Iterative Low Complexity Factorization for Projective Reconstruction

Hanno Ackermann and Kenichi Kanatani

Department of Computer Science
Okayama University, 700-8530 Okayama, Japan
{hanno,kanatani}@suri.it.okayama-u.ac.jp

Abstract. We present a highly efficient method for estimating the structure and motion from image sequences taken by uncalibrated cameras. The basic principle is to do projective reconstruction first followed by Euclidean upgrading. However, the projective reconstruction step dominates the total computational time, because we need to solve eigenproblems of matrices whose size depends on the number of frames or feature points. In this paper, we present a new algorithm that yields the same solution using only matrices of *constant* size irrespective of the number of frames or points. We demonstrate the superior performance of our algorithm, using synthetic and real video images.

1 Introduction

Various methods are known for recovering the structure and motion of a rigid body from image sequences taken by uncalibrated cameras [4]. A popular approach is to reconstruct the structure up to projectivity first and then to modify it into a correct Euclidean shape. The first stage is called *projective reconstruction*, the second *Euclidean upgrading*.

Some algorithms recover the projective structure by utilizing so-called multilinear constraints, which relate corresponding points in different images. For two images, this constraint is expressed by the fundamental matrix. For three and four views, the trifocal and quadrifocal tensors relate corresponding points, respectively. The use of multilinear constraints means, however, that particular pairs, triplets, or quadruples of images are chosen. For affine reconstruction, on the other hand, the popular Tomasi-Kanade factorization [11] handles all available data in a uniform way. Following this idea, Sturm and Triggs [9] extended the factorization formalism to perspective projection. However, their algorithm still relies on pair-wise epipolar constraints.

Later, Heyden et al. [5] and Mahamud and Hebert [7] introduced new factorization algorithms for projective reconstruction. They utilized the fact that the joint coordinates of rigidly moving 3-D points are constrained to be in a 4-dimensional subspace. They estimated the unknown projective depths by minimizing the distances of these joint coordinates to the subspace, alternating the subspace fitting and the projective depth optimization. These algorithms are summarized in [6].

However, it has been recognized that the projective depth optimization requires a long computation time due to the necessity to iteratively solve eigenproblems of matrices whose size depends on the number of frames or the number of points. Recently, Ackermann and Kanatani [1] proposed a simple scheme for accelerating the projective depth optimization, replacing the eigenvalue computation with the power method coupled with further acceleration techniques. They demonstrated that the computation time can significantly reduce.

This work further improves their method. We show that the projective depth optimization can be done by solving eigenproblems of matrices of *constant* size irrespective of the number of frames or points. In Section 2, we analyze two projective reconstruction algorithms, geometrically dual to each other [1]: one is due to Mahamud and Hebert [7]; the other due to Heyden et al. [5]. In Section 3, we show how the complexity of these algorithms can be reduced. We also point out that the use of SVD for subspace fitting is not necessary at each iteration. Using synthetic and real video images, we demonstrate in Section 4 that our algorithm outperforms the method of Ackermann and Kanatani [1]. Section 5 concludes this paper.

2 Projective Structure and Motion

2.1 Rank Constraint

Suppose N rigidly moving points are tracked through M consecutive frames taken by an uncalibrated camera (possibly with varying intrinsic parameters). The perspective projection of the αth point onto the κth frame can be modeled by

$$z_{\kappa\alpha}\boldsymbol{x}_{\kappa\alpha} = \Pi_\kappa X_\alpha, \qquad \boldsymbol{x}_{\kappa\alpha} = \begin{pmatrix} u_{\kappa\alpha}/f_0 \\ v_{\kappa\alpha}/f_0 \\ 1 \end{pmatrix}, \tag{1}$$

where $(u_{\kappa\alpha}, v_{\kappa\alpha})$ denote the measured point positions, and f_0 is a scale factor for numerical stabilization [3]. Here, $z_{\kappa\alpha}$ are scalars called *projective depths*, Π_κ are 3×4 projection (or camera) matrices consisting of the intrinsic and extrinsic parameters of the κth frame, and X_α are the 3-dimensional positions of the αth point in homogeneous coordinates. If we define the $3M \times N$ matrix W, the $3M \times 4$ matrix Π, and the $4 \times N$ matrix X by

$$W = \begin{pmatrix} z_{11}\boldsymbol{x}_{11} & \cdots & z_{1N}\boldsymbol{x}_{1N} \\ \vdots & \ddots & \vdots \\ z_{M1}\boldsymbol{x}_{M1} & \cdots & z_{MN}\boldsymbol{x}_{MN} \end{pmatrix}, \quad \Pi = \begin{pmatrix} \Pi_1 \\ \vdots \\ \Pi_M \end{pmatrix}, \quad X = \begin{pmatrix} X_1 \cdots X_N \end{pmatrix}, \tag{2}$$

the first of Eqs. (1) for $\kappa = 1, ..., M$ and $\alpha = 1, ..., N$ is written simply as

$$W = \Pi X. \tag{3}$$

Since Π is $3M \times 4$ and X is $4 \times N$, the matrix W generally has rank 4.

Let $\boldsymbol{w}_\alpha$ be the αth column of W. The absolute scale of the projective depths is indeterminate, because multiplying $z_{\kappa\alpha}$, $\kappa = 1, ..., M$, by a constant c is equivalent to multiplying the homogeneous coordinate vector X_α, whose scale is indeterminate, by c. Hence, we can normalize each $\boldsymbol{w}_\alpha$ to $\|\boldsymbol{w}_\alpha\| = 1$. Now, we rewrite $\boldsymbol{w}_\alpha$ as follows:

$$\boldsymbol{w}_\alpha = D_\alpha \boldsymbol{\xi}_\alpha, \tag{4}$$

$$D_\alpha \equiv \begin{pmatrix} \boldsymbol{x}_{1\alpha}/\|\boldsymbol{x}_{1\alpha}\| & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \boldsymbol{x}_{M\alpha}/\|\boldsymbol{x}_{M\alpha}\| \end{pmatrix}, \quad \boldsymbol{\xi}_\alpha \equiv \begin{pmatrix} z_{1\alpha}\|\boldsymbol{x}_{1\alpha}\| \\ \vdots \\ z_{M\alpha}\|\boldsymbol{x}_{M\alpha}\| \end{pmatrix}. \tag{5}$$

We see that $\boldsymbol{\xi}_\alpha$ is a unit vector:

$$\|\boldsymbol{\xi}_\alpha\|^2 = z_{1\alpha}^2\|\boldsymbol{x}_{1\alpha}\|^2 + \cdots + z_{M\alpha}^2\|\boldsymbol{x}_{M\alpha}\|^2 = \|\boldsymbol{w}_\alpha\|^2 = 1. \tag{6}$$

Alternatively, let $\boldsymbol{w}_\kappa^i$ be the $(3\kappa + i)$th row of W for $i = 1, 2, 3$. This time, we adopt normalization $\|\boldsymbol{w}_\kappa^1\|^2 + \|\boldsymbol{w}_\kappa^2\|^2 + \|\boldsymbol{w}_\kappa^3\|^2 = 1$ for each κ. We rewrite $\boldsymbol{w}_\kappa^i$ as

$$\boldsymbol{w}_\kappa^i = D_\kappa^i \boldsymbol{\xi}_\kappa, \tag{7}$$

$$D_\kappa^i \equiv \begin{pmatrix} x_{\kappa 1}^i/\|\boldsymbol{x}_{\kappa 1}\| & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & x_{\kappa M}^i/\|\boldsymbol{x}_{\kappa M}\| \end{pmatrix}, \quad \boldsymbol{\xi}_\kappa \equiv \begin{pmatrix} z_{\kappa 1}\|\boldsymbol{x}_{\kappa 1}\| \\ \vdots \\ z_{\kappa N}\|\boldsymbol{x}_{\kappa N}\| \end{pmatrix}, \tag{8}$$

where $x_{\kappa\alpha}^i$ denotes the ith component of $\boldsymbol{x}_{\kappa\alpha}$. We see that $\boldsymbol{\xi}_\kappa$ is a unit vector:

$$\|\boldsymbol{\xi}_\kappa\|^2 = z_{\kappa 1}^2\|\boldsymbol{x}_{\kappa 1}\|^2 + \cdots + z_{\kappa M}^2\|\boldsymbol{x}_{\kappa M}\|^2 = \|\boldsymbol{w}_\kappa^1\|^2 + \|\boldsymbol{w}_\kappa^2\|^2 + \|\boldsymbol{w}_\kappa^3\|^2 = 1. \tag{9}$$

In the following, we consider two methods. In one, we regard each column $\boldsymbol{w}_\alpha$ of W as a point in $3M$ dimensions; in the other, each triplet of rows $\{\boldsymbol{w}_\kappa^1, \boldsymbol{w}_\kappa^2, \boldsymbol{w}_\kappa^3\}$ of W is regarded as a point in $3N$ dimensions. The first, which we call the *primal* method, uses Eq. (4) for computing W; the second, which we call the *dual* method, uses Eq. (7). Both methods utilize the rank constraint on W.

2.2 Primal Method

Ideally, i.e., if the projective depths $z_{\kappa\alpha}$ are correctly chosen, and if there is no noise in the data $\boldsymbol{x}_{\kappa\alpha}$, the matrix W should have rank 4. Hence, all columns $\boldsymbol{w}_\alpha$ of W should be constrained to be in a 4-dimensional space $\mathcal{L}$. Let U be the $3M \times 4$ matrix consisting of its orthonormal basis vectors as its columns. Then, we can identify U with Π and X with $U^\top W$, which ensures $W = \Pi X$ due to the orthonormality of U. We define the *reprojection error* by

$$\epsilon = f_0 \sqrt{\frac{1}{MN}\sum_{\kappa=1}^{M}\sum_{\alpha=1}^{N}\|\boldsymbol{x}_{\kappa\alpha} - Z[\Pi_\kappa X_\alpha]\|^2}, \tag{10}$$

where $Z[\,\cdot\,]$ denotes normalization to make the third component 1. This reprojection error should ideally be 0, but if the projective depths $z_{\kappa\alpha}$ are not correct or there is noise in the data $\boldsymbol{x}_{\kappa\alpha}$, it may not exactly be zero.

For starting the algorithm, the unknown projective depths $z_{\kappa\alpha}$ are initialized to 1, which amounts to assuming affine cameras. Then, we estimate the $\mathcal{L}$ by least squares. Namely, we regard the four eigenvectors of $WW^\top$ for the largest four eigenvalues, or equivalently the four left singular vectors of W for the largest four singular values, as an orthonormal basis of $\mathcal{L}$.

Since the assumed $z_{\kappa\alpha}$ may not be correct, the columns $\boldsymbol{w}_\alpha$ of W may not exactly be in the subspace $\mathcal{L}$. So, we update $z_{\kappa\alpha}$ so as to minimize the distance of each $\boldsymbol{w}_\alpha$ from $\mathcal{L}$. Since $\boldsymbol{w}_\alpha$ is a unit vector, the solution is obtained by maximizing its orthogonal projection onto $\mathcal{L}$, namely maximizing

$$\left\|U^\top \boldsymbol{w}_\alpha\right\|^2 = \left\|U^\top D_\alpha \boldsymbol{\xi}_\alpha\right\|^2 = \boldsymbol{\xi}_\alpha^\top (D_\alpha^\top U U^\top D_\alpha)\boldsymbol{\xi}_\alpha, \tag{11}$$

subject to $\|\boldsymbol{\xi}_\alpha\| = 1$. The solution is given by the unit eigenvector of $D_\alpha^\top U U^\top D_\alpha$ for the largest eigenvalue. Since the sign of the eigenvector $\boldsymbol{\xi}_\alpha$ is indeterminate, we choose the one for which the sum of its components is non-negative.

Next, we update each column $\boldsymbol{w}_\alpha$ of W by Eq. (4) and identify $\mathit{\Pi}$ with U and X with $U^\top W$. From the updated W, we compute the matrix U for the new basis of $\mathcal{L}$. Then, we update $\boldsymbol{\xi}_\alpha$ by minimizing Eq. (11) and repeat this procedure until the reprojection error in Eq. (10) becomes smaller than some pre-defined threshold $\epsilon_{\min}$. This process is a kind of EM algorithm, so the convergence is guaranteed.

2.3 Dual Method

Since the matrix W should ideally have rank 4, all its rows $\boldsymbol{w}_\kappa^i$ should also be constrained to be in a 4-dimensional space $\mathcal{L}^\star$. Let V be the $N \times 4$ matrix consisting of its orthonormal basis vectors as its columns. Then, we can identify $V^\top$ with X and $\mathit{\Pi}$ with WV, which ensures $W = \mathit{\Pi} X$ due to the orthonormality of V.

For starting the algorithm, the unknown projective depths $z_{\kappa\alpha}$ are again initialized to 1. Then, we estimate the subspace $\mathcal{L}^\star$ by least squares. Namely, we regard the four eigenvectors of $W^\top W$ for the largest four eigenvalues, or equivalently the four right singular vectors of W for the largest four singular values, as an orthonormal basis of $\mathcal{L}^\star$.

Since the assumed $z_{\kappa\alpha}$ may not be correct, the rows $\boldsymbol{w}_\kappa^i$ of W may not exactly be in the subspace $\mathcal{L}^\star$. So, we update $z_{\kappa\alpha}$ so as to minimize the sum of the square distances of the triplet $\boldsymbol{w}_\kappa^i$, $i = 1, 2, 3$, from $\mathcal{L}^\star$. Since the sum of squares of $\boldsymbol{w}_\kappa^i$, $i = 1, 2, 3$, is normalized to 1, the solution is obtained by maximizing the sum of the squares of their orthogonal projections onto $\mathcal{L}^\star$,

$$\sum_{i=1}^{3} \left\|V^\top \boldsymbol{w}_\kappa^i\right\|^2 = \sum_{i=1}^{3} \left\|V^\top D_\kappa^i \boldsymbol{\xi}_\kappa\right\|^2 = \boldsymbol{\xi}_\kappa^\top \Big(\sum_{i=1}^{3} D_\kappa^i V V^\top D_\kappa^i\Big)\boldsymbol{\xi}_\kappa, \tag{12}$$

subject to $\|\boldsymbol{\xi}_\kappa\| = 1$. The solution is given by the unit eigenvector of $\sum_{i=1}^{3} D_\kappa^i V V^\top D_\kappa^i$ for the largest eigenvalue. Again, we choose the sign of $\boldsymbol{\xi}_\kappa$ in such a way that the sum of its components is non-negative.

Next, we update each row $\boldsymbol{w}_\kappa^i$ of W by Eq. (7) and identify X with $V^\top$ and $\mathit{\Pi}$ with WV. From the updated W, we compute the matrix V for the new basis of $\mathcal{L}$. Then, we update $\boldsymbol{\xi}_\kappa$ by minimizing Eq. (12) and repeat this procedure until the reprojection error in Eq. (10) becomes smaller than some pre-defined threshold $\epsilon_{\min}$. As in the case of the primal method, this process is guaranteed to converge.

3 Complexity Analysis and Efficiency Improvement

3.1 Subspace Fitting

The basis of the subspace $\mathcal{L}$ for the primal method is obtained by computing the eigenvectors of $WW^\top$, and the basis of the subspace $\mathcal{L}^\star$ for the dual method is obtained by computing the eigenvectors of $W^\top W$. Alternatively, they are obtained by computing the singular vectors of W. We first analyze the complexity of this subspace fitting computation.

For the primal method, computing $WW^\top$ requires $(3M)^2N$ operations, where we regard multiplication followed by addition/subtraction as one operation[1]. Its eigenvalue computation takes approximately $(3M)^3$ operations, so we need about $(3M)^2(3M+N)$ operations in total. For the dual method, computing $W^\top W$ requires $3MN^2$ operations, and its eigenvalue computation takes approximately N^3 operations. So, we need about $N^2(3M+N)$ operations in total. The singular value decomposition (SVD) of W, on the other hand, requires about $(3M)^2N$ or $3MN^2$ operations, depending on whether $3M$ is smaller or larger than N. In whichever case, SVD is obviously more efficient.

This subspace fitting is repeated each time the projective depths are updated. However, we can expect that the basis computed in each step does not differ much from the preceding one. Hence, we can save the SVD computation time if the new basis can be updated by a small number of operations. For this, we use the power method [1]: the basis vectors are multiplied by $WW^\top$ (primal) or $W^\top W$ (dual) followed by Gram-Schmidt orthogonalization [2].

As pointed out earlier, computing $WW^\top$ and $W^\top W$ requires $(3M)^2N$ and $3MN^2$ operations, respectively. Multiplying a $3M$-dimensional vector by $WW^\top$ costs $(3M)^2$ operations, and multiplying an N-dimensional vector by $W^\top W$ costs N^2 operations. Hence, the complexity of the power method is $(3M)^2N + 4k(3M)^2$ and $3MN^2 + 4kN^2$ for the primal and dual methods, respectively, where k is the number of iterations for the power method to converge.

Alternatively, multiplication by $WW^\top$ can be replaced by multiplication by $W^\top$ ($3MN$ operations) followed multiplication by W ($3MN$ operations). Similarly, multiplication by $W^\top W$ can be replaced by multiplication by W ($3MN$

[1] We disregard the fact that for computing the sum of products the number of addition/subtraction is smaller than the number of multiplication by one.

operations) followed multiplication by $W^\top$ ($3MN$ operations). Hence, this alternative is advantageous if

$$(3M)^2N + 4k(3M)^2 \geq 24kMN, \quad \text{or} \quad M \geq \frac{8k}{3 + 12k/N}, \tag{13}$$

$$3MN^2 + 4kN^2 \geq 24kMN, \quad \text{or} \quad N \geq \frac{24k}{3 + 4k/M}, \tag{14}$$

for the primal and dual methods, respectively. Equation (13) is satisfied for $M \geq 3k$, and Eq. (14) is satisfied for $N \geq 8k$. According to our experiments, the power method, whose convergence is theoretically guaranteed [2], almost always converges after one or two iterations to sufficient precision for both the primal and the dual methods. So, we adopt this alternative scheme.

3.2 Projective Depth Optimization

For computing the projective depths $z_{\kappa\alpha}$, we need to compute the dominant eigenvector (i.e., the eigenvector for the largest eigenvalue) of an $M \times M$ matrix for the primal method and of an $N \times N$ matrix for the dual method. Since solving an eigenproblem has approximately cubic complexity, the projective depth optimization dominates the entire projective reconstruction computation when the number N of points or the number M of frames is large. To resolve this problem, Ackermann and Kanatani [1] introduced a special variant of the power method for eigenvalue computation and demonstrated that this can reduce the computation time significantly. We now introduce a method that results in even better performance.

For the primal method, we need to compute the dominant eigenvector $\boldsymbol{\xi}_\alpha$ of the $M \times M$ matrix of $A_\alpha \equiv D_\alpha^\top U U^\top D_\alpha$ (see Eq. (11)), which can be written as

$$A_\alpha = (U^\top D_\alpha)^\top \underbrace{(U^\top D_\alpha)}_{\equiv\, C_\alpha} = C_\alpha^\top C_\alpha, \tag{15}$$

where C_α is a $4 \times M$ matrix. However, $\boldsymbol{\xi}_\alpha$ can be obtained from the dominant eigenvector $\boldsymbol{\eta}_\alpha$ of the 4×4 matrix

$$\tilde{A}_\alpha = C_\alpha C_\alpha^\top, \tag{16}$$

in the form of $\boldsymbol{\xi}_\alpha = C_\alpha^\top \boldsymbol{\eta}_\alpha$. In fact, if $C_\alpha C_\alpha^\top \boldsymbol{\eta}_\alpha = \lambda \boldsymbol{\eta}_\alpha$, we have $C_\alpha^\top C_\alpha C_\alpha^\top \boldsymbol{\eta}_\alpha = \lambda C_\alpha^\top \boldsymbol{\eta}_\alpha$, or $A_\alpha \boldsymbol{\xi} = \lambda \boldsymbol{\xi}_\alpha$.

Once C_α is evaluated, computing A_α costs $4M^2$ additional operations. Its eigenproblem has a complexity of M^3. Computing $\tilde{A}_\alpha$ costs $16M$ additional operations, and its eigenproblem has a *constant* complexity of 64. Hence, it is advantageous to solve the eigenproblem of $\tilde{A}_\alpha$ if $4M^2 + M^3 \geq 16M + 64$, or $M \geq 4$. Theoretically, we may be able to accelerate the dominant eigenvector computation of $\tilde{A}_\alpha$ by the power method, but it has no practical merit; the standard tool is sufficiently fast for such a small size.

For the dual method, we need to compute the dominant eigenvector $\boldsymbol{\xi}_\kappa$ of the $N \times N$ matrix $B_\kappa \equiv \sum_{i=1}^{3} D_\kappa^i V V^\top D_\kappa^i$ (see Eq. (12)), which can be written as

$$B_\kappa = \begin{pmatrix} D_\kappa^1 V & D_\kappa^2 V & D_\kappa^3 V \end{pmatrix} \underbrace{\begin{pmatrix} V^\top D_\kappa^1 \\ V^\top D_\kappa^2 \\ V^\top D_\kappa^3 \end{pmatrix}}_{\equiv\, C_\kappa} = C_\kappa^\top C_\kappa, \tag{17}$$

where C_κ is a $12 \times N$ matrix. As in the case of the primal method, however, $\boldsymbol{\xi}_\kappa$ can be obtained from the dominant eigenvector $\boldsymbol{\eta}_\kappa$ of the 12×12 matrix

$$\tilde{B}_\kappa = C_\kappa C_\kappa^\top, \tag{18}$$

in the form of $\boldsymbol{\xi}_\kappa = C_\kappa^\top \boldsymbol{\eta}_\kappa$.

Once C_κ is evaluated, computing B_κ costs $12N^2$ additional operations. Its eigenproblem has a complexity of N^3. Computing $\tilde{B}_\alpha$ costs $144N$ additional operations, and its eigenproblem has a *constant* complexity of 1728. Hence, it is advantageous to solve the eigenproblem of $\tilde{B}_\alpha$ if $12N^2 + N^3 \geq 144N + 1728$, or $N \geq 12$. As in the case of the primal method, we need not consider further speedup; the standard tool is sufficient.

In the above analysis, we disregarded the fact that the $M \times M$ matrix A_α and the $N \times N$ matrix B_κ are symmetric and hence only their upper-triangular and diagonal elements need to be evaluated. This reduces the evaluation cost to nearly 50%, yet the advantage of dealing with the 4×4 matrix $\tilde{A}_\alpha$ and the 12×12 matrix $\tilde{B}_\kappa$ is unchanged. The actual procedure of the above computation is summarized in Algorithms (1) and (2).

In Algorithms (1) and (2), the reprojection error ϵ (Step 6) is evaluated at each iteration step, but this can be omitted; we can simply stop when all values cease to change by setting an appropriate threshold. This can further speed up the computation, since the reprojection error evaluation takes time proportional to the number of points and the number of frames. This also avoids setting too low a reprojection error ϵ, which may never be reached. The reprojection error evaluation in Step 6 is solely for comparing the convergence performance for a common threshold, which is the theme of this paper.

4 Experimental Results

4.1 Synthetic Example

We now experimentally evaluate the efficiency of our algorithm. Figure 1 shows six out of 256 frames of a synthetic image sequence. The object consists of 256 points inside a cuboid. They are perspectively projected onto 512×512 frames with focal length $f = 600$. Using this motion sequence, we compare our algorithm with the original form [6], which we call the *prototype*, and the method of Ackermann and Kanatani [1], which we call the *power method* for short. For our algorithm, we set the reprojection error threshold to $\epsilon_{\min} = 0.1$ (pixels) and

Algorithm 1 Fast Projective Reconstruction: Primal Method

Input: Data vectors $\boldsymbol{x}_{\kappa\alpha}$, their norms $\|\boldsymbol{x}_{\kappa\alpha}\|$, the normalized vectors $N[\boldsymbol{x}_{\kappa\alpha}] = \boldsymbol{x}_{\kappa\alpha}/\|\boldsymbol{x}_{\kappa\alpha}\|$ for $\kappa = 1, .., M$, $\alpha = 1, .., N$, the minimal reprojection error $\epsilon_{\min}$, and the threshold δ for terminating the subspace update.
Output: Projection matrices Π_κ and 3-D points X_α.
Initialize the projective depths to $z_{\kappa\alpha} = 1$.
Compute the matrix W by stacking the products $z_{\kappa\alpha}\boldsymbol{x}_{\kappa\alpha}$ into the vectors $\boldsymbol{w}_\alpha$ and normalizing them to unit length.
Compute the matrix U of the subspace $\mathcal{L}$ consisting of the first four *left* singular vectors u_1, u_2, u_3, u_4 of W.
while ϵ (in Eq. (10)) $> \epsilon_{\min}$ **do**
  for $\alpha = 1$ to N **do**
    Compute the matrix $C_\alpha = U^\top D_\alpha$.
    Compute the matrix $\tilde{A}_\alpha$.
    Compute the dominant eigenvector η_α of $\tilde{A}_\alpha$.
    Compute $\boldsymbol{\xi}_\alpha = N[C_\alpha^\top \eta_\alpha]$.
    if $\sum_{i=1}^{M} \xi_{\alpha i} < 0$ **then**
      $\boldsymbol{\xi}_\alpha = -\boldsymbol{\xi}_\alpha$.
    end if
  end for
  Update the matrix W.
  Compute the projection matrices Π_κ given by the row-triplets of $\Pi = U$.
  Compute the 3-D positions X_α given by the columns of $X = \Pi^\top W$.
  repeat
    Assign $u_i^0 \leftarrow u_i$, $i = 1, ..., 4$.
    Compute $\tilde{u}_i = W\left(W^\top u_i^0\right)$, $i = 1, .., 4$
    Let u_i, $i = 1, .., 4$, be their Gram-Schmidt orthonormalization.
  until $\max_{i=1}^{4} \sqrt{1 - \sum_{l=1}^{4} \left(u_i^\top u_l^0\right)^2} < 10^{-\delta}$.
end while

the subspace fitting threshold to $\delta = 1$. For the power method, we use the same parameters as in [1].

The computation time for the primal method is approximately linear in the number N of the points because the same computation is repeated for each point, while the optimization of each point is generally nonlinear in the number M of the frames because the eigenvalue computation is involved. The computation time for the dual method, on the other hand, is approximately linear in M but generally nonlinear in N, because the roles of points and frames are interchanged. So, we created two series of test sequences.

For the primal method, we fixed the number of points to $N = 256$ and varied M from 32 to 512 by inserting intermediate frames (the first and last frames are the same as in Fig. 1). For the dual method, the number of frames was fixed to $M = 256$, and N is varied from 32 to 512 by randomly adding new points. We used Intel Core2Duo 1.8GHz CPU having 1GB main memory with Linux as operating system.

Algorithm 2 Fast Projective Reconstruction: Dual Method

Input: Data vectors $\boldsymbol{x}_{\kappa\alpha}$, their norms $\|\boldsymbol{x}_{\kappa\alpha}\|$, the normalized vectors $N[\boldsymbol{x}_{\kappa\alpha}] = \boldsymbol{x}_{\kappa\alpha}/\|\boldsymbol{x}_{\kappa\alpha}\|$ for $\kappa = 1, .., M$, $\alpha = 1, .., N$, the minimal reprojection error $\epsilon_{\min}$, and the threshold δ for terminating the subspace update.
Output: Projection matrices Π_κ and 3-D points X_α.
Initialize the projective depths to $z_{\kappa\alpha} = 1$.
Compute the matrix W by stacking the products $z_{\kappa\alpha}\boldsymbol{x}_{\kappa\alpha}$ into the vectors $\boldsymbol{w}_\kappa^i$ and normalize them to unit length.
Compute the matrix V of the subspace $\mathcal{L}^\star$ consisting of the first four *right* singular vectors v_1, v_2, v_3, v_4 of W.
while ϵ (in Eq. (10)) $> \epsilon_{\min}$ **do**
  for $\kappa = 1$ to M **do**
    Compute the matrix $C_\kappa = \begin{pmatrix} V^\top D_\kappa^1 \\ V^\top D_\kappa^2 \\ V^\top D_\kappa^3 \end{pmatrix}$.
    Compute the matrix $\tilde{B}_\kappa$.
    Compute the dominant eigenvector η_κ of $\tilde{B}_\kappa$.
    Compute $\boldsymbol{\xi}_\kappa = N[C_\kappa^\top \eta_\kappa]$.
    if $\sum_{i=1}^N \xi_{\kappa i} < 0$ **then**
      $\boldsymbol{\xi}_\kappa = -\boldsymbol{\xi}_\kappa$.
    end if
  end for
  Update the matrix W.
  Compute the 3-D positions X_α given by the columns of $X = V$.
  Compute the projection matrices Π_κ given by the row-triplets of $\Pi = WX^\top$.
  repeat
    Assign $v_i^0 \leftarrow v_i$, $i = 1, ..., 4$.
    Compute $\tilde{v}_i = W^\top (Wv_i)$, $i = 1, .., 4$.
    Let v_i, $i = 1, .., 4$, be their Gram-Schmidt orthonormalization.
  until $\max_{i=1}^4 \sqrt{1 - \sum_{l=1}^4 \left(v_i^\top v_l^0\right)^2} < 10^{-\delta}$.
end while

Figure 2 shows the results for the primal and the dual methods on the left and right, respectively. The solid lines indicate the prototype, the dashed lines the power method, and the dotted lines our algorithm. The prototype clearly performs very poorly both for the primal and the dual methods, as compared to which the power method runs dramatically fast. As we can see from the figure, however, our algorithm is even faster than the power method.

4.2 Other Examples

Figure 3(a) shows a sequence of 11 frames (six decimated frames are displayed here) of 231 points on a cylindrical surface perspectively projected onto an image frame of size 512×512 (pixels) with focal length $f = 600$ (pixels). We set the reprojection error threshold to $\epsilon_{\min} = 0.1$ (pixels) as in the previous examples.

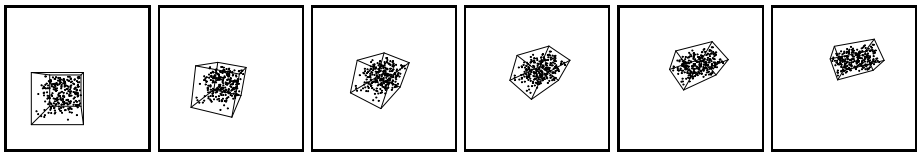

Fig. 1. Synthetic image sequence of 256 points through 256 frames (six frames decimated).

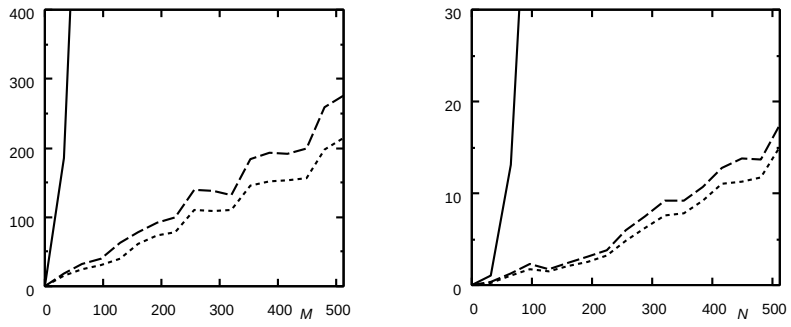

Fig. 2. Computation time (sec) for the synthetic sequence in Fig. 1 using the prototype (solid lines), the power method (dashed lines), and our algorithm (dotted lines). The left is for the primal method with $N = 256$, and the right is for the dual method with $M = 256$.

Figure 3(b) shows a real video sequence of 200 frames (six decimated frames are displayed here) of 16 points detected by KLT (the Kanade-Lucas-Tomasi tracker) [10]; we manually interfered whenever tracking failed. The frame size is 640×480 (pixels). For this sequence, we have found that the reprojection error ϵ cannot not be reduced to less than 2.1 pixels however many times the computation is iterated. This indicates that the points tracked by KLT have uncertainty of about 2.1 pixels. In fact, we visually observed that the tracked points had a few pixel fluctuations over the sequence. So, we set the reprojection error threshold to $\epsilon_{\min} = 2.1$ (pixels).

Table 1 lists the computation times (sec) for the two sequences. If the prototype is used, the primal method is more efficient than the dual method for the sequence in Fig. 3(a), because the number of points is large while the number of frames is small. For the real video sequence in Fig. 3(b), in contrast, the number of points is small while the number of frames large, so the primal method requires much longer time than the dual method.

If the power method is used, however, the dual method is faster than the primal method for both sequences. In particular, the reduction of computation time is significant for the sequence in Fig. 3(a). We can also see that the computation time of the primal method for the real video sequence in Fig. 3(b) dramatically reduces by using the power method. Yet, as we can see, our algorithm further improves efficiency.

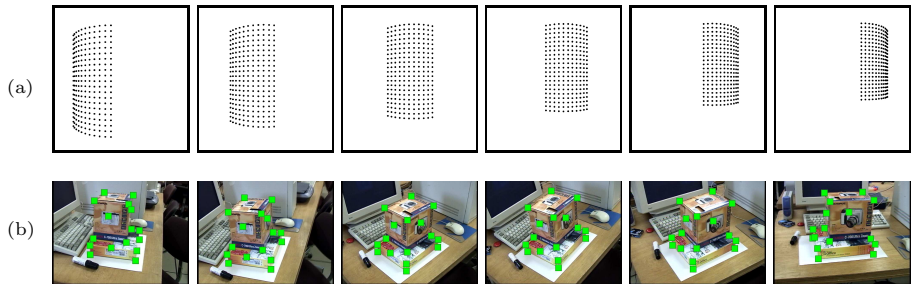

Fig. 3. (a) Synthetic image sequence of 256 points through 256 frames (six frames decimated). (b) Real video sequence of 16 points through 200 frames (six frames decimated).

Table 1. Computation times (sec) for the sequences in Figs. 3(a) (left side) and 3(b) (right side).

	Fig. 3(a)		Fig. 3(b)	
	Primal	Dual	Primal	Dual
prototype	2.385	4.029	107.338	0.073
power method	0.873	0.050	0.772	0.062
our algorithm	0.801	0.036	0.690	0.056

As for the choice of the primal vs. the dual methods, the latter is almost always preferable for practical applications whatever algorithm is used. This is because the number of frames can be arbitrarily large if a video camera is used, while the number of trackable feature points are very much limited.

5 Conclusions

We reformulated existing projective reconstruction algorithms, which solve the eigenproblem of very large size matrices, into a form that involves only small *constant* size matrices irrespective of the number of points or frames. Using synthetic and real video sequences, we demonstrated that our algorithm requires a very short time as compared with existing methods. The memory requirement also reduces significantly.

Our algorithm is based solely 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 [4, 8] and that we need to incorporate various statistical optimization techniques, a typical one being iterative refinement called *bundle adjustment* [12]. The focus of this paper is efficiency, so we have not gone into the accuracy issue, which crucially depends on the quality of the tracking data. To do statistical optimization, however, we need a good initial value, which our algorithm can very efficiently provide.

Acknowledgments: The authors thank Akinobu Mori of Canon, Inc. for initiating this project. They also thank Hirotaka Niitsuma of Okayama University for helpful discussions. This work is partly supported by Mitsubishi Precision Co., Ltd.

References

1. H. Ackermann and K. Kanatani, Robust and efficient 3-D reconstruction by self-calibration, *Proc. IAPR Conf. Machine Vision Applications*, Tokyo, Japan, May 2007, pp. 178–181.
2. T. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns-Hopkins University Press, 3rd Ed., Baltimore, MD, U.S.A., 1996.
3. R. Hartley, In defense of the 8-point algorithm, *IEEE Trans. Patt. Anal. Mach. Intell.*, **19**-6 (1997), 580–593.
4. R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Cambridge University Press, Cambridge, U.K., 2000.
5. A. Heyden, R. Berthilsson, and G. Sparr, An iterative factorization method for projective structure and motion from image sequences, *Image Vis. Comput.*, **17**-13 (1999), 981–991.
6. K. Kanatani, Latest progress of 3-D reconstruction from multiple camera images, in: X. P. Guo (Ed.), *Robotics Research Trends*, Nova Science, Hauppauge, NY, U.S.A., 2008, pp. 33–75.
7. S. Mahamud and M. Hebert, Iterative projective reconstruction from multiple views, *Proc. IEEE Conf. Comput. Vis. Patt. Recog.*, Hinton Head Island, SC, U.S.A., June 2000, Vol. 2, pp. 2430–2437.
8. M. Pollefeys, R. Koch, and L. Van Gool, Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters, *Int. J. Comput. Vis.*, **32**-1 (1999), 7–25.
9. P. Sturm and B. Triggs, A factorization based algorithm for multi-image projective structure and motion, *Proc. 4th Euro. Conf. Comput. Vis.*, Cambridge, U.K., April 1996, Vol. 2, pp. 709–720.
10. C. Tomasi and T. Kanade, *Detection and Tracking of Point Features*, CMU Tech. Rept. CMU-CS-91132, 1991; `http://vision.stanford.edu/~birch/klt/`.
11. C. Tomasi and T. Kanade, Shape and motion from image streams under orthography: A factorization method, *Int. J. Comput. Vis.*, **9**-2 (1992), 137–154.
12. B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. Fitzgibbon, Bundle adjustment—A modern synthesis, B. Triggs, A. Zisserman, and R. Szeliski (Eds.), *Vision Algorithms: Theory and Practice*, Springer, Berlin, 2000, pp. 298–375.