Vol. 49 No. SIG 6(CVIM 20) IPSJ Transactions on Computer Vision and Image Media Mar. 2008

Regular Paper

Fast Projective Reconstruction: Toward Ultimate Efficiency

Hanno Ackermann[†1] and Kenichi Kanatani[†1]

We accelerate the time-consuming iterations for projective reconstruction, a key component of self-calibration for computing 3-D shapes from feature point tracking over a video sequence. We first summarize the algorithms of the primal and dual methods for projective reconstruction. Then, we replace the eigenvalue computation in each step by the power method. We also accelerate the power method itself. Furthermore, we introduce the SOR method for accelerating the subspace fitting involved in the iterations. Using simulated and real video images, we demonstrate that the computation sometimes becomes several thousand times faster.

1. Introduction

Various techniques have been proposed in the past for self-calibration: 3-D reconstruction from point correspondences over multiple images taken by uncalibrated cameras[6]. This paper quests for an efficient self-calibration implementation.

From among many different existing approaches, we choose the most widely adopted "two-stage method", which consists of projective reconstruction for computing a 3-D shape up to projectivity and Euclidean upgrading that transforms it to a correct shape. This approach is supported by rigorous mathematical analysis based on projective geometry[6].

Early studies of projective reconstruction were based on a particular reference frame, usually the first image, whose camera model was assumed to be in a canonical form; the camera models of the other frames were estimated relative to it. Today, however, the mainstream approach is iterative "factorization" consisting of simultaneous computations to all frames. This approach is known to be numerically stable.

One of the earliest of such ideas was by Christy and Horaud[1], who assumed affine cameras and iteratively introduced perspective effects, using the Tomasi-Kanade factorization[23]. Then, Sturm and Triggs[20] and Triggs[24] introduced a direct approach based on the fundamental matrices computed between frames (see also Deguchi[2]). Extending the Tomasi-Kanade factorization[23], they used the singular value decomposition (SVD) to factorize a matrix consisting of data and unknowns (no unknowns are involved if the camera is affine).

Later, a new type of iterative method that does not require fundamental matrices was presented in many different forms[9),13),14),27]. One of the major differences from earlier ones is that the algebraic interpretation of "factorization of a matrix by SVD" is now replaced by the geometric interpretation[⋆1] of "fitting a subspace to high-dimensional data ". In this paper, we adopt the formulations of Mahamud and Hebert[13] and Heyden et al.[9].

For the second stage of Euclidean upgrading, the mainstream nowadays is the use of the "dual absolute quadric constraint" introduced by Triggs[25], for which many different assumptions have been used[7),8),16),18),19].

Assuming that the camera aspect ratio is 1 with no image skew and using a modified version[10] of the method of Seo and Heyden[18], we did many experiments and found that the iterations of projective reconstruction far exceed the Euclidean upgrading in computation time[10]; the latter is basically an analytical procedure, and the computation time is very short even if iterations are added to improve the robustness[10]. Hence, efficiency improvement of self-calibration largely depends on how to speed up the projective reconstruction. This paper concentrates on this issue.

Our close examination has revealed that the reason for projective reconstruction to take long time is that each iteration requires eigenvalue computation of a matrix whose size depends on the number of tracked points or the number of frames that we observe. As a result, the computation time quickly grows as the number of points or frames increases.

†1 Okayama University

⋆1 In fact, this is mathematically the essence of the Tomasi-Kanade factorization[23]; the term "factorization" is very misleading[11),12].

The key idea comes from the observation that the eigenvector computed in each step should not be very different from the value computed in the preceding step and hence it need not be computed from scratch; a large reduction of computation time is expected if the preceding value is updated with a small number of operations. To do this, we introduce the "power method" for eigenvalue computation.

The power method itself is an iterative process, so we introduce an extrapolation technique to accelerate the convergence. We can also introduce the scheme of successive overrelaxation (SOR), well known in physics, for further speedup. Using simulated and real video images, we demonstrate that these techniques significantly accelerate the computation, sometimes several thousand times faster.

2. Projective Reconstruction

Suppose we track N points over M image frames. Let $(x_{\kappa\alpha}, y_{\kappa\alpha})$ be the image coordinates of the αth point in the κth frame. The fundamental equation of projective reconstruction is[6)]

$$z_{\kappa\alpha}\boldsymbol{x}_{\kappa\alpha} = \boldsymbol{\Pi}_\kappa \boldsymbol{X}_\alpha, \quad \boldsymbol{x}_{\kappa\alpha} = \begin{pmatrix} x_{\kappa\alpha}/f_0 \\ y_{\kappa\alpha}/f_0 \\ 1 \end{pmatrix}, \tag{1}$$

where f_0 is an appropriate constant[⋆1]. In Eqs. (1), $\boldsymbol{\Pi}_\kappa$ is a 3×4 camera matrix (unknown), and $z_{\kappa\alpha}$ is a constant called the "projective depth" (unknown). The 4-D vector $\boldsymbol{X}_\alpha$ (unknown) consists of homogeneous coordinates of the αth point in the scene. Our task is to determine $z_{\kappa\alpha}$, $\boldsymbol{\Pi}_\kappa$, and $\boldsymbol{X}_\alpha$ from the data $\boldsymbol{x}_{\kappa\alpha}$.

To solve this problem, we modify the method of Mahamud and Hebert[13)], which we call the *primal method*, and the method of Heyden et al.[9)], which we call the *dual method*, in the same framework, using corresponding symbols and notations (see Ref. 10) for the details). Hereafter, we denote the inner product of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ by $(\boldsymbol{a}, \boldsymbol{b})$.

2.1 Primal Method (Prototype)

Input:

$\boldsymbol{x}_{\kappa\alpha}$, $\kappa = 1, ..., M$, $\alpha = 1, ..., N$,

admissible reprojection error $E_{\min}$ (pixels).

Output:

$\boldsymbol{\Pi}_\kappa$, $\kappa = 1, ..., M$, $\boldsymbol{X}_\alpha$, $\alpha = 1, ..., N$.

Computation:

(1) Initialize the projective depths to $z_{\kappa\alpha} = 1$.

(2) Let $\boldsymbol{p}_\alpha$ be the $3M$-D vector that vertically aligns $z_{1\alpha}\boldsymbol{x}_{1\alpha}$, ..., $z_{M\alpha}\boldsymbol{x}_{M\alpha}$ as its components. Then, normalize it to a unit vector.

(3) Compute the $3M \times 3M$ matrix

$$\boldsymbol{M} = \sum_{\alpha=1}^{N} \boldsymbol{p}_\alpha \boldsymbol{p}_\alpha^\top. \tag{2}$$

(4) Compute the unit eigenvectors $\boldsymbol{u}_1$, $\boldsymbol{u}_2$, $\boldsymbol{u}_3$, and $\boldsymbol{u}_4$ of $\boldsymbol{M}$ for the largest four eigenvalues.

(5) Compute the 3×4 camera matrix $\boldsymbol{\Pi}_\kappa$ by

$$\boldsymbol{\Pi}_\kappa = \begin{pmatrix} \boldsymbol{u}_{1\kappa}^* & \boldsymbol{u}_{2\kappa}^* & \boldsymbol{u}_{3\kappa}^* & \boldsymbol{u}_{4\kappa}^* \end{pmatrix}, \tag{3}$$

where $\boldsymbol{u}_{i\kappa}^*$ is a 3-D vector whose first, second, and third components are, respectively, the $(3(\kappa-1)+1)$st, $(3(\kappa-1)+2)$nd, and $(3(\kappa-1)+3)$rd components of $\boldsymbol{u}_i$.

(6) Do the following computations for $\alpha = 1$, ..., N.

(a) Compute the unit eigenvector $\boldsymbol{\xi}_\alpha$ for the largest eigenvalue of the $M \times M$ matrix $\boldsymbol{A}^\alpha = (A^\alpha_{\kappa\lambda})$ defined by

$$A^\alpha_{\kappa\lambda} = \frac{\sum_{k=1}^{4} (\boldsymbol{x}_{\kappa\alpha}, \boldsymbol{u}_{k\kappa}^*)(\boldsymbol{x}_{\lambda\alpha}, \boldsymbol{u}_{k\lambda}^*)}{\|\boldsymbol{x}_{\kappa\alpha}\| \cdot \|\boldsymbol{x}_{\lambda\alpha}\|}. \tag{4}$$

The sign of $\boldsymbol{\xi}_\alpha = (\xi_{\kappa\alpha})$ is chosen so that

$$\sum_{\kappa=1}^{M} \xi_{\kappa\alpha} \geq 0. \tag{5}$$

(b) From the resulting $\boldsymbol{\xi}_\alpha = (\xi_{\kappa\alpha})$, determine the projective depths $z_{\kappa\alpha}$ by

$$z_{\kappa\alpha} = \frac{\xi_{\kappa\alpha}}{\|\boldsymbol{x}_{\kappa\alpha}\|}. \tag{6}$$

(c) Using the computed $z_{\kappa\alpha}$, recompute the vectors $\boldsymbol{p}_\alpha$ and normalize them to unit norm.

(d) Compute the 3-D positions $\boldsymbol{X}_\alpha = (X^k_\alpha)$ by

$$X^k_\alpha = (\boldsymbol{p}_\alpha, \boldsymbol{u}_k), \quad k = 1 \sim 4. \tag{7}$$

(7) Compute the reprojection error

$$E = f_0 \sqrt{\frac{1}{MN} \sum_{\kappa=1}^{M} \sum_{\alpha=1}^{N} \|\boldsymbol{x}_{\kappa\alpha} - \mathcal{Z}[\boldsymbol{\Pi}_\kappa \boldsymbol{X}_\alpha]\|^2}, \tag{8}$$

where $\mathcal{Z}[\,\cdot\,]$ denotes normalization of a vector to make the third component 1.

(8) If $E < E_{\min}$, stop. Else, go back to Step (3).

⋆1 This is for scaling $x_{\kappa\alpha}/f_0$ and $y_{\kappa\alpha}/f_0$ into the order of 1 for numerical stability[4)]. In our experiment, we set $f_0 = 600$ pixels.

2.2 Dual Method (Prototype)

Input:

$\boldsymbol{x}_{\kappa\alpha}$, $\kappa = 1, ..., M$, $\alpha = 1, ..., N$,
admissible reprojection error $E_{\min}$ (pixels).

Output:

$\boldsymbol{\Pi}_\kappa$, $\kappa = 1, ..., M$, $\boldsymbol{X}_\alpha$, $\alpha = 1, ..., N$.

Computation:

(1) Initialize the projective depths to $z_{\kappa\alpha} = 1$.

(2) Compute the following N-D vectors $\boldsymbol{q}_\kappa^i$:

$$\boldsymbol{q}_\kappa^1 = \Big(\frac{z_{\kappa1}x_{\kappa1}}{f_0}, \frac{z_{\kappa2}x_{\kappa2}}{f_0}, ..., \frac{z_{\kappa N}x_{\kappa N}}{f_0}\Big)^\top,$$
$$\boldsymbol{q}_\kappa^2 = \Big(\frac{z_{\kappa1}y_{\kappa1}}{f_0}, \frac{z_{\kappa2}y_{\kappa2}}{f_0}..., \frac{z_{\kappa N}y_{\kappa N}}{f_0}\Big)^\top,$$
$$\boldsymbol{q}_\kappa^3 = (z_{\kappa1}, z_{\kappa2}, ..., z_{\kappa N})^\top. \quad (9)$$

For each κ, multiply $\boldsymbol{q}_\kappa^i$, $i = 1, 2, 3$, by a common constant so that

$$\sum_{i=1}^{3} \|\boldsymbol{q}_\kappa^i\|^2 = 1. \quad (10)$$

(3) Compute the $N \times N$ matrix

$$\boldsymbol{N} = \sum_{\kappa=1}^{M}\sum_{i=1}^{3} \boldsymbol{q}_\kappa^i \boldsymbol{q}_\kappa^{i\top}. \quad (11)$$

(4) Compute the unit eigenvectors $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, $\boldsymbol{v}_3$, and $\boldsymbol{v}_4$ of $\boldsymbol{N}$ for the largest four eigenvalues.

(5) Compute the 3-D positions $\boldsymbol{X}_\alpha = (X_\alpha^k)$ by

$$X_\alpha^k = (\text{the } \alpha\text{th component of } \boldsymbol{v}_k), \quad k = 1 \sim 4. \quad (12)$$

(6) Do the following computations for $\kappa = 1, ..., M$.

(a) Compute the unit eigenvector $\boldsymbol{\xi}_\kappa$ for the largest eigenvalue of the $N \times N$ matrix $\boldsymbol{B}^\kappa = (B_{\alpha\beta}^\kappa)$ defined by

$$B_{\alpha\beta}^\kappa = \frac{(\boldsymbol{v}_\alpha^*, \boldsymbol{v}_\beta^*)(\boldsymbol{x}_{\kappa\alpha}, \boldsymbol{x}_{\kappa\beta})}{\|\boldsymbol{x}_{\kappa\alpha}\| \cdot \|\boldsymbol{x}_{\kappa\beta}\|}, \quad (13)$$

where $\boldsymbol{v}_\alpha^*$ is a 4-D vector consisting of the αth components of $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, $\boldsymbol{v}_3$, and $\boldsymbol{v}_4$ in that order. The sign of $\boldsymbol{\xi}_\kappa$ is chosen so that

$$\sum_{\alpha=1}^{N} \xi_{\kappa\alpha} \geq 0. \quad (14)$$

(b) From the resulting $\boldsymbol{\xi}_\kappa = (\xi_{\kappa\alpha})$, determine the projective depths $z_{\kappa\alpha}$ by Eq. (6).

(c) Using the computed $z_{\kappa\alpha}$, recompute the vectors $\boldsymbol{q}_\kappa^i$, $i = 1, 2, 3$, and normalize them in the form of Eq. (10).

(d) Compute the 3×4 camera matrix $\boldsymbol{\Pi}_\kappa = (\Pi_{\kappa(ij)})$ by

$$\Pi_{\kappa(ij)} = (\boldsymbol{q}_\kappa^i, \boldsymbol{v}_j). \quad (15)$$

(7) Compute the reprojection error E in Eq. (8).

(8) If $E < E_{\min}$, stop. Else, go back to Step (3).

2.3 Comparison with Factorization

In the Steps (3) and (4) of the primal and the dual methods, a 4-D subspace is fitted to the data, and in the Step (6) the projective depths $z_{\kappa\alpha}$ are updated[10]. The algorithm iterates these two procedures. In this sense, it is a kind of EM algorithm. Setting $z_{\kappa\alpha} = 1$ in the Step (1) is equivalent to assuming an affine camera.

If we compute the camera matrix $\boldsymbol{\Pi}_\kappa$ in the Step (5) of the primal method and stop there, or equivalently, if we compute the 3-D positions $\boldsymbol{X}_\alpha$ in the Step (5) of the dual method and stop there, the computation is essentially the first stage[⋆1] ("affine reconstruction") of the Tomasi-Kanade factorization[23]. Hence, the above procedure is a natural extension of the Tomasi-Kanade factorization.

The second stage of the Tomasi-Kanade factorization is Euclidean upgrading using the "metric constraint" based on a particular affine camera model such as orthographic, weak perspective, or paraperspective[11], while for self-calibration, the "dual absolute quadric constraint[25]" is used[10]. They have similar computational structures (basically least squares) and the same computational complexity.

This observation shows that the difference between self-calibration and factorization is whether we do eigenvalue computation in Step (6)-(a) and iterate Steps (3)–(8) till the reprojection error converges. Hence, speedup of self-calibration depends on speedup of eigenvalue computation.

We can also see that the reprojection error evaluation in Step (7) is not necessary; we can simply stop when all values do not differ much (by setting an appropriate threshold) after update. This may further speed up the computation, since the reprojection error evaluation takes time proportional to the number of points and the number of frames. Also, we can avoid setting too low a reprojection error which may never be reached. In this paper, however, we

⋆1 In either case, the computation reduces to matrix factorization by SVD, hence the name "factorization".

retain this evaluation simply for comparing the convergence performance of the two algorithms for a common threshold, which is the theme of this paper.

3. Acceleration by the Power Method

As we pointed out earlier, the eigenvector computed in each iteration step is nearly the same as in the preceding step, so the computation is expected to speed up if the preceding value is updated with a small number of operations. To do this, we use the "power method"[3].

Let $\boldsymbol{T}$ be an $n \times n$ positive semi-definite symmetric matrix with eigenvalues $\lambda_1 \geq \cdots \geq \lambda_n$ (≥ 0), and $\boldsymbol{u}_1$, ..., $\boldsymbol{u}_n$ the corresponding unit eigenvectors. Then, $\boldsymbol{T}^k$ has eigenvalues $\lambda_1^k \geq \cdots \geq \lambda_n^k$ with the same eigenvectors. Hence, if arbitrary linearly independent vectors $\boldsymbol{v}_1$, ..., $\boldsymbol{v}_m$ are multiplied by $\boldsymbol{T}^k$, they are magnified in the directions of those eigenvectors that have large eigenvalues.

It follows that for large k, the m-D subspace spanned by $\boldsymbol{T}^k\boldsymbol{v}_1$, ..., $\boldsymbol{T}^k\boldsymbol{v}_m$ converges to the subspace spanned by the eigenvectors $\boldsymbol{u}_1$, ..., $\boldsymbol{u}_m$. For $m = 1$, in particular, the direction of $\boldsymbol{T}^k\boldsymbol{v}$ for an arbitrary $\boldsymbol{v}$ converges to that of $\boldsymbol{u}_1$ as $k \to \infty$. These facts have been used to speed up the Tomasi-Kanade factorization computation[5),15),28)].

In this paper, we use the power method for the eigenvalue computation in the Step (6)-(a) of the primal and the dual methods and for the 4-D subspace fitting in the Step (4). The actual procedure goes as follows ($\mathcal{N}[\,\cdot\,]$ denotes normalization of a vector to unit norm).

3.1 Primal Method (Power)

Input:

$\boldsymbol{x}_{\kappa\alpha}$, $\kappa = 1$, ..., M, $\alpha = 1$, ..., N,
admissible reprojection error $E_{\min}$ (pixels),
constants d and e for stopping the power method.

Output:

$\boldsymbol{\Pi}_\kappa$, $\kappa = 1$, ..., M, $\boldsymbol{X}_\alpha$, $\alpha = 1$, ..., N.

Computation:

(1) Initialize the projective depths to $z_{\kappa\alpha} = 1$.

(2) Compute the following M-D vector $\boldsymbol{\xi}_\alpha^0$.

$$\boldsymbol{\xi}_\alpha^0 = \mathcal{N}\Big[\begin{pmatrix} \|\boldsymbol{x}_{1\alpha}\|z_{1\alpha} \\ \|\boldsymbol{x}_{2\alpha}\|z_{2\alpha} \\ \vdots \\ \|\boldsymbol{x}_{M\alpha}\|z_{M\alpha} \end{pmatrix}\Big]. \tag{16}$$

(3) Let $\boldsymbol{p}_\alpha$ be the $3M$-D vector that vertically aligns $z_{1\alpha}\boldsymbol{x}_{1\alpha}$, ..., $z_{M\alpha}\boldsymbol{x}_{M\alpha}$ as its components. Then, normalize it to a unit vector.

(4) Compute the $3M \times N$ matrix

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{p}_1 & \boldsymbol{p}_2 & \cdots & \boldsymbol{p}_N \end{pmatrix}. \tag{17}$$

(5) Compute the SVD of $\boldsymbol{P}$ as follows:

$$\boldsymbol{P} = \boldsymbol{U}\mathrm{diag}(\sigma_1, \sigma_2, \cdots)\boldsymbol{V}^\top. \tag{18}$$

(6) Let $\boldsymbol{u}_1$, $\boldsymbol{u}_2$, $\boldsymbol{u}_3$, and $\boldsymbol{u}_4$ be the first four columns of the matrix $\boldsymbol{U}$.

(7) Compute the 3×4 camera matrix $\boldsymbol{\Pi}_\kappa$ in Eq. (3).

(8) Do the following computations for $\alpha = 1$, ..., N.

(a) Compute the $M \times M$ matrix $\boldsymbol{A}^\alpha = (A^\alpha_{\kappa\lambda})$ defined by Eq. (4).

(b) Compute $\boldsymbol{\xi}_\alpha$ by

$$\boldsymbol{\xi}_\alpha = \mathcal{N}[\boldsymbol{A}^\alpha\boldsymbol{\xi}_\alpha^0]. \tag{19}$$

(c) Let $\boldsymbol{\xi}_\alpha^0 \leftarrow \boldsymbol{\xi}_\alpha$ and go back to Step (b). Repeat this until $\|\boldsymbol{\xi}_\alpha - \boldsymbol{\xi}_\alpha^0\| < 10^{-d}$.

(d) From the resulting $\boldsymbol{\xi}_\alpha = (\xi_{\kappa\alpha})$, determine the projective depths $z_{\kappa\alpha}$ by Eq. (6).

(e) Using the computed $z_{\kappa\alpha}$, recompute the vectors $\boldsymbol{p}_\alpha$ and normalize them to unit norm.

(f) Compute the 3-D positions $\boldsymbol{X}_\alpha = (X^k_\alpha)$ by Eq. (7).

(9) Compute the reprojection error E in Eq. (8).

(10) If $E < E_{\min}$, stop. Else, recompute the matrix $\boldsymbol{P}$ in Eq. (17).

(11) Compute the following N-D vectors $\tilde{\boldsymbol{v}}_k$ and $3M$-D vectors $\tilde{\boldsymbol{u}}_k$, $k = 1 \sim 4$.

$$\tilde{\boldsymbol{v}}_k = \boldsymbol{P}^\top\boldsymbol{u}_k, \qquad \tilde{\boldsymbol{u}}_k = \boldsymbol{P}\tilde{\boldsymbol{v}}_k. \tag{20}$$

(12) Let $\hat{\boldsymbol{u}}_1$, $\hat{\boldsymbol{u}}_2$, $\hat{\boldsymbol{u}}_3$, and $\hat{\boldsymbol{u}}_4$ be the orthonormal system obtained by the Schmidt orthogonalization of $\tilde{\boldsymbol{u}}_1$, $\tilde{\boldsymbol{u}}_2$, $\tilde{\boldsymbol{u}}_3$, and $\tilde{\boldsymbol{u}}_4$.

(13) If $\max_{k=1}^4 \sqrt{1 - \sum_{l=1}^4 (\hat{\boldsymbol{u}}_k, \boldsymbol{u}_l)^2} < 10^{-e}$, let $\boldsymbol{u}_k \leftarrow \tilde{\boldsymbol{u}}_k$, $k = 1 \sim 4$, and go back to Step (7). Else, let $\boldsymbol{u}_k \leftarrow \tilde{\boldsymbol{u}}_k$, $k = 1 \sim 4$, and go back to Step (11).

3.2 Dual Method (Power)

Input:

$\boldsymbol{x}_{\kappa\alpha}$, $\kappa = 1$, ..., M, $\alpha = 1$, ..., N,
admissible reprojection error $E_{\min}$ (pixels),
constants d and e for stopping the power method.

Output:

$\boldsymbol{\Pi}_\kappa$, $\kappa = 1$, ..., M, $\boldsymbol{X}_\alpha$, $\alpha = 1$, ..., N.

Computation:

(1) Initialize the projective depths to $z_{\kappa\alpha} = 1$.

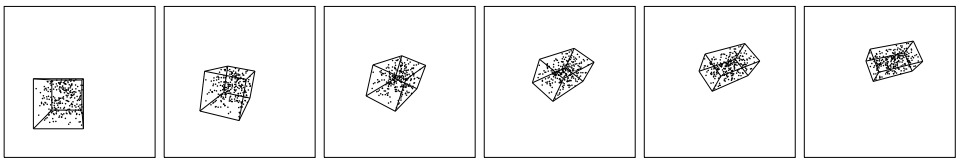

Fig. 1 Simulated image sequence of 265 points through 256 frames (6 frames decimated).

(2) Compute the following N-D vector $\boldsymbol{\xi}_\alpha^0$.

$$\boldsymbol{\xi}_\kappa^0 = \mathcal{N}\Big[\begin{pmatrix} \|\boldsymbol{x}_{\kappa 1}\| z_{\kappa 1} \\ \|\boldsymbol{x}_{\kappa 2}\| z_{\kappa 2} \\ \vdots \\ \|\boldsymbol{x}_{\kappa M}\| z_{\kappa N} \end{pmatrix}\Big]. \tag{21}$$

(3) Compute the N-D vectors $\boldsymbol{q}_\kappa^i$ in Eqs. (9), and normalize them so that Eq. (10) holds.

(4) Compute the $N \times 3M$ matrix

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{q}_1^1 & \boldsymbol{q}_1^2 & \boldsymbol{q}_1^3 & \boldsymbol{q}_2^1 & \cdots & \boldsymbol{q}_M^3 \end{pmatrix}. \tag{22}$$

(5) Compute the SVD of $\boldsymbol{Q}$ as follows[⋆1].

$$\boldsymbol{Q} = \boldsymbol{V}\mathrm{diag}(\sigma_1, \sigma_2, ...)\boldsymbol{U}^\top \tag{23}$$

(6) Let $\boldsymbol{v}_1$, $\boldsymbol{v}_2$, $\boldsymbol{v}_3$, and $\boldsymbol{v}_4$ be the first four columns of the matrix $\boldsymbol{V}$.

(7) Compute the 3-D positions $\boldsymbol{X}_\alpha = (X_\alpha^k)$ by Eq. (12).

(8) Do the following computations for $\kappa = 1$, ..., M.

(a) Compute the $N \times N$ matrix $\boldsymbol{B}^\kappa = (B_{\alpha\beta}^\kappa)$ defined by Eq. (13).

(b) Compute $\boldsymbol{\xi}_\kappa$ by

$$\boldsymbol{\xi}_\kappa = \mathcal{N}[\boldsymbol{B}^\kappa \boldsymbol{\xi}_\kappa^0]. \tag{24}$$

(c) Let $\boldsymbol{\xi}_\kappa^0 \leftarrow \boldsymbol{\xi}_\kappa$ and go back to Step (b). Repeat this until $\|\boldsymbol{\xi}_\kappa - \boldsymbol{\xi}_\kappa^0\| < 10^{-d}$.

(d) From the resulting $\boldsymbol{\xi}_\kappa = (\xi_{\kappa\alpha})$, determine the projective depths $z_{\kappa\alpha}$ by Eq. (6).

(e) Using the computed $z_{\kappa\alpha}$, recompute the vectors $\boldsymbol{q}_\kappa^i$ and normalize them in the form of Eq. (10).

(f) Compute the 3×4 camera matrix $\boldsymbol{\Pi}_\kappa = (\Pi_{\kappa(ij)})$ by Eq. (15).

(9) Compute the reprojection error E in Eq. (8).

(10) If $E < E_{\min}$, stop. Else, recompute the matrix $\boldsymbol{Q}$ in Eq. (22).

(11) Compute the following $3M$-D vectors $\tilde{\boldsymbol{u}}_k$ and N-D vectors $\tilde{\boldsymbol{v}}_k$, $k = 1 \sim 4$.

$$\tilde{\boldsymbol{u}}_k = \boldsymbol{Q}^\top \boldsymbol{v}_k, \quad \tilde{\boldsymbol{v}}_k = \boldsymbol{Q}\tilde{\boldsymbol{u}}_k. \tag{25}$$

(12) Let $\hat{\boldsymbol{v}}_1$, $\hat{\boldsymbol{v}}_2$, $\hat{\boldsymbol{v}}_3$, and $\hat{\boldsymbol{v}}_4$ be the orthonormal system obtained by the Schmidt orthogonalization of $\tilde{\boldsymbol{v}}_1$, $\tilde{\boldsymbol{v}}_2$, $\tilde{\boldsymbol{v}}_3$, and $\tilde{\boldsymbol{v}}_4$.

(13) If $\max_{k=1}^4 \sqrt{1 - \sum_{l=1}^4 (\hat{\boldsymbol{v}}_k, \boldsymbol{v}_l)^2} < 10^{-e}$, let $\boldsymbol{v}_k \leftarrow \tilde{\boldsymbol{v}}_k$, $k = 1 \sim 4$ and go back to Step (7). Else, let $\boldsymbol{v}_k \leftarrow \tilde{\boldsymbol{v}}_k$, $k = 1 \sim 4$, and go back to Step (11).

3.3 Effects of the Power Method

We note that if the matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$ ($= \boldsymbol{P}^\top$) are defined by Eqs. (17) and (22), Eqs. (2) and (11) are rewritten as $\boldsymbol{M} = \boldsymbol{P}\boldsymbol{P}^\top$ and $\boldsymbol{N} = \boldsymbol{Q}\boldsymbol{Q}^\top$. So, the eigenvalue computation of $\boldsymbol{M}$ and $\boldsymbol{N}$ in the Step (3) of the prototype can be replaced by the SVD of $\boldsymbol{P}$ and $\boldsymbol{Q}$ (Step (5)). This can generally reduce the computation time. In Step (11), we decompose the multiplication by $\boldsymbol{M}$ and $\boldsymbol{N}$ into the multiplications by $\boldsymbol{P}^\top$ followed by $\boldsymbol{P}$ and the multiplication by $\boldsymbol{Q}^\top$ followed by $\boldsymbol{Q}$, respectively; this can again reduce the number of arithmetic operations in general.

3.4 Experiment 1

Figure 1 shows a simulated image sequence of 256 points [⋆2] projected onto 256 camera frames of 600×600 pixels with focal length 600 pixels. Here, only six decimated frames are shown.

From this sequence, we computed projective reconstruction. The constants for stopping the power method were set to $e = 1$ and $d = 5$. The iterations were stopped when the reprojection error E became below 0.1 pixels. The computation time (sec) and the number of iterations are listed in **Table 1**. We used Pentium 4 3.4 GHz for the CPU with 2 GB main memory and Linux for the OS.

We can see that the power method reduces the computation time to 5% (primal) and 31% (dual) as compared to the prototype. However, the number of iterations slightly increases, because the power method is terminated before complete convergence. Yet, the overall computation time is drastically curtailed, because

⋆1 The singular values σ_1, σ_2, ... and the matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are the same as those in Eq. (18).

⋆2 The box is for visual ease only and not used for the computation.

Table 1 The computation time (sec) and the number of iterations for the sequence in Fig. 1 until the reprojection error is 0.1 pixels.

	primal		dual	
	time	iter.	time	iter.
prototype	85277	579	835	10
power	4036	602	257	28

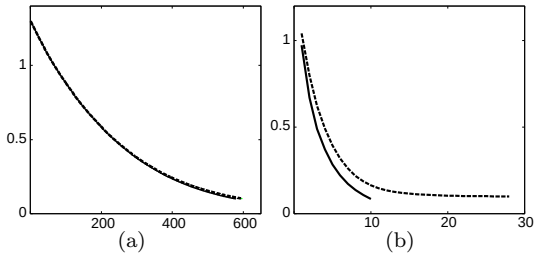

Fig. 2 Reprojection error vs. the number of iterations. The solid lines are for the prototype; the dashed lines are for the power method. (a) Primal method. (b) Dual method.

eigenvalue computation in each step is replaced by matrix vector multiplications.

To see this more closely, we plot the reprojection error E for the number of iterations in **Fig. 2**(a) (primal) and Fig. 2(b) (dual). The solid lines are for the prototype; the dashed lines are for the power method. We see that the reprojection error monotonically decreases for both the primal and the dual methods. We also see that the power method slows down the convergence, for the dual method in particular. Nevertheless, the overall computation time significantly reduces.

3.5 Comparing the Two Methods

As we can see from Table 1, the dual method is more efficient than the primal for the sequence in Fig. 1. However, the efficiency depends on the number N of points and the number M of frames. So, we need closer examination.

The primal method repeats eigenvalue computation for each point, while the dual method repeats eigenvalue computation for each frame. So, the computation time is nearly linear in N (primal) and in M (dual).

The complexity of eigenvalue computation is very difficult to evaluate, depending on the algorithm used. If we roughly estimate it to be $O(n^3)$ for an $n \times n$ matrix, the complexity is $O(NM^3)$ (primal) and $O(MN^3)$ (dual). Since the power method consists of vector-matrix multiplication, it is about $O(NM^2)$ (primal) and $O(MN^2)$ (dual) if terminated after a small number of iterations.

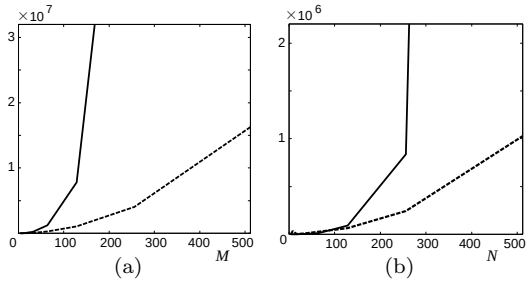

Fig. 3 Growth of the execution time (sec) for the prototype (solid lines) and the power method (dashed lines). (a) The primal method (N = 256). The horizontal axis is for the number M of frames. (b) The dual method (M = 256). The horizontal axis is for the number N of feature points.

In order to confirm this, we changed the number N of points and the number M of frames of the example in Fig. 1 (we added/removed points and intermediate frames) and observed the dependence of the computation time on M and N. **Figure** 3(a) shows the computation time of the primal method for different M (N = 256); Fig. 3(b) is for the dual method for different N (M = 256). The solid lines are for the prototype; the dashed lines for the power method.

We also tested various combinations of N and M. Fitting a polynomial in N and M to the computation time (we fitted a line to its logarithm by least square), we obtained

$$T_p \approx 1.407 N^{0.94} M^{1.7},$$
$$T_d \approx 0.163 M^{0.95} N^{1.7}, \quad (26)$$

for the primal and the dual methods, respectively (in ms). We can confirm that the computation time is approximately proportional to N (primal) and to M (dual), as expected. It is also proportional to $M^{1.7}$ (primal) and to $N^{1.7}$ (dual), which is very close to our prediction of M^2 (primal) and N^2 (dual).

We conclude that i) the primal method is favorable for a large number of points, ii) the dual method is favorable for a large number of frames, and iii) for comparable numbers of points and frames, the dual method is far more efficient.

4. Further Acceleration

4.1 Power Method Acceleration

The power method itself is an iterative process. As is well known in numerical analy-

sis[17], the convergence of an iterative process can be accelerated if the convergence rate is known. The technique is known by such names as "Eitken's δ^2" and the "Richardson extrapolation". A typical example is the "Romberg integration", which accelerates numerical integration in the course of increasing the sampling points. However, these are for scalar computations, while the power method is a vector computation. Hence, the known techniques cannot be directly applied to the power method. So, we devised the following method.

Consider the unit eigenvector $\boldsymbol{w}_1$ of a positive semidefinite symmetric matrix $\boldsymbol{T}$ for the largest eigenvalue λ_1. If we start from $\boldsymbol{\xi}^0$ and write $\boldsymbol{\xi}^k = \mathcal{N}[\boldsymbol{T}^k\boldsymbol{\xi}^0]$, we have, when $\boldsymbol{\xi}^0$ is close to $\boldsymbol{w}_1$ or k is large,

$$\boldsymbol{\xi}^k \approx \boldsymbol{w}_1 + C\gamma^k\boldsymbol{w}_2, \quad \gamma = \frac{\lambda_2}{\lambda_1}, \tag{27}$$

where $\boldsymbol{w}_2$ is the eigenvector for the second largest eigenvalue λ_2, and C is some constant. Eliminating $\boldsymbol{w}_2$ from this and the same relation for $k+1$, we obtain

$$\boldsymbol{w}_1 \approx \frac{\boldsymbol{\xi}^{k+1} - \gamma\boldsymbol{\xi}^k}{1-\gamma}. \tag{28}$$

If γ is known, this allows us to predict $\boldsymbol{w}_1$ from $\boldsymbol{\xi}^{k+1}$ and $\boldsymbol{\xi}^k$. If γ is unknown, we can estimate it from Eq. (27) by

$$\gamma \approx \frac{\|\boldsymbol{\xi}^{k+1} - \boldsymbol{\xi}^k\|}{\|\boldsymbol{\xi}^k - \boldsymbol{\xi}^{k-1}\|}. \tag{29}$$

Using this formula, we estimate γ from $\boldsymbol{\xi}^{k-1}$, $\boldsymbol{\xi}^k$, and $\boldsymbol{\xi}^{k+1}$ and replace $\boldsymbol{\xi}^{k+1}$ by $\mathcal{N}[(\boldsymbol{\xi}^{k+1} - \gamma\boldsymbol{\xi}^k)/(1-\gamma)]$, accelerating the iterations every other step (from $\boldsymbol{\xi}^0$ and $\boldsymbol{\xi}^1$ to $\boldsymbol{\xi}^2$, from $\boldsymbol{\xi}^2$ and $\boldsymbol{\xi}^3$ to $\boldsymbol{\xi}^4$, ...). This can be applied to the power method computation of the projective depth vectors $\boldsymbol{\xi}_\alpha$ and $\boldsymbol{\xi}_\kappa$.

The above idea can be extended to higher order correction, using a longer history of iterations[21], but this turned out not to be effective according to our experiments. We can use, instead, the steepest descent and the conjugate gradient methods[21], but again they turned out not to be so effective. So, we consider only the above described acceleration technique.

4.2 Acceleration by SOR

The successive overrelaxation (SOR) is a technique for convergence acceleration well known in physics: for a sequence ξ^1, ξ^2, ..., we accelerate ξ^k in the form

$$\xi^k \leftarrow \xi^{k-1} + \omega(\xi^k - \xi^{k-1}), \tag{30}$$

where ω (> 1) is called the *acceleration constant*. It has empirically been known that this scheme works well in many iterative problems, but an appropriate value of ω is very difficult to find except for special types of linear computation; it is usually set by experience. We apply this scheme to accelerate the projective depth vector $\boldsymbol{\xi}$ ($= \boldsymbol{\xi}_\alpha$ or $\boldsymbol{\xi}_\kappa$) each time the fitted subspace is updated: Using the value $\boldsymbol{\xi}'$ in the previous iteration, we replace $\boldsymbol{\xi}$ by $\mathcal{N}[\boldsymbol{\xi}' + \omega(\boldsymbol{\xi} - \boldsymbol{\xi}')]$.

5. Experiment 2

Table 2 lists the computation time and the number of iterations for the sequence in Fig. 1, using 1) the power method, 2) the accelerated power method, 3) SOR without accelerating the power method, and 4) SOR with the accelerated power method.

We set $e = 1$ and $d = 1$ for stopping the accelerated power method and $\omega = 1.9$ for SOR. The iterations were stopped when the reprojection error E became less than 0.1 pixels. In Table 2, the "efficiency index" means the ratio of the longest computation time to the shortest.

Figure 4 plots the reprojection error E for the number of iterations for the (a) primal and the (b) dual methods. The solid lines are for the power method; the dashed lines are for the accelerated power method; the dotted lines are for combining the power method and SOR; the chained lines are for combining the accelerated power method and SOR. For the primal method, only the solid and dotted lines are displayed; the dashed and the chained lines overlap the solid and the dotted line, respectively.

Figure 5(a) plots the computation time of the primal method for different M ($N = 256$); Fig. 5(b) is for the dual method for different N ($M = 256$). The solid lines are for the power method; the dashed lines are for the acceler-

Table 2 The computation time (sec) and the number of iterations for the sequence in Fig. 1 until the reprojection error becomes below 0.1 pixels (pm = power method, acc.pm = accelerated power method).

	primal		dual	
	time	iter.	time	iter.
pm	4036	602	257	28
acc.pm	4083	599	**48**	11
pm+SOR	2146	315	139	7
acc.pm+SOR	2112	312	76	14

Efficiency index = 1777

ated power method; the dotted lines are for the power method and SOR; the chained lines are for the accelerated power method and SOR. These plots reflect the iteration behavior shown in Fig. 4.

From these, we observe that accelerating the power method does not exhibit much effect on the primal method, while the dual method significantly improves. Closely examining the value of γ in Eqs. (27), we found that the power method converges very quickly (γ is small) for the primal method but slowly (γ is large) for the dual. This explains why the acceleration is more effective for the dual method.

On the other hand, SOR is effective for both. However, it is less effective than power method acceleration, and their simultaneous use reduces the effect of the accelerated power method.

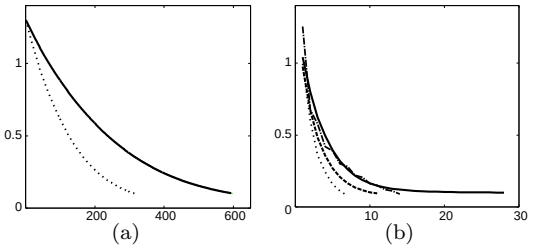

Fig. 4 Reprojection error vs. the number of iterations for (a) the primal and (b) the dual methods. The solid lines are for the power method; the dashed lines are for the accelerated power method; the dotted lines are for the power method and SOR; the chained lines are for the accelerated power method and SOR. In (a), only the solid and dotted lines are displayed; the dashed and the chained lines overlap the solid and the dotted line, respectively.

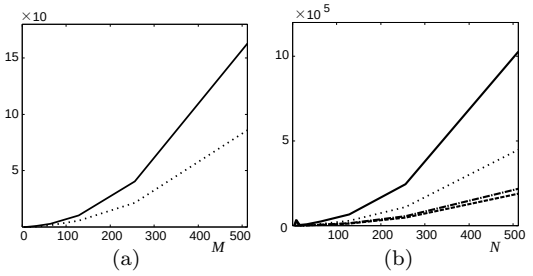

Fig. 5 Growth of the execution time (sec). The solid lines are for the power method; the dashed lines are for the accelerated power method; the dotted lines are for the power method and SOR; the chained lines are for the accelerated power method and SOR. (a) The primal method (N = 256). The horizontal axis is for the number M of frames. (b) The dual method (M = 256). The horizontal axis is for the number N of feature points.

Overall, the dual method with power method acceleration performs the best and is about 1800 times faster than the prototype primal method.

6. Experiment 3

Figure 6(a) shows six frames decimated from a simulated 11 frame sequence (600×600 pixels with focal length 600 pixels). Fig. 6(b) shows six frames decimated from a 200 frame video sequence (640×480 pixels), tracking 16 points as marked there. They were specified by hand in the first frame and tracked through the rest of the frames by the Kanade-Lucus-Tomasi (KLT) algorithm[22]; we manually intervened whenever the tracking failed.

Table 3 lists the computation time (sec) and the number of iterations for the sequence in **Fig. 3**(a) until the reprojection error becomes below 0.1 pixels for different methods.

For this sequence, the number of points is large (N = 231), but the number of frames is small (M = 11), so the prototype primal method is more efficient than the prototype dual method in spite of the fact that the number of iterations is smaller for the latter. This is because one step of the prototype dual method requires a large amount of computation, which is reduced by the use of the power method. As a result, the dual method becomes faster than the primal if the power method is used.

We also see from Table 3 that the power method acceleration has no effect on the primal method. This is because the power method iterations converge very quickly for the primal method, as we observed in Experiment 2. However, SOR has some effect.

For the dual method, on the other hand, the convergence of the power method is very slow, so the power method acceleration is very effective, making the computation 14 times faster. However, SOR has an adverse effect, and combining it with the accelerated power method is still inferior to the accelerated power method only.

Table 4 lists the computation time (sec) and the number of iterations for the sequence in Fig. 6(b) until the reprojection error becomes below 2.01 pixels for different methods; due to the tracking accuracy limitation of the KLT, it did not reduce in further iterations.

For this sequence, the number of frames is

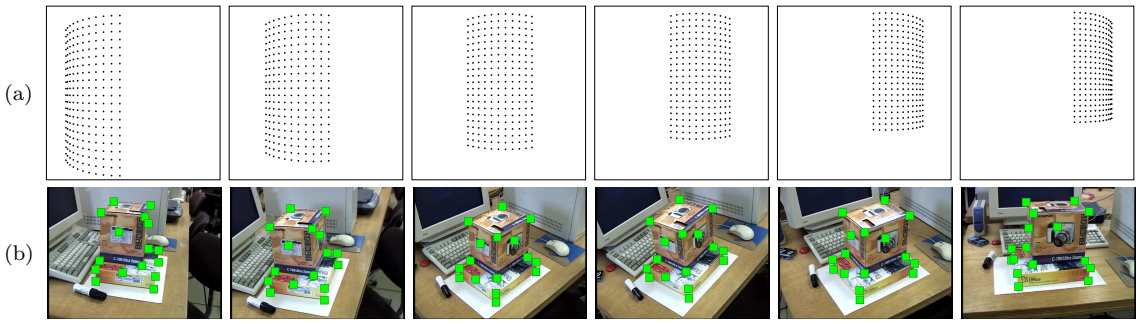

Fig. 6 (a) Simulated image sequence of 256 points through 256 frames (6 frames decimated). (b) Real video sequence (6 frames decimated), tracking 16 points through 200 frames.

Table 3 The computation time (sec) and the number of iterations for the sequence in Fig. 6(a) until the reprojection error becomes below 0.1 pixels (pm = power method, acc.pm = accelerated power method).

	primal		dual	
	time	iter.	time	iter.
prototype	3.84	89	7.15	3
pm	2.24	90	1.25	3
acc.pm	2.37	90	**0.51**	3
pm+SOR	1.12	47	10.76	21
acc.pm+SOR	1.27	47	5.73	31

Efficiency index = 14

Table 4 The computation time (sec) and the number of iterations for the sequence in Fig. 6(b) until the reprojection error is 2.01 pixels (pm = power method, acc.pm = accelerated power method).

	primal		dual	
	time	iter.	time	iter.
prototype	2153.3	300	0.26	5
pm	82.8	315	0.87	8
acc.pm	81.3	314	**0.26**	5
pm+SOR	43.4	165	1.85	15
acc.pm+SOR	42.6	165	1.48	31

Efficiency index = 8282

large ($M = 200$), so the primal method takes a vast amount of time if the prototype is used. The power method curtailed it to about 4% (26 times faster). Since the power method converges very quickly for the primal method, its acceleration has little effect, while SOR can further speed up the computation.

Nevertheless, all these cannot compare with the dual method, because the number of points is very small ($N = 16$). Already, the prototype converges so quickly that the use of the power method has an adverse effect. However, acceleration of the power method recovers the original efficiency, and the computation is about 8000 times faster than the prototype primal method.

For the dual method, SOR has an adverse effect, and its combination with power method acceleration gains only a little.

7. Conclusions

We presented various techniques for accelerating the projective reconstruction computation for self-calibration. Using simulated and real video images, we observed the following:

(1) The use of the power method increases the number of iterations, but the total computation time dramatically decreases, because the complexity of each step reduces. This is especially conspicuous for the primal method.

(2) Acceleration of the power method has little effect on the primal method, because the second largest eigenvalue is very small and hence the convergence is quick even without acceleration. In contrast, it has a significant effect on the dual method, for which the second largest eigenvalue is large.

(3) SOR is effective for both the primal and the dual methods, but the effect is relatively small for the dual method.

(4) The primal method is favorable for a very large number of points over a small number of frames; the dual method is preferable for a small number of points over a very large number of frames.

In real applications, we can obtain as many consecutive frames as we wish, using a video camera, but the number of available points is usually limited, because feature point tracking over a long sequence is a difficult task. In such situations, the best choice is the dual method combined with power method acceleration.

The self-calibration algorithm that we have

studied in this paper is based on the algebraic structure of the problem. It has often been pointed out that the reconstruction accuracy quickly deteriorates as the noise in the input data becomes larger[6),16)] and that we need optimization search known as *bundle adjustment*[26)].

The focus of this paper is efficiency, so in this paper we have not gone into the accuracy issue, which crucially depends on the quality of the tracking data. In the presence of large noise, however, bundle adjustment is a practical choice. To start bundle adjustment, we need a good initial value, which the algorithm described in this paper can provide.

According to our experiences, however, we have observed that fairly good reconstruction can be obtained even without bundle adjustment if the environment is controlled with human interventions and if the aim is display rather than precise measurement.

Acknowledgments The authors thank Akinobu Mori of Canon, Inc. for collaborating on this project while he was with us. They also thank Isao Miyagawa of NTT for helpful comments. This work is partly supported by Mitsubishi Precision Co., Ltd.

References

1) Christy, C. and Horaud, R.: Euclidean shape and motion from multiple perspective views by affine iterations, *IEEE Trans. Patt. Anal. Mach. Intell.*, Vol.18, No.11, pp.1098–1104 (1996).
2) Deguchi, K.: Factorization method for structure from perspective multi-view images, *IEICE Trans. Inf & Syst.*, Vol.E81-D, No.11, pp.1281–1289 (1998).
3) Golub, T.H. and Van Loan, C.F.: *Matrix Computations*, 3rd Ed., Johns-Hopkins University Press, Baltimore, MD, U.S.A. (1996).
4) Hartley, R.I.: In defense of the eight-point algorithm, *IEEE Trans. Patt. Anal. Mach. Intell.*, Vol.19, No.6, pp.580–593 (1997).
5) Hartley, R. and Schaffalitzky, R.: PowerFactorization: 3D reconstruction with missing or uncertain data, *Proc. Australia-Japan Advanced Workshop on Computer Vision*, Adelaide, Australia, pp.1–9 (2003).
6) Hartley, R. and Zisserman, A.: *Multiple View Geometry in Computer Vision*, Cambridge University Press, Cambridge, U.K. (2000).
7) Heyden, A. and Åström, K.: Euclidean reconstruction from image sequences with varying and unknown focal length and principal point, *Proc. IEEE Conf. Comput. Vis. Patt. Recog.*, Puerto Rico, pp.438–443 (1997).
8) Heyden, A. and Åström, K.: Flexible calibration: Minimal cases for auto-calibration, *Proc. 7th Int. Conf. Comput. Vis.*, Kerkyra, Greece, Vol.1, pp.350–355 (1999).
9) Heyden, A., Berthilsson, R. and Sparr, G.: An iterative factorization method for projective structure and motion from image sequences, *Image Vis. Comput.*, Vol.17, No.13, pp.981–991 (1999).
10) Kanatani, K.: Latest progress of 3-D reconstruction from multiple camera images, In Xing P. Guo (Ed.): *Robotics Research Trends*, pp. 33–75, Nova Science Publishers, Hauppauge, NY, U.S.A. (2008).
11) Kanatani, K. and Sugaya, Y.: Factorization without factorization: Complete recipe, *Mem. Fac. Eng. Okayama Univ.*, Vol.38, Nos.1&2, pp.61–72 (2004).
12) Kanatani, K., Sugaya, Y. and Ackermann, H.: Uncalibrated factorization using a variable symmetric affine camera, *IEICE Trans. Inf. & Sys.*, Vol.E90-D, No.5, pp.851–858 (2007).
13) Mahamud, S. and Hebert, M.: Iterative projective reconstruction from multiple views, *Proc. IEEE Conf. Comput. Vis. Patt. Recog.*, Hilton Head Island, SC, U.S.A., Vol.2, pp.430–437 (2000).
14) Mahamud, S., Hebert, M., Omori, Y. and Ponce, J.: Provably-convergent iterative methods for projective structure from motion, *Proc. IEEE Conf. Comput. Vis. Patt. Recog.*, Kauai, HI, U.S.A., Vol.1, pp.1018–1025 (2001).
15) Morita, T. and Kanade, K.: A sequential factorization method for recovering shape and motion from image streams, *IEEE Trans. Patt. Anal. Mach. Intell.*, Vol.19, No.8, pp.858–867 (1997).
16) Pollefeys, M., Koch, R. and Van Gool, L.: Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters, *Int. J. Comput. Vis.*, Vol.32, No.1, pp.7–25 (1999).
17) Ralston, R.: *A First Course in Numerical Analysis*, McGraw-Hill, New York, NY, U.S.A. (1965).
18) Seo, Y. and Heyden, A.: Auto-calibration by linear iteration using the DAC equation, *Image Vis. Comput.*, Vol.22, No.11, pp.919–926 (2004).
19) Seo, Y. and Hong, K.-S.: A linear metric reconstruction by complex eigen-decomposition, *IEICE Trans. Inf. & Syst.*, Vol.E84-D, No.12, pp.1626–1632 (2001).
20) Sturm, P. and Triggs, B.: A factorization based algorithm for multi-image projective

structure and motion, *Proc. 4th Euro. Conf. Comput. Vis.*, Vol.2, pp. 709–720, Cambridge, U.K. (1996).
21) Taira H. and Kanatani, K.: Comparison of eigenvalue computation speed for time-varying large-size symmetric matrices, *IEICE Tech. Rep.*, Vol.107, No.384, PRMU-135, pp. 1–6 (in Japanese).
22) Tomasi, C. and Kanade, T.: *Detection and Tracking of Point Features*, CMU Tech. Rep. CMU-CS-91-132 (1991). http://vision.stanford.edu/˜birch/klt/
23) Tomasi, C. and Kanade, T.: Shape and motion from image streams under orthography—A factorization method, *Int. J. Comput. Vis.*, Vol.9, No.2, pp.137–154 (1992).
24) Triggs, B.: Factorization methods for projective structure and motion, *Proc. IEEE Conf. Comput. Vis. Patt. Recog.*, San Francisco, CA, U.S.A., pp.845–851 (1996).
25) Triggs, B.: Autocalibration and the absolute quadric, *Proc. IEEE Conf. Compt. Vis. Patt. Recog.*, San Juan, Puerto Rico, pp.609–614 (1997).
26) Triggs, B., McLauchlan, P.F., Hartley, R.I. and Fitzgibbon, A.: Bundle adjustment—A modern synthesis, in Triggs, B., Zisserman, A. and Szeliski R. (Eds.), *Vision Algorithms: Theory and Practice*, Springer, Berlin (2000).
27) Ueshiba, T. and Tomita, F.: A factorization method for perspective and Euclidean reconstruction from multiple perspective views via iterative depth estimation, *Proc. 5th Euro. Conf. Comput. Vis.*, Freiburg, Germany, Vol.1, pp.296–310 (1998).
28) Vidal, R. and Hartley, R.: Motion segmentation with missing data using PowerFactorization and GPCA, *Proc. IEEE Comput. Vis. Patt. Recog.*, Washington, D.C., Vol.2, pp.310–316 (2004).

(Received May 10, 2007)
(Accepted December 5, 2007)

(Editor in Charge: *Ken Masuda*)

Hanno Ackermann studied Computer Engineering at the University of Mannheim, Mannheim, Germany. Currently, he is working for his Ph.D. at the Department of Computer Science, Okayama Universisity, Okayama, Japan. His research interests include image processing, computer vision and pattern recognition.

Kenichi Kanatani received his B.E., M.S., andPh.D. in applied mathematics from the University of Tokyo in 1972, 1974 and 1979, respectively. After serving as Professor of computer science at Gunma University, Gunma, Japan, he is currently Professor of computer science at Okayama University, Okayama, Japan. He is an IEEE Fellow.