Factorization methods for structure from motion

By Takeo Kanade and Daniel D. Morris

Robotics Institute, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh PA 15213-3891, USA

In this article we present an overview of factorization methods for recovering structure and motion from image sequences. We distinguish these methods from general non-linear algorithms primarily by their bilinear formulation in motion and shape parameters. The bilinear formulation makes possible powerful and efficient solution techniques including Singular Value Decomposition. We show how factorization methods apply under various affine camera models and under the perspective camera model, and then we review factorization methods for various features including points, lines, directional point features and line segments. An extension to these methods enables them to segment and recover motion and shape for multiple independently moving objects. Finally we illustrate the generality of the factorization methods with two applications outside structure from motion.

1. Introduction

As we watch the video output from a camera moving in a three dimensional (3D) scene, our minds naturally obtain a "feeling for" or estimate of the motion of the camera as well as an idea of the geometry of the scene. Humans are well adapted to recovering geometry and motion from image sequences. It has been a computer vision goal to perform a similar task: from an image sequence taken by a camera undergoing unknown motion, extract the 3D shape of the scene as well as the camera motion. This is called the *structure from motion* problem.

The main challenge for structure from motion, as in many vision tasks, lies not creating a model of the physics of the task, but rather lies in estimating the parameters of the model. The image formation processes including the optics of the camera are well understood and can be accurately modelled such that if the scene geometry and camera motion are known, the resulting images can be reliably calculated. However, since it is only the images that are known, solving the structure from motion task corresponds to inverting the equations modelling image formation to obtain model parameters. Mathematically this can be well defined modulo ambiguities, but the non-linearity and high sensitivity to parameter variations causes many direct solutions to be numerically ill-conditioned, or else results in computationally complex non-linear approaches with many convergence hurdles.

In this article we discuss a class of algorithms known as *factorization* methods for structure from motion. These algorithms depend on the mathematical

possibility of decomposing a set of image measurements into the product of two separate factors:

$$image\ sequence \Leftrightarrow motion \times shape.$$

Intuitively the projected images are considered to result from two factors: the relative *motion* between the camera and the object and the object *shape*. These are composed in a bilinear form such that if either motion or shape is constant, then the image sequence will be a linear function of the other. The motion parameters refer to all of those parameters describing the interaction between the camera and the object; namely the relative orientation and translation of the object and intrinsic camera calibration parameters. These parameters may vary from image to image in the sequence, but are the same for all features in a single image. The shape parameters describe the 3D geometric characteristics of the object and are assumed to remain constant over the sequence. Typically the 3D coordinates of features on the surface of the object are used to specify shape.

The factorization method takes advantage of the bilinear formulation to decompose the image measurements into the relevant motion and shape components. This can be achieved by a number of techniques including Singular Value Decomposition (SVD) and can be performed much more efficiently and with better convergence properties than general non-linear optimizations. The factorization method has been demonstrated to be a powerful approach to extracting structure and motion from image sequences.

The factorization method was first introduced by Tomasi & Kanade (1990,1992) for the orthographic case. It has since inspired numerous extensions and generalizations. In this report we review the basic the factorization method and its development along several research directions. Important developments include the extension to alternate camera models, both affine and perspective, the extension of feature models from points to lines and to directional point feature models, a multi-body case and the creation of alternative solution techniques. There have also been applications of factorization methods in other domains.

2. The Factorization Method: Fundamentals

The use of features is a key factor in making the factorization and other structure from motion methods tractable and general. It is assumed that there exists a set of features on the object that are tracked throughout the image sequence providing a complete set of feature coordinates in all images. This assumption enables the method to focus on geometric considerations and ignore all of the image processing tasks. Object shape is interpreted to mean the 3D location of the features with respect to a reference frame affixed to the object. Object motion is the rotation and translation of this reference frame with respect to the camera, and the image sequence means simply the coordinates of the projected features in the images. The use of features enables factorization methods to be used whenever a feature set is available, irrespective of object motion, shape or illumination.

The core element of factorization that distinguishes it from similar methods is its strong dependence on a bilinear formulation of structure and motion. With appropriate choice of coordinates it is possible to encode both affine and perspective

camera projections in the general bilinear form:

$$\mathbf{w}_{fp} = P_f M_f \mathbf{s}_p. \tag{2.1}$$

The feature coordinate vector $\mathbf{w}_{fp}$ for image f and feature p is formed as the linear sum of the product of motion parameters in matrix M_f and shape parameters in vector $\mathbf{s}_p$ weighted with the constant weighting or projection matrix P_f. The projection and motion parameters will vary across the frames of the image sequence but are the same for each object feature whereas the shape parameters will vary between features but be identical for all frames. In the first stage of factorization the projection parameters are included in the motion parameters. The bilinear form then permits a convenient and efficient decomposition of image measurements into motion and shape.

The other important and distinguishing feature of factorization methods are their uniform treatment of features and images from a large sequence. Many algorithms are designed to recover structure from the minimum number of images. Extension to the multi-image case typically requires partitioning the sequence into subsets, solving the subsets and combining the results, or else selecting a "special image" (or subset of images) to which all other images are compared in obtaining motion and shape. The partitioning approach suffers from not applying consistent constraints across the sequence, and the "special image" approach results in high sensitivity to noise in measurements of that image. Factorization, on the other hand, provides true batch processing of the images with global application of constraints over all images. Neither image pairs nor special images are relied on by the algorithm.

Early work on obtaining structure from feature correspondences starting from Longuet-Higgins (1981) and Tsai & Huang (1984) focussed on obtaining camera orientation and object shape from minimal numbers of views. In 1990 Tomasi & Kanade demonstrated a working scheme for structure from motion for many images and features that they called the *factorization algorithm*. The authors recently learned that Kontsevich *et al.* (1987) presented a paper proposing a mathematical formulation essentially the same as the factorization algorithm. In the remainder of § 2 we describe the basic factorization algorithm.

(*a*) *Constructing the Equations*

We assume that there is a set of P features on an object that are projected into F images with coordinates $\{\mathbf{w}_{fp} = (u_{fp}, v_{fp})^T | f = 1, \ldots, F, p = 1, \ldots, P\}$ as illustrated in figure 1. Initially we formulate the equations describing feature projection assuming an orthographic camera model. These equations are further generalized to other affine models in § 3(*a*), and to the perspective case in § 3(*b*).

The primary constraint on the object is its rigidity. All of the object features can be described by their constant 3D positions in a reference frame affixed to the object. Each feature has coordinates given by a 3×1 vector $\mathbf{s}_p$ for $p = 1, \ldots, P$. Object motion is described by a rotation, with matrix R_f, and translation, $\mathbf{t}_f = (t_{fx}, t_{fy}, t_{fz})^T$, of this reference frame with respect to the camera in each image, f. A feature point p in image f will thus have position $\mathbf{s}_p^f = R_f \mathbf{s}_p + \mathbf{t}_f$ with respect to the camera. Under orthography with the z axis along the optical axis, image feature $\mathbf{w}_{fp}$ is given by:

$$\mathbf{w}_{fp} = M_f \mathbf{s}_p + \mathbf{w}'_f, \tag{2.2}$$

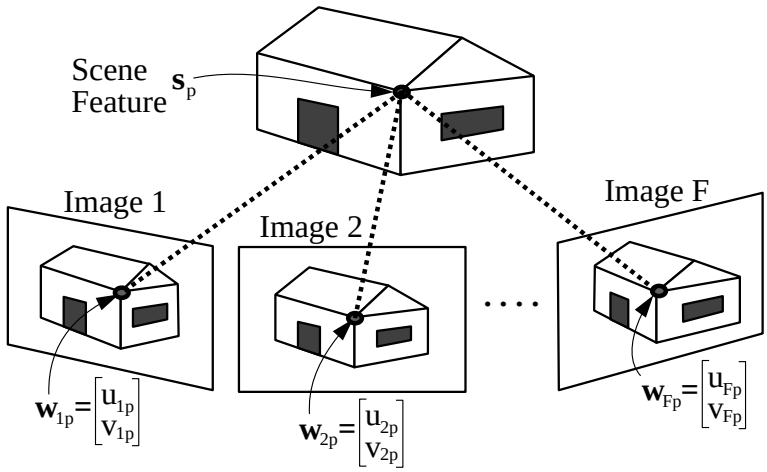

Figure 1. Feature tracked in an image sequence

where M_f consists of the top two rows of the rotation matrix R_f, and $\mathbf{w}'_f = (t_{fx}, t_{fy})^T$ is the image displacement between the origins of the world reference frame and the object reference frame. We then choose and coordinate system fixed to object and express the features in this new frame by displacing them by $\mathbf{w}'_f$ to get $\mathbf{w}^o_{fp} = \mathbf{w}^w_{fp} \Leftrightarrow \mathbf{w}'_f$. In this object centered frame equation (2.2) becomes:

$$\mathbf{w}^o_{fp} = M_f \mathbf{s}_p \tag{2.3}$$

for $f = 1, \ldots, F, p = 1, \ldots, P$. By stacking rows and columns, these equations can be written compactly in the form:

$$W = MS \tag{2.4}$$

where

$$W = \begin{bmatrix} \mathbf{w}^o_{11} & \mathbf{w}^o_{12} & \cdots & \mathbf{w}^o_{1P} \\ \mathbf{w}^o_{21} & \mathbf{w}^o_{22} & \cdots & \mathbf{w}^o_{2P} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{w}^o_{F1} & \mathbf{w}^o_{F2} & \cdots & \mathbf{w}^o_{FP} \end{bmatrix}_{2F \times P},$$

$$M = \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_F \end{bmatrix}_{2F \times 3} \quad \text{and} \quad S = \begin{bmatrix} \mathbf{s}_1 & \mathbf{s}_2 & \cdots & \mathbf{s}_P \end{bmatrix}_{3 \times P}.$$

Equation (2.4) contains the core of the factorization algorithm. It states that the feature locations in object centered coordinates can be expressed as the product of a motion matrix and a shape matrix projected onto the image. The key property of this equation is the following theorem.

Theorem 1. *In the ideal case with no corrupting noise, the measurement matrix W has rank given by:*

$$\text{rank}(W) \leq 3. \tag{2.5}$$

Proof. This result is evident since M has three columns and the columns of W are linear combinations of these. It also follows as an extension of work by Ullman (1979) who he showed that three points and three frames are sufficient to recover shape and motion under orthography. Here this property is applied to the whole image sequence. ■

(*b*) *Solving the equations*

An important reason for the success and popularity of the factorization algorithm is its convergence to a globally optimum affine solution. We describe here the original solution technique proposed by Kontsevich *et al.* (1987) and Tomasi & Kanade (1990). Since the rank constraint only applies to the ideal case where feature positions are measured with no error, when there is noise in the measurement, a norm squared error can be defined:

$$E_{SVD}(M,S) = \|W \Leftrightarrow MS\|^2 \tag{2.6}$$

where M and S are constrained to be of the form given in equation (2.4). This least squares approximation can be achieved by performing SVD on the measurement matrix:

$$W = U\Sigma V^T. \tag{2.7}$$

We let $\tilde{\Sigma}$ be equal to the top left 3×3 block of Σ containing the three largest singular values. These singular values correspond to the principal components of W. Then selecting U_3 and V_3 to be the first three columns of U and V respectively, we obtain the least squares approximation to W given by: $\hat{W} = U_3\tilde{\Sigma}V_3^T$. This decomposition step results in a matrix U_3 which measures inter-image differences that are common for all features, namely the object motion, and a matrix V_3 that measures intra-image structure that is common in all images, namely the object shape.

From here it is simple to factor this into motion and shape matrices. An arbitrary solution is to choose: $\hat{M} = U_3\tilde{\Sigma}^{1/2}$, and $\hat{S} = \tilde{\Sigma}^{1/2}V_3^T$. However, this solution is unique only up to an affine transformation, since the motion and shape can be transformed by any 3×3 invertible matrix A, with their product remaining the same: $W = \hat{M}\hat{S} = \hat{M}AA^{-1}\hat{S} = MS$. The Euclidean solution for the orthographic case which is unique up to a rotation and reflection is obtained by finding transformation A that the rows of each M_f in M are orthonormal. Solutions for other affine models are obtained by imposing the camera projection constraints and the initial orientation information on the motion matrix as described in § 3.

By using SVD, the factorization algorithm remains numerically stable and is guaranteed to converge to the global minimum of E_{SVD}. It thus provides a reliable algorithm for obtaining shape and motion from an image sequence; a major step forward in structure from motion research.

3. Factorization Under Various Camera Models

The first step of the factorization algorithm recovers affine shape and motion. Euclidean shape and motion are recovered by choosing a camera model and finding the matrix A such that the motion matrix $M = \hat{M}A$ satisfies this model. This is because M implicitly incorporates both rotation and projection information. In this section we described various affine camera models used with factorization

and give constraint equations for each. We then turn our attention to the full perspective factorization recently proposed by Triggs (1996), Sturm & Triggs (1996) and Deguchi (1997).

(*a*) *Affine Models*

The general affine camera projection model is defined in homogeneous coordinates by the projection equation:

$$\lambda \bar{\mathbf{w}}_{fp} = P_{Af} \bar{\mathbf{s}}_p. \tag{3.1}$$

Here an object point $\bar{\mathbf{s}} = (\mathbf{s}^T, 1)^T$ is projected onto the image plane to point $\bar{\mathbf{w}} = (\mathbf{w}^T, 1)^T$ by the projection matrix P_A. The projection is in the form:

$$P_{Af} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ 0 & 0 & 0 & p_{34} \end{bmatrix} \equiv \begin{bmatrix} M_{f\,2\times3} & \mathbf{w}'_{f\,2\times1} \\ 0_{1\times3} & \lambda \end{bmatrix}. \tag{3.2}$$

Restricting consideration to points within the affine subspace not at infinity, equation (3.1) can be scaled by $1/\lambda$ and so can be expressed in the form:

$$\mathbf{w}_{fp} = M_f \mathbf{s}_p + \mathbf{w}'_f. \tag{3.3}$$

Note that up to a choice of origin, $\mathbf{w}'_f$, this is precisely the form in which SVD decomposition obtains motion and shape.

The matrix M_f can be decomposed into a camera calibration matrix and a rotation matrix in the form:

$$M_f = K_{Af} R_f = \frac{1}{z_f} \begin{bmatrix} 1 & s \\ 0 & a \end{bmatrix} \begin{bmatrix} \mathbf{i}_{f\,1\times3} \\ \mathbf{j}_{f\,1\times3} \end{bmatrix}. \tag{3.4}$$

Here z_f is the depth in focal lengths of the object, s is the skew parameter, a is the aspect ratio, and vectors $\mathbf{i}_f$ and $\mathbf{j}_f$ are the two top orthonormal rows of a rotation matrix. With this decomposition we obtain the identity:

$$M_f M_f^T = K_{Af} K_{Af}^T. \tag{3.5}$$

In the remainder of this section we reinterpret elements of K_{Af} according to particular sub-affine camera models. We find matrix A that transforms the recovered motion, $M = \hat{M}A$, to satisfy this identity up to a rotation and reflection. The rotation can be determined by choice of coordinate system, but the reflection is an inherent ambiguity of the projected affine model.

(i) Orthography

The orthographic camera model has the parameters given by:

$$K_{Af} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{3.6}$$

It is typically a good approximation for imaging when the object thickness and change in depth between images are small compared to its depth from the camera. It is in these cases that depth recovery is difficult and may be sensitive to noise so an orthographic model is likely to be more reliable than more complicated models. To recover Euclidean shape with this camera we seek a matrix A that will enable $M_f = \hat{M}_f A$ to satisfy equation (3.5) for all the images. From this we

obtain the following equations:

$$\hat{M}_f A A^T \hat{M}_f^T = K_{Af} K_{Af}^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ for } f = 1, \ldots, F. \tag{3.7}$$

A linear set of equations are used to obtain $Q = AA^T$ as described by Morita & Kanade (1994), and then Q is split into AA^T.† Finally Euclidean motion and shape are recovered as: $M = \hat{M}A$ and $S = A^{-1}S$.

(ii) Weak Perspective

Weak perspective, also known as scaled orthography, has the following camera parameters:

$$K_{Af} = \frac{1}{z_f} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{3.8}$$

Weak perspective can be interpreted as an orthographic projection of the object features onto a plane through its centroid and parallel to the image plane, followed by a perspective projection of this plane onto the image plane. This perspective projection of the plane is simply a uniform scaling of the feature coordinates by the inverse of their distance from the camera measured in focal lengths which is here denoted as: $1/z_f$. Weak perspective thus models the scaling effects caused by depth changes between images. Hence it is appropriate for shallow objects making significant changes in distance from the cameras in image to image. The camera constraints for obtaining A become:

$$\hat{M}_f A A^T \hat{M}_f^T = K_{Af} K_{Af}^T = \frac{1}{z_f^2} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ for } f = 1, \ldots, F \tag{3.9}$$

Both the depths z_f and the transformation A can be solved for from these equations as described by Kontsevich *et al.* (1987) and Poelman (1995).

(iii) Paraperspective

The paraperspective model is another affine model and is a step closer to approximating perspective projection. Originally introduced by Ohta *et al.*(1981), it models the scaling of weak perspective as well as the apparent rotation resulting from an object moving to the edge of the image. Modelling this effect is useful when the object being viewed moves across the image. Poelman & Kanade (1994) and Poelman (1995) showed how the paraperspective model could be incorporated into the factorization formulation and derived the constraint equations:

$$\hat{M}_f A A^T \hat{M}_f^T = \frac{1}{z_f^2} \begin{bmatrix} 1 + x_f^2 & x_f y_f \\ x_f y_f & 1 + y_f^2 \end{bmatrix} \text{ for } f = 1, \ldots, F. \tag{3.10}$$

Here x_f and y_f are the image coordinates of the centroid of the object with respect to the center of the image and so can be measured in each image. It just remains for A and the depths to be recovered from these equations.

† If the recovered matrix, Q is not positive definite it cannot be factored in this form. Fortunately for typical image noise Q is generally positive definite.

(*b*) *Perspective Factorization*

When the object thickness is significant compared to its depth from the camera, affine models become poor approximations to the imaging process and perspective models should be used. Sturm & Triggs (1996), Triggs (1996) and Deguchi (1997) proposed a generalization of the factorization algorithm to recover motion and shape up to a projective transformation. Using calibration parameters this can be transformed into the Euclidean shape. Here we summarize their projective factorization formulation followed by a calibration step to obtain a perspective factorization method.

The general projective camera model is a linear transformation of homogeneous points in $\mathcal{P}^3$ onto points in the plane, $\mathcal{P}^2$. Thus it is defined by a 3×4 projection matrix P_P that maps object points, $\bar{\mathbf{s}} = (\mathbf{s}^T, s_4)^T$, onto the points in the image plane $\bar{\mathbf{w}} = (\mathbf{w}^T, w_3)^T$ where these points are described in homogeneous coordinates. This projection occurs up to an unknown scale factor λ_{fp} called the projective depth:

$$\lambda_{fp}\bar{\mathbf{w}}_{fp} = P_{Pf}\bar{\mathbf{s}}_p. \tag{3.11}$$

Gathering these equations for each point in each image into a measurement matrix of size $3F \times P$, we obtain the matrix equation:

$$W \equiv \begin{bmatrix} \lambda_{11}\bar{\mathbf{w}}_{11} & \lambda_{12}\bar{\mathbf{w}}_{12} & \cdots & \lambda_{1P}\bar{\mathbf{w}}_{1P} \\ \lambda_{21}\bar{\mathbf{w}}_{21} & \lambda_{22}\bar{\mathbf{w}}_{22} & \cdots & \lambda_{2P}\bar{\mathbf{w}}_{2P} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{F1}\bar{\mathbf{w}}_{F1} & \lambda_{F2}\bar{\mathbf{w}}_{F2} & \cdots & \lambda_{FP}\bar{\mathbf{w}}_{FP} \end{bmatrix} = \begin{bmatrix} P_1 \\ P_2 \\ \vdots \\ P_F \end{bmatrix} \begin{bmatrix} \bar{\mathbf{s}}_1 & \bar{\mathbf{s}}_2 & \cdots & \bar{\mathbf{s}}_P \end{bmatrix}. \tag{3.12}$$

With the correct projective depths, λ_{fp}, the measurement matrix will have at most rank 4 and can be decomposed using SVD as is done in the affine case. The key concept here is to view projective recovery as one of correctly rescaling the measurement matrix. While the projective depth for a single point is arbitrary, the projective depths for points in the same image have fixed ratios, and similarly the projective depths for a single feature appearing in multiple images will have fixed ratios. Thus the scale factors in the measurement matrix are only arbitrary up to a scale factor for each triple of rows, corresponding to features in the same image, and each column, corresponding to the same point in different images. Hence it is necessary to determine $FP \Leftrightarrow F \Leftrightarrow P$ scale factors in order to correctly re-scale W into the form of equation (3.12).

The projective depths can be determined modulo these arbitrary scales from the fundamental matrices and epipolar lines of features in the images. Sturm & Triggs (1996) derive the following result relating projective depths for point p in images i and j:

$$(\mathbf{F}_{ij}\bar{\mathbf{w}}_{jp})\,\lambda_{jp} = (\mathbf{e}_{ij} \wedge \bar{\mathbf{w}}_{ip})\,\lambda_{ip}, \tag{3.13}$$

where $\mathbf{F}_{ij}$ is the fundamental matrix and $\mathbf{e}_{ij}$ is the epipole. In part this is a restatement of the epipolar constraint: that the epipolar line of $\bar{\mathbf{w}}_{jp}$ (given by $\mathbf{F}_{ij}\bar{\mathbf{w}}_{jp}$) is the line joining the epipole $\mathbf{e}_{ij}$ and point $\bar{\mathbf{w}}_{ip}$ (here denoted by the join or cross product of these points, $\mathbf{e}_{ij} \wedge \bar{\mathbf{w}}_{ip}$). It also says that given the correct projective depths, these quantities are exactly equal, not just up to a scale factor. Using this relationship and knowledge of the epipoles and the fundamental matrices, all of the projective depths can be recovered modulo row and column scalings. The fundamental matrices and epipoles can be calculated from pairwise point corre-

spondences using standard algorithms such as the 8-point algorithm described well by Hartley (1995). Sturm and Triggs calculate these matrices along with projective depths by pairing images sequentially $\mathbf{F}_{12}, \mathbf{F}_{23}, \ldots, \mathbf{F}_{F-1,F}$. Calculating the depths by chaining together images can lead to an exponential growth or decrease their size. This does not change the rank 4 property, but may cause significant roundoff errors in SVD. Since rows and columns are only defined up to scale, they can be judiciously rescaled around unity.

The second step of factorization, following this rescaling and decomposition, consists of obtaining the Euclidean shape and motion using camera calibration parameters. The recovered projective shape and motion are unique only up to an arbitrary, invertible, 4×4 matrix H since:

$$W = PS = \hat{P}HH^{-1}\hat{S}. \tag{3.14}$$

If we know the camera calibration parameters then H can be obtained. Otherwise a non-linear method such as that described by Hartley (1993) could be used to perform camera calibration from the image sequence and so obtain the Euclidean shape.

Unlike the affine case, not all of the steps of the perspective factorization method work on all the images during the batch mode. The preprocessing to find epipolar geometry occurs across images in a pairwise manner. Hence it loses one of the benefits of affine factorization by not enforcing a globally consistent geometry. If instead of a sequence of pair-wise calculations, the epipolar geometry is estimated by comparing each image with the first image, this will ensure that the geometry is consistent, but then one image is arbitrarily selected as a special instance, creating greater sensitivity to errors. In practical terms, however, assuming that the epipolar geometry is known may not be a significantly greater assumption than assuming that points are registered.

4. Factorization with Alternative Features

(*a*) *Point Features*

In feature-based structure from motion, points are by far the most popular choice for features. As one of the most primitive features, point features enjoy many geometric advantages over other features. Points exist in all dimensions and are invariant to projective transformations; they remain points when projected onto an image and when rotated. Points are simple to describe and manipulate mathematically, and more complicated features can often be compactly described as a collection of point features. From the practical side, there are difficulties in detecting and registering points. Points are not measurable by a discrete pixel grid and so are typically defined by a template or window function. These templates do not have the invariant nature of point features under projection changes and so inaccuracies are introduced into the feature detection and registration. Despite this, points are still the feature of choice for most algorithms.

(*b*) *Line Features*

Another feature is the line. Lines share the primitive nature of points and invariance under projective transformations. While there are no ideal lines in real images, straight edges can be naturally modelled as lines, and in many cases can

be localized more accurately than point features. Hence line features are also attractive primitives for factorization. The main disadvantage of lines compared to points is that the projection of a line in 3D to a line in the image only identifies a plane on which the feature lies; whereas the projection of a point in 3D onto a point in the image identifies a line on which the feature lies. Thus line features provide less constraint information than point features.

(i) Lines with Affine Factorization

Quan & Kanade (1996) proposed a factorization algorithm for affine structure from line feature correspondences. The algorithm operates in four steps to retrieve shape and motion from a minimum of seven lines in three images. We briefly describe the algorithm in this section.

A line in $\mathcal{R}^3$ passing through a point $\mathbf{s}_0$ with direction $\mathbf{d}_s$ can be described in the form: $\mathbf{s} = \mathbf{s}_0 + \lambda \mathbf{d}_s$. Under an affine projection (3.2) this forms an image line:

$$\mathbf{w} = P_A \mathbf{s} = \mathbf{w}_0 + \lambda' \mathbf{d}_w, \tag{4.1}$$

where a point on the image line is $\mathbf{w}_0 = M\mathbf{s}_0 + \mathbf{w}'$ and the direction is given by:

$$\lambda'' \mathbf{d}_w = M \mathbf{d}_s. \tag{4.2}$$

Equation (4.2) relates the measured line directions, $\mathbf{d}_w$, to the affine motion, M, and the 3D line directions $\mathbf{d}_s$. Matrix M thus projects line directions in $\mathcal{R}^3$ to directions in $\mathcal{R}^2$ up to a scale factor. An alternative interpretation is to consider M as projecting a point in $\mathcal{P}^2$ to a point in $\mathcal{P}^1$ using homogeneous coordinates. Thus it has the same form as the projective equation (3.11) except that it is a 1D camera instead of a 2D camera. It can be solved in an analogous manner to perspective factorization. First the measurement matrix is formed:

$$W_D = \begin{bmatrix} \lambda_{11}\mathbf{d}_{u11} & \lambda_{12}\mathbf{d}_{u12} & \cdots & \lambda_{1P}\mathbf{d}_{u1P} \\ \lambda_{21}\mathbf{d}_{u21} & \lambda_{22}\mathbf{d}_{u22} & \cdots & \lambda_{2P}\mathbf{d}_{u2P} \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_{F1}\mathbf{d}_{uF1} & \lambda_{F2}\mathbf{d}_{uF2} & \cdots & \lambda_{FP}\mathbf{d}_{uFP} \end{bmatrix} \tag{4.3}$$

and it is rescaled appropriately. It can then be factored into affine motion and line directions:

$$W_D = M_D D_D = \begin{bmatrix} M_1 \\ M_2 \\ \vdots \\ M_F \end{bmatrix} \begin{bmatrix} \mathbf{d}_{s1} & \mathbf{d}_{s2} & \cdots & \mathbf{d}_{sP} \end{bmatrix}. \tag{4.4}$$

The second step of the algorithm is to recover the translation vectors $\mathbf{t}_p$, and the third step is to recover points through which the lines pass, $\mathbf{x}_p$. Both of these are described by Quan & Kanade (1996) and can be achieved by solving sets of linear equations with SVD.

The final step is the conversion from affine results to Euclidean shape and motion. This is done by applying the constraints for whichever affine camera is being used in the same was as for point features described in § 3(*a*). With this algorithm a minimum seven line directions in three images are needed to recover motion and shape, which is more than the four point features in three images needed by the point-based affine factorization. Also shape recovery for lines is typically more sensitive to noise than for point features.

(ii) Lines with Perspective Factorization

There has not yet been a direct extension from affine to perspective factorization for lines. Sturm & Triggs (1996) proposed an alternate route in which lines are represented by two points. A pair of points for each line are chosen in the first image and, using known epipolar geometry, the corresponding point positions on the lines in the rest of the images are calculated. The projective depths can also be calculated and then the perspective factorization for points (see §3(*b*)) can be applied to recover shape and motion.

As in perspective factorization for points, the epipolar geometry needs to be established, and since this is typically calculated from point correspondences, point features must be tracked in addition. On a deeper level, however, an important aspect of line features is lost by converting each into two point features. A line gives a strong one-dimensional constraint orthogonal to its length and no constraint along its length. Modelling a line with two point features results in an equal constraint or weighting along and perpendicular to the line. Thus simply using end-points to represent lines adds an artificial factor in the least squares that may harm the line fitting.

(*c*) *Generalized Points and Line Segments*

Up to this point factorization has been viewed as a least squares optimization algorithm; the configuration that minimizes the norm squared error from equation (2.6) is deemed the best solution. Poelman (1995) extended this to a weighted least squares optimization where each image point is given a separate weight so that unreliable or occluded points can be given less influence or discarded. The least squares solution is arbitrary in the sense that many error measures could be used in place of norm squared error for E_{SVD}, and it is unclear which will give superior results. The norm square error is chosen mostly for its convenient numerical properties including the SVD solution of E_{SVD}.

An alternative approach taken by Morris & Kanade (1998) is to make probabilistic assumptions about the data and then find the Maximum Likelihood solution. True feature locations, $\mathbf{x}_{fp}$, are assumed to have known 2D Gaussian probability densities in the image plane with covariances C_{fp} around the measured features, $\mathbf{w}_{fp}$:

$$\rho_{fp}(\mathbf{x}_{fp}) = k_{fp} \exp(\Leftrightarrow\frac{1}{2}(\mathbf{w}_{fp} \Leftrightarrow\mathbf{x}_{fp})^T C_{fp}^{-1}(\mathbf{w}_{fp} \Leftrightarrow\mathbf{x}_{fp})). \tag{4.5}$$

Assuming independence this results in a total density function given by the product of these: $\rho_T = \prod_{fp} \rho_{fp}(\mathbf{x}_{fp})$. The Maximum Likelihood solution to this is obtained by minimizing the cost function, E_B, given by:

$$E_B = \sum_{fp} \frac{1}{2}(\mathbf{w}_{fp} \Leftrightarrow\mathbf{x}_{fp})^T C_{fp}^{-1}(\mathbf{w}_{fp} \Leftrightarrow\mathbf{x}_{fp}). \tag{4.6}$$

If feature position is given by the affine model, $\mathbf{x}_{fp} = M_f \mathbf{s}_p$, and all features have unit covariance matrices, then E_B and E_{SVD} are equivalent up to scale. This implies that the original factorization algorithm is the Maximum Likelihood solution when feature uncertainty is modelled as having independent, identical Gaussian distribution. Other choices of covariance matrices are possible permitting the use of arbitrary Gaussian densities for feature positions. In particular directional uncertainty of feature position may be modelled as illustrated in figure 2(a). The

Figure 2. (a) Directional feature uncertainty modelling, (b) Line segment modeling with endpoints having large directional uncertainties along line length.

new cost E_B can no longer be minimized with SVD, but Morris & Kanade (1998) present an efficient bilinear algorithm for achieving this and recovering motion and shape.

This generalized point feature model can be extended to model line segments as well. Practical line registration algorithms actually work with line segments which are fit to edges in the images. The positions of the recovered line endpoints may correspond to different parts of the edge in any given image, but they are restricted to fall within the physical limits of the object. Thus, by using line segments which correspond to a particular region of the infinite line, one can extract more constraints than are available for infinite lines. Line segments can be defined by modelling their end-points as Gaussian density functions with large uncertainty along the length of the line and small uncertainty perpendicular to the line, as illustrated in figure 2(b). The same factorization algorithm will thus work with line segments modelled in this way.

Using the generalized point formulation increases the flexibility and accuracy of factorization and still leverages the bilinear nature of the problem, but it loses the guarantee of global convergence. Yet the additional flexibility is the ability to naturally handle missing features. The original formulation required and iterative "hallucination" scheme by Tomasi & Kanade (1992) to handle missing features, but this negates much of the gains in the SVD formulation. With generalized features the a missing feature is given zero probability.

5. Extension to the Multi-Body Case

A common and basic assumption of most structure from motion algorithms is that features belong to a single rigid object. When there are multiple independently moving objects in a scene with features tracked on each, the additional challenge becomes to segment features and then recover object motion and shape. The difficulty introduced by multiple objects (particularly if the number is unknown) is that typically segmentation depends on object motion estimates and object motion estimation depends on a prior segmentation of the features. Approaches to solving this cyclic dilemma have generally relied on recursive clustering techniques such as by Boult & Brown (1991) and Gear (1994). Costeira & Kanade (1995) showed that this multi-body problem could be described in a bilinear formulation, and then decomposed and solved segmentation and recovery of motion and shape at the same time using a factorization method. Their algorithm, which uses an affine approximation, does not need to know the number of shapes nor does it involve recursive application.

First the problem is formulated in a bilinear form. Since individual objects may have different translations, both rotation and translation must be incorporated in the motion matrix. This is achieved using homogeneous coordinates for object

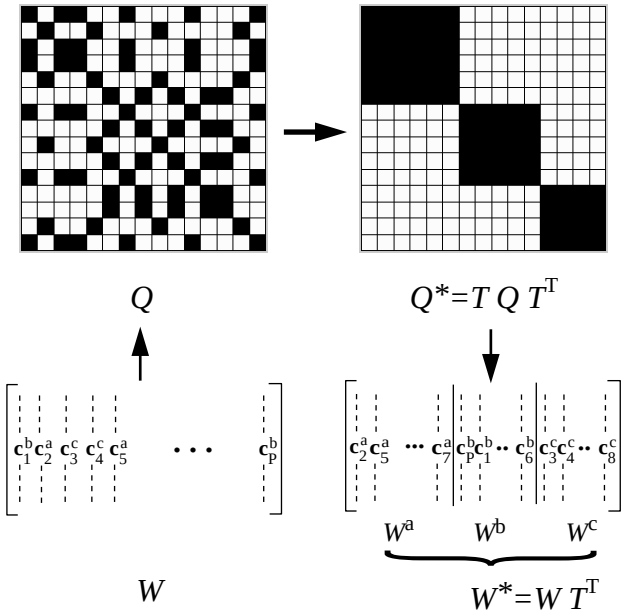

Figure 3. The shape interaction matrix Q, is transformed into block diagonal form, Q^*, with each block representing a separate object—in this case three objects. Using the same transformation T, the columns, $\mathbf{c}_i^{object}$, of W can be reordered into a segmented form.

points: $\bar{\mathbf{s}} = (\mathbf{s}^T, 1)^T$, so that the projection equation becomes:

$$\mathbf{w}_{fp} = \begin{bmatrix} M_f & \mathbf{w}'_f \end{bmatrix} \bar{\mathbf{s}}. \tag{5.1}$$

A measurement matrix, W, analogous to that in equation (2.4) can be formed by stacking features in rows and columns. The only difference is that since points are not in object centered coordinates, matrix W produced by a single rigid object will now have rank at most 4. When there are multiple rigid objects, the columns of W, representing features, are from each of the objects and are mixed in an unknown manner. The rank of W increases with each independently moving rigid object. Let us assume that $r = rank(W)$. The goal of segmentation is to permute the columns of W into the form:

$$W^* = \begin{bmatrix} W^a & W^b & \cdots \end{bmatrix} \tag{5.2}$$

where each component, W^a, W^b, etc., is at most rank 4.

The first step of decomposition is to perform SVD on W to obtain $W = U\Sigma V^T$. We let V_r consist of the first r columns of V. We then form the *shape interaction matrix* Q given by:

$$Q = V_r V_r^T. \tag{5.3}$$

The shape interaction matrix captures the structural information of the objects and its entries are invariant to object translation and rotation. The p'th row and p'th column correspond to the p'th feature, $\mathbf{s}_p$. Costeira & Kanade (1995) showed that in the zero noise case Q can be transformed into a block diagonal form Q^* by applying a transform, which here we denote by T, such that:

$$Q^* = TQT^T. \tag{5.4}$$

T is a square matrix whose columns are a permutation of the identity matrix. Pre-multiplying by T thus permutes rows of Q, and post-multiplying by T^T permutes columns of Q. An iterative technique described by Costeira & Kanade (1995) is used to find the transform, T, that results in the block diagonal matrix Q^* illustrated in figure 3. The rows and columns of each block in Q^* identify those features whose motions in the image sequence are linearly dependent on each other and independent of other features, and hence are part of the same rigid object.

The effect of post-multiplying Q or W by T is to reorder the respective columns of Q or W from their initial sequential ordering by feature index to a new segmented ordering where columns from the same feature are adjacent. Thus the segmented measurement and shape matrices are obtained as:

$$W^* = WT^T \qquad (5.5)$$
$$S^* = ST^T.$$

Subsequent to this the standard factorization procedure must be applied in an analogous way to that described in § 3.

The multi-body factorization method performs a bilinear decomposition into motion and shape components and so can eliminate the independent motions of the objects. Then using only the linear dependences of rows and columns in the shape interaction matrix, the object features can be segmented. The elimination of motion leads to a simpler segmentation algorithm than that of Gear (1994) who iteratively scaled and permuted columns of the measurement matrix based on linear dependences to obtain segmentation. A weakness of the factorization-based algorithm is that analysis is performed assuming zero noise. With noise thresholds and other approximations must be introduced.

6. Factorization in Other Domains

There are other situations outside the structure from motion domain that can be formulated as a bilinear decomposition and the factorization paradigm may be a powerful tool for tackling the problems. Here we describe two such problems.

(*a*) *Force/Torque Sensor Calibration*

Multi-axis force and torque sensing devices are used in applications such as haptic interfaces. They are designed to convert a force and/or torque into an electrical signal. Figure 4 illustrates a simple two degree of freedom force sensor in which the four strain gauges measure bending of the cantilever along approximately orthogonal directions. The device requires calibration because of sensitivities of strain gauges are different and they may not be placed precisely 90 degrees apart. Calibration typically requires applying a force and measuring it and the response, and then repeating this in many directions. The transformation equation is defined:

$$C\mathbf{z} = \mathbf{m} \qquad (6.1)$$

for a vector $\mathbf{m}$ containing force and torque components, vector $\mathbf{z}$ containing sensor output, and calibration matrix C. The sensor shown in figure 4 only measures forces and so here $\mathbf{m}$ contains just force components. The equation can be ex-

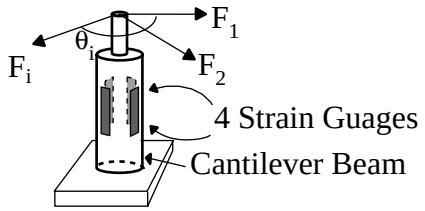

Figure 4. Two degree of freedom force sensor

pressed as:

$$\begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \end{bmatrix} \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ z_4 \end{bmatrix} = \begin{bmatrix} m_x \\ m_y \end{bmatrix} \equiv m_o \begin{bmatrix} \sin\theta \\ \cos\theta \end{bmatrix}. \tag{6.2}$$

A least squares solution for the calibration parameters can easily be formulated given multiple corresponding $\mathbf{m}_i$ and $\mathbf{z}_i$ vectors. The main drawback with this approach is that the applied force vector, $\mathbf{m}_i = m_o[\ \sin\theta_i \ \ \cos\theta_i\ ]^T$, must be known. The magnitude of the force, m_o, for multiple measurements can easily be made constant by suspending a known weight from the tip of the force sensor. However, the direction, θ_i, of the force must be painstakingly acquired for each electronic measurement of $\mathbf{z}_i$.

Voyles *et al.* (1997) show that the calibration task can be recast in a bilinear form, and then solved using the factorization technique. Equation (6.1) is restated in the form:

$$\mathbf{z}_i^T = \mathbf{m}_i^T S, \tag{6.3}$$

for the i'th measurement, and where C is the pseudo-inverse of S. With n measurements this equation can be stacked to form a matrix equation:

$$Z = MS \tag{6.4}$$

where

$$Z = \begin{bmatrix} \mathbf{z}_1^T \\ \vdots \\ \mathbf{z}_n^T \end{bmatrix} \text{ and } M = \begin{bmatrix} \mathbf{m}_1^T \\ \vdots \\ \mathbf{m}_n^T \end{bmatrix}. \tag{6.5}$$

We assume here only that the sensor data Z is available, and that the magnitude of the force components, m_o, is constant and known. The key advantage with this approach is that the directions, θ_i, need not be measured. This equation is thus analogous to bilinear camera equation (2.4) and can be solved using SVD decomposition. The necessary requirements for this to be a useful least squares solution technique is that the rank of Z be less than the number of columns, and that there is a tractable technique to determine matrix A that removes the ambiguity inherent in the solution: $MS = \hat{M}A^{-1}A\hat{S}$. Voyles *et al.*(1997) show how these conditions are met in their force/torque sensor domain.

The work by Voyles *et al.* can be seen as a recasting of the problem from the linear domain, which needed careful measurements of force data, to a bilinear problem in which both force/torque and calibration parameters are calculated. This makes the task of performing measurements much simpler and faster and

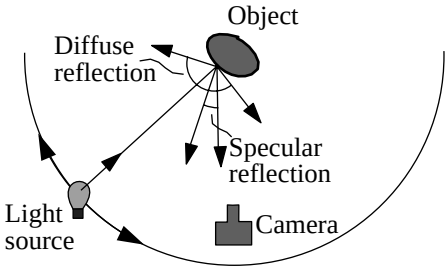

Figure 5. Experimental arrangement to model surface reflectance properties of an object.

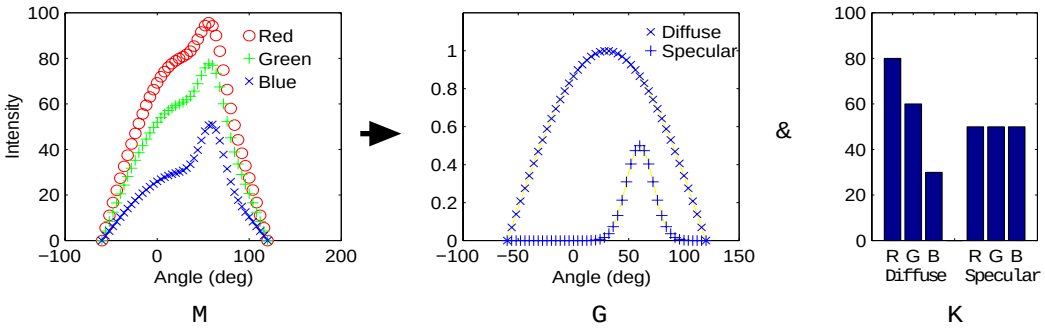

Figure 6. The measurement matrix M contains the intensity of the RGB components of the reflected light for a single pixel as a function of illumination angle. It is decomposed into two matrices $M = GK$: where the geometry matrix, G, contains the relative intensities of diffuse and specular components for each angle, and the colour matrix, K, contains the RGB components for the diffuse and specular reflectance.

at the same time they report comparable or even slightly improved results using the factorization method.

(*b*) *Object Colour Modelling from Image Sequences*

Colour image sequences taken by a fixed camera and moving light source can be used to recover the surface reflectance properties of an object and surface orientation of the object. Sato & Ikeuchi (1995) and Sato (1997) present a carefully performed experiment in which a single light source illuminates an object from multiple points along a circular trajectory as illustrated in figure 5. Images are taken from a fixed camera.

The red, green and blue (RGB) components are extracted for each object pixel in each image. A measurement matrix is formed from the RGB components for a single pixel and its corresponding pixels in all the images:

$$M = \begin{bmatrix} R_1 & G_1 & B_1 \\ \vdots & \vdots & \vdots \\ R_n & G_n & B_n \end{bmatrix} \tag{6.6}$$

for n images. Sato & Ikeuchi (1995) derive the following decomposition of this matrix into two matrices:

$$M = GK \tag{6.7}$$

where G is a $n \times 2$ geometry matrix and K is a 2×3 colour matrix. The ge-

ometry matrix, which contains surface normal information, identifies the relative contribution of diffuse and specular reflectance to the total reflection. Its form is determined using a Lambertian model for diffuse reflection components and a simplified Torrence Sparrow model for specular reflection components. The colour matrix contains the RGB intensities of the diffuse and specular reflection.

The decomposition in equation (6.7) could be solved by first finding one of the matrices G or K, and then obtaining the least squares solution for the other, or it could be decomposed using the factorization method. Sato & Ikeuchi (1995) follow the former approach, using alternative techniques to obtain K and solving for G with linear least squares method. They could have used SVD to obtain the best rank 2 approximation to M, and then decomposed it into $\hat{G}$ and $\hat{K}$ which are defined up to a 2×2 invertible such that $G = \hat{G}A$ and $K = A^{-1}\hat{K}$. The correct transformation A is determined by applying geometric or colour constraints in an analogous way to that described in § 3. Figure 6 shows a synthetic example of the RGB components of a single pixel and their decomposition into the geometry and colour matrices.

Whether a linear least squares or a bilinear decomposition approach is better depends on the availability of information describing the recovered components. If one of the two components is easily and accurately determined by alternate means, then the linear least squares may be best. However if neither components are accurately known and there are sufficient constraints that can be applied on one or both of the components, then the factorization method may be the simpler and more accurate approach.

7. Summary

The factorization method works in several steps. First the equations must be formulated in a bilinear form. This is achieved for the affine case either by working in an object-based coordinate system or by using homogeneous coordinates. In the perspective case it is necessary to find a rescaling of the homogeneous coordinates in order to formulate the matrix equation. The decomposition then produces only the affine or projective shape and motion. The next step consisted of imposing, and possibly recovering, calibration and orientation constraints to obtain Euclidean motion.

Since its inception in the late 1980's, the factorization method has proven to be a powerful approach for the structure from motion problem. The factorization method has performance improvements over direct non-linear methods, with guaranteed global convergence for SVD calculations and rapid convergence in other bilinear formulations. Its reliability is also due to uniformly applying constraints, such as the rigidity constraint, over all features and images, and avoiding the use of special images.

Finally factorization methods have found application in areas beyond structure from motion where a bilinear decomposition into two components can be formulated. Constraint information on the recovered components is then used to transform these components into the desired form for the problem. These methods have much greater power and flexibility than direct linear methods.

References

Boult, T. & Brown, L. 1991 Factorization-based segmentation of motions. In *Proc. IEEE Workshop on Visual Motion*, 179–86, Princeton, NJ, USA.

Costeira, J. & Kanade, T. 1995 A multi-body factorization method for motion analysis. In *Proc. Fifth Int. Conf. Computer Vision*, 1071–1076, Cambridge, MA, USA.

Deguchi, K. 1997 Factorization method for structure from multiple perspective images. Tech. Rep. Meip7-97001-02, Dept. Math. Eng. Infor. Phys., U. Tokyo.

Gear, C. W. 1994 Feature grouping in moving objects. In *Proc. Workshop Mot. Non-rigid Artic. Obj.*, 214–19, Austin, TX, USA.

Hartley, R. 1993 Euclidean reconstruction from uncalibrated views. In *Proc. DARPA–ESPRIT Workshop App. Invariants in Comp. Vision*, 187–202, Azores, Portugal.

Hartley, R. 1995 In defence of the 8-point algorithm. In *Proc. Fifth Int. Conf. Computer Vision*, 1064–70, Cambridge, MA, USA.

Kontsevich, L. L., Kontsevich, M. L. & Shen, A. K. 1987 Two algorithms for reconstructing shapes. *Avtometriya* 76–81. (Transl. Optoelec., Instrum. Data Proc., no. 5. 76–81, 1987).

Longuet-Higgins, H. 1981 A computer algorithm for reconstructing a scene from two projections. *Nature* **293**, 133–35.

Morita, T. & Kanade, T. 1994 A sequential factorization method for recovering shape and motion from image streams. Tech. Rep. CMU-CS-94-158, Carnegie Mellon U.

Poelman, C. & Kanade, T. 1994 A paraperspective factorization method for shape and motion recovery. In *Proc. Third Euro. Conf. Computer Vision*, vol. 2, 97–108, Stockholm, Sweden.

Morris, D. D. & Kanade, T. 1998 A unified factorization algorithm for points, line segments and planes with uncertainty models. In *Proc. Sixth Int. Conf. Computer Vision*, 696–702, Bombay, India.

Ohta, Y., Maenobu, K., & Sakai, T. 1981 Obtaining surface orientation from texels under perspective projection. In *Proc. 7th Int. Joint Conf. on AI*, 746–51.

Poelman, C. 1995 *The Paraperspective and Projective Factorization Methods for Recovering Shape and Motion*. Ph.D. thesis, Carnegie Mellon U., Pittsburgh, PA, USA.

Quan, L. & Kanade, T. 1996 A factorization method for affine structure from line correspondences. In *Proc. Comp. Vision Patt. Recog.*, 803–8, San Fran., CA, USA.

Sato, Y. & Ikeuchi, K. 1995 Reflectance analysis for 3d computer graphics model generation. Tech. Rep. CMU-CS-95-146, Carnegie Mellon U.

Sato, Y. 1997 *Object Shape and reflectance modeling from Color image sequence*. Ph.D. thesis, Carnegie Mellon U., Pittsburgh, PA, USA.

Sturm, P. & Triggs, B. 1996 A factorization based algorithm for multi-image projective structure and motion. In *Proc. Euro. Conf. Computer Vision*, 709–20, Cambridge, UK.

Tomasi, C. & Kanade, T. 1990 Shape and motion without depth. In *Proc. Third Int. Conf. Computer Vision*, 91–5, Osaka, Japan.

Tomasi, C. & Kanade, T. 1992 Shape and motion from image streams under orthography: a factorization method. *Int. J. Comp. Vision* **9**, 137–54.

Triggs, B. 1996 Factorization methods for projective structure and motion. In *Proc. Comp. Vision Patt. Recog.*, 845–851, San Fran., CA, USA.

Tsai, R. Y. & Huang, T. S. 1984 Uniqueness and estimation of three-dimensional motion parameters of rigid objects with curved surfaces. *IEEE Trans. Patt. Anal. Mach. Intelligence* **6**, 13–27.

Ullman, S. 1979 *The Interpretation of Visual Motion*. Cambridge, MA, USA: MIT Press.

Voyles, R. M., Morrow, J. D., & Khosla, P. K. 1997 The shape from motion approach to rapid and precise force/torque sensor calibration. *J. Dynamic Sys., Measurement and Control* **119**, 229–35.