$ such that the reprojection error

$$\sum_{ij} \left|\left|\mathbf{x}_j^i - (\mathtt{M}^i \mathbf{X}_j + \mathbf{t}^i)\right|\right|^2$$

is minimized over $\{\mathtt{M}^i, \mathbf{t}^i, \mathbf{X}_j\}$, with $\mathtt{M}^i$ a 2×3 matrix, $\mathbf{X}_j$ a 3-vector, and $\mathbf{x}_j^i = (x_j^i, y_j^i)^\mathsf{T}$ and $\mathbf{t}^i$ are 2-vectors.

Algorithm

(i) **Computation of translations.** Each translation $\mathbf{t}^i$ is computed as the centroid of points in image i, namely

$$\mathbf{t}^i = \langle \mathbf{x}^i \rangle = \frac{1}{n} \sum_j \mathbf{x}_j^i.$$

(ii) **Centre the data**. Centre the points in each image by expressing their coordinates with respect to the centroid:

$$\mathbf{x}_j^i \leftarrow \mathbf{x}_j^i - \langle \mathbf{x}^i \rangle.$$

Henceforth work with these centred coordinates.

(iii) **Construct the** $2m \times n$ **measurement matrix** $\mathtt{W}$ from the centred data, as defined in (18.5), and compute its SVD $\mathtt{W} = \mathtt{UDV}^\mathsf{T}$.

(iv) Then the matrices $\mathtt{M}^i$ are obtained from the first three columns of $\mathtt{U}$ multiplied by the singular values:

$$\begin{bmatrix} \mathtt{M}^1 \\ \mathtt{M}^2 \\ \vdots \\ \mathtt{M}^m \end{bmatrix} = \begin{bmatrix} \sigma_1 \mathbf{u}_1 & \sigma_2 \mathbf{u}_2 & \sigma_3 \mathbf{u}_3 \end{bmatrix}.$$

The vectors $\mathbf{t}^i$ are as computed in step (i) and the 3D structure is read from the first three columns of $\mathtt{V}$

$$\begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \ldots & \mathbf{X}_n \end{bmatrix} = \begin{bmatrix} \mathbf{v}_1 & \mathbf{v}_2 & \mathbf{v}_3 \end{bmatrix}^\mathsf{T}.$$

Algorithm 18.1. *The factorization algorithm to determine the MLE for an affine reconstruction from n image correspondences over m views (under Gaussian image noise).*

$\langle \mathbf{x}^i \rangle = \frac{1}{n} \sum_j \mathbf{x}_j^i$ and $\langle \mathbf{X} \rangle = \frac{1}{n} \sum_j \mathbf{X}_j$. The origin of the 3D frame is arbitrary, so may be chosen to coincide with the centroid $\langle \mathbf{X} \rangle$, in which case $\langle \mathbf{X} \rangle = \mathbf{0}$ and

$$\mathbf{t}^i = \langle \mathbf{x}^i \rangle. \tag{18.3}$$

It follows that if we measure the image coordinates with respect to a coordinate origin based at the centroid of the projected points, then $\mathbf{t}^i = \mathbf{0}$. Thus, we replace each $\mathbf{x}_j^i$ by $\mathbf{x}_j^i - \langle \mathbf{x}^i \rangle$. Henceforth we will assume that this has been done, and work with the centred coordinates. With respect to these new coordinates $\mathbf{t}^i = \mathbf{0}$, and so (18.2) reduces to

$$\min_{\mathtt{M}^i, \mathbf{X}_j} \sum_{ij} \left|\left|\mathbf{x}_j^i - \hat{\mathbf{x}}_j^i\right|\right|^2 = \min_{\mathtt{M}^i, \mathbf{X}_j} \sum_{ij} \left|\left|\mathbf{x}_j^i - \mathtt{M}^i \mathbf{X}_j\right|\right|^2. \tag{18.4}$$

The minimization problem now has a very simple form when written as a matrix. The *measurement* matrix $\mathtt{W}$ is the $2m \times n$ matrix composed of the centred coordinates of the measured image points

$$\mathtt{W} = \begin{bmatrix} \mathbf{x}_1^1 & \mathbf{x}_2^1 & \dots & \mathbf{x}_n^1 \\ \mathbf{x}_1^2 & \mathbf{x}_2^2 & \dots & \mathbf{x}_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_1^m & \mathbf{x}_2^m & \dots & \mathbf{x}_n^m \end{bmatrix}. \tag{18.5}$$

Since each $\mathbf{x}_j^i = \mathtt{M}^i \mathbf{X}_j$, the complete set of equations may be written as

$$\mathtt{W} = \begin{bmatrix} \mathtt{M}^1 \\ \mathtt{M}^2 \\ \vdots \\ \mathtt{M}^m \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \dots & \mathbf{X}_n \end{bmatrix}.$$

In the presence of noise this equation will not be satisfied exactly, so instead we seek a matrix $\hat{\mathtt{W}}$ as close as possible to $\mathtt{W}$ in Frobenius norm, such that $\hat{\mathtt{W}}$ may be decomposed as

$$\hat{\mathtt{W}} = \begin{bmatrix} \hat{\mathbf{x}}_1^1 & \hat{\mathbf{x}}_2^1 & \dots & \hat{\mathbf{x}}_n^1 \\ \hat{\mathbf{x}}_1^2 & \hat{\mathbf{x}}_2^2 & \dots & \hat{\mathbf{x}}_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \hat{\mathbf{x}}_1^m & \hat{\mathbf{x}}_2^m & \dots & \hat{\mathbf{x}}_n^m \end{bmatrix} = \begin{bmatrix} \mathtt{M}^1 \\ \mathtt{M}^2 \\ \vdots \\ \mathtt{M}^m \end{bmatrix} \begin{bmatrix} \mathbf{X}_1 & \mathbf{X}_2 & \dots & \mathbf{X}_n \end{bmatrix}. \tag{18.6}$$

In this case it may be verified that

$$\left|\left|\mathtt{W} - \hat{\mathtt{W}}\right|\right|_{\mathrm{F}}^2 = \sum_{ij}\left(\mathtt{W}_{ij} - \hat{\mathtt{W}}_{ij}\right)^2 = \sum_{ij}\left|\left|\mathbf{x}_j^i - \hat{\mathbf{x}}_j^i\right|\right|^2 = \sum_{ij}\left|\left|\mathbf{x}_j^i - \mathtt{M}^i\mathbf{X}_j\right|\right|^2$$

Comparing this with (18.4) we find that minimizing the required geometric error is equivalent to finding such a $\hat{\mathtt{W}}$ as close as possible to $\mathtt{W}$ in Frobenius norm.

Note that a matrix $\hat{\mathtt{W}}$ satisfying (18.6) is the product of a $2m \times 3$ *motion matrix* $\hat{\mathtt{M}}$, and a $3 \times n$ *structure matrix* $\hat{\mathtt{X}}$; consequently $\hat{\mathtt{W}} = \hat{\mathtt{M}}\hat{\mathtt{X}}$ has rank 3. In other words we seek a rank 3 matrix which is closest to $\mathtt{W}$ in Frobenius norm. Such a matrix may be determined by the SVD of $\mathtt{W}$ truncated to rank 3. In more detail, if the SVD of $\mathtt{W} = \mathtt{UDV}^\mathsf{T}$ then $\hat{\mathtt{W}} = \mathtt{U}_{2m\times 3}\mathtt{D}_{3\times 3}\mathtt{V}_{3\times n}^\mathsf{T}$ is the rank 3 matrix which is closest to $\mathtt{W}$ in the Frobenius norm, where $\mathtt{U}_{2m\times 3}$ consists of the first 3 columns of $\mathtt{U}$, $\mathtt{V}_{3\times n}^\mathsf{T}$ consists of the first 3 rows of $\mathtt{V}^\mathsf{T}$, and $\mathtt{D}_{3\times 3}$ is the diagonal matrix containing the first 3 singular values, $\mathtt{D}_{3\times 3} = \mathrm{diag}(\sigma_1, \sigma_2, \sigma_3)$.

Note that the choice of $\hat{\mathtt{M}}$ and $\hat{\mathtt{X}}$ is not unique. For example $\hat{\mathtt{M}}$ may be chosen as $\hat{\mathtt{M}} = \mathtt{U}_{2m\times 3}\mathtt{D}_{3\times 3}$ and $\hat{\mathtt{X}} = \mathtt{V}_{3\times n}^\mathsf{T}$, or as $\hat{\mathtt{M}} = \mathtt{U}_{2m\times 3}$, $\hat{\mathtt{X}} = \mathtt{D}_{3\times 3}\mathtt{V}_{3\times n}^\mathsf{T}$ since in either case $\hat{\mathtt{W}} = \hat{\mathtt{M}}\hat{\mathtt{X}} = \mathtt{U}_{2m\times 3}\mathtt{D}_{3\times 3}\mathtt{V}_{3\times n}^\mathsf{T}$.

Affine ambiguity. In fact for any such choice there is an additional ambiguity since an arbitrary 3×3 rank 3 matrix $\mathtt{A}$ may be inserted in the decomposition as $\hat{\mathtt{W}} = \hat{\mathtt{M}}\mathtt{A}\mathtt{A}^{-1}\hat{\mathtt{X}} = (\hat{\mathtt{M}}\mathtt{A})(\mathtt{A}^{-1}\hat{\mathtt{X}})$. This means that the camera matrices $\mathtt{M}^i$, which are obtained from $\hat{\mathtt{M}}$, and the 3D points $\mathbf{X}_j$, which are obtained from $\hat{\mathtt{X}}$, are determined up to multiplication by a common matrix $\mathtt{A}$. In other words the MLE reconstruction is affine.

This affine reconstruction may be upgraded to a metric reconstruction by supplying metric information on the scene as described in section 10.4.2(*p*272), or by using auto-calibration methods as described in chapter 19, or a combination of the two. Note that in the case of affine cameras only three internal parameters need be specified (compared to five for projective cameras) and the auto-calibration task is correspondingly simpler.

18.2.1 Affine multiple view tensors

The factorization algorithm provides an optimal method for computing the affine multiview tensors from image point correspondences. These tensors are the affine fundamental matrix, affine trifocal tensor, and affine quadrifocal tensor. In each case the algorithm determines the camera matrices up to an overall affine ambiguity. The tensors may then be computed directly from the camera matrices (as for instance in chapter 17). The affine ambiguity of 3-space is irrelevant when computing the tensors since they are unaffected by affine transformations of 3-space. In fact it is not necessary to compute the full SVD of $\mathtt{W}$ since only the $\mathtt{U}$ part of the decomposition is required. If the number of points n is large compared with the number of views then very great savings can be made in the computation of the SVD by not determining $\mathtt{V}$ (see table A4.1(*p*587)).

An alternative to using the SVD is to use the eigenvalue decomposition of $\mathtt{W}\mathtt{W}^{\mathsf{T}}$, since $\mathtt{W}\mathtt{W}^{\mathsf{T}} = (\mathtt{U}\mathtt{D}\mathtt{V}^{\mathsf{T}})(\mathtt{U}\mathtt{D}\mathtt{V}^{\mathsf{T}})^{\mathsf{T}} = \mathtt{U}\mathtt{D}^2\mathtt{U}^{\mathsf{T}}$. In the case of three views (computation of the trifocal tensor) the matrix $\mathtt{W}\mathtt{W}^{\mathsf{T}}$ has dimension only 9×9. Thus this approach can mean significant savings. However, it is numerically inferior, since forming $\mathtt{W}\mathtt{W}^{\mathsf{T}}$ causes the condition number of the matrix to be squared (see the discussion of SVD in [Golub-89]). Since we need just the three largest eigenvectors, that may not be such a problem in this case. However, the savings of this approach will not be so great given an implementation of the SVD that avoids computing $\mathtt{V}$.

The factorization method may be used to compute any of the multiple-view affine tensors. For the affine fundamental matrix, however, algorithm 14.1(*p*351) described in chapter 14 is more direct. The results of both the methods are identical.

18.2.2 Triangulation and reprojection using subspaces

The factorization algorithm also provides an optimal method for computing the images of new points or of points not observed in all views. Again the affine ambiguity of 3-space is irrelevant.

A column of $\mathtt{W}$ is the set of all corresponding image points for the point $\mathbf{X}_j$ and is referred to as a point's *trajectory*. The rank 3 decomposition (18.6) of $\hat{\mathtt{W}}$ as $\hat{\mathtt{W}} = \hat{\mathtt{M}}\hat{\mathtt{X}}$ shows that all trajectories lie in a 3 dimensional subspace. In particular the trajectory (i.e. all image projections) of a new point $\mathbf{X}$ may be obtained as $\hat{\mathtt{M}}\mathbf{X}$. This is simply a linear weighting of the three columns of $\hat{\mathtt{M}}$.

Suppose we have observed a new point $\mathbf{X}$ in some (not all) views, and wish to predict its projection in the other views. This is carried out in two steps: first triangulation to find the pre-image $\mathbf{X}$, and then reprojection as $\hat{\mathtt{M}}\mathbf{X}$ to generate its image in all views. Note that the projected points will not coincide exactly with the measured (noisy) points. In the triangulation step we wish to find the point $\mathbf{X}$ that minimizes reprojection error, and this corresponds to finding the point in the linear subspace spanned by the columns of $\hat{\mathtt{M}}$ closest to the trajectory. This closest point is found by projecting the trajectory onto the subspace (in a similar manner to algorithm 4.7(p130)).

In more detail suppose we have computed a set of affine cameras $\{\mathtt{M}^i, \mathbf{t}^i\}$ then the triangulation problem may be solved linearly for any number of views. The image points $\mathbf{x}^i = \mathtt{M}^i\mathbf{X} + \mathbf{t}^i$ give a pair of linear equations $\mathtt{M}^i\mathbf{X} = \mathbf{x}^i - \mathbf{t}^i$ in the entries of $\mathbf{X}$. Given sufficiently many such equations (arising from known values of $\mathbf{x}^i$) one can find the linear least-squares solution for $\mathbf{X}$, using algorithm A5.1(p589), the pseudo-inverse (see result A5.1(p590)) or algorithm A5.3(p591). Note that if the data $\mathbf{x}^i$ is centred using the same transformation applied in step (ii) of algorithm 18.1, then the translation vectors $\mathbf{t}^i$ in the affine triangulation method may be taken to be zero.

In practice triangulation and reprojection provides a method of ‘filling in’ points that are missed during tracking or multiple view matching.

18.2.3 Affine Reconstruction by Alternation

Suppose a set of image coordinates $\mathbf{x}_j^i$ are given as in algorithm 18.1, and we wish to perform affine reconstruction. We have seen that affine triangulation may be carried out linearly. Thus, if the affine camera matrices represented by $\{\mathtt{M}^i, \mathbf{t}^i\}$ are known, then the optimal point positions $\mathbf{X}_j$ may be computed by a linear least-squares method such as the normal equations method of algorithm A5.3(p591).

Conversely, if the points $\mathbf{X}_j$ are known, then the equations $\mathbf{x}_j^i = \mathtt{M}^i\mathbf{X}_j + \mathbf{t}^i$ are linear in $\mathtt{M}^i$ and $\mathbf{t}^i$. So it is once again possible simply to solve for $\{\mathtt{M}^i, \mathbf{t}^i\}$ by linear least-squares.

This suggests a method of affine reconstruction in which linear least-squares methods are used to solve alternately for the points $\mathbf{X}_j$ and the cameras $\{\mathtt{M}^i, \mathbf{t}^i\}$. This method of alternation is not to be recommended as a general method for reconstruction, or for solving optimization problems in general. In the case of affine reconstruction, however, it can be proven to converge rapidly to the optimal solution, starting from a random starting point. This method of affine reconstruction has the advantage of working with missing data, or with covariance-weighted data which algorithm 18.1 will not, though in the missing data or covariance-weighted case, global optimal convergence is not guaranteed in all cases.

18.3 Non-rigid factorization

Throughout the book it has been assumed that we are viewing a rigid scene and that only the relative motion between the camera and scene is to be modelled. In this section we relax this assumption and consider the problem of recovering a reconstruction for a deforming object. It will be shown that if the deformation is modelled as a linear combination over basis shapes then the reconstruction *and the basis shapes* may be recovered with a simple modification of the factorization algorithm of section 18.2.

An example where this type of situation arises is in a sequence of images of a per-

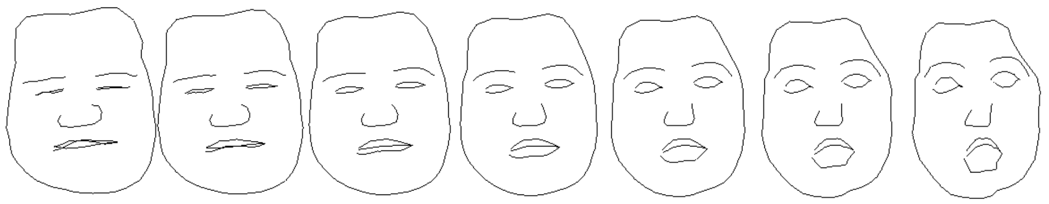


Fig. 18.1. **Shape basis**. *A face template is represented by N equally spaced 2D points (here $N = 140$). The central face of the seven is the mean shape and the faces to the left or right are generated by adding or subtracting, respectively, the basis shape that accounts for the maximum variation in the training set. In this case the basis spans expressions from surprised to disgruntled. Facial expressions are learnt by tracking the face of an actor with the template whilst he changes expression but does not vary his head pose. Each frame of the training sequence generates a set of N 2D points, and the coordinates of these are rewritten as a $2N$-vector. A $2N \times f$ matrix is then composed from these vectors, where f is the number of training frames, and the basis shapes are computed from the singular vectors of this matrix. Figure courtesy of Richard Bowden.*

son's head which moves and also changes expression. The motion of the head may be modelled as a rigid rotation, and the change of expression relative to the fixed head may be modelled as a linear combination over basis sets. For example the mouth outline may be represented by a set of points.

Suppose the set of n scene points $\mathbf{X}_j$ may be represented as a linear combination of l basis shapes $\mathtt{B}_k$ so that at a particular time i:

$$\begin{bmatrix} \mathbf{X}_1^i & \mathbf{X}_2^i & \dots & \mathbf{X}_n^i \end{bmatrix} = \sum_{k=1}^{l} \alpha_k^i \begin{bmatrix} \mathbf{B}_{1k} & \mathbf{B}_{2k} & \dots & \mathbf{B}_{nk} \end{bmatrix} = \sum_k \alpha_k^i \mathtt{B}_k$$

where here both the scene points $\mathbf{X}_j^i$ and the basis points $\mathbf{B}_{jk}$ are inhomogeneous points represented by 3-vectors, and $\mathtt{B}_k$ is a $3 \times n$ matrix. Typically the number of basis shapes, l, is much smaller than the number of points, n. The coefficients α_k^i may be different at each time i, and the resulting differing combination of basis shapes generates the deformation. An example is shown in figure 18.1.

In the forward model of image generation each view i is acquired by an affine camera and gives the image points

$$\mathbf{x}_j^i = \mathtt{M}^i \sum_k \alpha_k^i \mathbf{B}_{jk} + \mathbf{t}^i.$$

It will again be assumed that image point matches are available for all views. Our goal is to estimate cameras $\{\mathtt{M}^i, \mathbf{t}^i\}$ and 3D structure $\{\alpha_k^i, \mathbf{B}_{jk}\}$ from the measured image points $\{\mathbf{x}_j^i\}$, such that the distance between the estimated image points $\hat{\mathbf{x}}_j^i = \mathtt{M}^i \sum_k \alpha_k^i \mathbf{B}_{jk} + \mathbf{t}^i$ and measured image points is minimized

$$\min_{\mathtt{M}^i, \mathbf{t}^i, \alpha_k^i, \mathbf{B}_{jk}} \sum_{ij} \left|\left| \mathbf{x}_j^i - \hat{\mathbf{x}}_j^i \right|\right|^2 = \min_{\mathtt{M}^i, \mathbf{t}^i, \alpha_k^i, \mathbf{B}_{jk}} \sum_{ij} \left|\left| \mathbf{x}_j^i - (\mathtt{M}^i \sum_k \alpha_k^i \mathbf{B}_{jk} + \mathbf{t}^i) \right|\right|^2.$$

As in affine factorization the translation may be eliminated by centring the measured image points, and it will be assumed from here on that this has been done. Then the problem reduces to

$$\min_{\mathtt{M}^i,\alpha_k^i,\mathbf{B}_{jk}} \sum_{ij} \left\|\mathbf{x}_j^i - \hat{\mathbf{x}}_j^i\right\|^2 = \min_{\mathtt{M}^i,\alpha_k^i,\mathbf{B}_{jk}} \sum_{ij} \left\|\mathbf{x}_j^i - \mathtt{M}^i \sum_k \alpha_k^i \mathbf{B}_{jk}\right\|^2. \tag{18.7}$$

The complete set of equations $\hat{\mathbf{x}}_j^i = \mathtt{M}^i \sum_k \alpha_k^i \mathbf{B}_{jk}$ may be written

$$\hat{\mathtt{W}} = \begin{bmatrix} \mathtt{M}^1\left(\alpha_1^1\mathtt{B}_1 + \alpha_2^1\mathtt{B}_2 + \ldots \alpha_l^1\mathtt{B}_l\right) \\ \mathtt{M}^2\left(\alpha_1^2\mathtt{B}_1 + \alpha_2^2\mathtt{B}_2 + \ldots \alpha_l^2\mathtt{B}_l\right) \\ \vdots \\ \mathtt{M}^m\left(\alpha_1^m\mathtt{B}_1 + \alpha_2^m\mathtt{B}_2 + \ldots \alpha_l^m\mathtt{B}_l\right) \end{bmatrix} = \begin{bmatrix} \alpha_1^1\mathtt{M}^1 & \alpha_2^1\mathtt{M}^1 & \ldots & \alpha_l^1\mathtt{M}^1 \\ \alpha_1^2\mathtt{M}^2 & \alpha_2^2\mathtt{M}^2 & \ldots & \alpha_l^2\mathtt{M}^2 \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^m\mathtt{M}^m & \alpha_2^m\mathtt{M}^m & \ldots & \alpha_l^m\mathtt{M}^m \end{bmatrix} \begin{bmatrix} \mathtt{B}_1 \\ \mathtt{B}_2 \\ \vdots \\ \mathtt{B}_l \end{bmatrix} \tag{18.8}$$

This rearrangement shows that the $2m \times n$ matrix $\hat{\mathtt{W}}$ may be decomposed as a product of a $2m \times 3l$ motion matrix $\hat{\mathtt{M}}$ and $3l \times n$ structure matrix $\hat{\mathtt{B}}$, and consequently $\hat{\mathtt{W}}$ has maximum rank $3l$.

As in rigid factorization a rank $3l$ decomposition may be obtained from the measurement matrix $\mathtt{W}$ by truncating the SVD of $\mathtt{W}$ to rank $3l$. Also, as in rigid factorization, the decomposition $\hat{\mathtt{W}} = \hat{\mathtt{M}}\hat{\mathtt{B}}$ is not uniquely defined since an arbitrary $3l \times 3l$ rank $3l$ matrix $\mathtt{A}$ may be inserted in the decomposition as $\hat{\mathtt{W}} = \hat{\mathtt{M}}\mathtt{A}\mathtt{A}^{-1}\hat{\mathtt{B}} = (\hat{\mathtt{M}}\mathtt{A})(\mathtt{A}^{-1}\hat{\mathtt{B}})$. In the rigid case this resulted in a straightforward affine ambiguity in the reconstruction. However, in the non-rigid case there is the additional requirement that the motion matrix has the replicated block structure of (18.8), and we return to this below. This block structure is not required for determining a point's image motion, as will now be discussed.

18.3.1 Subspaces and tensors

In the case of rigid factorization (18.6), as discussed in section 18.2.2, the trajectories lie in a 3 dimensional subspace, and any trajectory may be generated as a linear combination of the columns of $\hat{\mathtt{M}}$ (the $2m \times 3$ motion matrix). Similarly in the case of non-rigid factorization, (18.8), the trajectories lie in a $3l$ dimensional subspace, and any trajectory may be generated as a linear combination of the columns of $\hat{\mathtt{M}}$ (the $2m \times 3l$ motion matrix).

Suppose we observe a new point in a subset of the views, how many images are required before its position can be predicted in all the other views? This is simply a question of triangulation: in rigid factorization a 3-space point has 3 degrees of freedom, and must be observed in two views to obtain the necessary 3 measurements. In non-rigid factorization we need to specify $3l$ degrees of freedom (the number of rows in the $\hat{\mathtt{B}}$ matrix), and this requires $3l/2$ images. For example, if $l = 2$ the subspace is six dimensional (the columns of the $\hat{\mathtt{B}}$ matrix are 6-vectors), and given the image position in three views, the image position in all views is then determined by an analogue of affine triangulation (section 18.2.2) even though the object is deforming.

Independently moving objects. Low-rank factorization methods also arise when there are independently moving objects in the scene. For instance suppose the scene is divided into two objects, each moving independently of the other, and viewed by

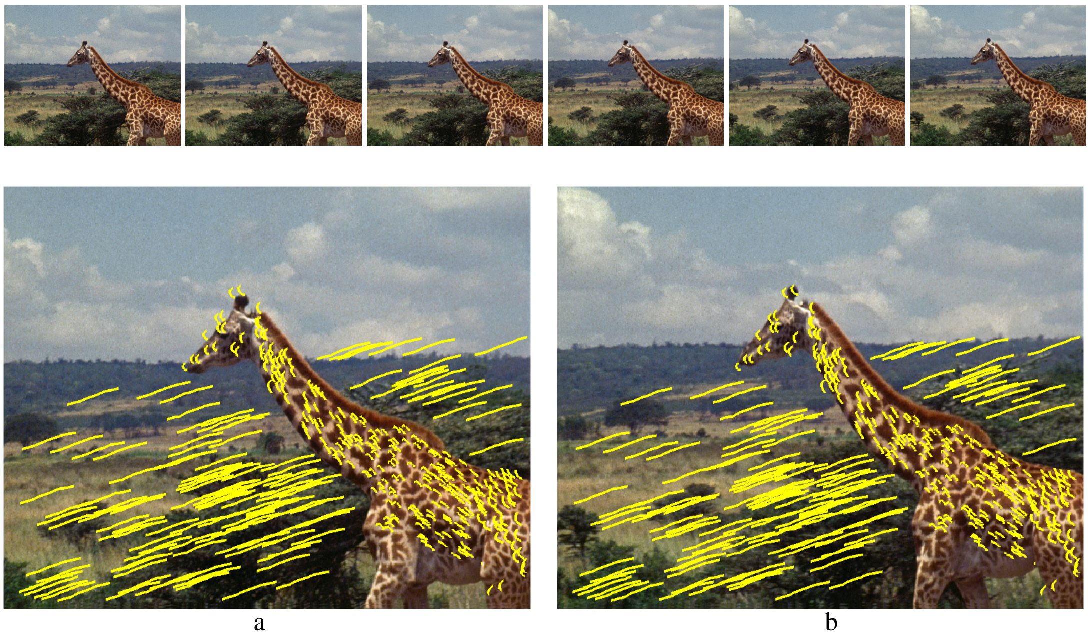


Fig. 18.2. **Non-rigid motion sequence**. *Top row: alternate frames from a sequence in which a giraffe gracefully walks and flexes its neck, whilst the camera pans to match its speed. Bottom row: point tracks showing the motion over (a) the 10 previous, and (b) the 10 forthcoming frames. These tracks are computed using non-rigid factorization and lie in a six dimensional subspace. Note the very different trajectories of the (rigid) background from the (deforming) foreground. Yet these are all spanned by the six basis vectors of the motion matrix. The rank can be accounted for as follows: the sequence motion is effectively that of two planes of points moving independently relative to the camera. The background is a rigid object represented by a plane and contributes 2 to the rank. The giraffe in the foreground is represented as a non-rigid object by a set of $l = 2$ planar basis shapes and contributes 4 to the rank. Figures courtesy of Andrew Fitzgibbon and Aeron Morgan.*

a moving affine camera. In this case, the columns of the measurement matrix corresponding to points on one object will have rank 3, and those corresponding to the other object will also have rank 3. The total rank of the measurement matrix will be 6. In degenerate configurations in which one object's points all lie in a plane, its contribution to the rank will be only 2. This multibody factorization problem has been studied in some depth in [Costeira-98].

An example of point tracks residing in a low dimensional subspace is shown in figure 18.2.

The existence of the analogue of the fundamental matrix and trifocal tensor depends on the dimension of the subspace. For example suppose the subspace has odd dimension (e.g. $l = 3$ so it is 9 dimensional) then given point measurements in $\lfloor 3l/2 \rfloor$ views (e.g. 4 views) the corresponding point in any other view is constrained to a line, the analogue of an epipolar line, since there is one fewer measurement than degrees of freedom of the subspace. However, if the dimension is even (e.g. $l = 2$ so it is 6 dimensional) then given point measurements in $l/2$ views (e.g. 3 views) the corresponding point in any other view is completely determined. Multi-view tensors can be built for the non-rigid $l > 1$ subspaces using methods similar to those developed in chapter 17 for $l = 1$ 3-space points.

18.3.2 Recovering the camera motion

In rigid-factorization the camera matrices are obtained relatively easily from the motion matrix $\hat{\mathtt{M}}$ – all that is required is to remove a global affine ambiguity specified by a 3×3 matrix $\mathtt{A}$ as described on page 438.

In the non-rigid case, the analogous problem is not so straightforward. It is a simple matter to obtain the motion matrix:

(i) Construct the $2m \times n$ measurement matrix $\mathtt{W}$ from the centred data, as defined in (18.5), and compute its SVD $\mathtt{W} = \mathtt{UDV}^{\mathsf{T}}$.
(ii) Then the motion matrix $\hat{\mathtt{M}}$ is obtained from the first $3l$ columns of $\mathtt{U}$ multiplied by the singular values as $\hat{\mathtt{M}} = \left[\begin{array}{cccc} \sigma_1\mathbf{u}_1 & \sigma_2\mathbf{u}_2 & \ldots & \sigma_{3l}\mathbf{u}_{3l} \end{array} \right]$,

but the matrix obtained by this route will *not* in general have the block structure of (18.8). As in the case of rigid-factorization the motion matix is determined up to post-multiplication by a matrix $\mathtt{A}$, which here is $3l \times 3l$. The task is then to determine $\mathtt{A}$ such that $\hat{\mathtt{M}}\mathtt{A}$ has the required block structure of (18.8) *and* also to remove the usual affine ambiguity such that each block conforms to any available constraints on the camera calibration (for example identical internal parameters over all views).

Various methods for determining $\mathtt{A}$ have been investigated (see [Brand-01, Torresani-01]), but these do not impose the full block structure, and currently there is not a satisfactory solution to this problem. Once an initial solution has been obtained by some means, then the correct form can be imposed by bundle adjustment of (18.7).

18.4 Projective factorization

The affine factorization method does not apply directly to projective reconstruction. It was observed in [Sturm-96], however, that if one knows the "projective depth" of each of the points then the structure and camera parameters may be estimated by a simple factorization algorithm similar in style to the affine factorization algorithm.

Consider a set of image points $\mathbf{x}_j^i = \mathtt{P}^i\mathbf{X}_j$. This equation representing the projective mapping is to be interpreted as true only up to a constant factor. Writing these constant factors explicitly, we have $\lambda_j^i\mathbf{x}_j^i = \mathtt{P}^i\mathbf{X}_j$. In this equation, and henceforth in the description of the projective factorization algorithm, the notation $\mathbf{x}_j^i$ means the 3-vector $(x_j^i, y_j^i, 1)^{\mathsf{T}}$ representing an image point. Thus the third coordinate is equal to unity, and x_j^i and y_j^i are the actual measured image coordinates. Provided that each point is visible in every view, so that $\mathbf{x}_j^i$ is known for all i, j, the complete set of equations may be written as a single matrix equation as follows:

$$\left[\begin{array}{cccc} \lambda_1^1\mathbf{x}_1^1 & \lambda_2^1\mathbf{x}_2^1 & \ldots & \lambda_n^1\mathbf{x}_n^1 \\ \lambda_1^2\mathbf{x}_1^2 & \lambda_2^2\mathbf{x}_2^2 & \ldots & \lambda_n^m\mathbf{x}_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_1^m\mathbf{x}_1^m & \lambda_2^m\mathbf{x}_2^m & \ldots & \lambda_n^m\mathbf{x}_n^m \end{array} \right] = \left[\begin{array}{c} \mathtt{P}^1 \\ \mathtt{P}^2 \\ \vdots \\ \mathtt{P}^m \end{array} \right] [\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n] \ . \qquad (18.9)$$

Objective

Given a set of n image points seen in m views:

$$\mathbf{x}_j^i \; ; \; i = 1, \ldots, m, \; j = 1, \ldots, n$$

compute a projective reconstruction.

Algorithm

(i) Normalize the image data using isotropic scaling as in section 4.4.4(p107).

(ii) Start with an initial estimate of the projective depths λ_j^i. This may be obtained by techniques such as an initial projective reconstruction, or else by setting all $\lambda_j^i = 1$.

(iii) Normalize the depths λ_j^i by multiplying rows and columns by constant factors. One method is to do a pass setting the norms of all rows to 1, then a similar pass on columns.

(iv) Form the $3m \times n$ measurement matrix on the left of (18.9), find its nearest rank-4 approximation using the SVD and decompose to find the camera matrices and 3D points.

(v) **Optional iteration.** Reproject the points into each image to obtain new estimates of the depths and repeat from step (ii).

Algorithm 18.2. *Projective reconstruction through factorization.*

This equation is true only if the correct weighting factors λ_j^i are applied to each of the measured points $\mathbf{x}_j^i$. For the present, let us assume that these depths are known. As with the affine factorization algorithm, we would like the matrix on the left – denote it by $\mathtt{W}$ – to have rank 4, since it is the product of two matrices with 4 columns and rows respectively. The actual measurement matrix can be corrected to have rank 4 by using the SVD. Thus, if $\mathtt{W} = \mathtt{UDV}^\mathsf{T}$, all but the first four diagonal entries of $\mathtt{D}$ are set to zero resulting in $\hat{\mathtt{D}}$. The corrected measurement matrix is $\hat{\mathtt{W}} = \mathtt{U}\hat{\mathtt{D}}\mathtt{V}^\mathsf{T}$. The camera matrices are retrieved from $[\mathtt{P}_1^\mathsf{T}, \mathtt{P}_2^\mathsf{T}, \ldots, \mathtt{P}_m^\mathsf{T}]^\mathsf{T} = \mathtt{U}\hat{\mathtt{D}}$ and the points from $[\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n] = \mathtt{V}^\mathsf{T}$. Note that this factorization is not unique, and in fact we may interpose an arbitrary 4×4 projective transformation $\mathtt{H}$ and its inverse between the two matrices on the right of (18.9), reflecting the fact that reconstruction has a projective ambiguity.

The steps of the projective factorization method are summarized in algorithm 18.2.

18.4.1 Choosing the depths

The weighting factors λ_j^i are called the *projective depths* of the points. The justification of this terminology is the relation of these λ_j^i to the actual depths if camera matrices are known in a Euclidean frame. Refer to section 6.2.3(p162) and in particular figure 6.6(p162). The main difficulty with this projective factorization algorithm is that we need to know these projective depths up front, but we do not have this knowledge. There are various techniques to estimate the depths.

(i) Start with an initial projective reconstruction obtained by other means, such as those discussed in section 18.6 below. Then compute λ_j^i by reprojecting the 3D points.

(ii) Start with initial depths all equal to 1, compute the reconstruction and reproject to obtain a new estimate of the depths. This step may be repeated to obtain improved estimates. However, there is no guarantee that the procedure will converge to a global minimum.

The original paper [Sturm-96] gives a method of computing the depths by stringing together pairwise estimates of the depth obtained from the fundamental matrix, or the trifocal tensor. This method is quite similar to obtaining an initial projective reconstruction by stringing together triples of images (see section 18.6), whilst ensuring that the scale factors are consistent for a common projective reconstruction.

18.4.2 What is being minimized?

In the case of noise, or incorrect values for λ_j^i, the equations (18.9) are not satisfied exactly. We determine a corrected measurement matrix $\hat{\mathtt{W}}$ that is closest to $\mathtt{W}$ in Frobenius norm, subject to having rank 4. Denoting the entries of this matrix as $\hat{\lambda}_j^i \hat{\mathbf{x}}_j^i$, then the computed solution minimizes the expression

$$\|\mathtt{W} - \hat{\mathtt{W}}\|^2 = \sum_{ij} \|\lambda_j^i \mathbf{x}_j^i - \hat{\lambda}_j^i \hat{\mathbf{x}}_j^i\|^2 = \sum_{ij} (\lambda_j^i x_j^i - \hat{\lambda}_j^i \hat{x}_j^i)^2 + (\lambda_j^i y_j^i - \hat{\lambda}_j^i \hat{y}_j^i)^2 + (\lambda_j^i - \hat{\lambda}_j^i)^2 \tag{18.10}$$

Because of the last term, at a minimum $\hat{\lambda}_j^i$ must be close to λ_j^i. Assuming they are equal, (18.10) reduces to $\sum_{ij} (\lambda_j^i)^2 \|\mathbf{x}_j^i - \hat{\mathbf{x}}_j^i\|^2$. Noting that $\|\mathbf{x}_j^i - \hat{\mathbf{x}}_j^i\|$ is the geometric distance between the measured and estimated points, what is being minimized is a weighted sum-of-squares geometric distance, where each point is being weighted by λ_j^i. If all the geometric depths λ_j^i are close to equal, then the factorization method minimizes an approximation to geometric distance scaled by the common value of λ_j^i.

18.4.3 Normalizing the depths

Projective depths as defined here are not unique. Indeed suppose that $\lambda_j^i \mathbf{x}_j^i = \mathtt{P}^i \mathbf{X}_j$. If we replace $\mathtt{P}^i$ by $\alpha^i \mathtt{P}^i$ and $\mathbf{X}_j$ by $\beta_j \mathbf{X}_j$, then we find that

$$(\alpha^i \beta_j \lambda_j^i) \mathbf{x}_j^i = (\alpha^i \mathtt{P}^i)(\beta_j \mathbf{X}_j) \ .$$

In other words, the projective depths λ_j^i may be replaced by multiplying the i-th row of (18.9) by a factor α^i and the j-th column by a factor β_j. In the light of the previous paragraph, the closer all λ_j^i are to unity, the more exactly the error expression represents geometric distance. Therefore, it is advantageous to renormalize the values of the λ_j^i so that they are as close to unity as possible, by multiplying rows and columns of the measurement matrix by constant values α^i and β_j. A simple heuristic manner of doing this is to multiply each row by a factor α^i so that it has unit norm, followed by a similar pass normalizing the columns. The row and column passes may be iterated.

18.4.4 Normalizing the image coordinates

As with most numerical algorithms involving homogeneous representations of image coordinates described in this book, it is important to normalize the image coordinates. A reasonable scheme is the isotropic normalization method described in section 4.4.4-(p107). One can see the necessity of normalization quite clearly in this case. Consider two image points $\mathbf{x} = (200, 300, 1)^\mathsf{T}$ and $\hat{\mathbf{x}} = (250, 375, 1)^\mathsf{T}$. Obviously these points are very far apart in a geometric sense. However, the error expression (18.10) measures not geometric error, but the distance between homogeneous vectors $\|\lambda\mathbf{x} - \hat{\lambda}\hat{\mathbf{x}}\|$. Choosing $\lambda = 1.25$ and $\hat{\lambda} = 1.0$, the error is $\|(250, 375, 1.25)^\mathsf{T} - (250, 375, 1)^\mathsf{T}\|$ which is proportionally quite small. On the other hand, the distance between points $\mathbf{x} = (200, 300, 1)^\mathsf{T}$ and $\hat{\mathbf{x}} = (199, 301, 1)^\mathsf{T}$, which are much closer geometrically can not be made so small by choice of λ and $\hat{\lambda}$ (except for small values). The reader may observe that if the points are scaled down by a factor of 200, then this anomalous situation no longer occurs. In short, with normalized coordinates, the error is a closer approximation to geometric error.

18.4.5 When is the assumption $\lambda^i_j = 1$ reasonable?

According to result 6.1(p162), if camera matrices are normalized such that $p_{31}^2 + p_{32}^2 + p_{33}^2 = 1$, and 3D points are normalized to have last coordinate $\mathrm{T} = 1$, then λ_{ij} defined by $\lambda^i_j(x^i_j, y^i_j, 1) = \mathtt{P}^i\mathbf{X}_j$ are the true depths of the points from the camera in a Euclidean frame. If all points are equidistant from the cameras throughout a sequence then we may reasonably assume that each $\lambda^i_j = 1$, for (18.9) will have at least the solution where $\mathtt{P}^i$ and $\mathbf{X}_j$ are the true cameras and points, normalized in the manner just stated. More generally, suppose that points are located at different depths, but each point $\mathbf{X}_j$ remains at approximately the same depth d_j from the cameras through the whole sequence. In this case a solution will exist with all $\lambda^i_j = 1$ in which the computed $\mathbf{X}_j = d_j^{-1}(\mathrm{X}_j, \mathrm{Y}_j, \mathrm{Z}_j, 1)^\mathsf{T}$. Similarly, by allowing multiplication of the camera matrices by a factor, we find

- *If the ratios of true depths of the different 3D points $\mathbf{X}_j$ remain approximately constant during a sequence, then the assumption $\lambda^i_j = 1$ is a good first approximation to projective depth.*

This is for instance the case of an aerial image camera pointing straight down from constant altitude.

18.5 Projective reconstruction using planes

It was seen in section 17.5.2 that if four points visible in each view are known to be coplanar then the computation of the multifocal tensors relating the image points becomes significantly more simple. A major advantage is that a tensor satisfying all its constraints may be computed using a linear algorithm. We now continue with that particular line of investigation, and show that the use of linear techniques extends to estimation of motion and structure for any number of views.

The condition that four of the image correspondences are derived from coplanar points is equivalent to knowing the homographies between the images induced by a plane in space, since a homography may be computed from the four points. It is only the homographies that are important in the following approach. These homographies may be computed from four or more point correspondences, or line correspondences, or estimated directly from the images by direct correlation methods.

What do the plane-plane homographies tell us? The key to projective reconstruction using planes is the observation that knowledge of homographies between the images means we know the first 3×3 part of the camera matrices:

$$\mathtt{P} = \left[\begin{array}{c|c} & t_1 \\ \mathtt{M} & t_2 \\ & t_3 \end{array} \right]$$

Hence, it remains only to compute their last columns, namely the vectors $\mathbf{t}$.

Since we are interested at this point only in obtaining a *projective* reconstruction of the scene, we may suppose that the plane inducing the homographies is the plane at infinity, with points $\mathbf{X}_j = (\mathrm{X}_j, \mathrm{Y}_j, \mathrm{Z}_j, 0)^\mathsf{T}$. Camera matrices may be written in the form $\mathtt{P}^i = [\mathtt{M}^i | \mathbf{t}^i]$, where $\mathtt{M}$ is a 3×3 matrix and $\mathbf{t}^i$ is a column vector. A reasonable assumption is that the camera centres do not lie on the plane inducing the homographies (for otherwise the homographies will be degenerate). This means that the matrix $\mathtt{M}$ is non-singular. For simplicity, the first camera may be assumed to have the form $\mathtt{P}^1 = [\mathtt{I} \mid \mathbf{0}]$, where $\mathtt{I}$ is the identity matrix.

Now, if $\mathbf{x}_j^i$ is the point in image i corresponding to the 3D point $\mathbf{X}_j = (\mathrm{X}_j, \mathrm{Y}_j, \mathrm{Z}_j, 0)^\mathsf{T}$ lying on the homography-inducing plane, then

$$\mathbf{x}_j^1 = \mathtt{P}^1 (\mathrm{X}_j, \mathrm{Y}_j, \mathrm{Z}_j, 0)^\mathsf{T} = (\mathrm{X}_j, \mathrm{Y}_j, \mathrm{Z}_j)^\mathsf{T}$$

whereas

$$\mathbf{x}_j^i = \mathtt{M}^i (\mathrm{X}_j, \mathrm{Y}_j, \mathrm{Z}_j)^\mathsf{T} = \mathtt{M}^i \mathbf{x}_j^1.$$

Thus $\mathtt{M}^i$ represents the homography from the first image to the i-th image induced by the plane. Conversely, if $\mathtt{M}^i$ is the known plane-induced homography that maps a point in the first image to its matching point in the i-th image, then the set of camera matrices can be assumed to have the form $\mathtt{P}^i = [\mathtt{M}^i | \mathbf{t}^i]$, where the $\mathtt{M}^i$ are known and their scale is fixed, but the final columns $\mathbf{t}^i$ are not.

Known camera orientation. We have just shown that knowledge of homographies implies the knowledge of the left-hand 3×3 submatrix of each camera matrix. The same will hold if we know the orientation (and calibration) of all the cameras. For instance, a reasonable approach to reconstruction, knowing the calibration of each camera, is to estimate the orientation of each camera separately from the translation (for example from two or more scene vanishing points). Once the orientation ($\mathtt{R}^i$) and calibration ($\mathtt{K}^i$) of each camera is known, the left-hand block of each camera matrix is $\mathtt{K}^i \mathtt{R}^i$.

18.5.1 Direct solution for structure and translation

We describe two separate methods for computation of the projective structure given plane-induced homographies between images. The first method solves for the 3D points and the camera motion simultaneously by solving a single linear system. Suppose point $\mathbf{X} = (\mathrm{X}, \mathrm{Y}, \mathrm{Z}, 1)^\mathsf{T}$ is not on the plane at infinity, that is, the plane inducing the homographies.

The equation for point projection is

$$\lambda \mathbf{x} = \mathtt{P}\mathbf{X} = [\mathtt{M}|\mathbf{t}]\mathbf{X} = [\mathtt{M}|\mathbf{t}]\begin{pmatrix} \widetilde{\mathbf{X}} \\ 1 \end{pmatrix}$$

where the (unknown) scale factor λ has been explicitly written. More precisely, we may write

$$\lambda \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{bmatrix} \mathbf{m}_1^\mathsf{T} & t_1 \\ \mathbf{m}_2^\mathsf{T} & t_2 \\ \mathbf{m}_3^\mathsf{T} & t_3 \end{bmatrix} \begin{pmatrix} \widetilde{\mathbf{X}} \\ 1 \end{pmatrix} = \begin{pmatrix} \mathbf{m}_1^\mathsf{T}\widetilde{\mathbf{X}} + t_1 \\ \mathbf{m}_2^\mathsf{T}\widetilde{\mathbf{X}} + t_2 \\ \mathbf{m}_3^\mathsf{T}\widetilde{\mathbf{X}} + t_3 \end{pmatrix}$$

where $\mathbf{m}_i^\mathsf{T}$ is the i-th row of the matrix $\mathtt{M}$.

The unknown scale factor λ may be eliminated by taking the vector product of the two sides of this equation, resulting in

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \times \begin{pmatrix} \mathbf{m}_1^\mathsf{T}\widetilde{\mathbf{X}} + t_1 \\ \mathbf{m}_2^\mathsf{T}\widetilde{\mathbf{X}} + t_2 \\ \mathbf{m}_3^\mathsf{T}\widetilde{\mathbf{X}} + t_3 \end{pmatrix} = \mathbf{0}.$$

This provides two independent equations

$$\begin{aligned} x(\mathbf{m}_3^\mathsf{T}\widetilde{\mathbf{X}} + t_3) - (\mathbf{m}_1^\mathsf{T}\widetilde{\mathbf{X}} + t_1) &= 0 \\ y(\mathbf{m}_3^\mathsf{T}\widetilde{\mathbf{X}} + t_3) - (\mathbf{m}_2^\mathsf{T}\widetilde{\mathbf{X}} + t_2) &= 0 \end{aligned}$$

which are linear in the unknowns $\widetilde{\mathbf{X}} = (\mathrm{X}, \mathrm{Y}, \mathrm{Z})^\mathsf{T}$ and $\mathbf{t} = (t_1, t_2, t_3)^\mathsf{T}$. The equations may be written as

$$\begin{bmatrix} x\mathbf{m}_3^\mathsf{T} - \mathbf{m}_1^\mathsf{T} & -1 & 0 & x \\ y\mathbf{m}_3^\mathsf{T} - \mathbf{m}_2^\mathsf{T} & 0 & -1 & y \end{bmatrix} \begin{pmatrix} \widetilde{\mathbf{X}} \\ t_1 \\ t_2 \\ t_3 \end{pmatrix} = \mathbf{0}.$$

Thus, each measured point $\mathbf{x}_j^i = \mathtt{P}^i\mathbf{X}_j$ generates a pair of equations, and with m views involving n points a $2nm$ set of equations in $3n + 3m$ unknowns is generated in this way. These equations may be solved by linear or linear least-squares techniques to obtain the structure and motion.

A few remarks about this method are offered.

(i) In contrast to factorization methods (section 18.2) we do not need all points to be visible in all views. Only equations corresponding to the measured points are used.

(ii) Since it is assumed that points have final coordinate equal to one, it is necessary to exclude points that lie on the plane at infinity (the plane inducing the homography) which have final coordinate equal to zero. A test to detect points lying on or close to the plane is necessary.

(iii) Both points and cameras are computed at once. For a large number of points and cameras this may be a very large estimation problem. However, if the point tracks have a banded form, then sparse solution techniques may be used to solve the equation set efficiently, as in section A6.7(p613).

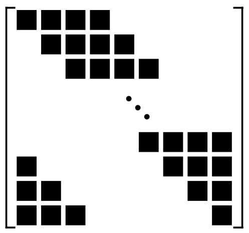
This method and its implementation are discussed in depth in [Rother-01, Rother-03]. The details given here are different from those given in [Rother-01], where the structure and motion computation is carried out in a specific projective frame related to the matched points on the plane, involving a coordinate change in the images.

18.5.2 Direct motion estimation

The second method for planar reconstruction knowing homographies solves for the camera matrices first and subsequently computes the point positions.

We start from the set of camera matrices which again can be assumed to have the form $\mathtt{P}^i = [\mathtt{H}^i | \mathbf{t}^i]$, where the $\mathtt{H}^i$ are known and their scale is fixed, but the final columns $\mathbf{t}^i$ are not. We may assume that $\mathtt{P}^1 = [\mathtt{I} \mid \mathbf{0}]$, so that $\mathbf{t}^1 = \mathbf{0}$. The set of all remaining $\mathbf{t}^i$ have $3m - 4$ degrees of freedom, since the $\mathbf{t}^i$ are defined only up to a common scale. Now assume that several point or line correspondences across two or more views are known (three views are required for lines). These correspondences must derive from 3D points or lines that do not lie in the reference plane (used to compute the $\mathtt{H}^i$). Each point correspondence across two views leads to a linear equation in the entries of the fundamental matrix. Similarly, correspondences of points or lines across three or four views lead to linear equations in the entries of the trifocal or quadrifocal tensor.

The key point (as explained in section 17.5.2) is that we may express the entries of the fundamental matrix (or trifocal or quadrifocal tensor) linearly in the entries of the vectors $\mathbf{t}^i$. Therefore each linear relation induced by a point or line correspondence may be related back to a linear relationship in terms of the entries of the $\mathbf{t}^i$. Thus, for example, a correspondence across views i, j and k gives rise to a set of linear equations in the entries of the three vectors $\mathbf{t}^i$, $\mathbf{t}^j$ and $\mathbf{t}^k$. A set of correspondences across many views can be broken down into correspondences across sets of consecutive views. Thus, for example, a single point correspondence across $m > 4$ views will give a set of equations of the form



where each row represents a set of equations derived from a quadrifocal tensor relationship. Each black square represents a block with 3 columns corresponding to one of the vectors $\mathbf{t}^i$. In the diagram above, we choose to wrap the equations around from the last to the first view to add greater rigidity. Otherwise, the values of the $\mathbf{t}^i$ can drift from the first to the last view. Other schemes for selecting groups of views are possible, and it is not necessary to restrict to consecutive views.

Linear relations may be generated between any subset of sufficiently many images (2, 3 or 4 depending on which tensor is used to generate the equations). One must trade off the added stability of the solution against the added computational cost of adding more equations. A mixture of bifocal, trifocal and quadrifocal constraints may be used in generating the set of all equations, and it is not necessary that all points be visible in all views.

Numbers of equations generated. Let the total number of views be m. Consider a subset of s views ($s = 2, 3$ or 4) and let n point correspondences be given between these views. We briefly consider the problem of reconstruction from this subset of s views in isolation. From these point correspondences we can generate a set of equations $\mathtt{A}\mathbf{t}'$ between the entries $\mathbf{t}'$ of the s views, and thence estimate the values of the $3s$ entries of $\mathbf{t}'$. In doing this, we can assume that the first view has $\mathbf{t} = \mathbf{0}$, and the vectors $\mathbf{t}$ from the remaining $s - 1$ views are only determined up to a common scale. Thus, the $\mathtt{A}$ occurring in the equation set $\mathtt{A}\mathbf{t}'$ has a right null-space of dimension at least 4, corresponding to the 4 degrees of freedom of the solution. In general, then:

Result 18.1. *Ignoring the effects of noisy data, the total rank of the set of equations generated from $n \geq 2$ point correspondences in s views is $3s - 4$. This is independent of the number of point (or line) correspondences used to generate them.*

To be exact, the argument above showed that the rank was *at most* $3s - 4$. For 2-view, 3-view and 4-view correspondences this is equal to 2, 5 or 8 respectively. However, as long as there are two correspondences the maximum rank is achieved. This is because two points are sufficient for reconstruction from $s = 2, 3$ or 4 views as shown by the counting arguments of section 17.5.2.

Now consider the total set of m views. The total number of retrievable parameters of all the $\mathbf{t}^i$ is $3m - 4$. Therefore, for a solution to be possible, the number of equations must exceed $3m - 4$, which gives the following result.

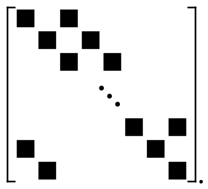
Result 18.2. *If S subsets of s_k views are chosen from among m views, then in order to solve for all the vectors $\mathbf{t}^i$ representing final columns of the camera matrices, it is necessary that*

$$\sum_{k=1}^{S} (3s_k - 4) \geq 3m - 4 \ .$$

One can verify that if 2-view correspondences are to be used, involving equations derived from the fundamental matrix constraints, then it is not sufficient to use just pairs of consecutive views in a configuration such as

$$\begin{bmatrix} \blacksquare & \blacksquare & & & & \\ & \blacksquare & \blacksquare & & & \\ & & \blacksquare & \blacksquare & & \\ & & & \ddots & & \\ & & & & \blacksquare & \blacksquare \\ \blacksquare & & & & & \blacksquare \end{bmatrix}$$

for in this case, the total number of equations generated is $m(3s - 4) = m(3 \cdot 2 - 4) = 2m$, whereas the total number of equations required is $3m - 4$. Thus for $m > 4$ there are not enough equations. This is related to the fact that the fundamental matrices between consecutive views are not sufficient to define the structure of the sequence of views. It is necessary to add additional constraints from non-consecutive views, such as



Note though that the discussion of section 15.4(*p*383) suggests that it is preferable to use trifocal or quadrifocal constraints over triplets or quadruplets of views. Implementation details for this method are given in [Kaucic-01].

18.6 Reconstruction from sequences

In this final section we bring together several ideas from earlier in the book. The objective here is to compute a reconstruction from a sequence of frames provided by a video. There are three stages to this problem: (i) compute corresponding features throughout the sequence; (ii) compute an initial reconstruction which may be used as a starting point for (iii) bundle adjustment (as described in section 18.1).

Here the features we will consider are interest points, though others such as lines could equally well be used. The correspondence problem is exacerbated because an interest point feature will generally not appear in all of the images, and often will be missing from consecutive images. Bundle adjustment, however, is not hindered by missing correspondences.

There are several advantages of a video sequence over an arbitrary set of images: (i) there is an ordering on the images; (ii) the distance between camera centres (the baseline) for successive frames is small. The small baseline is important because it enables possible feature matches between successive images to be obtained and assessed more easily. Matches are more easily obtained because the image points do not move "far" between views so a proximity search region can be employed; matches are more easily assessed (as to whether they arise from the same point in 3-space) because nearby images are similar in appearance. The disadvantage of a small baseline is that the 3D structure is estimated poorly. However, this disadvantage is mitigated by tracking over many views in the sequence so that the effective baseline is large.

An overview of the method is given in algorithm 18.3. There are several strategies that may be used to obtain the initial reconstruction, though this area is still to some extent a black art. Three possibilities are:

1. Extending the baseline. Suppose a reasonable number of scene points are visible throughout the sequence. Correspondences may be carried through from the first to the last frame using the pairwise matches (from $\mathtt{F}$), or the triplet matched points (from $\mathcal{T}$). Indeed if the baseline between consecutive frames is small (compared to the structure depth), then pairwise matches may be obtained using homography computation (algorithm 4.6(*p*123)) – this provides a stronger matching constraint (point to point) than $\mathtt{F}$ (point to line).

A trifocal tensor can then be estimated from corresponding points in the first, middle (say), and end frames of the sequence. This tensor determines a projective reconstruc-

Objective

Given a sequence of frames in a video, compute correspondences and a reconstruction of the scene structure and the camera for each frame.

Algorithm

(i) **Interest points:** Compute interest points in each image.
(ii) **2 view correspondences:** Compute interest point correspondences and F between consecutive frames using algorithm 11.4(*p*291) (frames may be omitted if the baseline motion is too small).
(iii) **3 view correspondences:** Compute interest point correspondences and $\mathcal{T}$ between all consecutive image triplets using algorithm 16.4(*p*401).
(iv) **Initial reconstruction:** See text.
(v) **Bundle adjust** the cameras and 3D structure for the complete sequence.
(vi) **Auto-calibration:** see chapter 19 (optional).

Algorithm 18.3. *Overview of reconstruction from a sequence of images.*

tion for those points and frames. The cameras for the intermediate frames may then be estimated by resectioning, and the scene points not visible throughout the sequence estimated by triangulation.

2. Hierarchical merging of sub-sequences. The idea here is to partition the sequence into manageable sub-sequences (there can be several hierarchical layers of partitioning). A projective reconstruction is then computed for each sub-sequence and these reconstructions are “zipped” (merged) together.

Consider the problem of merging two triplets which overlap by two views. It is a simple matter to extend the correspondences over the views: a correspondence which exists across the triplet 1-2-3 and also across the triplet 2-3-4 may be extended to the frames 1-2-3-4, since the pair 2-3 overlaps for the triplets. The camera matrices and 3D structure are then computed for the frames 1-2-3-4, for example by first resectioning and then bundle adjustment. This process is extended by merging neighbouring groups of frames until camera matrices and correspondences are established throughout the sequence. In this manner error can be distributed evenly over the sequence.

3. Incremental bundle adjustment. A fresh bundle adjustment is carried out as the correspondences from each new frame are added. The disadvantage of this method is the computational expense and also the possibility that error systematically accumulates.

Of course these three methods may be combined together. For example, the sequence can be partitioned into a sub-sequence where common points are visible, and a reconstruction built for the sub-sequence using the extended baseline method. These sub-sequences may then be combined hierarchically.

In this manner structure and cameras may be computed automatically for sequences consisting of hundreds of frames. These reconstructions may form the basis for such tasks as navigation (determining the camera/ego-position) and virtual model genera-

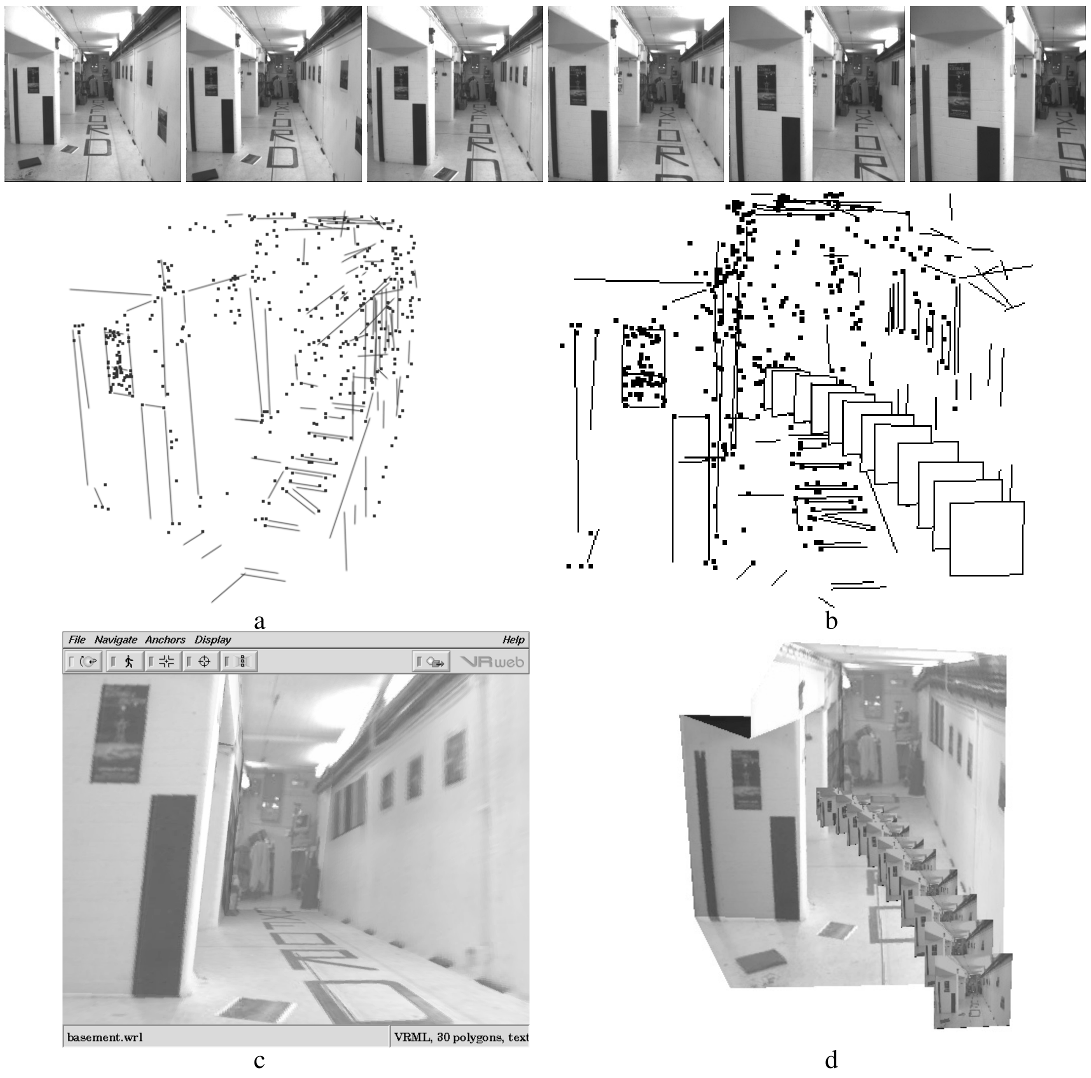
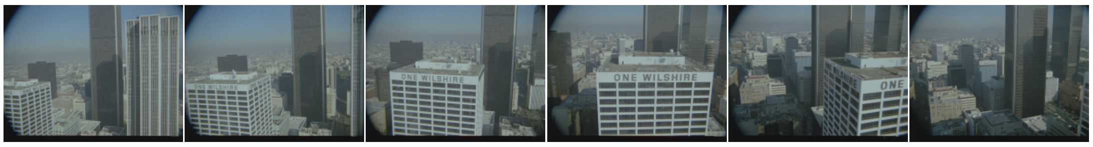


Fig. 18.3. **Corridor sequence**. *(a) A three dimensional reconstruction of points and lines in the scene, and (b) cameras (represented by their image planes) computed automatically from the images. A texture mapped triangulated graphical model is then automatically constructed as described in [Baillard-99]. (c) A rendering of the scene from a novel viewpoint, different from any in the sequence. (d) VRML model of the scene with the cameras represented by their image planes (texture mapped with the original images from the sequence).*

tion. Often it is necessary first to compute a metric reconstruction from the projective one, using the methods described in chapter 10 and chapter 19. Metric reconstruction and virtual model generation is illustrated in the following examples.

Example 18.3. Corridor sequence

A camera is mounted on a mobile vehicle for this sequence. The vehicle moves along the floor turning to the left. The forward translation in this sequence makes structure recovery difficult, due to the small baseline for triangulation. In this situation, the benefit of using all frames in the sequence is significant. Figure 18.3 shows the recovered structure. △



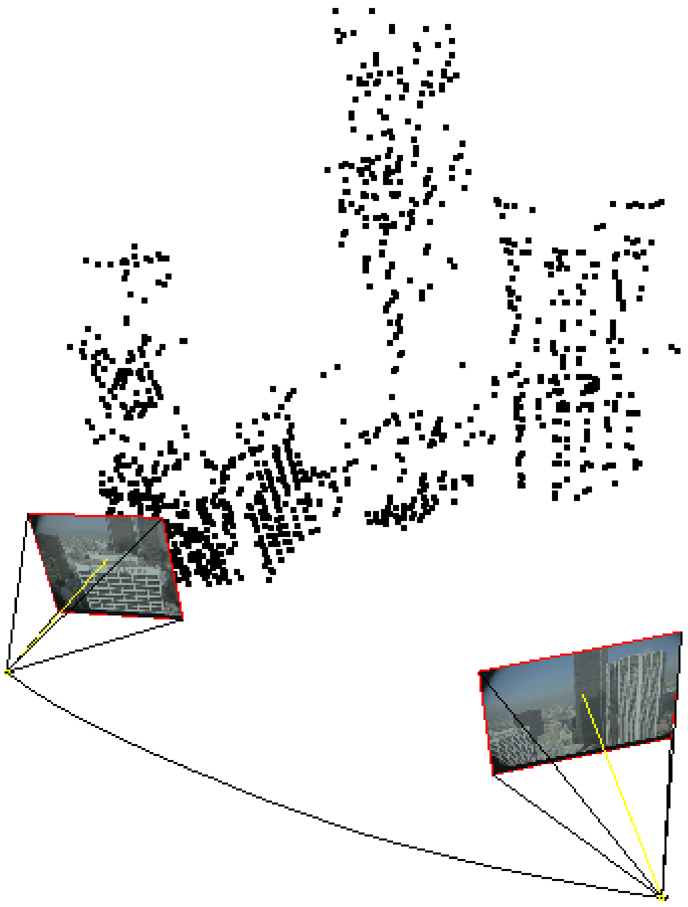


Fig. 18.4. **Wilshire:** *3D points and cameras for 350 frames of a helicopter shot. Cameras are shown for just the start and end frames for clarity, with the camera path plotted between.*

Example 18.4. "Wilshire" sequence
This is a helicopter shot of Wilshire Boulevard, Los Angeles. In this case reconstruction is hampered by the repeated structure in the scene – many of the feature points (for example those on the skyscraper windows) have very similar intensity neighbourhoods, so correlation-based tracking produces many false candidates. However, the robust geometry-guided matching successfully rejects the incorrect correspondences. Figure 18.4 shows the structure. △

18.7 Closure

It is probably fair to say that no fully satisfactory technique for reconstruction from a sequence of projective images is known, and many ad-hoc techniques have been used, with reasonable success. Four views is the limit for the closed-form solutions based on multiview tensors. For larger numbers of views there is no such neat mathematical formulation of the problem. One exception to this is the m-view technique based on duality (see chapter 20), but this techniques is limited to six to eight points, depending on which dual tensor (fundamental matrix, trifocal tensor or quadrifocal tensor) is used. Most sequences will contain many more matched points than this.

18.7.1 Literature

The Tomasi–Kanade algorithm was first proposed for orthographic projection, [Tomasi-92], but was later extended to paraperspective [Poelman-94]. It has been extended to lines and conics, e.g. [Kahl-98a], but the MLE property no longer applies, and it is unclear what is being minimized in the affine reconstruction. Others have investigated subspace methods for multiple affine views in the case of planes [Irani-99], and the case of multiple objects moving independently [Boult-91, Gear-98]. Non-rigid factorization was formulated by [Brand-01, Torresani-01], though the elements of the idea are present in [Bascle-98]. Affine reconstruction with uncertainty (covariance-weighted data) has been discussed by Irani and Anandan [Irani-00, Anandan-02] The method of affine reconstruction by alternating triangulation and camera estimation is mentioned in [Huynh-03], under the name “PowerFactorization.”

The extension of factorization to projective cameras is due to Sturm and Triggs [Sturm-96]. Methods of iteration using this approach have been proposed by [Heyden-97a, Triggs-96].

A method of computing multiple cameras based on a plane homography was employed in [Cross-99], initializing the $\mathbf{t}^i$ vectors using planar auto-calibration.

Methods for obtaining an initial projective reconstruction from a sequence are described in [Avidan-98, Beardsley-94, Beardsley-96, Fitzgibbon-98a, Laveau-96a, Nister-00, Sturm-97b]. [Torr-99], and more recently [Pollefeys-02], discuss the important problem of scene and motion degeneracies that may be encounted in sequence reconstruction.

18.7.2 Notes and exercises

(i) The affine factorization algorithm can be applied to obtain a reconstruction in situations where a set of cameras $\{\mathtt{P}^i\}$ have a common third row, even though the cameras are not affine. The third row is the principal plane of the camera (see section 6.2(p158)) and the condition of a common third row is equivalent to coplanar principal planes. For example if a camera translates in a direction perpendicular to its principal axis, then all the camera centres will lie on a plane, and the principal planes are coplanar. The affine factorization algorithm can be applied in this case because the set of cameras can be transformed as $\mathtt{P}^i\mathtt{H}_{4\times 4}$ to the affine form by a 4×4 homography $\mathtt{H}$ satisfying $\mathbf{P}^3\mathtt{H}_{4\times 4} = (0,0,0,1)$, where $\mathbf{P}^3$ is the last row of $\mathtt{P}^i$.

More generally, if the camera centres are restricted to a plane then the images may be synthetically rotated such that the cameras effectively have coplanar principal planes. For example in the case of planar motion (section 19.8) or single axis rotation (section 19.9(*p*490)) if all the images are rotated such that the principal axis is parallel to the rotation axis (by applying a homography to each image which maps the horizon to infinity in the case of a vertical rotation axis), then the principal planes of all the cameras are parallel. However, if the cameras are not actually affine, then the algorithm will not give the ML estimate of the reconstruction.