Memoirs of the Faculty of Engineering, Okayama University, Vol.38, Nos.1&2, pp.61-72, March 2004

Factorization without Factorization: Complete Recipe

Kenichi KANATANI* and Yasuyuki SUGAYA
Department of Information Technology, Okayama University
Okayama 700-8530 Japan

(Received February 13, 2004)

The Tomasi-Kanade *factorization* for reconstructing the 3-D shape of the feature points tracked through a video stream is widely regarded as based on factorization of a matrix by SVD (singular value decomposition). This paper points out that the core principle is the affine camera approximation to the imaging geometry and that SVD is merely one means of numerical computation. We first describe the geometric structure of the problem and then give a complete programming scheme for 3-D reconstruction.

1. Introduction

The *factorization* method of Tomasi and Kanade [12] is one of the best known techniques for 3-D reconstruction from feature points tracked through a video stream. This method computes the camera motion and the 3-D shape of the scene by approximating the camera imaging geometry by an *affine transformation*. Its execution is very easy, requiring only linear analysis of the trajectories of the feature points. The reconstructed shape has sufficient accuracy for many practical applications for which very high-precision is not required. Also, the solution can be used as an initial value for more sophisticated iterative methods based on rigorous perspective projection [1].

After more than ten years since its birth, however, there is still a wide-spread misunderstanding that *it is a method for reconstructing 3-D by matrix factorization using SVD (singular value decomposition)*. In reality, however, the underlying principle is only the affine approximation to the camera imaging; factorization by SVD is nothing but an (optional) means for numerically computing the least-squares solution.

This fact should be so obvious to those using this method that this need not particularly be mentioned. For those who did not use this method, however, explaining this may help them really understand what the Tomasi-Kanade factorization is.

One cause of the misunderstanding may be that the actual procedure for the factorization method is not given in most literature except for characterizing it as "factorization by SVD". In truth, the core of this method lies in what is known as the *metric condition* and *least-squares optimization* of the motion and shape.

The purpose of this paper is two-fold:

1. We explain the *geometric* structure of the factorization method, thereby showing that matrix factorization by SVD does not play any essential role.
2. From this viewpoint, we describe a complete computational procedure for this method, detailing specific optimization steps for individual camera models.

In the following, we first describe the underlying geometric structure and the general framework of the computation. Then, we describe the actual algorithm in detail. The derivation is summarized in the appendix.

2. Affine Camera Model

Suppose we track N feature points over M frames. Let $(x_{\kappa\alpha}, y_{\kappa\alpha})$ be the coordinates of the αth point in the κth frame[1]. Stacking all the coordinates vertically, we represent the entire trajectory by the following $2M$-D *trajectory vector*:

$$\boldsymbol{p}_\alpha = \begin{pmatrix} x_{1\alpha} & y_{1\alpha} & x_{2\alpha} & y_{2\alpha} & \cdots & x_{M\alpha} & y_{M\alpha} \end{pmatrix}^\top. \quad (1)$$

For convenience, we identify the frame number κ with "time" and refer to the κth frame as "time κ".

We regard the XYZ camera coordinate system as the world frame[2], relative to which the scene is moving. Consider a 3-D coordinate system fixed to the

*E-mail kanatani@suri.it.okayama-u.ac.jp

[1]The coordinate origin is arbitrary, e.g., at the upper-left corner of the image, as long as orthographic projection is assumed. If weak perspective or paraperspective projection is assumed, however, we must specify the principal point (the point that corresponds to the camera optical axis), typically at the center of the image frame, and take it as the coordinate origin, because we are approximating the perspective projection.

[2]The mathematical structure is the same if we regard the camera as moving relative to a stationary scene. However, if we consider multiple motions [4, 5, 6, 7, 9, 10], we need to take the camera as a reference. So, using the camera-based world frame is more consistent than using the object-based world frame.

scene, and let $\boldsymbol{t}_\kappa$ and $\{\boldsymbol{i}_\kappa, \boldsymbol{j}_\kappa, \boldsymbol{k}_\kappa\}$ be, respectively, its origin and basis vectors at time κ. If the αth point has coordinates $(a_\alpha, b_\alpha, c_\alpha)$ with respect to this coordinate system, its position with respect to the world frame at time κ is

$$\boldsymbol{r}_{\kappa\alpha} = \boldsymbol{t}_\kappa + a_\alpha \boldsymbol{i}_\kappa + b_\alpha \boldsymbol{j}_\kappa + c_\alpha \boldsymbol{k}_\kappa. \tag{2}$$

We assume an affine camera, which generalizes orthographic, weak perspective, and paraperspective projections [8]: the 3-D point $\boldsymbol{r}_{\kappa\alpha}$ is projected onto the image position

$$\begin{pmatrix} x_{\kappa\alpha} \\ y_{\kappa\alpha} \end{pmatrix} = \boldsymbol{A}_\kappa \boldsymbol{r}_{\kappa\alpha} + \boldsymbol{b}_\kappa, \tag{3}$$

where $\boldsymbol{A}_\kappa$ and $\boldsymbol{b}_\kappa$ are, respectively, a 2×3 matrix and a 2-D vector determined by the position and orientation of the camera and its internal parameters at time κ. Substituting Eq. (2), we have

$$\begin{pmatrix} x_{\kappa\alpha} \\ y_{\kappa\alpha} \end{pmatrix} = \tilde{\boldsymbol{m}}_{0\kappa} + a_\alpha \tilde{\boldsymbol{m}}_{1\kappa} + b_\alpha \tilde{\boldsymbol{m}}_{2\kappa} + c_\alpha \tilde{\boldsymbol{m}}_{3\kappa}, \tag{4}$$

where $\tilde{\boldsymbol{m}}_{0\kappa}$, $\tilde{\boldsymbol{m}}_{1\kappa}$, $\tilde{\boldsymbol{m}}_{2\kappa}$, and $\tilde{\boldsymbol{m}}_{3\kappa}$ are 2-D vectors determined by the position and orientation of the camera and its internal parameters at time κ. From Eq. (4), the trajectory vector $\boldsymbol{p}_\alpha$ in Eq. (1) can be written in the form

$$\boldsymbol{p}_\alpha = \boldsymbol{m}_0 + a_\alpha \boldsymbol{m}_1 + b_\alpha \boldsymbol{m}_2 + c_\alpha \boldsymbol{m}_3, \tag{5}$$

where $\boldsymbol{m}_0$, $\boldsymbol{m}_1$, $\boldsymbol{m}_2$, and $\boldsymbol{m}_3$ are the $2M$-D vectors obtained by stacking $\tilde{\boldsymbol{m}}_{0\kappa}$, $\tilde{\boldsymbol{m}}_{1\kappa}$, $\tilde{\boldsymbol{m}}_{2\kappa}$, and $\tilde{\boldsymbol{m}}_{3\kappa}$ vertically over the M frames, respectively.

Eq. (5) implies that the trajectory vectors $\{\boldsymbol{p}_\alpha\}$ are constrained to be in the *4-D subspace* spanned by $\{\boldsymbol{m}_0, \boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{m}_3\}$ in $\mathcal{R}^{2M}$. This fact is called the *subspace constraint* [4]. Moreover, the coefficient of $\boldsymbol{m}_0$ in eq. (5) is identically 1, meaning that $\{\boldsymbol{p}_\alpha\}$ are constrained to be in the *3-D affine space* passing through $\boldsymbol{m}_0$ and spanned by $\{\boldsymbol{m}_1, \boldsymbol{m}_2, \boldsymbol{m}_3\}$. This fact is called the *affine space constraint* [5]. These geometric interpretations play a central role in recovering missing data [11] and segmenting independently moving multiple objects into individual motions [4, 5, 6, 7, 9, 10].

3. Flow of Computation

From the above observation, the procedure for reconstructing 3-D from the data $\{\boldsymbol{p}_\alpha\}$ consists of the following four stages:

1. Fit a 3-D affine space to $\{\boldsymbol{p}_\alpha\}$ by least squares.
2. From the fitted space, compute the vectors $\boldsymbol{m}_0$, $\boldsymbol{m}_1$, $\boldsymbol{m}_2$, and $\boldsymbol{m}_3$.
3. From the resulting vectors $\boldsymbol{m}_0$, $\boldsymbol{m}_1$, $\boldsymbol{m}_2$, and $\boldsymbol{m}_3$, compute the position $\boldsymbol{t}_\kappa$ and the orientation $\{\boldsymbol{i}_\kappa, \boldsymbol{j}_\kappa, \boldsymbol{k}_\kappa\}$ of the scene coordinate system.
4. Compute the scene coordinates $(a_\alpha, b_\alpha, c_\alpha)$ of individual feature points.

What is known as "factorization by SVD" is a typical numerical scheme for the first stage. The core of the procedure is the second stage and is known as the *metric condition*, which depends on what camera model is assumed. The third stage is non-linear optimization, because we need to incorporate the constraint that $\{\boldsymbol{i}_\kappa, \boldsymbol{j}_\kappa, \boldsymbol{k}_\kappa\}$ be an orthonormal system. The solution is given by SVD [2]. The fourth stage is the simplest: we only need to minimize

$$J = \sum_{\alpha=1}^{N} \|\boldsymbol{p}_\alpha - \boldsymbol{m}_0 - a_\alpha \boldsymbol{m}_1 - b_\alpha \boldsymbol{m}_2 - c_\alpha \boldsymbol{m}_3\|^2. \tag{6}$$

This is a quadratic minimization in $(a_\alpha, b_\alpha, c_\alpha)$, so the solution is obtained by solving a linear equation (the *normal equation*).

4. Affine Space Fitting

Since the absolute position of the scene coordinate system is arbitrary, we take its origin at the centroid of the N feature points. Then, $\sum_{\alpha=1}^{N} a_\alpha = \sum_{\alpha=1}^{N} b_\alpha = \sum_{\alpha=1}^{N} c_\alpha = 0$, so eq. (5) implies that $\boldsymbol{m}_0$ coincides with the centroid of $\{\boldsymbol{p}_{\kappa\alpha}\}$

$$\boldsymbol{p}_C = \frac{1}{N} \sum_{\alpha=1}^{N} \boldsymbol{p}_\alpha. \tag{7}$$

Let

$$\boldsymbol{p}'_\alpha = \boldsymbol{p}_\alpha - \boldsymbol{p}_C, \tag{8}$$

and define the (*second-order*) *moment matrix*

$$\boldsymbol{C} = \sum_{\alpha=1}^{N} \boldsymbol{p}'_\alpha \boldsymbol{p}'^{\top}_\alpha. \tag{9}$$

As is well known, the affine space that fits $\{\boldsymbol{p}_\alpha\}$ optimally (in the sense of least squares) passes through the centroid $\boldsymbol{p}_C$ and is spanned by the unit eigenvectors $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3\}$ of $\boldsymbol{C}$ for the largest three eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3$. It follows that the vectors $\boldsymbol{m}_1$, $\boldsymbol{m}_2$, and $\boldsymbol{m}_3$ can be expressed as a linear combination of the basis $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3\}$ in the form

$$\boldsymbol{m}_j = \sum_{i=1}^{3} A_{ij} \boldsymbol{u}_i. \tag{10}$$

It can be shown, however, that what we need for 3-D reconstruction is not the matrix $\boldsymbol{A} = (A_{ij})$ itself but the *metric matrix*

$$\boldsymbol{T} = \boldsymbol{A}\boldsymbol{A}^{\top}. \tag{11}$$

The constraint on the matrix $\boldsymbol{T}$ depends on the assumed camera model and is called the *metric condition*.

5. Why "Factorization"?

As we have seen, matrix factorization by SVD is not necessary for 3-D reconstruction. However, we can use it as a *numerical means*. In fact, we may define the $2M \times N$ *observation matrix*

$$\boldsymbol{W} = \begin{pmatrix} \boldsymbol{p}'_1 & \boldsymbol{p}'_2 & \cdots & \boldsymbol{p}'_N \end{pmatrix} \tag{12}$$

that has $\boldsymbol{p}'_1$, $\boldsymbol{p}'_2$, ..., $\boldsymbol{p}'_N$ as its column. Let

$$\boldsymbol{W} = \boldsymbol{U}_N \mathrm{diag}(\sigma_1, \sigma_2, ..., \sigma_N) \boldsymbol{V}_N^\top \tag{13}$$

be its SVD, where $\boldsymbol{U}_N$ and $\boldsymbol{V}_N$ are, respectively, $2M \times N$ and $N \times N$ matrices consisting of an orthonormal system of columns, and σ_1, σ_2, ..., σ_N (≥ 0) are the singular values (diag($\cdots$) denotes the diagonal matrix having $\cdots$ as its diagonal elements). It is easily seen that the ith column of $\boldsymbol{U}_N$ is the unit eigenvalue of the moment matrix $\boldsymbol{C}$ in eq. (9) for the eigenvalue $\lambda_i = \sigma_i^2$ [2, 3]. Thus, the eigendecomposition of the moment matrix $\boldsymbol{C}$ is mathematically equivalent to the SVD of the observation matrix $\boldsymbol{W}$.

The computational complexity of SVD of the $2M \times N$ matrix $\boldsymbol{W}$ usually depends on $\min(2M, N)$. So, SVD will be computationally more efficient when the number $2M$ of frames is much larger than the number N of feature points.

From eq. (5), we have $\boldsymbol{p}'_\alpha = a_\alpha \boldsymbol{m}_1 + b_\alpha \boldsymbol{m}_2 + c_\alpha \boldsymbol{m}_3$. Substituting this into eq. (12), we have

$$\begin{aligned} \boldsymbol{W} &= \begin{pmatrix} \boldsymbol{m}_1 & \boldsymbol{m}_2 & \boldsymbol{m}_3 \end{pmatrix} \begin{pmatrix} a_1 & a_2 & \cdots & a_N \\ b_1 & b_2 & \cdots & b_N \\ c_1 & c_2 & \cdots & c_N \end{pmatrix} \\ &= \boldsymbol{M}\boldsymbol{S}, \end{aligned} \tag{14}$$

where $\boldsymbol{M}$ is the $2M \times 3$ *motion matrix* having $\boldsymbol{m}_i$ as its ith column, and $\boldsymbol{S}$ is the $3 \times N$ *shape matrix* having $(a_\alpha, b_\alpha, c_\alpha)^\top$ as its αth column. In this notation, eq. (10) can be rewritten as

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{A}, \tag{15}$$

where $\boldsymbol{U}$ is a $2M \times 3$ matrix consisting of the first three column of the matrix $\boldsymbol{U}_N$.

Tomasi and Kanade [12] expressed the affine space constraint (5) as eq. (14) and called it the *rank 3 constraint*[3]. Then, they applied SVD to the observation matrix $\boldsymbol{W}$ in the form of eq. (13), determining the motion matrix $\boldsymbol{M}$ in the form of eq. (15) using the metric condition.

From this originates the interpretation that their method is to do matrix factorization by SVD. However, we should not mix up the mathematical structure of the problem with numerical means for solving it. As we noted earlier, SVD is simply an (optional) numerical means for affine space fitting.

[3]If we do not center the coordinate system at the centroid, we can obtain what is known as the *rank 4 constraint*: $\begin{pmatrix} \boldsymbol{p}_1 & \cdots & \boldsymbol{p}_N \end{pmatrix} = \begin{pmatrix} \boldsymbol{m}_0 & \boldsymbol{m}_1 & \boldsymbol{m}_2 & \boldsymbol{m}_3 \end{pmatrix} \begin{pmatrix} 1 & \cdots & 1 \\ a_1 & \cdots & a_N \\ b_1 & \cdots & b_N \\ c_1 & \cdots & c_N \end{pmatrix}$, or $\tilde{\boldsymbol{W}} = \tilde{\boldsymbol{M}}\tilde{\boldsymbol{S}}$. This is a more faithful expression of our affine space constraint (5).

6. Main Procedure

We first describe the main procedure for 3-D reconstruction independent of individual camera models. Then, we detail the subprocedures that depend on the assumed camera models. The derivation is summarized in the appendix.

The important thing to note is that *two* solutions are obtained and that the corresponding 3-D shapes are *mirror images* of each other. The two solutions cannot be distinguished under the affine camera model: *both result in identical images when projected.*

Input:

- $2M$-D trajectory vectors $\{\boldsymbol{p}_\alpha\}$, $\alpha = 1, ..., N$ (M is the number of frames, N is the number of feature points).
- the average scene depth Z_c and the focal lengths $\{f_\kappa\}$ for all the frames (arbitrarily assigned if unknown).

Output:

- the translation vectors $\{\boldsymbol{t}_\kappa\}$ (the positions of the centroid of the feature points).
- the 3-D positions $\{\hat{\boldsymbol{r}}_\alpha\}$ and $\{\hat{\boldsymbol{r}}'_\alpha\}$ (mirror images of each other) of the individual feature points with respect to the first frame.
- the 3-D rotations $\{\boldsymbol{R}_\kappa\}$ and $\{\boldsymbol{R}'_\kappa\}$ (the orientations of the scene coordinate system).

Main procedure[4]**:**

— *Fitting an affine space* —

1. Compute the centroid $\boldsymbol{p}_C$ of the trajectory vectors $\{\boldsymbol{p}_\alpha\}$ by eq. (7).
2. Let $\tilde{t}_{x\kappa}$ and $\tilde{t}_{y\kappa}$ be the $(2(\kappa-1)+1)$th and $(2(\kappa-1)+2)$th components of $\boldsymbol{p}_C$, respectively.
3. Fit a 3-D affine space to the trajectory vectors $\{\boldsymbol{p}_\alpha\}$, and let $\{\boldsymbol{u}_1, \boldsymbol{u}_2, \boldsymbol{u}_3\}$ be its basis.
4. Let $\boldsymbol{U}$ be the $2M \times 3$ matrix having $\boldsymbol{u}_1$, $\boldsymbol{u}_2$, and $\boldsymbol{u}_3$ as its columns, and let $\boldsymbol{u}^\dagger_{\kappa(a)}$ be the $(2(\kappa-1)+a)$th column of $\boldsymbol{U}^\top$, $\kappa = 1, ..., M$, $a = 1, 2$.

— *Computing the metric matrix* —

5.* Compute the 3×3 metric matrix $\boldsymbol{T}$.

6. Compute the eigenvalues $\{\lambda_1, \lambda_2, \lambda_3\}$ of $\boldsymbol{T}$ and the corresponding orthonormal system $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}$ of unit eigenvectors.

   # $\boldsymbol{T}$ is a positive definite symmetric matrix; its eigenvalues $\{\lambda_1, \lambda_2, \lambda_3\}$ are all positive.

— *Computing translation* —

7.* Compute the translation vectors $\boldsymbol{t}_\kappa = (t_{x\kappa}, t_{y\kappa}, t_{z\kappa})^\top$.

— *Computing rotation* —

[4]The mark $*$ indicates that the computation depends the assumed camera model.

8. Compute the following $2M$-D vectors:

$$\boldsymbol{m}_i = \sqrt{\lambda_i} \begin{pmatrix} (\boldsymbol{u}^\dagger_{1(1)}, \boldsymbol{v}_i) \\ (\boldsymbol{u}^\dagger_{1(2)}, \boldsymbol{v}_i) \\ (\boldsymbol{u}^\dagger_{2(1)}, \boldsymbol{v}_i) \\ \vdots \\ (\boldsymbol{u}^\dagger_{M(2)}, \boldsymbol{v}_i) \end{pmatrix}, \quad i = 1, 2, 3. \tag{16}$$

9. Let $\boldsymbol{M}$ be the $2M \times 3$ motion matrix having $\boldsymbol{m}_1$, $\boldsymbol{m}_2$, and $\boldsymbol{m}_3$ as its columns, and let $\boldsymbol{m}^\dagger_{\kappa(a)}$ be the the $(2(\kappa-1)+a)$th column of $\boldsymbol{M}^\top$, $\kappa = 1, ..., M$, $a = 1, 2$.

10.* Compute the rotations $\{\boldsymbol{R}_\kappa\}$.

— *Optimizing the shape* —

11.* Recompute the motion matrix $\boldsymbol{M}$ as follows:

$$\boldsymbol{M} = \sum_{\kappa=1}^{M} \Pi_\kappa^\top \boldsymbol{R}_\kappa. \tag{17}$$

\# $\Pi_\kappa = (\Pi_{\kappa(ij)})$ is a $3 \times 2M$ matrix that depends on the assumed camera model.

12. Compute the 3-D shape vectors $\{\boldsymbol{s}_\alpha\}$ as follows:

$$\boldsymbol{s}_\alpha = (\boldsymbol{M}^\top \boldsymbol{M})^{-1} \boldsymbol{M}^\top (\boldsymbol{p}_\alpha - \boldsymbol{p}_C). \tag{18}$$

— *Computing the mirror image* —

13.* Compute $\{\boldsymbol{s}'_\alpha\}$ and $\{\boldsymbol{R}'_\kappa\}$ as follows:

$$\boldsymbol{s}'_\alpha = -\boldsymbol{s}_\alpha, \quad \boldsymbol{R}'_\kappa = \boldsymbol{\Omega}_\kappa \boldsymbol{R}_\kappa. \tag{19}$$

\# $\boldsymbol{\Omega}_\kappa$ is a rotation matrix that depends on the assumed camera model.

— *Computing the 3-D positions* —

14. Compute the two sets of solutions $\{\hat{\boldsymbol{r}}_\alpha\}$, $\{\hat{\boldsymbol{r}}'_\alpha\}$ as follows:

$$\hat{\boldsymbol{r}}_\alpha = \frac{Z_c}{t_{z1}}(\boldsymbol{R}_1 \boldsymbol{s}_\alpha + \boldsymbol{t}_1), \quad \hat{\boldsymbol{r}}'_\alpha = \frac{Z_c}{t_{z1}}(\boldsymbol{R}'_1 \boldsymbol{s}'_\alpha + \boldsymbol{t}_1). \tag{20}$$

7. Subprocedures

The steps with the mark $*$ in the preceding section depend on the assumed camera model. Here, we consider three projection models and give corresponding subprocedures.

7.1 Orthographic projection

By *orthographic projection* (Fig. 1), a point (X, Y, Z) in the scene is projected onto a point (x, y) in the image such that

$$\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} X \\ Y \end{pmatrix}. \tag{21}$$

The steps with $*$ are computed as follows (the derivation is given in the appendix).

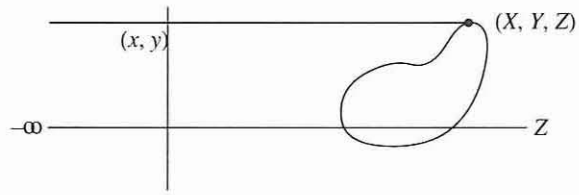

Figure 1: Orthographic projection.

Computing the metric matrix

1. Define the following $3 \times 3 \times 3 \times 3$ tensor $\mathcal{B} = (B_{ijkl})$:

$$\begin{aligned} B_{ijkl} = \sum_{\kappa=1}^{M} \Big[& (\boldsymbol{u}^\dagger_{\kappa(1)})_i (\boldsymbol{u}^\dagger_{\kappa(1)})_j (\boldsymbol{u}^\dagger_{\kappa(1)})_k (\boldsymbol{u}^\dagger_{\kappa(1)})_l \\ & + (\boldsymbol{u}^\dagger_{\kappa(2)})_i (\boldsymbol{u}^\dagger_{\kappa(2)})_j (\boldsymbol{u}^\dagger_{\kappa(2)})_k (\boldsymbol{u}^\dagger_{\kappa(2)})_l \\ & + \frac{1}{4} \Big((\boldsymbol{u}^\dagger_{\kappa(1)})_i (\boldsymbol{u}^\dagger_{\kappa(2)})_j + (\boldsymbol{u}^\dagger_{\kappa(2)})_i (\boldsymbol{u}^\dagger_{\kappa(1)})_j \Big) \\ & \Big((\boldsymbol{u}^\dagger_{\kappa(1)})_k (\boldsymbol{u}^\dagger_{\kappa(2)})_l + (\boldsymbol{u}^\dagger_{\kappa(2)})_k (\boldsymbol{u}^\dagger_{\kappa(1)})_l \Big) \Big]. \end{aligned} \tag{22}$$

\# $(\boldsymbol{u}^\dagger_{\kappa(a)})_i$ denotes the ith component of the 3-D vector $\boldsymbol{u}^\dagger_{\kappa(a)}$.

2. Define the following 6×6 symmetric matrix $\boldsymbol{B}$:

$$\boldsymbol{B} = \begin{pmatrix} B_{1111} & B_{1122} & B_{1133} \\ B_{2211} & B_{2222} & B_{2233} \\ B_{3311} & B_{3322} & B_{3333} \\ \sqrt{2}B_{2311} & \sqrt{2}B_{2322} & \sqrt{2}B_{2333} \\ \sqrt{2}B_{3111} & \sqrt{2}B_{3122} & \sqrt{2}B_{3133} \\ \sqrt{2}B_{1211} & \sqrt{2}B_{1222} & \sqrt{2}B_{1233} \\ \sqrt{2}B_{1123} & \sqrt{2}B_{1131} & \sqrt{2}B_{1112} \\ \sqrt{2}B_{2223} & \sqrt{2}B_{2231} & \sqrt{2}B_{2212} \\ \sqrt{2}B_{3323} & \sqrt{2}B_{3331} & \sqrt{2}B_{3312} \\ 2B_{2323} & 2B_{2331} & 2B_{2312} \\ 2B_{3123} & 2B_{3131} & 2B_{3112} \\ 2B_{1223} & 2B_{1231} & 2B_{1212} \end{pmatrix}. \tag{23}$$

3. Compute the following 6-D vector $\boldsymbol{c}$:

$$\boldsymbol{c} = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \end{pmatrix}^\top. \tag{24}$$

4. Compute the 6-D vector $\boldsymbol{\tau} = (\tau_i)$ by solving the following linear equation:

$$\boldsymbol{B}\boldsymbol{\tau} = \boldsymbol{c}. \tag{25}$$

5. Compute the metric matrix $\boldsymbol{T}$ as follows:

$$\boldsymbol{T} = \begin{pmatrix} \tau_1 & \tau_6/\sqrt{2} & \tau_5/\sqrt{2} \\ \tau_6/\sqrt{2} & \tau_2 & \tau_4/\sqrt{2} \\ \tau_5/\sqrt{2} & \tau_4/\sqrt{2} & \tau_3 \end{pmatrix}. \tag{26}$$

Computing translation

1. Let $t_{z\kappa} = Z_c$, $\kappa = 1, ..., 2M$.
2. Let $t_{x\kappa} = \tilde{t}_{x\kappa}$ and $t_{y\kappa} = \tilde{t}_{y\kappa}$, $\kappa = 1, ..., 2M$.

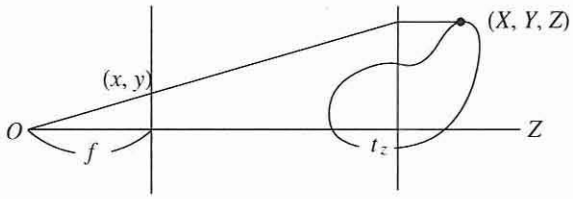

Figure 2: Weak perspective projection.

Computing rotation

1. For each κ, compute the following SVD:

$$\begin{pmatrix} \boldsymbol{m}^{\dagger}_{\kappa(1)} & \boldsymbol{m}^{\dagger}_{\kappa(2)} & \boldsymbol{0} \end{pmatrix} = \boldsymbol{V}_{\kappa}\boldsymbol{\Lambda}_{\kappa}\boldsymbol{U}_{\kappa}^{\top}. \quad (27)$$

2. Compute the rotation matrices $\{\boldsymbol{R}_{\kappa}\}$ as follows:

$$\boldsymbol{R}_{\kappa} = \boldsymbol{U}_{\kappa}\mathrm{diag}(1, 1, \det(\boldsymbol{V}_{\kappa}\boldsymbol{U}_{\kappa}^{\top}))\boldsymbol{V}_{\kappa}^{\top} \quad (28)$$

Matrices Π_{κ} and $\boldsymbol{\Omega}_{\kappa}$

$$\Pi_{\kappa} = \begin{pmatrix} 0 & \cdots & 0 & \overset{(2\kappa-1)}{1} & \overset{(2\kappa)}{0} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \end{pmatrix}, \quad (29)$$

$$\boldsymbol{\Omega}_{\kappa} = \mathrm{diag}(-1, -1, 1). \quad (30)$$

7.2 Weak perspective projection

By *weak perspective projection* (Fig. 2) [8], a point (X, Y, Z) in the scene is projected onto a point (x, y) in the image such that[5]

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{f}{t_z}\begin{pmatrix} X \\ Y \end{pmatrix}, \quad (31)$$

where f is the focal length of the camera and t_z is the third component of the origin $\boldsymbol{t}$ of the scene coordinate system.

If weak perspective projection is assumed, the focal lengths $\{f_{\kappa}\}$, $\kappa = 1, ..., M$, for all the frames need to be specified. If they are unknown, they can be arbitrarily assigned; the reconstructed shape does not depend on their values. This is because the depth t_z is indeterminate and is normalized to be the user-assigned value Z_c in the end. The steps with $*$ are computed as follows (the derivation is given in the appendix).

Computing the metric matrix

1. Compute the following $3 \times 3 \times 3 \times 3$ tensor $\mathcal{B} = (B_{ijkl})$:

$$\begin{aligned} B_{ijkl} = \sum_{\kappa=1}^{M} \Big[& (\boldsymbol{u}^{\dagger}_{\kappa(1)})_i(\boldsymbol{u}^{\dagger}_{\kappa(1)})_j(\boldsymbol{u}^{\dagger}_{\kappa(1)})_k(\boldsymbol{u}^{\dagger}_{\kappa(1)})_l \\ & -(\boldsymbol{u}^{\dagger}_{\kappa(1)})_i(\boldsymbol{u}^{\dagger}_{\kappa(1)})_j(\boldsymbol{u}^{\dagger}_{\kappa(2)})_k(\boldsymbol{u}^{\dagger}_{\kappa(2)})_l \\ & -(\boldsymbol{u}^{\dagger}_{\kappa(2)})_i(\boldsymbol{u}^{\dagger}_{\kappa(2)})_j(\boldsymbol{u}^{\dagger}_{\kappa(1)})_k(\boldsymbol{u}^{\dagger}_{\kappa(1)})_l \\ & +(\boldsymbol{u}^{\dagger}_{\kappa(2)})_i(\boldsymbol{u}^{\dagger}_{\kappa(2)})_j(\boldsymbol{u}^{\dagger}_{\kappa(2)})_k(\boldsymbol{u}^{\dagger}_{\kappa(2)})_l \\ & +\frac{1}{4}\Big((\boldsymbol{u}^{\dagger}_{\kappa(1)})_i(\boldsymbol{u}^{\dagger}_{\kappa(2)})_j(\boldsymbol{u}^{\dagger}_{\kappa(1)})_k(\boldsymbol{u}^{\dagger}_{\kappa(2)})_l \\ & +(\boldsymbol{u}^{\dagger}_{\kappa(2)})_i(\boldsymbol{u}^{\dagger}_{\kappa(1)})_j(\boldsymbol{u}^{\dagger}_{\kappa(1)})_k(\boldsymbol{u}^{\dagger}_{\kappa(2)})_l \\ & +(\boldsymbol{u}^{\dagger}_{\kappa(1)})_i(\boldsymbol{u}^{\dagger}_{\kappa(2)})_j(\boldsymbol{u}^{\dagger}_{\kappa(2)})_k(\boldsymbol{u}^{\dagger}_{\kappa(1)})_l \\ & +(\boldsymbol{u}^{\dagger}_{\kappa(2)})_i(\boldsymbol{u}^{\dagger}_{\kappa(1)})_j(\boldsymbol{u}^{\dagger}_{\kappa(2)})_k(\boldsymbol{u}^{\dagger}_{\kappa(1)})_l\Big)\Big]. \end{aligned} \quad (32)$$

2. Compute the 6×6 symmetric matrix $\boldsymbol{B}$ in eq. (23).
3. Compute the 6-D unit eigenvector $\boldsymbol{\tau} = (\tau_i)$ of $\boldsymbol{B}$ for the smallest eigenvalue.
4. Compute the metric matrix $\boldsymbol{T}$ in eq. (26).
5. If $\det \boldsymbol{T} < 0$, let $\boldsymbol{T} \leftarrow -\boldsymbol{T}$.

Computing translation

1. Compute $t_{z\kappa}$ as follows:

$$t_{z\kappa} = f_{\kappa}\sqrt{\frac{2}{(\boldsymbol{u}^{\dagger}_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^{\dagger}_{\kappa(1)}) + (\boldsymbol{u}^{\dagger}_{\kappa(2)}, \boldsymbol{T}\boldsymbol{u}^{\dagger}_{\kappa(2)})}}. \quad (33)$$

2. Compute $t_{x\kappa}$ and $t_{y\kappa}$ as follows:

$$t_{x\kappa} = \frac{t_{z\kappa}}{f_{\kappa}}\tilde{t}_{x\kappa}, \quad t_{y\kappa} = \frac{t_{z\kappa}}{f_{\kappa}}\tilde{t}_{y\kappa}. \quad (34)$$

Computing rotation

1. For each κ, compute the following SVD:

$$\frac{t_{z\kappa}}{f_{\kappa}}\begin{pmatrix} \boldsymbol{m}^{\dagger}_{\kappa(1)} & \boldsymbol{m}^{\dagger}_{\kappa(2)} & \boldsymbol{0} \end{pmatrix} = \boldsymbol{V}_{\kappa}\boldsymbol{\Lambda}_{\kappa}\boldsymbol{U}_{\kappa}^{\top}. \quad (35)$$

2. Compute the rotation matrices $\{\boldsymbol{R}_{\kappa}\}$ by eq. (28).

Matrices Π_{κ} and $\boldsymbol{\Omega}_{\kappa}$

$$\Pi_{\kappa} = \frac{f_{\kappa}}{t_{z\kappa}}\begin{pmatrix} 0 & \cdots & 0 & \overset{(2\kappa-1)}{1} & \overset{(2\kappa)}{0} & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 \end{pmatrix}, \quad (36)$$

$$\boldsymbol{\Omega}_{\kappa} = \mathrm{diag}(-1, -1, 1). \quad (37)$$

7.3 Paraperspective projection

By *paraperspective projection* (Fig. 3) [8], a point (X, Y, Z) in the scene is projected onto a point (x, y) in the image such that[6]

$$\begin{pmatrix} x \\ y \end{pmatrix} = \frac{f}{t_z}\left(\begin{pmatrix} X \\ Y \end{pmatrix} + \left(1 - \frac{Z}{t_z}\right)\begin{pmatrix} t_x \\ t_y \end{pmatrix}\right), \quad (38)$$

where t_x, t_y, and t_z are the tree components of the origin $\boldsymbol{t}$ of the scene coordinate system.

[5] We assume that the principal point (the point that corresponds to the camera optical axis) is specified, typically at the center of the image frame, and take it as the coordinate origin. This is because we are approximating the perspective projection. See footnote 1.

[6] As in the case of weak perspective projection, we assume that the principal point is specified, typically at the center of the image frame, and take it as the coordinate origin.

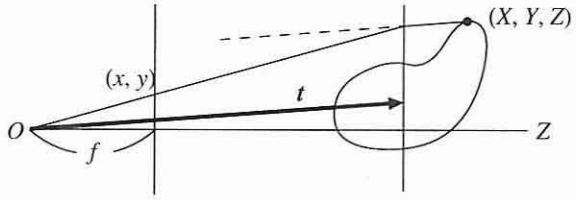

Figure 3: Paraperspective projection.

If paraperspective projection is assumed, the focal lengths $\{f_\kappa\}$, $\kappa = 1, ..., M$, for all the frames need to be specified, but they can be assigned arbitrarily if they are unknown. The reconstructed shape does not depend on their values as in the case of weak perspective projection. The steps with $*$ are computed as follows (the derivation is given in the appendix).

Computing the metric matrix

1. Let

$$\alpha_\kappa = \frac{1}{1+\tilde{t}_{x\kappa}^2/f_\kappa^2}, \quad \beta_\kappa = \frac{1}{1+\tilde{t}_{y\kappa}^2/f_\kappa^2}, \quad \gamma_\kappa = \frac{\tilde{t}_{x\kappa}\tilde{t}_{y\kappa}}{f_\kappa^2}, \quad \kappa = 1, ..., M. \tag{39}$$

2. Define the following $3 \times 3 \times 3 \times 3$ tensor $\mathcal{B} = (B_{ijkl})$:

$$\begin{aligned} B_{ijkl} = \sum_{\kappa=1}^{M} \Big[& (\gamma_\kappa^2+1)\alpha_\kappa^2 (\boldsymbol{u}^\dagger_{\kappa(1)})_i (\boldsymbol{u}^\dagger_{\kappa(1)})_j (\boldsymbol{u}^\dagger_{\kappa(1)})_k (\boldsymbol{u}^\dagger_{\kappa(1)})_l \\ & +(\gamma_\kappa^2+1)\beta_\kappa^2 (\boldsymbol{u}^\dagger_{\kappa(2)})_i (\boldsymbol{u}^\dagger_{\kappa(2)})_j (\boldsymbol{u}^\dagger_{\kappa(2)})_k (\boldsymbol{u}^\dagger_{\kappa(2)})_l \\ & +(\boldsymbol{u}^\dagger_{\kappa(1)})_i (\boldsymbol{u}^\dagger_{\kappa(2)})_j (\boldsymbol{u}^\dagger_{\kappa(1)})_k (\boldsymbol{u}^\dagger_{\kappa(2)})_l \\ & +(\boldsymbol{u}^\dagger_{\kappa(1)})_i (\boldsymbol{u}^\dagger_{\kappa(2)})_j (\boldsymbol{u}^\dagger_{\kappa(2)})_k (\boldsymbol{u}^\dagger_{\kappa(1)})_l \\ & +(\boldsymbol{u}^\dagger_{\kappa(2)})_i (\boldsymbol{u}^\dagger_{\kappa(1)})_j (\boldsymbol{u}^\dagger_{\kappa(1)})_k (\boldsymbol{u}^\dagger_{\kappa(2)})_l \\ & +(\boldsymbol{u}^\dagger_{\kappa(2)})_i (\boldsymbol{u}^\dagger_{\kappa(1)})_j (\boldsymbol{u}^\dagger_{\kappa(2)})_k (\boldsymbol{u}^\dagger_{\kappa(1)})_l \\ & -\alpha_\kappa\gamma_\kappa (\boldsymbol{u}^\dagger_{\kappa(1)})_i (\boldsymbol{u}^\dagger_{\kappa(1)})_j (\boldsymbol{u}^\dagger_{\kappa(1)})_k (\boldsymbol{u}^\dagger_{\kappa(2)})_l \\ & -\alpha_\kappa\gamma_\kappa (\boldsymbol{u}^\dagger_{\kappa(1)})_i (\boldsymbol{u}^\dagger_{\kappa(1)})_j (\boldsymbol{u}^\dagger_{\kappa(2)})_k (\boldsymbol{u}^\dagger_{\kappa(1)})_l \\ & -\alpha_\kappa\gamma_\kappa (\boldsymbol{u}^\dagger_{\kappa(1)})_i (\boldsymbol{u}^\dagger_{\kappa(2)})_j (\boldsymbol{u}^\dagger_{\kappa(1)})_k (\boldsymbol{u}^\dagger_{\kappa(1)})_l \\ & -\alpha_\kappa\gamma_\kappa (\boldsymbol{u}^\dagger_{\kappa(2)})_i (\boldsymbol{u}^\dagger_{\kappa(1)})_j (\boldsymbol{u}^\dagger_{\kappa(1)})_k (\boldsymbol{u}^\dagger_{\kappa(1)})_l \\ & -\beta_\kappa\gamma_\kappa (\boldsymbol{u}^\dagger_{\kappa(2)})_i (\boldsymbol{u}^\dagger_{\kappa(2)})_j (\boldsymbol{u}^\dagger_{\kappa(1)})_k (\boldsymbol{u}^\dagger_{\kappa(2)})_l \\ & -\beta_\kappa\gamma_\kappa (\boldsymbol{u}^\dagger_{\kappa(2)})_i (\boldsymbol{u}^\dagger_{\kappa(2)})_j (\boldsymbol{u}^\dagger_{\kappa(2)})_k (\boldsymbol{u}^\dagger_{\kappa(1)})_l \\ & -\beta_\kappa\gamma_\kappa (\boldsymbol{u}^\dagger_{\kappa(1)})_i (\boldsymbol{u}^\dagger_{\kappa(2)})_j (\boldsymbol{u}^\dagger_{\kappa(2)})_k (\boldsymbol{u}^\dagger_{\kappa(2)})_l \\ & -\beta_\kappa\gamma_\kappa (\boldsymbol{u}^\dagger_{\kappa(2)})_i (\boldsymbol{u}^\dagger_{\kappa(1)})_j (\boldsymbol{u}^\dagger_{\kappa(2)})_k (\boldsymbol{u}^\dagger_{\kappa(2)})_l \\ & +(\gamma_\kappa^2-1)\alpha_\kappa\gamma_\kappa (\boldsymbol{u}^\dagger_{\kappa(1)})_i (\boldsymbol{u}^\dagger_{\kappa(1)})_j (\boldsymbol{u}^\dagger_{\kappa(2)})_k (\boldsymbol{u}^\dagger_{\kappa(2)})_l \\ & +(\gamma_\kappa^2-1)\alpha_\kappa\gamma_\kappa (\boldsymbol{u}^\dagger_{\kappa(2)})_i (\boldsymbol{u}^\dagger_{\kappa(2)})_j (\boldsymbol{u}^\dagger_{\kappa(1)})_k (\boldsymbol{u}^\dagger_{\kappa(1)})_l \Big]. \end{aligned} \tag{40}$$

3. Compute the 6×6 symmetric matrix $\boldsymbol{B}$ in eq. (23).
4. Compute the 6-D unit eigenvector $\boldsymbol{\tau} = (\tau_i)$ of $\boldsymbol{B}$ for the smallest eigenvalue.
5. Compute the metric matrix $\boldsymbol{T}$ in eq. (26).
6. If $\det \boldsymbol{T} < 0$, let $\boldsymbol{T} \leftarrow -\boldsymbol{T}$.

Computing translation

1. Compute $t_{z\kappa}$ as follows:

$$t_{z\kappa} = f_\kappa \sqrt{\frac{2}{\alpha_\kappa(\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(1)}) + \beta_\kappa(\boldsymbol{u}^\dagger_{\kappa(2)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(2)})}} \tag{41}$$

2. Compute $t_{x\kappa}$ and $t_{y\kappa}$ by eqs. (34).

Computing rotation

1. For each κ, compute the vectors $\{\boldsymbol{r}^\dagger_{\kappa(1)}, \boldsymbol{r}^\dagger_{\kappa(2)}, \boldsymbol{r}^\dagger_{\kappa(3)}\}$ as follows:

$$\begin{aligned} \boldsymbol{r}^\dagger_{\kappa(3)} &= \frac{t_{z\kappa}}{f_\kappa\Big(1+(t_{x\kappa}/t_{z\kappa})^2+(t_{y\kappa}/t_{z\kappa})^2\Big)} \Big(\frac{t_{z\kappa}}{f_\kappa}\boldsymbol{m}^\dagger_{\kappa(1)} \times \boldsymbol{m}^\dagger_{\kappa(2)} - \frac{t_{x\kappa}}{t_{z\kappa}}\boldsymbol{m}^\dagger_{\kappa(1)} - \frac{t_{y\kappa}}{t_{z\kappa}}\boldsymbol{m}^\dagger_{\kappa(2)}\Big), \\ \boldsymbol{r}^\dagger_{\kappa(1)} &= \frac{t_{z\kappa}}{f_\kappa}\boldsymbol{m}^\dagger_{\kappa(1)} + \frac{t_{x\kappa}}{t_{z\kappa}}\boldsymbol{r}^\dagger_{\kappa(3)}, \\ \boldsymbol{r}^\dagger_{\kappa(2)} &= \frac{t_{z\kappa}}{f_\kappa}\boldsymbol{m}^\dagger_{\kappa(2)} + \frac{t_{y\kappa}}{t_{z\kappa}}\boldsymbol{r}^\dagger_{\kappa(3)}. \end{aligned} \tag{42}$$

2. For each κ, compute the following SVD:

$$\begin{pmatrix} \boldsymbol{r}^\dagger_{\kappa(1)} & \boldsymbol{r}^\dagger_{\kappa(2)} & \boldsymbol{r}^\dagger_{\kappa(3)} \end{pmatrix} = \boldsymbol{V}_\kappa \boldsymbol{\Lambda}_\kappa \boldsymbol{U}_\kappa^\top. \tag{43}$$

3. Compute the rotation matrices $\{\boldsymbol{R}_\kappa\}$ by eq. (28).

Matrices Π_κ and $\boldsymbol{\Omega}_\kappa$

$$\Pi_\kappa = \frac{f_\kappa}{t_{z\kappa}} \begin{pmatrix} & & \overset{(2\kappa-1)}{} & \overset{(2\kappa)}{} & \\ 0 \cdots 0 & & 1 & 0 & 0 \cdots 0 \\ 0 \cdots 0 & & 0 & 1 & 0 \cdots 0 \\ 0 \cdots 0 & & -t_{x\kappa}/t_{z\kappa} & -t_{y\kappa}/t_{z\kappa} & 0 \cdots 0 \end{pmatrix}, \tag{44}$$

$$\boldsymbol{\Omega}_\kappa = \frac{2\boldsymbol{t}_\kappa\boldsymbol{t}_\kappa^\top}{\|\boldsymbol{t}_\kappa\|^2} - \boldsymbol{I}. \tag{45}$$

8. Concluding Remarks

We have pointed out that the principle of the Tomasi-Kanade factorization is the affine camera approximation to the imaging geometry and that matrix factorization by SVD is nothing but a numerical means for computing a least-squares solution. We have clarified the geometric structure of the problem and given a complete computational procedure[7] for 3-D reconstruction based on orthographic projection, weak perspective projection, and paraperspective projection. The derivation of the procedure is summarized in the appendix.

[7]The C program source is available at:
`http://www.suri.it.okayama-u.ac.jp/e-program.html`

Acknowledgments. This work was supported in part by the Ministry of Education, Culture, Sports, Science and Technology, Japan, under a Grant in Aid for Scientific Research C(2) (No. 15500113), the Support Center for Advanced Telecommunications Technology Research, and Kayamori Foundation of Informational Science Advancement. The authors thank Dr. Jun Fujiki of the National Institute of Advanced Industrial Science and Technology, Japan, for helpful comments and our graduate student, Mr. Takayuki Tsubouchi, for doing experiments for us.

References

[1] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, Cambridge, U.K., 2000.
[2] K. Kanatani, *Geometric Computation for Machine Vision*, Oxford University Press, Oxford, U.K., 1993.
[3] K. Kanatani, *Statistical Optimization for Geometric Computation: Theory and Practice*, Elsevier, Amsterdam, The Netherlands, 1996.
[4] K. Kanatani, Motion segmentation by subspace separation and model selection, *Proc. 8th Int. Conf. Comput. Vision*, July 2001, Vancouver, Canada, Vol. 2, pp. 301–306.
[5] K. Kanatani, Evaluation and selection of models for motion segmentation, *Proc. 7th Euro. Conf. Comput. Vision*, Copenhagen, Denmark, June 2002, pp. 335–349.
[6] K. Kanatani and C. Matsunaga, Estimating the number of independent motions for multibody segmentation, *Proc. 5th Asian Conf. Comput. Vision*, January 2002, Melbourne, Australia, Vol. 1, pp. 7–12.
[7] K. Kanatani, Motion segmentation by subspace separation: Model selection and reliability evaluation, *Int. J. Image Graphics*, **2**-2 (2002-4), 179–197.
[8] C. J. Poelman and T. Kanade, A paraperspective factorization method for shape and motion recovery, *IEEE Trans. Patt. Anal. Mach. Intell.*, **19**-3 (1997-3), 206–218.
[9] Y. Sugaya and K. Kanatani, Automatic camera model selection for multibody motion segmentation, *Proc. IAPR Workshop on Machine Vision Applications*, December 2002, Nara, Japan, pp. 412–415.
[10] Y. Sugaya and K. Kanatani, Outlier removal for motion tracking by subspace separation, *IEICE Trans. Inf. & Syst.*, **E86-D**-6 (2003-6), 1095–1102.
[11] Y. Sugaya and K. Kanatani, Extending interrupted feature point tracking for 3-D affine reconstruction, *IEICE Trans. Inf. & Syst.*, **E87-D** (2004), to appear.
[12] C. Tomasi and T. Kanade, Shape and motion from image streams under orthography—A factorization method, *Int. J. Comput. Vision*, **9**-2 (1992-10), 137–154.

Appendix: Derivations

A. Orthographic Projection

The orthographic projection equation (21) can be rewritten as

$$\begin{pmatrix} x \\ y \end{pmatrix} = \boldsymbol{\Pi} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}, \tag{46}$$

where $\boldsymbol{\Pi}$ is the following 2×3 projection matrix:

$$\boldsymbol{\Pi} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}. \tag{47}$$

Computing the metric matrix

In terms of the projection matrix (47), vectors $\tilde{\boldsymbol{m}}_{1\kappa}$, $\tilde{\boldsymbol{m}}_{2\kappa}$, and $\tilde{\boldsymbol{m}}_{3\kappa}$ in eq. (4) are written as follows:

$$\tilde{\boldsymbol{m}}_{1\kappa} = \boldsymbol{\Pi}\boldsymbol{i}_\kappa, \quad \tilde{\boldsymbol{m}}_{2\kappa} = \boldsymbol{\Pi}\boldsymbol{j}_\kappa, \quad \tilde{\boldsymbol{m}}_{3\kappa} = \boldsymbol{\Pi}\boldsymbol{k}_\kappa. \tag{48}$$

Let

$$\boldsymbol{R}_\kappa = \begin{pmatrix} \boldsymbol{i}_\kappa & \boldsymbol{j}_\kappa & \boldsymbol{k}_\kappa \end{pmatrix} \tag{49}$$

be the matrix having the orthonormal system $\{\boldsymbol{i}_\kappa, \boldsymbol{j}_\kappa, \boldsymbol{k}_\kappa\}$ as columns. This is a rotation matrix that describes the orientation of the scene coordinate system at time κ. Using this, we can write eq. (48) as follows:

$$\begin{pmatrix} \tilde{\boldsymbol{m}}_{1\kappa} & \tilde{\boldsymbol{m}}_{2\kappa} & \tilde{\boldsymbol{m}}_{3\kappa} \end{pmatrix} = \boldsymbol{\Pi}\boldsymbol{R}_\kappa. \tag{50}$$

Since $\boldsymbol{R}_\kappa$ is an orthogonal matrix, we have

$$\begin{aligned} &\begin{pmatrix} \tilde{\boldsymbol{m}}_{1\kappa} & \tilde{\boldsymbol{m}}_{2\kappa} & \tilde{\boldsymbol{m}}_{3\kappa} \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{m}}_{1\kappa} & \tilde{\boldsymbol{m}}_{2\kappa} & \tilde{\boldsymbol{m}}_{3\kappa} \end{pmatrix}^\top \\ &\quad = \boldsymbol{\Pi}\boldsymbol{R}_\kappa\boldsymbol{R}_\kappa^\top\boldsymbol{\Pi}^\top = \boldsymbol{\Pi}\boldsymbol{\Pi}^\top = \boldsymbol{I}. \end{aligned} \tag{51}$$

In terms of the vectors $\boldsymbol{m}^\dagger_{\kappa(a)}$ defined in Step 9 of the main procedure, the above equation can be written as

$$\begin{pmatrix} \boldsymbol{m}^\dagger_{\kappa(1)} & \boldsymbol{m}^\dagger_{\kappa(2)} \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{m}^\dagger_{\kappa(1)} & \boldsymbol{m}^\dagger_{\kappa(2)} \end{pmatrix} = \boldsymbol{I}. \tag{52}$$

Equating individual elements on both sides, we obtain

$$\|\boldsymbol{m}^\dagger_{\kappa(1)}\|^2 = \|\boldsymbol{m}^\dagger_{\kappa(2)}\|^2 = 1, \quad (\boldsymbol{m}^\dagger_{\kappa(1)}, \boldsymbol{m}^\dagger_{\kappa(2)}) = 0. \tag{53}$$

Since $\boldsymbol{M}^\top = \boldsymbol{A}^\top\boldsymbol{U}^\top$ from eq. (15), we obtain

$$\boldsymbol{m}^\dagger_{\kappa(a)} = \boldsymbol{A}^\top\boldsymbol{u}^\dagger_{\kappa(a)}, \tag{54}$$

where $\boldsymbol{u}^\dagger_{\kappa(a)}$ are the vectors defined in Step 4 of the main procedure. Substituting eq. (54) into eqs. (53), we obtain

$$\begin{aligned} &(\boldsymbol{A}^\top\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{A}^\top\boldsymbol{u}^\dagger_{\kappa(1)}) = (\boldsymbol{A}^\top\boldsymbol{u}^\dagger_{\kappa(2)}, \boldsymbol{A}^\top\boldsymbol{u}^\dagger_{\kappa(2)}) = 1, \\ &(\boldsymbol{A}^\top\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{A}^\top\boldsymbol{u}^\dagger_{\kappa(2)}) = 0. \end{aligned} \tag{55}$$

In terms of the metric matrix $\boldsymbol{T}$ (eq. (11)), these are rewritten as

$$\begin{aligned} &(\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(1)}) = (\boldsymbol{u}^\dagger_{\kappa(2)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(2)}) = 1, \\ &(\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(2)}) = 0. \end{aligned} \tag{56}$$

So, we compute the matrix $\boldsymbol{T}$ that minimizes

$$\begin{aligned} K = \sum_{\kappa=1}^{M} \Big[&\Big((\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(1)}) - 1 \Big)^2 \\ &+ \Big((\boldsymbol{u}^\dagger_{\kappa(2)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(2)}) - 1 \Big)^2 + (\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(2)})^2 \Big]. \end{aligned} \tag{57}$$

Differentiating this with respect to T_{ij} and setting the result 0, we obtain the simultaneous linear equations

$$\sum_{i,j,k,l=1}^{3} B_{ijkl} T_{kl} = \delta_{ij}, \tag{58}$$

where the tensor $\mathcal{B} = (B_{ijkl})$ is defined by eq. (23). In terms of the 6×6 symmetric matrix $\boldsymbol{B}$ given in eq. (23), the 6-D vector $\boldsymbol{\tau}$ defined via eq. (26), and the 6-D vector $\boldsymbol{c}$ defined by eq. (24), the above equation can be expressed in the form given in eq. (25). Its solution $\boldsymbol{\tau}$ determines the matrix $\boldsymbol{T}$ in the form of eq. (26).

Computing translation

The depth information is lost through orthographic projection, so we place the origin $\boldsymbol{t}_\kappa = (t_{x\kappa}, t_{y\kappa}, t_{z\kappa})^\top$ of the scene coordinate system in the distance Z_c specified by the user from the XY plane. We also identify $(t_{x\kappa}, t_{y\kappa})$ with the centroid of the projected feature positions, namely $(\tilde{t}_{x\kappa}, \tilde{t}_{y\kappa})$.

Computing rotation

The metric matrix $\boldsymbol{T}$ is expressed in terms of its eigenvalues $\{\lambda_1, \lambda_2, \lambda_3\}$ and the corresponding orthonormal system $\{\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{v}_3\}$ of eigenvectors as follows [2]:

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \boldsymbol{v}_3 \end{pmatrix} \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3) \begin{pmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \boldsymbol{v}_3 \end{pmatrix}^\top. \tag{59}$$

Since this equals $\boldsymbol{A}\boldsymbol{A}^\top$, we have

$$\begin{aligned} \boldsymbol{A} &= \pm \begin{pmatrix} \boldsymbol{v}_1 & \boldsymbol{v}_2 & \boldsymbol{v}_3 \end{pmatrix} \mathrm{diag}(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \sqrt{\lambda_3}) \boldsymbol{Q} \\ &= \pm \begin{pmatrix} \sqrt{\lambda_1}\boldsymbol{v}_1 & \sqrt{\lambda_2}\boldsymbol{v}_2 & \sqrt{\lambda_3}\boldsymbol{v}_3 \end{pmatrix} \boldsymbol{Q}, \end{aligned} \tag{60}$$

where $\boldsymbol{Q}$ is an arbitrary rotation matrix, which corresponds to the orientation indeterminacy of the scene coordinate system. The double-sign $\pm$ reflects the mirror-image indeterminacy of the solution. Here, we pick out one solution by choosing $+$ and $\boldsymbol{Q} = \boldsymbol{I}$ (the identity). Then, the ith column of the motion matrix $\boldsymbol{M}$ is

$$\boldsymbol{m}_i = \begin{pmatrix} \boldsymbol{u}_{1(1)}^{\dagger\top} \\ \boldsymbol{u}_{1(2)}^{\dagger\top} \\ \boldsymbol{u}_{2(1)}^{\dagger\top} \\ \boldsymbol{u}_{2(2)}^{\dagger\top} \\ \vdots \\ \boldsymbol{u}_{M(2)}^{\dagger\top} \end{pmatrix} \sqrt{\lambda_i}\boldsymbol{v}_i, \tag{61}$$

which is rewritten as eq. (16).

If we transpose both sides of eq. (50) and let $\boldsymbol{r}_{\kappa(i)}^\dagger$ be the ith column of $\boldsymbol{R}_\kappa^\top$, we obtain from the definition of $\boldsymbol{m}_{\kappa(a)}^\dagger$ and eq. (17)

$$\begin{pmatrix} \boldsymbol{m}_{\kappa(1)}^\dagger & \boldsymbol{m}_{\kappa(2)}^\dagger \end{pmatrix} = \boldsymbol{R}_\kappa^\top \boldsymbol{\Pi}^\top = \begin{pmatrix} \boldsymbol{r}_{\kappa(1)}^\dagger & \boldsymbol{r}_{\kappa(2)}^\dagger \end{pmatrix}. \tag{62}$$

Hence, we obtain

$$\boldsymbol{r}_{\kappa(1)}^\dagger = \boldsymbol{m}_{\kappa(1)}^\dagger, \quad \boldsymbol{r}_{\kappa(2)}^\dagger = \boldsymbol{m}_{\kappa(2)}^\dagger. \tag{63}$$

Since $\{\boldsymbol{r}_{\kappa(1)}^\dagger, \boldsymbol{r}_{\kappa(2)}^\dagger, \boldsymbol{r}_{\kappa(3)}^\dagger\}$ is a right-handed orthonormal system, we should theoretically have $\boldsymbol{r}_{\kappa(3)}^\dagger = \boldsymbol{r}_{\kappa(1)}^\dagger \times \boldsymbol{r}_{\kappa(2)}^\dagger$. In the presence of noise in the data, however, the computed vectors $\boldsymbol{r}_{\kappa(1)}^\dagger$ and $\boldsymbol{r}_{\kappa(2)}^\dagger$ may not necessarily be orthogonal or of unit length. So, we fit an exact orthonormal system to them. If we compute the SVD in the form of eq. (27), the matrix $\boldsymbol{V}_\kappa \boldsymbol{U}_\kappa^\top$ is the best approximation to $\boldsymbol{R}_\kappa^\top$ in the sense of least squares [2]. In order to prevent the determinant from being -1, we modify it in the form of eq. (28) [2].

Computing the shape

Combining the matrix Π_κ in eq. (29) and eqs. (49) and (50), we can rewrite the motion matrix $\boldsymbol{M}$ defined in Step 9 of the main procedure in the form of eq. (17). Introducing the 3-D shape vector $\boldsymbol{s}_\alpha = (a_\alpha, b_\alpha, c_\alpha)^\top$, we can write eq. (6) as

$$J = \sum_{\alpha=1}^{N} \|\boldsymbol{p}_\alpha - \boldsymbol{m}_0 - \boldsymbol{M}\boldsymbol{s}_\alpha\|^2. \tag{64}$$

Since $\boldsymbol{m}_0$ is the centroid of $\{\boldsymbol{p}_\alpha\}$, the least-squares solution is given by eq. (18).

Computing the mirror image solution

We are considering a solution corresponding to the sign $+$ in eq. (60). If we choose $-$ instead, the vectors $\boldsymbol{m}_i$ in eq. (16) will change their signs, and hence the motion matrix $\boldsymbol{M}$ and the vectors $\boldsymbol{m}_i$ will also change their signs. If the motion matrix $\boldsymbol{M}$ changes its sign, the shape vectors $\boldsymbol{s}_\alpha$ given by eq. (18) also change their signs.

If the vectors $\tilde{\boldsymbol{m}}_{1\kappa}$, $\tilde{\boldsymbol{m}}_{2\kappa}$, and $\tilde{\boldsymbol{m}}_{3\kappa}$ change their signs, eq. (50) implies that we obtain a mirror image solution $\boldsymbol{R}'_\kappa$ for the rotation such that $\boldsymbol{\Pi}\boldsymbol{R}_\kappa = -\boldsymbol{\Pi}\boldsymbol{R}'_\kappa$. Transposing both sides and letting $\boldsymbol{\Omega}_\kappa = \boldsymbol{R}'_\kappa \boldsymbol{R}_\kappa^\top$, we obtain

$$\boldsymbol{\Omega}_\kappa \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \end{pmatrix}. \tag{65}$$

This means that $\boldsymbol{\Omega}_\kappa$ is a rotation matrix that maps vectors $(1,0,0)^\top$ and $(0,1,0)^\top$ onto $(-1,0,0)^\top$ and $(0,-1,0)^\top$, respectively. So, $\boldsymbol{\Omega}_\kappa = \mathrm{diag}(-1,-1,1)$.

B. Weak Perspective Projection

Weak perspective projection equation (31) can be written in the form of eq. (46) if we define the projection matrix $\boldsymbol{\Pi}$ by

$$\boldsymbol{\Pi} = \frac{f}{t_z} \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}. \tag{66}$$

Computing the metric matrix

For weak perspective projection, the vectors $\tilde{\boldsymbol{m}}_{0\kappa}$, $\tilde{\boldsymbol{m}}_{1\kappa}$, $\tilde{\boldsymbol{m}}_{2\kappa}$, and $\tilde{\boldsymbol{m}}_{3\kappa}$ in eq. (4) can be written as

$$\begin{aligned} \tilde{\boldsymbol{m}}_{0\kappa} &= \boldsymbol{\Pi}_\kappa \boldsymbol{t}_\kappa, \quad & \tilde{\boldsymbol{m}}_{1\kappa} &= \boldsymbol{\Pi}_\kappa \boldsymbol{i}_\kappa, \\ \tilde{\boldsymbol{m}}_{2\kappa} &= \boldsymbol{\Pi}_\kappa \boldsymbol{j}_\kappa, \quad & \tilde{\boldsymbol{m}}_{3\kappa} &= \boldsymbol{\Pi}_\kappa \boldsymbol{k}_\kappa, \end{aligned} \tag{67}$$

where $\mathbf{\Pi}_\kappa$ is the matrix obtained by replacing f and t_z, respectively, by the values f_κ and $t_{z\kappa}$ at time κ in eq. (66). From the definition of the rotation matrix $\boldsymbol{R}_\kappa$ in eq. (49), we obtain

$$\begin{pmatrix} \tilde{\boldsymbol{m}}_{1\kappa} & \tilde{\boldsymbol{m}}_{2\kappa} & \tilde{\boldsymbol{m}}_{3\kappa} \end{pmatrix} = \mathbf{\Pi}_\kappa \boldsymbol{R}_\kappa. \tag{68}$$

Since $\boldsymbol{R}_\kappa$ is an orthogonal matrix, we have

$$\begin{aligned} &\begin{pmatrix} \tilde{\boldsymbol{m}}_{1\kappa} & \tilde{\boldsymbol{m}}_{2\kappa} & \tilde{\boldsymbol{m}}_{3\kappa} \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{m}}_{1\kappa} & \tilde{\boldsymbol{m}}_{2\kappa} & \tilde{\boldsymbol{m}}_{3\kappa} \end{pmatrix}^\top \\ &= \mathbf{\Pi}_\kappa \boldsymbol{R}_\kappa \boldsymbol{R}_\kappa^\top \mathbf{\Pi}_\kappa^\top = \mathbf{\Pi}_\kappa \mathbf{\Pi}_\kappa^\top = \frac{f_\kappa^2}{t_{z\kappa}^2} \boldsymbol{I}. \end{aligned} \tag{69}$$

In terms of the vectors $\boldsymbol{m}^\dagger_{\kappa(a)}$ defined in Step 9 of the main procedure, the above equation can be rewritten as

$$\begin{pmatrix} \boldsymbol{m}^\dagger_{\kappa(1)} & \boldsymbol{m}^\dagger_{\kappa(2)} \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{m}^\dagger_{\kappa(1)} & \boldsymbol{m}^\dagger_{\kappa(2)} \end{pmatrix} = \frac{f_\kappa^2}{t_{z\kappa}^2} \boldsymbol{I}. \tag{70}$$

Equating individual elements on both sides, we obtain

$$\|\boldsymbol{m}^\dagger_{\kappa(1)}\|^2 = \|\boldsymbol{m}^\dagger_{\kappa(2)}\|^2 = \frac{f_\kappa^2}{t_{z\kappa}^2}, \quad (\boldsymbol{m}^\dagger_{\kappa(1)}, \boldsymbol{m}^\dagger_{\kappa(2)}) = 0. \tag{71}$$

Recalling that $\boldsymbol{M}^\top = \boldsymbol{A}^\top \boldsymbol{U}^\top$ from eq. (15) and introducing the vectors $\boldsymbol{u}^\dagger_{\kappa(a)}$ defined in Step 4 of the main procedure, we can express the vector $\boldsymbol{m}^\dagger_{\kappa(a)}$ in the form of eq. (54). Substituting it into eqs. (71), we obtain

$$\begin{aligned} &(\boldsymbol{A}^\top \boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{A}^\top \boldsymbol{u}^\dagger_{\kappa(1)}) = (\boldsymbol{A}^\top \boldsymbol{u}^\dagger_{\kappa(2)}, \boldsymbol{A}^\top \boldsymbol{u}^\dagger_{\kappa(2)}) = \frac{f_\kappa^2}{t_{z\kappa}^2}, \\ &(\boldsymbol{A}^\top \boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{A}^\top \boldsymbol{u}^\dagger_{\kappa(2)}). \end{aligned} \tag{72}$$

In terms of the metric matrix $\boldsymbol{T}$ (eq. (11)), these are rewritten as

$$\begin{aligned} &(\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(1)}) = (\boldsymbol{u}^\dagger_{\kappa(2)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(2)}) = \frac{f_\kappa^2}{t_{z\kappa}^2}, \\ &(\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(2)}) = 0. \end{aligned} \tag{73}$$

So, we compute the matrix $\boldsymbol{T}$ that minimizes

$$\begin{aligned} K = \sum_{\kappa=1}^{M} \Big[\Big(&(\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(1)}) - (\boldsymbol{u}^\dagger_{\kappa(2)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(2)}) \Big)^2 \\ &+ (\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(2)})^2 \Big]. \end{aligned} \tag{74}$$

If we define the tensor $\mathcal{B} = (B_{ijkl})$ by eq. (32), we have

$$K = \sum_{i,j,k,l=1}^{3} B_{ijkl} T_{ij} T_{kl}. \tag{75}$$

Note that eqs. (73) imply that the matrix $\boldsymbol{T}$ has scale indeterminacy. In fact, doubling $\boldsymbol{T}$ means dividing $t_{z\kappa}$ by $\sqrt{2}$. Then, the matrix $\mathbf{\Pi}_\kappa$ in eq. (66) is multiplied by $\sqrt{2}$, so eq. (68) implies that the vectors $\tilde{\boldsymbol{m}}_{1\kappa}$, $\tilde{\boldsymbol{m}}_{2\kappa}$, and $\tilde{\boldsymbol{m}}_{3\kappa}$ are all multiplied by $\sqrt{2}$. We can see from eq. (5), however, that this magnification can be compensated for by dividing a_α, b_α, and c_α by $\sqrt{2}$, resulting in a solution compatible with all the data. So, we do not lose generality if we normalize $\boldsymbol{T}$ into $\|\boldsymbol{T}\| = 1$.

In terms of the 6×6 symmetric matrix $\boldsymbol{B}$ given in eq. (23) and the 6-D vector $\boldsymbol{\tau}$ defined via eq. (26), we can write eq. (75) as the following quadratic form:

$$K = (\boldsymbol{\tau}, \boldsymbol{B}\boldsymbol{\tau}). \tag{76}$$

Since K is a sum of squares, $\boldsymbol{B}$ is a positive semi-definite symmetric matrix. From the definition of the vector $\boldsymbol{\tau}$, the condition $\|\boldsymbol{T}\| = 1$ is equivalent to the condition $\|\boldsymbol{\tau}\| = 1$. So, we compute the unit vector $\boldsymbol{\tau}$ that minimizes the quadratic form K in $\boldsymbol{\tau}$, and the solution is the unit eigenvector of $\boldsymbol{B}$ for the smallest eigenvalue [3]. From it, we obtain the solution $\boldsymbol{T}$ in the form of eq. (26). However, eigenvectors have signature indeterminacy, and the matrix $\boldsymbol{T}$ should be positive semi-definite. So, if $\det \boldsymbol{T} < 0$, we change the sign of $\boldsymbol{T}$.

Computing translation

If the metric matrix $\boldsymbol{T}$ is computed, the first of eqs. (73) gives $t_{z\kappa}$ in the form of eq. (33), which is arranged so as to preserve the symmetry between $\boldsymbol{u}^\dagger_{\kappa(1)}$ and $\boldsymbol{u}^\dagger_{\kappa(2)}$. The remaining components $t_{x\kappa}$ and $t_{y\kappa}$ are determined from the first of eqs. (67) in the form of eqs. (34).

Computing rotation

As in the case of orthographic projection, the ith column of the motion matrix $\boldsymbol{M}$ is given in the form of eq. (16), and the vectors $\boldsymbol{m}^\dagger_{\kappa(a)}$ are determined accordingly.

If we transpose both sides of eq. (68) and let $\boldsymbol{r}^\dagger_{\kappa(i)}$ be the ith column of $\boldsymbol{R}_\kappa^\top$, we obtain from the definition of $\boldsymbol{m}^\dagger_{\kappa(a)}$

$$\begin{pmatrix} \boldsymbol{m}^\dagger_{\kappa(1)} & \boldsymbol{m}^\dagger_{\kappa(2)} \end{pmatrix} = \boldsymbol{R}_\kappa^\top \mathbf{\Pi}_\kappa^\top = \frac{f_\kappa}{t_{z\kappa}} \begin{pmatrix} \boldsymbol{r}^\dagger_{\kappa(1)} & \boldsymbol{r}^\dagger_{\kappa(2)} \end{pmatrix}. \tag{77}$$

Hence, we obtain

$$\boldsymbol{r}^\dagger_{\kappa(1)} = \frac{t_{z\kappa}}{f_\kappa} \boldsymbol{m}^\dagger_{\kappa(1)}, \quad \boldsymbol{r}^\dagger_{\kappa(2)} = \frac{t_{z\kappa}}{f_\kappa} \boldsymbol{m}^\dagger_{\kappa(2)}. \tag{78}$$

Since $\{\boldsymbol{r}^\dagger_{\kappa(1)}, \boldsymbol{r}^\dagger_{\kappa(2)}, \boldsymbol{r}^\dagger_{\kappa(3)}\}$ is a right-handed orthonormal system, we should theoretically have $\boldsymbol{r}^\dagger_{\kappa(3)} = \boldsymbol{r}^\dagger_{\kappa(1)} \times \boldsymbol{r}^\dagger_{\kappa(2)}$. In the presence of noise in the data, however, the computed vectors $\boldsymbol{r}^\dagger_{\kappa(1)}$ and $\boldsymbol{r}^\dagger_{\kappa(2)}$ may not necessarily be orthogonal or of unit length. So, we fit an exact orthonormal system to them. As in the case of orthographic projection, if we compute the SVD in the form of eq. (27), the best approximation to $\boldsymbol{R}_\kappa$ is given by eq. (28) [2].

Computing the shape

Combining the matrix Π_κ in eq. (36) and eqs. (49) and (68), we can rewrite the motion matrix $\boldsymbol{M}$ defined in Step 9 of the main procedure in the form of eq. (17). Introducing the 3-D shape vector $\hat{\boldsymbol{s}}_\alpha = (a_\alpha, b_\alpha, c_\alpha)^\top$, we can write eq. (6) in the form of eq. (64), from which the least-squares solution is given in the form of eq. (18).

Computing the mirror image solution

As in the case of orthographic projection, the solution $\boldsymbol{s}_\alpha$ for the shape vector gives rise to its mirror image solution $-\boldsymbol{s}_\alpha$, for which the vectors $\boldsymbol{m}^\dagger_{\kappa(a)}$ change their signs. Then, eq. (68) implies that we obtain a mirror image solution $\boldsymbol{R}'_\kappa$ for the rotation such that $\boldsymbol{\Pi}_\kappa \boldsymbol{R}_\kappa = -\boldsymbol{\Pi}_\kappa \boldsymbol{R}'_\kappa$. Transposing both sides and letting $\boldsymbol{\Omega}_\kappa = \boldsymbol{R}'_\kappa \boldsymbol{R}_\kappa^\top$, we obtain

$$\boldsymbol{\Omega}_\kappa \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 0 \end{pmatrix} = \begin{pmatrix} -1 & 0 \\ 0 & -1 \\ 0 & 0 \end{pmatrix}. \tag{79}$$

This means that $\boldsymbol{\Omega}_\kappa$ is a rotation matrix that maps vectors $(1,0,0)^\top$ and $(0,1,0)^\top$ onto $(-1,0,0)^\top$ and $(0,-1,0)^\top$, respectively. So, $\boldsymbol{\Omega}_\kappa = \mathrm{diag}(-1,-1,1)$.

C. Paraperspective Projection

If we define the projection matrix $\boldsymbol{\Pi}$ by

$$\boldsymbol{\Pi} = \frac{f}{t_z} \begin{pmatrix} 1 & 0 & -t_x/t_z & t_x/f \\ 0 & 1 & -t_y/t_z & t_y/f \end{pmatrix}, \tag{80}$$

paraperspective projection equation (38) can be rewritten as

$$\begin{pmatrix} x \\ y \end{pmatrix} = \boldsymbol{\Pi} \begin{pmatrix} X \\ Y \\ Z \\ f \end{pmatrix}. \tag{81}$$

Computing the metric matrix

Eq. (2) can be rewritten as

$$\begin{pmatrix} \boldsymbol{r}_{\kappa\alpha} \\ f_\kappa \end{pmatrix} = \begin{pmatrix} \boldsymbol{t}_\kappa \\ f_\kappa \end{pmatrix} + \begin{pmatrix} \boldsymbol{i}_\kappa \\ 0 \end{pmatrix} + \begin{pmatrix} \boldsymbol{j}_\kappa \\ 0 \end{pmatrix} + \begin{pmatrix} \boldsymbol{k}_\kappa \\ 0 \end{pmatrix}. \tag{82}$$

Hence, the vectors $\tilde{\boldsymbol{m}}_{0\kappa}$, $\tilde{\boldsymbol{m}}_{1\kappa}$, $\tilde{\boldsymbol{m}}_{2\kappa}$, and $\tilde{\boldsymbol{m}}_{3\kappa}$ in eq. (4) can be written as

$$\tilde{\boldsymbol{m}}_{0\kappa} = \boldsymbol{\Pi}_\kappa \begin{pmatrix} \boldsymbol{t}_\kappa \\ f_\kappa \end{pmatrix}, \quad \tilde{\boldsymbol{m}}_{1\kappa} = \boldsymbol{\Pi}_\kappa \begin{pmatrix} \boldsymbol{i}_\kappa \\ 0 \end{pmatrix},$$
$$\tilde{\boldsymbol{m}}_{2\kappa} = \boldsymbol{\Pi}_\kappa \begin{pmatrix} \boldsymbol{j}_\kappa \\ 0 \end{pmatrix}, \quad \tilde{\boldsymbol{m}}_{3\kappa} = \boldsymbol{\Pi}_\kappa \begin{pmatrix} \boldsymbol{k}_\kappa \\ 0 \end{pmatrix}, \tag{83}$$

where $\boldsymbol{\Pi}_\kappa$ is the matrix obtained by replacing f and t_z, respectively, by the values f_κ and $t_{z\kappa}$ at time κ in eq. (80). From the definition of the rotation matrix $\boldsymbol{R}_\kappa$ in eq. (49), we obtain

$$\begin{pmatrix} \tilde{\boldsymbol{m}}_{1\kappa} & \tilde{\boldsymbol{m}}_{2\kappa} & \tilde{\boldsymbol{m}}_{3\kappa} \end{pmatrix} = \boldsymbol{\Pi}_\kappa \begin{pmatrix} \boldsymbol{R}_\kappa \\ \boldsymbol{0}^\top \end{pmatrix}. \tag{84}$$

Since $\boldsymbol{R}_\kappa$ is an orthogonal matrix, we have

$$\begin{aligned} &\begin{pmatrix} \tilde{\boldsymbol{m}}_{1\kappa} & \tilde{\boldsymbol{m}}_{2\kappa} & \tilde{\boldsymbol{m}}_{3\kappa} \end{pmatrix} \begin{pmatrix} \tilde{\boldsymbol{m}}_{1\kappa} & \tilde{\boldsymbol{m}}_{2\kappa} & \tilde{\boldsymbol{m}}_{3\kappa} \end{pmatrix}^\top \\ &= \boldsymbol{\Pi}_\kappa \begin{pmatrix} \boldsymbol{R}_\kappa \\ \boldsymbol{0}^\top \end{pmatrix} \begin{pmatrix} \boldsymbol{R}_\kappa^\top & \boldsymbol{0}^\top \end{pmatrix} \boldsymbol{\Pi}_\kappa^\top \\ &= \boldsymbol{\Pi}_\kappa \begin{pmatrix} \boldsymbol{I} & \\ & 0 \end{pmatrix} \boldsymbol{\Pi}_\kappa^\top \\ &= \frac{f_\kappa^2}{t_{z\kappa}^2} \begin{pmatrix} 1 + t_{x\kappa}^2/t_{z\kappa}^2 & t_{x\kappa} t_{y\kappa}/t_{z\kappa}^2 \\ t_{x\kappa} t_{y\kappa}/t_{z\kappa}^2 & 1 + t_{y\kappa}^2/t_{z\kappa}^2 \end{pmatrix}. \end{aligned} \tag{85}$$

In terms of the vectors $\boldsymbol{m}^\dagger_{\kappa(a)}$ defined in Step 9 of the main procedure, the above equation can be rewritten as

$$\begin{aligned} &\begin{pmatrix} \boldsymbol{m}^\dagger_{\kappa(1)} & \boldsymbol{m}^\dagger_{\kappa(2)} \end{pmatrix}^\top \begin{pmatrix} \boldsymbol{m}^\dagger_{\kappa(1)} & \boldsymbol{m}^\dagger_{\kappa(2)} \end{pmatrix} \\ &= \frac{f_\kappa^2}{t_{z\kappa}^2} \begin{pmatrix} 1 + t_{x\kappa}^2/t_{z\kappa}^2 & t_{x\kappa} t_{y\kappa}/t_{z\kappa}^2 \\ t_{x\kappa} t_{y\kappa}/t_{z\kappa}^2 & 1 + t_{y\kappa}^2/t_{z\kappa}^2 \end{pmatrix}. \end{aligned} \tag{86}$$

Equating individual elements on both sides, we obtain

$$\begin{aligned} \|\boldsymbol{m}^\dagger_{\kappa(1)}\|^2 &= \frac{f_\kappa^2}{t_{z\kappa}^2}\Big(1 + \frac{\tilde{t}_{x\kappa}^2}{f_\kappa^2}\Big), \\ \|\boldsymbol{m}^\dagger_{\kappa(2)}\|^2 &= \frac{f_\kappa^2}{t_{z\kappa}^2}\Big(1 + \frac{\tilde{t}_{y\kappa}^2}{f_\kappa^2}\Big), \\ (\boldsymbol{m}^\dagger_{\kappa(1)}, \boldsymbol{m}^\dagger_{\kappa(2)}) &= \frac{\tilde{t}_{x\kappa} \tilde{t}_{y\kappa}}{t_{z\kappa}^2}, \end{aligned} \tag{87}$$

where we use the following identities resulting from the first of eqs. (83):

$$\begin{pmatrix} \tilde{t}_{x\kappa} \\ \tilde{t}_{y\kappa} \end{pmatrix} = \frac{f_\kappa}{t_{z\kappa}} \begin{pmatrix} t_{x\kappa} \\ t_{y\kappa} \end{pmatrix}. \tag{88}$$

Recalling that $\boldsymbol{M}^\top = \boldsymbol{A}^\top \boldsymbol{U}^\top$ from eq. (15) and introducing the vectors $\boldsymbol{u}^\dagger_{\kappa(a)}$ defined in Step 4 of the main procedure, we can express the vector $\boldsymbol{m}^\dagger_{\kappa(a)}$ in the form of eq. (54). Substituting it into eqs. (87) and defining α_κ, β_κ, and γ_κ by eqs. (39), we obtain

$$\begin{aligned} (\boldsymbol{A}^\top \boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{A}^\top \boldsymbol{u}^\dagger_{\kappa(1)}) &= \frac{f_\kappa^2}{\alpha_\kappa t_{z\kappa}^2}, \\ (\boldsymbol{A}^\top \boldsymbol{u}^\dagger_{\kappa(2)}, \boldsymbol{A}^\top \boldsymbol{u}^\dagger_{\kappa(2)}) &= \frac{f_\kappa^2}{\beta_\kappa t_{z\kappa}^2}, \\ (\boldsymbol{A}^\top \boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{A}^\top \boldsymbol{u}^\dagger_{\kappa(2)}) &= \frac{\gamma_\kappa f_\kappa^2}{t_{z\kappa}^2}. \end{aligned} \tag{89}$$

In terms of the metric matrix $\boldsymbol{T}$ (eq. (11)), these are rewritten as

$$\begin{aligned} &\alpha_\kappa (\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(1)}) = \beta_\kappa (\boldsymbol{u}^\dagger_{\kappa(2)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(2)}) \\ &= \frac{(\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(2)})}{\gamma_\kappa} = \frac{f_\kappa^2}{t_{z\kappa}^2}, \end{aligned} \tag{90}$$

from which we obtain the following two conditions:

$$\begin{aligned} &\alpha_\kappa (\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(1)}) = \beta_\kappa (\boldsymbol{u}^\dagger_{\kappa(2)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(2)}) \\ &\gamma_\kappa \Big(\alpha_\kappa (\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(1)}) + \beta_\kappa (\boldsymbol{u}^\dagger_{\kappa(2)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(2)}) \Big) \\ &= 2(\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(2)}). \end{aligned} \tag{91}$$

These conditions are arranged so as to preserve the symmetry between α_κ and α_β. We also multiply the second equation by γ_κ in order to prevent γ_κ from appearing in the denominator, since if $\tilde{t}_{x\kappa}$ or $\tilde{t}_{y\kappa}$ happens to be 0, we will have $\gamma_\kappa = 0$.

We now compute the matrix $\boldsymbol{T}$ that minimizes

$$K = \sum_{\kappa=1}^{M} \Big[\Big(\alpha_\kappa (\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(1)}) - \beta_\kappa (\boldsymbol{u}^\dagger_{\kappa(2)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(2)}) \Big)^2 + \Big(\gamma_\kappa \Big(\alpha_\kappa (\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(1)}) + \beta_\kappa (\boldsymbol{u}^\dagger_{\kappa(2)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(2)}) \Big) - 2(\boldsymbol{u}^\dagger_{\kappa(1)}, \boldsymbol{T}\boldsymbol{u}^\dagger_{\kappa(2)}) \Big)^2 \Big]. \quad (92)$$

If we define the tensor $\mathcal{B} = (B_{ijkl})$ by eq. (40), the above equation has the form of eq. (75). Since the matrix $\boldsymbol{T} = (T_{ij})$ has scale indeterminacy as in the case of weak perspective projection, we do not lose generality if we normalize it into $\|\boldsymbol{T}\| = 1$.

In terms of the 6×6 symmetric matrix $\boldsymbol{B}$ given in eq. (23) and the 6-D vector $\boldsymbol{\tau}$ defined via eq. (26), we can write eq. (92) in the form of eq. (76). Since the condition $\|\boldsymbol{T}\| = 1$ is equivalent to the condition $\|\boldsymbol{\tau}\| = 1$, we compute the unit vector $\boldsymbol{\tau}$ that minimizes the quadratic form K in $\boldsymbol{\tau}$, and the solution is given by the unit eigenvector of $\boldsymbol{B}$ for the smallest eigenvalue. From it, we obtain the solution $\boldsymbol{T}$ in the form of eq. (26). However, eigenvectors have signature indeterminacy, and the matrix $\boldsymbol{T}$ should be positive semi-definite. So, if $\det \boldsymbol{T} < 0$, we change the sign of $\boldsymbol{T}$.

Computing translation

If the metric matrix $\boldsymbol{T}$ is computed, $t_{z\kappa}$ is given from eq. (90). However, γ_κ in the denominator of the last term may happen to be 0. We also want to preserve the symmetry between α_κ and β_κ. So, we compute $t_{z\kappa}$ in the form of eq. (33). The remaining components $t_{x\kappa}$ and $t_{y\kappa}$ are determined from (88) in the form of eq. (34).

Computing rotation

As in the case of orthographic and weak perspective projections, the ith column of the motion matrix $\boldsymbol{M}$ is given in the form of eq. (16), and the vectors $\boldsymbol{m}^\dagger_{\kappa(a)}$ are determined accordingly.

If we transpose both sides of eq. (84) and let $\boldsymbol{r}^\dagger_{\kappa(i)}$ be the ith column of $\boldsymbol{R}^\top_\kappa$, we obtain from the definition of $\boldsymbol{m}^\dagger_{\kappa(a)}$

$$\begin{aligned} \begin{pmatrix} \boldsymbol{m}^\dagger_{\kappa(1)} & \boldsymbol{m}^\dagger_{\kappa(2)} \end{pmatrix} &= \begin{pmatrix} \boldsymbol{R}^\top_\kappa & \boldsymbol{0} \end{pmatrix} \boldsymbol{\Pi}^\top_\kappa \\ &= \frac{f_\kappa}{t_{z\kappa}} \begin{pmatrix} \boldsymbol{r}^\dagger_{\kappa(1)} - \frac{t_{x\kappa}}{t_{z\kappa}} \boldsymbol{r}^\dagger_{\kappa(3)} & \boldsymbol{r}^\dagger_{\kappa(2)} - \frac{t_{y\kappa}}{t_{z\kappa}} \boldsymbol{r}^\dagger_{\kappa(3)} \end{pmatrix}. \end{aligned} \quad (93)$$

Hence, we obtain

$$\begin{aligned} \frac{t_{z\kappa}}{f_\kappa} \boldsymbol{m}^\dagger_{\kappa(1)} &= \boldsymbol{r}^\dagger_{\kappa(1)} - \frac{t_{x\kappa}}{t_{z\kappa}} \boldsymbol{r}^\dagger_{\kappa(3)}, \\ \frac{t_{z\kappa}}{f_\kappa} \boldsymbol{m}^\dagger_{\kappa(2)} &= \boldsymbol{r}^\dagger_{\kappa(2)} - \frac{t_{y\kappa}}{t_{z\kappa}} \boldsymbol{r}^\dagger_{\kappa(3)}. \end{aligned} \quad (94)$$

Since $\{\boldsymbol{r}^\dagger_{\kappa(1)}, \boldsymbol{r}^\dagger_{\kappa(2)}, \boldsymbol{r}^\dagger_{\kappa(3)}\}$ is a right-handed orthonormal system, we obtain

$$\frac{t_{z\kappa}^2}{f_\kappa^2} \boldsymbol{m}^\dagger_{\kappa(1)} \times \boldsymbol{m}^\dagger_{\kappa(2)} = \boldsymbol{r}^\dagger_{\kappa(3)} + \frac{t_{x\kappa}}{t_{z\kappa}} \boldsymbol{r}^\dagger_{\kappa(1)} + \frac{t_{y\kappa}}{t_{z\kappa}} \boldsymbol{r}^\dagger_{\kappa(2)}. \quad (95)$$

Hence, the vectors $\{\boldsymbol{r}^\dagger_{\kappa(1)}, \boldsymbol{r}^\dagger_{\kappa(2)}, \boldsymbol{r}^\dagger_{\kappa(3)}\}$ are obtained by solving the following linear equations:

$$\begin{aligned} \boldsymbol{r}^\dagger_{\kappa(1)} - \frac{t_{x\kappa}}{t_{z\kappa}} \boldsymbol{r}^\dagger_{\kappa(3)} &= \frac{t_{z\kappa}}{f_\kappa} \boldsymbol{m}^\dagger_{\kappa(1)}, \\ \boldsymbol{r}^\dagger_{\kappa(2)} - \frac{t_{y\kappa}}{t_{z\kappa}} \boldsymbol{r}^\dagger_{\kappa(3)} &= \frac{t_{z\kappa}}{f_\kappa} \boldsymbol{m}^\dagger_{\kappa(2)}, \\ \frac{t_{x\kappa}}{t_{z\kappa}} \boldsymbol{r}^\dagger_{\kappa(1)} + \frac{t_{y\kappa}}{t_{z\kappa}} \boldsymbol{r}^\dagger_{\kappa(2)} + \boldsymbol{r}^\dagger_{\kappa(3)} &= \frac{t_{z\kappa}^2}{f_\kappa^2} \boldsymbol{m}^\dagger_{\kappa(1)} \times \boldsymbol{m}^\dagger_{\kappa(2)}. \end{aligned} \quad (96)$$

The solution is given by eqs. (42). In the presence of noise in the data, however, the computed $\{\boldsymbol{r}^\dagger_{\kappa(1)}, \boldsymbol{r}^\dagger_{\kappa(2)}, \boldsymbol{r}^\dagger_{\kappa(3)}\}$ may not necessarily be an exact orthonormal system. So, we fit an exact orthonormal system to them as in the case of orthographic and weak perspective projections. The SVD in the form of eq. (43) yields the best approximation to $\boldsymbol{R}_\kappa$ in the form of eq. (28).

Computing the shape

Combining the matrix Π_κ in eq. (44) and eqs. (49) and (84), we can rewrite the motion matrix $\boldsymbol{M}$ defined in Step 9 of the main procedure in the form of eq. (17) and obtain the least-squares solution eq. (18) as in the case of orthographic and weak perspective projections.

Computing the mirror image solution

As in the case of orthographic and weak perspective projections, the solution $\boldsymbol{s}_\alpha$ for the shape vector gives rise to its mirror image solution $-\boldsymbol{s}_\alpha$, for which the vectors $\boldsymbol{m}^\dagger_{\kappa(a)}$ change their signs. Then, eq. (84) implies that we obtain a mirror image solution $\boldsymbol{R}'_\kappa$ for the rotation such that $\boldsymbol{\Pi}_\kappa \begin{pmatrix} \boldsymbol{R}_\kappa \\ \boldsymbol{0}^\top \end{pmatrix} = -\boldsymbol{\Pi}_\kappa \begin{pmatrix} \boldsymbol{R}'_\kappa \\ \boldsymbol{0}^\top \end{pmatrix}$. Transposing both sides and letting $\boldsymbol{\Omega}_\kappa = \boldsymbol{R}'_\kappa \boldsymbol{R}^\top_\kappa$, we obtain

$$\begin{aligned} \boldsymbol{\Omega}_\kappa \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ -t_{x\kappa}/t_{z\kappa} & -t_{y\kappa}/t_{z\kappa} \end{pmatrix} \\ = \begin{pmatrix} -1 & 0 \\ 0 & -1 \\ t_{x\kappa}/t_{z\kappa} & t_{y\kappa}/t_{z\kappa} \end{pmatrix}. \end{aligned} \quad (97)$$

This means that $\boldsymbol{\Omega}_\kappa$ is a rotation matrix that maps vectors $(1, 0, -t_{x\kappa}/t_{z\kappa})^\top$ and $(1, 0, -t_{y\kappa}/t_{z\kappa})^\top$ onto $(-1, 0, t_{x\kappa}/t_{z\kappa})^\top$ and $(-1, 0, t_{y\kappa}/t_{z\kappa})^\top$, respectively. Noting that the vector $\boldsymbol{t}_\kappa = (t_{x\kappa}, t_{y\kappa}, t_{z\kappa})^\top$ is orthogonal to both $(1, 0, -t_{x\kappa}/t_{z\kappa})^\top$ and $(1, 0, -t_{y\kappa}/t_{z\kappa})^\top$, we conclude that $\boldsymbol{\Omega}_\kappa$ represents the rotation around axis $\boldsymbol{t}_\kappa$ by angle 180°. Hence, we obtain eq. (45).