Memoirs of the Faculty of Engineering, Okayama University, Vol. 45, pp. 1–9, January 2011

Bundle Adjustment for 3-D Reconstruction: Implementation and Evaluation

Kenichi KANATANI*
Department of Computer Science
Okayama University
Okayama 700-8530 Japan

Yasuyuki SUGAYA
Department of Information and Computer Sciences
Toyohashi University of Technology
Toyohashi, Aichi 441-8580 Japan

(Received November 15, 2010)

We describe in detail the algorithm of bundle adjustment for 3-D reconstruction from multiple images based on our latest research results. The main focus of this paper is on the handling of camera rotations and the efficiency of computation and memory usage when the number of variables is very large; an appropriate consideration of this is the core of the implementation of bundle adjustment. Computing the fundamental matrix from two views and reconstructing the 3-D structure from multiple views, we evaluate the performance of our algorithm and discuses technical issues of bundle adjustment implementation.

1. INTRODUCTION

Bundle adjustment is a fundamental technique for computing the 3-D structure of the scene from point correspondences over multiple images. The basic principle is to search the space of all the parameters, i.e., the coordinates of all 3-D points and the intrinsic and extrinsic camera parameters of all frames, in such a way that the images of the reconstructed 3-D points reprojected using the computed camera parameters agree with the input images as much as possible [11, 12, 15]. This computation is a complicated iterative procedure, and the details have not been well documented, partly because the implementation, the treatment of rotations in particular, differs from researcher to researcher. The purpose of this paper is to describe the bundle adjustment procedure in a way considered to be the most appropriate from the viewpoint of our latest research. In particular, we highlight the treatment of rotations and the efficiency of computation and memory usage when the number of variables is very large; an appropriate consideration of this is the core of the implementation of bundle adjustment. Computing the fundamental matrix from two views and reconstructing the 3-D structure from multiple views, we evaluate the performance of our algorithm and discuss technical issues of bundle adjustment implementation.

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

2. PERSPECTIVE PROJECTION

We model the camera imaging geometry by *perspective projection*, which projects a 3-D point (X, Y, Z) onto (x, y) on the image plane by the relationship

$$\begin{pmatrix} x \\ y \\ f_0 \end{pmatrix} \simeq \boldsymbol{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \tag{1}$$

where $\simeq$ denotes equality up to a nonzero constant multiplier, and f_0 is an appropriate scaling constant[1]. The 3×4 matrix $\boldsymbol{P}$ is called the *projection matrix*. If the camera with focal length f pixels and the principal point at (u_0, v_0) is placed at $\boldsymbol{t}$ with orientation $\boldsymbol{R}$ (rotation matrix) relative to the world coordinate system, the projection matrix $\boldsymbol{P}$ has the following expression [3] ($\boldsymbol{I}$ is the unit matrix):

$$\boldsymbol{P} = \boldsymbol{K}\boldsymbol{R}^\top \begin{pmatrix} \boldsymbol{I} & -\boldsymbol{t} \end{pmatrix},$$

$$\boldsymbol{K} = \begin{pmatrix} f/f_0 & 0 & u_0/f_0 \\ 0 & f/f_0 & v_0/f_0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{2}$$

Here, we are assuming that the aspect ratio is 1 with no image skew. Eq. (2) is known as the *matrix of intrinsic parameters*. In components, Eq. (1) is written

[1]The numerical error due to finite length computation is reduced if it is taken to be of the order of the image coordinates x and y [2]. In our system, we let $f_0 = 600$ (pixels).

as

$$x = f_0 \frac{P^{11}X + P^{12}Y + P^{13}Z + P^{14}}{P^{31}X + P^{32}Y + P^{33}Z + P^{34}},$$
$$y = f_0 \frac{P^{21}X + P^{22}Y + P^{23}Z + P^{24}}{P^{31}X + P^{32}Y + P^{33}Z + P^{34}}, \quad (3)$$

where P^{ij} denotes the (ij) element of $\boldsymbol{P}$.

Suppose we take M images of N points $(X_\alpha, Y_\alpha, Z_\alpha)$, $\alpha = 1, ..., N$, in the scene. Let $(x_{\alpha\kappa}, y_{\alpha\kappa})$ be the projection of the αth point onto the κth image. Let $\boldsymbol{P}_\kappa$ be the projection matrix of the κth image. We measure the discrepancy between the observed points $(x_{\alpha\kappa}, y_{\alpha\kappa})$ and the image positions predicted by the projection matrices $\boldsymbol{P}_\kappa$ by the sum E of square distances between them over all the images. From Eq. (3), we see that E, which is called the *reprojection error*, is given by

$$E = \sum_{\alpha=1}^{N}\sum_{\kappa=1}^{M} I_{\alpha\kappa}\Big[\Big(\frac{p_{\alpha\kappa}}{r_{\alpha\kappa}} - \frac{x_{\alpha\kappa}}{f_0}\Big)^2 + \Big(\frac{q_{\alpha\kappa}}{r_{\alpha\kappa}} - \frac{y_{\alpha\kappa}}{f_0}\Big)^2\Big], \quad (4)$$

where $I_{\alpha\kappa}$ is the *visibility index*, taking 1 if the αth point is visible in the κth image and 0 otherwise. In Eq. (4), we measure the distance on the image plane with f_0 as the unit of length and define $p_{\alpha\kappa}$, $q_{\alpha\kappa}$, and $r_{\alpha\kappa}$ as follows:

$$p_{\alpha\kappa} = P_\kappa^{11}X_\alpha + P_\kappa^{12}Y_\alpha + P_\kappa^{13}Z_\alpha + P_\kappa^{14},$$
$$q_{\alpha\kappa} = P_\kappa^{21}X_\alpha + P_\kappa^{22}Y_\alpha + P_\kappa^{23}Z_\alpha + P_\kappa^{24},$$
$$r_{\alpha\kappa} = P_\kappa^{31}X_\alpha + P_\kappa^{32}Y_\alpha + P_\kappa^{33}Z_\alpha + P_\kappa^{34}. \quad (5)$$

The task of bundle adjustment is to compute the 3-D coordinates $(X_\alpha, Y_\alpha, Z_\alpha)$ and the projection matrices $\boldsymbol{P}_\kappa$ that minimize Eq. (4) from the observations $(x_{\alpha\kappa}, y_{\alpha\kappa})$, $\alpha = 1, ..., N$, $\kappa = 1, ..., M$ [11, 12, 15].

3. CORRECTION OF VARIABLES

The basic principle of bundle adjustment computation is to iteratively modify the assumed values of $(X_\alpha, Y_\alpha, Z_\alpha)$ and $\boldsymbol{P}_\kappa$ so that the reprojection error E in Eq. (4) decreases. Let $(\Delta X_\alpha, \Delta Y_\alpha, \Delta Z_\alpha)$ be the correction of $(X_\alpha, Y_\alpha, Z_\alpha)$. The projection matrix $\boldsymbol{P}_\kappa$ is determined by the focal length f_κ, the principal point $(u_{0\kappa}, v_{0\kappa})$, the translation $\boldsymbol{t}_\kappa = (t_{\kappa1}, t_{\kappa2}, t_{\kappa3})^\top$, and the rotation $\boldsymbol{R}_\kappa$. Let Δf_κ, $(\Delta u_{0\kappa}, \Delta v_{0\kappa})$, and $(\Delta t_{\kappa1}, \Delta t_{\kappa2}, \Delta t_{\kappa3})^\top$ be the corrections of f_κ, $(u_{0\kappa}, v_{0\kappa})$, and $\boldsymbol{t}_\kappa$, respectively.

Expressing the correction of $\boldsymbol{R}_\kappa$ needs care. The orthogonality relationship $\boldsymbol{R}_\kappa\boldsymbol{R}_\kappa^\top = \boldsymbol{I}$ imposes three constraints on the nine elements of $\boldsymbol{R}_\kappa$, so $\boldsymbol{R}_\kappa$ has three degrees of freedom. However, *we do not need any 3-parameter expression of* $\boldsymbol{R}_\kappa$, because what we actually need is the expression of its "correction", i.e., the *rate of change*, which mathematically means *differentiation*. From $\boldsymbol{R}_\kappa\boldsymbol{R}_\kappa^\top = \boldsymbol{I}$, we see that the change $\Delta\boldsymbol{R}_\kappa$ of $\boldsymbol{R}_\kappa$ satisfies to a first approximation $\Delta\boldsymbol{R}_\kappa\boldsymbol{R}_\kappa^\top + \boldsymbol{R}_\kappa\Delta\boldsymbol{R}_\kappa^\top = \boldsymbol{O}$, hence $(\Delta\boldsymbol{R}_\kappa\boldsymbol{R}_\kappa^\top)^\top = -\Delta\boldsymbol{R}_\kappa\boldsymbol{R}_\kappa^\top$, which means that $\Delta\boldsymbol{R}_\kappa\boldsymbol{R}_\kappa^\top$ is an antisymmetric matrix. Thus, $\Delta\boldsymbol{R}_\kappa\boldsymbol{R}_\kappa^\top$ can be expressed in terms of some $\omega_{\kappa1}$, $\omega_{\kappa2}$, $\omega_{\kappa3}$ in the form

$$\Delta\boldsymbol{R}_\kappa\boldsymbol{R}_\kappa^\top = \begin{pmatrix} 0 & -\omega_{\kappa3} & \omega_{\kappa2} \\ \omega_{\kappa3} & 0 & -\omega_{\kappa1} \\ -\omega_{\kappa2} & \omega_{\kappa1} & 0 \end{pmatrix}. \quad (6)$$

It follows that the set of these first order changes of rotation, which are called *infinitesimal rotations* in mathematics, form a 3-D linear space spanned by $\omega_{\kappa1}$, $\omega_{\kappa2}$, $\omega_{\kappa3}$, which is known as the *Lie algebra*[2] $so(3)$ of the group of rotations $SO(3)$ [4].

Let us define the product $\boldsymbol{a} \times \boldsymbol{T}$ of a vector $\boldsymbol{a}$ and a matrix $\boldsymbol{T}$ to be the matrix consisting of the vector product of $\boldsymbol{a}$ and each column of $\boldsymbol{T}$. Then, the right-hand side of Eq. (6) is the product $\boldsymbol{\omega}_\kappa \times \boldsymbol{I}$ of the vector $\boldsymbol{\omega}_\kappa = (\omega_{\kappa1}, \omega_{\kappa2}, \omega_{\kappa3})^\top$ and the unit matrix $\boldsymbol{I}$. Note that the identities $(\boldsymbol{a}\times\boldsymbol{I})\boldsymbol{b} = \boldsymbol{a}\times\boldsymbol{b}$ and $(\boldsymbol{a}\times\boldsymbol{I})\boldsymbol{T} = \boldsymbol{a}\times\boldsymbol{T}$ hold. Multiplying Eq. (6) by $\boldsymbol{R}_\kappa$ from right, we have

$$\Delta\boldsymbol{R}_\kappa = \boldsymbol{\omega}_\kappa \times \boldsymbol{R}_\kappa. \quad (7)$$

If we divide this by the time interval Δt and take the limit of $\Delta t \to 0$, we obtain the instantaneous rate of change $d\boldsymbol{R}_\kappa/dt$ of $\boldsymbol{R}_\kappa$, and the vector $\boldsymbol{\omega}_\kappa$ is identified with the *angular velocity*, as is well known in physics. Equation (7), which some researchers call the *method of Lie algebra*, is the basic expression for optimization involving rotations. This is the standard approach in physics but does not seem to be well known in the computer vision community, where the use of the Euler angles, axis-wise rotations, and the quaternion representation may be more popular. However, if we parameterize $\boldsymbol{R}_\kappa$ itself by using these, differentiation with respect to the parameters results in rather complicated expressions. The use of Eq. (7) is the simplest and the most straightforward.

4. BUNDLE ADJUSTMENT PROCEDURE

4.1 Basic Principle

As mentioned above, there are $3N + 9M$ variables to adjust for reducing the reprojection error E: ΔX_α, ΔY_α, ΔZ_α, $\alpha = 1, ..., N$, Δf_κ, $\Delta t_{\kappa1}$, $\Delta t_{\kappa2}$, $\Delta t_{\kappa3}$, $\Delta u_{0\kappa}$, $\Delta v_{0\kappa}$, $\omega_{\kappa1}$, $\omega_{\kappa2}$, $\omega_{\kappa3}$, $\kappa = 1, ..., M$. Introducing serial numbers, let us denote them by $\Delta\xi_1$, $\Delta\xi_2$, ..., $\Delta\xi_{3N+9M}$. The first order change of E caused by $\Delta\xi_k$ is obtained by ignoring second and higher order terms in the expansion of E in $\Delta\xi_k$ and is called the "derivative" of E and denote by $\partial E/\partial\xi_k$. It has the

[2]Strictly, this is called a Lie algebra if the *commutator* operation is added [4]. Here, however, the commutator does not play any role.

following form:

$$\frac{\partial E}{\partial \xi_k} = 2\sum_{\alpha=1}^{N}\sum_{\kappa=1}^{M}\frac{I_{\alpha\kappa}}{r_{\alpha\kappa}^2}\Big[\Big(\frac{p_{\alpha\kappa}}{r_{\alpha\kappa}} - \frac{x_{\alpha\kappa}}{f_0}\Big) \Big(r_{\alpha\kappa}\frac{\partial p_{\alpha\kappa}}{\partial \xi_k} - p_{\alpha\kappa}\frac{\partial r_{\alpha\kappa}}{\partial \xi_k}\Big) + \Big(\frac{q_{\alpha\kappa}}{r_{\alpha\kappa}} - \frac{y_{\alpha\kappa}}{f_0}\Big)\Big(r_{\alpha\kappa}\frac{\partial q_{\alpha\kappa}}{\partial \xi_k} - q_{\alpha\kappa}\frac{\partial r_{\alpha\kappa}}{\partial \xi_k}\Big)\Big]. \quad (8)$$

If we introduce the Gauss-Netwon approximation, the second derivative of E is given by

$$\frac{\partial^2 E}{\partial \xi_k \partial \xi_l} = 2\sum_{\alpha=1}^{N}\sum_{\kappa=1}^{M}\frac{I_{\alpha\kappa}}{r_{\alpha\kappa}^4}\Big[\Big(r_{\alpha\kappa}\frac{\partial p_{\alpha\kappa}}{\partial \xi_k} - p_{\alpha\kappa}\frac{\partial r_{\alpha\kappa}}{\partial \xi_k}\Big) \Big(r_{\alpha\kappa}\frac{\partial p_{\alpha\kappa}}{\partial \xi_l} - p_{\alpha\kappa}\frac{\partial r_{\alpha\kappa}}{\partial \xi_l}\Big) + \Big(r_{\alpha\kappa}\frac{\partial q_{\alpha\kappa}}{\partial \xi_k} - q_{\alpha\kappa}\frac{\partial r_{\alpha\kappa}}{\partial \xi_k}\Big)\Big(r_{\alpha\kappa}\frac{\partial q_{\alpha\kappa}}{\partial \xi_l} - q_{\alpha\kappa}\frac{\partial r_{\alpha\kappa}}{\partial \xi_l}\Big)\Big]. \quad (9)$$

Equations (8) and (9) imply that evaluation of the first and the second derivatives $\partial E/\partial \xi_k$ and $\partial^2 E/\partial \xi_k \partial \xi_l$ requires *only the first derivatives* $\partial p_{\alpha\kappa}/\partial \xi_k$, $\partial q_{\alpha\kappa}/\partial \xi_k$, and $\partial r_{\alpha\kappa}/\partial \xi_k$. In the following, we derive them in turn.

4.2 Derivatives for 3-D Positions

Differentiating Eqs. (5), we obtain the derivatives of $p_{\alpha\kappa}$, $q_{\alpha\kappa}$, and $r_{\alpha\kappa}$ with respect to $(X_\beta, Y_\beta, Z_\beta)$ as follows, where $\delta_{\alpha\beta}$ denotes the Kronecker delta:

$$\begin{aligned}
&\frac{\partial p_{\alpha\kappa}}{\partial X_\beta} = \delta_{\alpha\beta}P_\kappa^{11}, \quad \frac{\partial p_{\alpha\kappa}}{\partial Y_\beta} = \delta_{\alpha\beta}P_\kappa^{12}, \quad \frac{\partial p_{\alpha\kappa}}{\partial Z_\beta} = \delta_{\alpha\beta}P_\kappa^{13},\\
&\frac{\partial q_{\alpha\kappa}}{\partial X_\beta} = \delta_{\alpha\beta}P_\kappa^{21}, \quad \frac{\partial q_{\alpha\kappa}}{\partial Y_\beta} = \delta_{\alpha\beta}P_\kappa^{22}, \quad \frac{\partial q_{\alpha\kappa}}{\partial Z_\beta} = \delta_{\alpha\beta}P_\kappa^{23},\\
&\frac{\partial r_{\alpha\kappa}}{\partial X_\beta} = \delta_{\alpha\beta}P_\kappa^{31}, \quad \frac{\partial r_{\alpha\kappa}}{\partial Y_\beta} = \delta_{\alpha\beta}P_\kappa^{32}, \quad \frac{\partial r_{\alpha\kappa}}{\partial Z_\beta} = \delta_{\alpha\beta}P_\kappa^{33}.
\end{aligned} \quad (10)$$

4.3 Derivatives for Focal Lengths

Differentiating $\boldsymbol{P}$ in Eqs. (2) with respect to f, we obtain

$$\begin{aligned}
\frac{\partial \boldsymbol{P}}{\partial f} &= \begin{pmatrix}1&0&0\\0&1&0\\0&0&0\end{pmatrix}\boldsymbol{R}^\top\begin{pmatrix}\boldsymbol{I} & -\boldsymbol{t}\end{pmatrix}\\
&= \begin{pmatrix}1&0&0\\0&1&0\\0&0&0\end{pmatrix}\boldsymbol{K}^{-1}\boldsymbol{K}\boldsymbol{R}^\top\begin{pmatrix}\boldsymbol{I} & -\boldsymbol{t}\end{pmatrix}\\
&= \begin{pmatrix}1&0&0\\0&1&0\\0&0&0\end{pmatrix}\frac{1}{f}\begin{pmatrix}1&0&-u_0/f_0\\0&1&-v_0/f_0\\0&0&f/f_0\end{pmatrix}\boldsymbol{P}\\
&= \frac{1}{f}\begin{pmatrix}1&0&-u_0/f_0\\0&1&-v_0/f_0\\0&0&0\end{pmatrix}\boldsymbol{P}\\
&= \frac{1}{f}\begin{pmatrix}P^{11}-u_0P^{31}/f_0 & P^{12}-u_0P^{32}/f_0 & P^{13}-u_0P^{33}/f_0 & P^{14}-u_0P^{34}/f_0\\ P^{21}-v_0P^{31}/f_0 & P^{22}-v_0P^{32}/f_0 & P^{23}-v_0P^{33}/f_0 & P^{24}-v_0P^{34}/f_0\\ 0&0&0&0\end{pmatrix}. \quad (11)
\end{aligned}$$

Hence, the derivatives of $p_{\alpha\kappa}$, $q_{\alpha\kappa}$, and $r_{\alpha\kappa}$ with respect to f_λ are given as follows:

$$\frac{\partial p_{\alpha\kappa}}{\partial f_\lambda} = \frac{\delta_{\kappa\lambda}}{f_\kappa}\Big(p_{\alpha\kappa} - \frac{u_0}{f_0}r_{\alpha\kappa}\Big), \quad \frac{\partial q_{\alpha\kappa}}{\partial f_\lambda} = \frac{\delta_{\kappa\lambda}}{f_\kappa}\Big(q_{\alpha\kappa} - \frac{v_0}{f_0}r_{\alpha\kappa}\Big), \quad \frac{\partial r_{\alpha\kappa}}{\partial f_\lambda} = 0. \quad (12)$$

4.4 Derivatives for Principal Points

Differentiating $\boldsymbol{P}$ in Eqs. (2) with respect to u_0, we obtain

$$\begin{aligned}
\frac{\partial \boldsymbol{P}}{\partial u_0} &= \begin{pmatrix}0&0&1\\0&0&0\\0&0&0\end{pmatrix}\boldsymbol{R}^\top\begin{pmatrix}\boldsymbol{I} & -\boldsymbol{t}\end{pmatrix}\\
&= \begin{pmatrix}0&0&1\\0&0&0\\0&0&0\end{pmatrix}\boldsymbol{K}^{-1}\boldsymbol{K}\boldsymbol{R}^\top\begin{pmatrix}\boldsymbol{I} & -\boldsymbol{t}\end{pmatrix}\\
&= \begin{pmatrix}0&0&1\\0&0&0\\0&0&0\end{pmatrix}\frac{1}{f}\begin{pmatrix}1&0&-u_0/f_0\\0&1&-v_0/f_0\\0&0&f/f_0\end{pmatrix}\boldsymbol{P}\\
&= \frac{1}{f_0}\begin{pmatrix}P^{31}&P^{32}&P^{33}&P^{34}\\0&0&0&0\\0&0&0&0\end{pmatrix}. \quad (13)
\end{aligned}$$

Similarly, we obtain

$$\begin{aligned}
\frac{\partial \boldsymbol{P}}{\partial v_0} &= \begin{pmatrix}0&0&0\\0&0&1\\0&0&0\end{pmatrix}\boldsymbol{R}^\top\begin{pmatrix}\boldsymbol{I} & -\boldsymbol{t}\end{pmatrix}\\
&= \frac{1}{f_0}\begin{pmatrix}0&0&0&0\\P^{31}&P^{32}&P^{33}&P^{34}\\0&0&0&0\end{pmatrix}. \quad (14)
\end{aligned}$$

Hence, the derivatives of $p_{\alpha\kappa}$, $q_{\alpha\kappa}$, and $r_{\alpha\kappa}$ with respect to $(u_{0\lambda}, v_{0\lambda})$ are given as follows:

$$\frac{\partial p_{\alpha\kappa}}{\partial u_{0\lambda}} = \frac{\delta_{\kappa\lambda}r_{\alpha\kappa}}{f_0}, \quad \frac{\partial q_{\alpha\kappa}}{\partial u_{0\lambda}} = 0, \quad \frac{\partial r_{\alpha\kappa}}{\partial u_{0\lambda}} = 0,$$

$$\frac{\partial p_{\alpha\kappa}}{\partial v_{0\lambda}} = 0, \quad \frac{\partial q_{\alpha\kappa}}{\partial v_{0\lambda}} = \frac{\delta_{\kappa\lambda}r_{\alpha\kappa}}{f_0}, \quad \frac{\partial r_{\alpha\kappa}}{\partial u_{0\lambda}} = 0. \quad (15)$$

4.5 Derivatives for Translations

From Eqs. (2), we see that only the fourth column of $\boldsymbol{P}$ contains $\boldsymbol{t}$ in the form

$$\begin{pmatrix}P^{14}\\P^{24}\\P^{34}\end{pmatrix} = -\boldsymbol{K}\boldsymbol{R}^\top\boldsymbol{t}$$

$$= -\begin{pmatrix} (fR^{11} + u_0R^{13})t_1 + (fR^{21} + u_0R^{23})t_2 \\ \qquad +(fR^{31} + u_0R^{33})t_3 \\ (fR^{12} + v_0R^{13})t_1 + (fR^{22} + v_0R^{23})t_2 \\ \qquad +(fR^{32} + v_0R^{33})t_3 \\ f_0(R^{13}t_1 + R^{23}t_2 + R^{33}t_3) \end{pmatrix}. \tag{16}$$

Hence, we obtain

$$\frac{\partial}{\partial t_1}\begin{pmatrix} P^{14} \\ P^{24} \\ P^{34} \end{pmatrix} = -\begin{pmatrix} fR^{11} + u_0R^{13} \\ fR^{12} + v_0R^{13} \\ f_0R^{13} \end{pmatrix},$$
$$\frac{\partial}{\partial t_2}\begin{pmatrix} P^{14} \\ P^{24} \\ P^{34} \end{pmatrix} = -\begin{pmatrix} fR^{21} + u_0R^{23} \\ fR^{22} + v_0R^{23} \\ f_0R^{23} \end{pmatrix},$$
$$\frac{\partial}{\partial t_3}\begin{pmatrix} P^{14} \\ P^{24} \\ P^{34} \end{pmatrix} = -\begin{pmatrix} fR^{31} + u_0R^{33} \\ fR^{32} + v_0R^{33} \\ f_0R^{33} \end{pmatrix}. \tag{17}$$

Introducing the vector operator $\nabla_{\boldsymbol{t}_\lambda}$ for differentiation with respect to $(t_{\lambda 1}, t_{\lambda 2}, t_{\lambda 3})$, we obtain from Eqs. (5)

$$\begin{aligned} \nabla_{\boldsymbol{t}_\lambda} p_{\alpha\kappa} &= -\delta_{\kappa\lambda}(f_\kappa \boldsymbol{r}^1_\kappa + u_0 \boldsymbol{r}^3_\kappa), \\ \nabla_{\boldsymbol{t}_\lambda} p_{\alpha\kappa} &= -\delta_{\kappa\lambda}(f_\kappa \boldsymbol{r}^2_\kappa + v_0 \boldsymbol{r}^3_\kappa), \\ \nabla_{\boldsymbol{t}_\lambda} p_{\alpha\kappa} &= -\delta_{\kappa\lambda} f_0 \boldsymbol{r}^3_\kappa, \end{aligned} \tag{18}$$

where we define $\boldsymbol{r}^1_\kappa$, $\boldsymbol{r}^2_\kappa$, and $\boldsymbol{r}^3_\kappa$ as follows:

$$\boldsymbol{r}^1_\kappa = \begin{pmatrix} R^{11}_\kappa \\ R^{21}_\kappa \\ R^{31}_\kappa \end{pmatrix}, \quad \boldsymbol{r}^2_\kappa = \begin{pmatrix} R^{12}_\kappa \\ R^{22}_\kappa \\ R^{32}_\kappa \end{pmatrix}, \quad \boldsymbol{r}^3_\kappa = \begin{pmatrix} R^{13}_\kappa \\ R^{23}_\kappa \\ R^{33}_\kappa \end{pmatrix}. \tag{19}$$

4.6 Derivatives for Rotations

The first order variation of the matrix $\boldsymbol{P}$ in Eqs. (2) is given by

$$\begin{aligned} \Delta\boldsymbol{P} &= \boldsymbol{K}(\boldsymbol{\omega} \times \boldsymbol{R})^\top \begin{pmatrix} \boldsymbol{I} & -\boldsymbol{t} \end{pmatrix} \\ &= \boldsymbol{K}\boldsymbol{R}^\top \begin{pmatrix} 0 & \omega_3 & -\omega_2 & \omega_2 t_3 - \omega_3 t_2 \\ -\omega_3 & 0 & \omega_1 & \omega_3 t_1 - \omega_1 t_3 \\ \omega_2 & -\omega_1 & 0 & \omega_1 t_2 - \omega_2 t_1 \end{pmatrix}, \end{aligned} \tag{20}$$

where we have used the identities $(\boldsymbol{\omega} \times \boldsymbol{R})^\top = -\boldsymbol{R}^\top(\boldsymbol{\omega} \times \boldsymbol{I})$ and $(\boldsymbol{\omega} \times \boldsymbol{I})\boldsymbol{t} = \boldsymbol{\omega} \times \boldsymbol{t}$. The derivatives $\partial\boldsymbol{P}/\partial\omega_1$, $\partial\boldsymbol{P}/\partial\omega_2$, and $\partial\boldsymbol{P}/\partial\omega_3$ are given as follows:

$$\frac{\partial \boldsymbol{P}}{\partial \omega_1} = \begin{pmatrix} 0 & -fR^{31} - u_0R^{33} & fR^{21} + u_0R^{23} \\ 0 & -fR^{32} - v_0R^{33} & fR^{22} + v_0R^{23} \\ 0 & -f_0R^{33} & f_0R^{23} \\ f(t_2R^{31} - t_3R^{21}) + u_0(t_2R^{33} - t_3R^{23}) \\ f(t_2R^{32} - t_3R^{22}) + v_0(t_2R^{33} - t_3R^{23}) \\ f_0(t_2R^{33} - t_3R^{23}) \end{pmatrix},$$
$$\frac{\partial \boldsymbol{P}}{\partial \omega_2} = \begin{pmatrix} fR^{31} + u_0R^{33} & 0 & -fR^{11} - u_0R^{13} \\ fR^{32} + v_0R^{33} & 0 & -fR^{12} - v_0R^{13} \\ f_0R^{33} & 0 & -f_0R^{13} \\ f(t_3R^{11} - t_1R^{31}) + u_0(t_3R^{13} - t_1R^{33}) \\ f(t_3R^{12} - t_1R^{32}) + v_0(t_3R^{13} - t_1R^{33}) \\ f_0(t_3R^{13} - t_1R^{33}) \end{pmatrix},$$
$$\frac{\partial \boldsymbol{P}}{\partial \omega_3} = \begin{pmatrix} -fR^{21} - u_0R^{23} & fR^{11} + u_0R^{13} & 0 \\ -fR^{22} - v_0R^{23} & fR^{12} + v_0R^{13} & 0 \\ -f_0R^{23} & f_0R^{13} & 0 \\ f(t_1R^{21} - t_2R^{11}) + u_0(t_1R^{23} - t_2R^{13}) \\ f(t_1R^{22} - t_2R^{12}) + v_0(t_1R^{23} - t_2R^{13}) \\ f_0(t_1R^{23} - t_2R^{13}) \end{pmatrix}, \tag{21}$$

Introducing the vector operator $\nabla_{\boldsymbol{\omega}_\lambda}$ for differentiation with respect to $(\omega_{\lambda 1}, \omega_{\lambda 2}, \omega_{\lambda 3})$, we obtain from Eqs. (5)

$$\begin{aligned} \nabla_{\boldsymbol{\omega}_\lambda} p_{\alpha\kappa} &= \delta_{\kappa\lambda}(f_\kappa \boldsymbol{r}^1_\kappa + u_{0\kappa} \boldsymbol{r}^3_\kappa) \times (\boldsymbol{X}_\alpha - \boldsymbol{t}_\kappa), \\ \nabla_{\boldsymbol{\omega}_\lambda} q_{\alpha\kappa} &= \delta_{\kappa\lambda}(f_\kappa \boldsymbol{r}^2_\kappa + v_{0\kappa} \boldsymbol{r}^3_\kappa) \times (\boldsymbol{X}_\alpha - \boldsymbol{t}_\kappa), \\ \nabla_{\boldsymbol{\omega}_\lambda} r_{\alpha\kappa} &= \delta_{\kappa\lambda} f_0 \boldsymbol{r}^3_\kappa \times (\boldsymbol{X}_\alpha - \boldsymbol{t}_\kappa), \end{aligned} \tag{22}$$

where we define $\boldsymbol{X}_\alpha = (X_\alpha, Y_\alpha, Z_\alpha)^\top$.

5. LEVENBERG-MARQUARDT METHOD

The Levenberg-Marquardt (LM) procedure that minimizes the reprojection error E go as follows [13]:

1. Provide initial values for $\boldsymbol{X}_\alpha$, f_κ, $(u_{0\kappa}, v_{0\kappa})$, $\boldsymbol{t}_\kappa$, and $\boldsymbol{R}_\kappa$, and compute the corresponding reprojection error E. Let $c = 0.0001$.

2. Compute the first and second derivatives $\partial E/\partial\xi_k$ and $\partial^2 E/\partial\xi_k\partial\xi_l$, k, $l = 1, ..., 3N + 9M$.

3. Solve the linear equation

$$\begin{pmatrix} (1+c)\partial^2 E/\partial\xi_1^2 & \partial^2 E/\partial\xi_1\partial\xi_2 & \cdots & \partial^2 E/\partial\xi_1\partial\xi_{3N+9M} \\ \partial^2 E/\partial\xi_2\partial\xi_1 & (1+c)\partial^2 E/\partial\xi_2^2 & \cdots & \partial^2 E/\partial\xi_2\partial\xi_{3N+9M} \\ \vdots & \vdots & \ddots & \vdots \\ \partial^2 E/\partial\xi_{3N+9M}\partial\xi_1 & \partial^2 E/\partial\xi_{3N+9M}\partial\xi_2 & \cdots & (1+c)\partial^2 E/\partial\xi_{3N+9M}^2 \end{pmatrix} \begin{pmatrix} \Delta\xi_1 \\ \Delta\xi_2 \\ \vdots \\ \Delta\xi_{3N+9M} \end{pmatrix}$$
$$= -\begin{pmatrix} \partial E/\partial\xi_1 \\ \partial E/\partial\xi_2 \\ \vdots \\ \partial E/\partial\xi_{3N+9M} \end{pmatrix}, \tag{23}$$

   for $\Delta\xi_k$, $k = 1, ..., 3N + 9M$.

4. Update $\boldsymbol{X}_\alpha$, f_κ, $(u_{0\kappa}, v_{0\kappa})$, and $\boldsymbol{t}_\kappa$, $\boldsymbol{R}_\kappa$ by

$$\begin{aligned} \tilde{\boldsymbol{X}}_\alpha &\leftarrow \boldsymbol{X}_\alpha + \Delta\boldsymbol{X}_\alpha, \\ \tilde{f}_\kappa &\leftarrow f_\kappa + \Delta f_\kappa, \quad (\tilde{u}_{0\kappa}, \tilde{v}_{0\kappa}) \leftarrow (u_{0\kappa}, v_{0\kappa}), \\ \tilde{\boldsymbol{t}}_\kappa &\leftarrow \boldsymbol{t}_\kappa + \Delta\boldsymbol{t}_\kappa, \quad \tilde{\boldsymbol{R}}_\kappa \leftarrow \mathcal{R}(\boldsymbol{\omega}_\kappa)\boldsymbol{R}_\kappa, \end{aligned} \tag{24}$$

where $\mathcal{R}(\boldsymbol{\omega}_\kappa)$ denotes the rotation by angle $\|\boldsymbol{\omega}_\kappa\|$ around axis $\mathcal{N}[\boldsymbol{\omega}_\kappa]$ screwwise (the *Rodriguez formula*[3])

5. Compute the reprojection error $\tilde{E}$ corresponding to $\tilde{\boldsymbol{X}}_\alpha$, $\tilde{f}_\kappa$, $(\tilde{u}_{0\kappa}, \tilde{v}_{0\kappa})$, $\tilde{\boldsymbol{t}}_\kappa$, and $\tilde{\boldsymbol{R}}_\kappa$. If $\tilde{E} > E$, let $c \leftarrow 10c$ and go back to Step 3.

6. Let

$$\boldsymbol{X}_\alpha \leftarrow \tilde{\boldsymbol{X}}_\alpha, \quad f_\kappa \leftarrow \tilde{f}, \quad (u_{0\kappa}, v_{0\kappa}) \leftarrow (\tilde{u}_{0\kappa}, \tilde{v}_{0\kappa}),$$
$$\boldsymbol{t}_\kappa \leftarrow \tilde{\boldsymbol{t}}_\kappa, \qquad \boldsymbol{R}_\kappa \leftarrow \tilde{\boldsymbol{R}}_\kappa. \tag{25}$$

Stop if $|\tilde{E} - E| \leq \delta$ for a small constant δ. Else, let $E \leftarrow \tilde{E}$, $c \leftarrow c/10$, and go back to Step 2.

6. IMPLEMENTATION TECHNIQUES

6.1 Removing Indeterminacy

Equation (23) does not have a unique solution, because for $c = 0$ the Hessian $\boldsymbol{H} = (\partial^2 E/\partial\xi_k \partial\xi_l)$ has determinant 0 at the solution. This is due to the well known fact that the absolute scale and 3-D position of the scene cannot be determined from images alone. In order to remove this ambiguity, we introduce the following normalization:

$$\boldsymbol{R}_1 = \boldsymbol{I}, \qquad \boldsymbol{t}_1 = \boldsymbol{0}, \qquad t_{22} = 1. \tag{26}$$

This means that we compute the 3-D position relative to the first camera and regard the Y component of the relative displacement of the second camera from the first camera as the unit of length. Imposing $\|\boldsymbol{t}_2\| = 1$ would be theoretically more general but difficult to treat in computation. Here, we assume that the second camera is displaced mostly in the Y direction from the first camera; we may impose $t_{21} = 1$ or $t_{23} = 1$ if we know that the camera displacement is mostly in the X or Y direction. Accordingly, the rows and columns corresponding to ω_{11}, ω_{12}, ω_{13}, Δt_{11}, Δt_{12}, Δt_{13}, and Δt_{22} are removed from the Hessian, and Eq. (23) is solved for the remaining $3N + 9M - 7$ unknowns.

The initial values of $\boldsymbol{X}_\alpha$, f_κ, $(u_{0\kappa}, v_{0\kappa})$, $\boldsymbol{t}_\kappa$, and $\boldsymbol{R}_\kappa$ for starting the LM iterations must be computed by some other means, e.g., the least squares, but if they are not computed with the constraint in Eqs. (26), we need to normalize the given $\boldsymbol{X}_\alpha$, $\boldsymbol{t}_\kappa$, $\boldsymbol{R}_\kappa$ to $\boldsymbol{X}'_\alpha$, $\boldsymbol{t}'_\kappa$, $\boldsymbol{R}'_\kappa$ as follows:

$$\boldsymbol{X}'_\alpha = \frac{1}{s}\boldsymbol{R}_1^\top \Big(\boldsymbol{X}_\alpha - \boldsymbol{t}_1\Big),$$
$$\boldsymbol{R}'_\kappa = \boldsymbol{R}_1^\top \boldsymbol{R}_\kappa, \qquad \boldsymbol{t}'_\kappa = \frac{1}{s}\boldsymbol{R}_1^\top (\boldsymbol{t}_\kappa - \boldsymbol{t}_1). \tag{27}$$

Here, we put $s = (\boldsymbol{j}, \boldsymbol{R}_1^\top (\boldsymbol{t}_2 - \boldsymbol{t}_1))$ and $\boldsymbol{j} = (0, 1, 0)^\top$.

[3]This is often written as $\exp(\boldsymbol{\omega}_\kappa \times \boldsymbol{I})$ and called the *exponential map* from the Lie algebra $so(3)$ to the Lie group $SO(3)$.

6.2 Efficient Computation and Memory Use

From the summation $\sum_{\alpha=1}^N \sum_{\kappa=1}^M$ in Eqs. (8) and (9), it appears that one needs to sum at most MN terms. However, the amount of computation significantly reduces if one notes the following. Consider $\partial E/\partial\xi_k$. From Eq. (8), it is seen that if $\Delta\xi_k$ is a component of the correction $\Delta\boldsymbol{X}_\beta$ of the βth point, only the term for $\alpha = \beta$ needs to be computed in the summation $\sum_{\alpha=1}^N$ due to the Kronecker delta $\delta_{\alpha\beta}$ in Eqs. (10). If $\Delta\xi_k$ corresponds to the correction of f_λ, $(u_{0\lambda}, v_{0\lambda})$, $\boldsymbol{t}_\lambda$, or $\boldsymbol{R}_\lambda$ for the λth image, only the term for $\kappa = \lambda$ needs to be computed in the summation $\sum_{\kappa=1}^M$ due to the Kronecker delta $\delta_{\kappa\lambda}$ in Eqs. (12), (15), (18), and (22). Thus, the summation $\sum_{\alpha=1}^N \sum_{\kappa=1}^M$ in Eq. (8) needs to be computed for either α or κ. Note that for the αth point, only those images that can view that point needs to be considered in the sum, and for the κth image, only those points that appear in that image needs to be considered in the sum.

The same holds for $\partial^2 E/\partial\xi_k\partial\xi_l$. Equation (9) is 0 if $\Delta\xi_k$ and $\Delta\xi_l$ are corrections of different points. If they correspond to the same point, only the term for that point needs to be computed in $\sum_{\alpha=1}^N$. Similarly, Eq. (9) is 0 if $\Delta\xi_k$ and $\Delta\xi_l$ are corrections of camera parameters for different images. If they correspond to the same image, only the terms for that image needs to be computed in $\sum_{\kappa=1}^M$. If one of $\Delta\xi_k$ and $\Delta\xi_l$ correspond to a point and the other to an image, only those terms for that point and that image need to be summed in $\sum_{\alpha=1}^N \sum_{\kappa=1}^M$ provided that that point appears in that image.

By these considerations, the computation time for evaluating $\partial E/\partial\xi_k$ and $\partial^2 E/\partial\xi_k\partial\xi_l$ can be limited to a minimum. However, the Hessian $\boldsymbol{H} = (\partial^2 E/\partial\xi_k\partial\xi_l)$ has $(3N + 9M)^2$ elements in total, and allocating memory space to them is difficult for large N and M. In order to store them in a minimum amount of space avoiding duplication and zero elements as much as possible, we define arrays E, F, and G of size $3N \times 3$, $3N \times 9M$, and $9M \times 9$, respectively, and store $\partial^2 E/\partial X_\alpha \partial Y_\alpha$, etc. in E, $\partial^2 E/\partial X_\alpha \partial f_\kappa$, etc. in F, and $\partial^2 E/\partial f_\kappa \partial u_{0\kappa}$, etc. in G. The total number of necessary array elements is $2NM + 9N + 81M$.

6.3 Decomposing the Linear Equations

After the normalization of Eqs. (26), the matrix in Eq. (23) is of size $(3N + 9M - 7) \times (3N + 9M - 7)$ corresponding to $3N + 9M - 7$ unknowns. For large N and M, we cannot allocate memory space to store intermediate values for numerical computation such as the LU or Cholesky decomposition. We resolve this difficulty by decomposing Eq. (23) into parameters for points and parameters for images. Equation (23) has

the form

$$\begin{pmatrix} \boldsymbol{E}_1^{(c)} & & & \boldsymbol{F}_1 \\ & \ddots & & \vdots \\ & & \boldsymbol{E}_N^{(c)} & \boldsymbol{F}_N \\ \boldsymbol{F}_1^\top & \cdots & \boldsymbol{F}_N^\top & \boldsymbol{G}^{(c)} \end{pmatrix} \begin{pmatrix} \Delta\boldsymbol{\xi}_P \\ \Delta\boldsymbol{\xi}_F \end{pmatrix} = - \begin{pmatrix} \boldsymbol{d}_P \\ \boldsymbol{d}_F \end{pmatrix}, \tag{28}$$

where $\Delta\boldsymbol{\xi}_P$ is the $3N$-D vector corresponding to the 3-D coordinates and $\Delta\boldsymbol{\xi}_F$ is the $(9M-7)$-D vector corresponding to the camera parameters. Likewise, $\boldsymbol{d}_P$ and $\boldsymbol{d}_F$ are the $3N$-D and $(9M-7)$-D parts of the vector on the right hand of Eq. (23). The submatrices $\boldsymbol{E}_\alpha^{(c)}$, $\alpha = 1, ..., N$, are of size 3×3 and contain the second derivatives of E with respect to $(X_\alpha, Y_\alpha, Z_\alpha)$; the superscript (c) indicates that the diagonal elements are multiplied by $(1+c)$. The submatrices $\boldsymbol{F}_\alpha$ are of size $3 \times (9M-7)$ and contain the second derivatives of E with respect to $(X_\alpha, Y_\alpha, Z_\alpha)$ and the camera parameters of the frames where it appears. The submatrices $\boldsymbol{G}^{(c)}$ are of size $(9M-7) \times (9M-7)$ and contain the second derivatives of E with respect to camera parameters, with the diagonal elements multiplied by $(1+c)$. Equation (28) is decomposed into the following two parts:

$$\begin{pmatrix} \boldsymbol{E}_1^{(c)} & & \\ & \ddots & \\ & & \boldsymbol{E}_N^{(c)} \end{pmatrix} \Delta\boldsymbol{\xi}_P + \begin{pmatrix} \boldsymbol{F}_1 \\ \vdots \\ \boldsymbol{F}_N \end{pmatrix} \Delta\boldsymbol{\xi}_F = -\boldsymbol{d}_P,$$

$$\begin{pmatrix} \boldsymbol{F}_1^\top & \cdots & \boldsymbol{F}_N^\top \end{pmatrix} \Delta\boldsymbol{\xi}_P + \boldsymbol{G}^{(c)} \Delta\boldsymbol{\xi}_F = -\boldsymbol{d}_F. \tag{29}$$

Solving the first equation for $\Delta\boldsymbol{\xi}_P$ and substituting it into the second, we obtain the following $9M-7$-D linear equation for $\Delta\boldsymbol{\xi}_F$ alone:

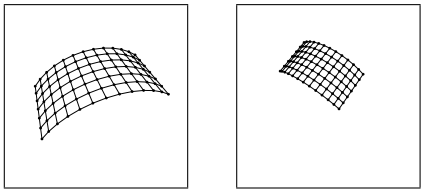
$$\Big(\boldsymbol{G}^{(c)} - \sum_{\alpha=1}^{N} \boldsymbol{F}_\alpha^\top \boldsymbol{E}_\alpha^{(c)-1} \boldsymbol{F}_\alpha\Big) \Delta\boldsymbol{\xi}_F = \sum_{\alpha=1}^{N} \boldsymbol{F}_\alpha^\top \boldsymbol{E}_\alpha^{(c)-1} \nabla_\alpha E - \boldsymbol{d}_F, \quad \nabla_\alpha E \equiv \begin{pmatrix} \partial E/\partial X_\alpha \\ \partial E/\partial Y_\alpha \\ \partial E/\partial Z_\alpha \end{pmatrix}. \tag{30}$$

Solving this for $\Delta\boldsymbol{\xi}_F$ and substituting it to the second of Eqs. (29), we can determine $\Delta\boldsymbol{\xi}_P$. The correction of αth point is given in the form

$$\begin{pmatrix} \Delta X_\alpha \\ \Delta Y_\alpha \\ \Delta Z_\alpha \end{pmatrix} = -\boldsymbol{E}_\alpha^{(c)-1} (\boldsymbol{F}_\alpha \Delta\boldsymbol{\xi}_F + \nabla_\alpha E). \tag{31}$$

6.4 Convergence Decision

Even with the above techniques, the LM iterations require considerable computation time. In numerically solving equations, it may be a common practice to continue iterations until all significant digits of the unknowns are unaltered, but the number of unknowns for bundle adjustment may become hundreds and thousands, requiring a very long time for complete convergence. However, the purpose of bundle adjustment is to find a solution with a small reprojection error, so it makes sense to stop if the reprojection error decreases less than a specified amount. The LM algorithm in Sec. 10 is described that way. For stopping the iterations when the decrease of the reprojection error is less than ϵ pixels per point, we can set the constant δ in Sec. 10 to $\delta = n\epsilon^2/f_0^2$, where $n = \sum_{\alpha=1}^{N} \sum_{\kappa=1}^{M} I_{\alpha\kappa}$ is the total number of observed points in all images. We set $\epsilon = 0.01$ pixels in our experiments.



(a)

	Kanatani and Sugaya [9]	bundle adjustment
0	0.0000000000000	0.2740543086661
1	0.1071688468318	0.1083766529404
2	0.1071686014356	0.1076009069457
3	0.1071686015030	0.1076005713017
4	0.1071686013682	0.1071718714030
5	0.1071686015030	0.1071686014673
6	0.1071686013682	0.1071686014580
7	0.1071686016378	0.1071686014580

(b)

Figure 1: (a) Simulated images of a grid surface in the scene. (b) Typical example of the decrease of the reprojection error for $\sigma = 0.1$ pixels.

7. EXPERIMENTS

7.1 Two-View Reconstruction

Figure 1(a) shows two simulated images of a grid surface taken from different angles. The image size is assumed to be 600×600 pixels with focal lengths $f = f' = 600$ pixels. We added independent Gaussian noise of mean 0 and standard deviation $\sigma = 0.1$ pixels to the x and y coordinates of the grid points in the two images and computed from them the fundamental matrix by least squares (or Hartley's' 8-point algorithm [2]). From the obtained fundamental matrix, we estimated the focal lengths and the relative translation and rotation of the two cameras and reconstructed the 3-D coordinates of the grid points by the procedure described in [10]. Since theoretically the principal point (u_0, v_0) cannot be estimated from two views [10], we assumed it to be at the center of the frame. The number of parameters is 280: two for the focal lengths, two for the relative translation, two for the relative rotation, and 273 for the 3-D coordinates of the 91 grid point. We evaluated the the

reprojection error e per point in pixel in the form

$$e = f_0 \sqrt{\frac{E}{N-7}}, \tag{32}$$

where the number 7 in $N-7$ is the degree of the freedom of the focal lengths, the translation, and the relative rotation ($N = 91$). As is well known in statistics [5], $e^2/f_0^2\sigma^2$ is subject to a χ^2 distribution with $N-7$ degrees of freedom for independent Gaussian noise of mean 0 and standard deviation σ and hence has expectation $N-7$. Thus, Eq. (32) gives an estimate of σ of the added noise.

As a comparison, we tested the method of Kanatani and Sugaya [9] by starting from the same initial values. The result is listed in Fig. 1(b), where we continued computation indefinitely without convergence judgment. The method of Kanatani and Sugaya [9] orthogonally projects the observed point correspondences onto the 3-D manifold (hyperbolic surface with one sheet) in the 4-D joint $xyx'y'$ space and iteratively optimizes the fundamental matrix using the EFNS of Kanatani and Sugaya [8]. As can be seen from Fig. 1(b), it reaches the same solution as bundle adjustment, confirming its optimality. Moreover, it converges after two iterations, while bundle adjustment requires around five iterations. Thus, the combination of fundamental matrix computation using the method of [9] and 3-D reconstruction using the method of [10] is better than bundle adjustment, as far as two-view reconstruction is concerned. However, the focal lengths computed by the method of [10] can be imaginary (the values in square roots can become negative) in the presence of large noise. In such a case, we need to start bundle adjustment from an appropriate guess of the focal lengths.

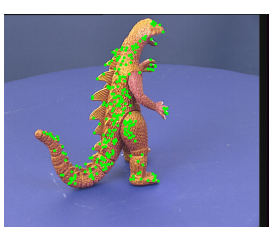
7.2 Multi-View Reconstruction

We tested our method using the real video sequence provided by the University of Oxford[4]. It consists of 36 frames tacking feature points (4983 in total) over 2 to 21 consecutive frames, and the projection matrix $\boldsymbol{P}_\kappa$ is estimated for each frame. Figure 2(a) shows one frame with tracked feature points.

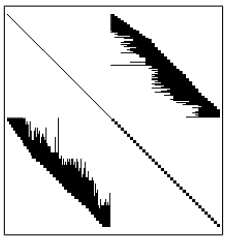
Since the number of unknowns is 15266, the Hessian has around 200 millions elements. We cannot allocate memory space to store them directly, but with the scheme described in Sec. 10, they can be stored in around 4000 array cells (about a five thousandth). Since each point is visible only in a limited number of images, most of the Hessian elements are 0. Figure 2(b) shows the sparsity pattern of the Hessian for 1000 decimated points; nonzero elements are indicated in black (about 13%).

We first estimated the focal lengths f_κ, the principal points $(u_{0\kappa}, v_{0\kappa})$, the translations $\boldsymbol{t}_\kappa$, and the

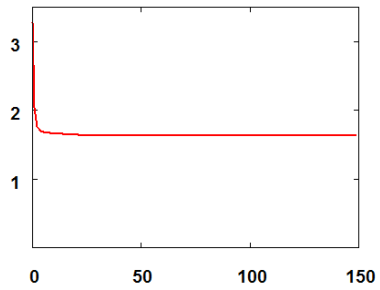
[4] `http://www.robots.ox.ac.uk/~vgg/data.html`



(a)



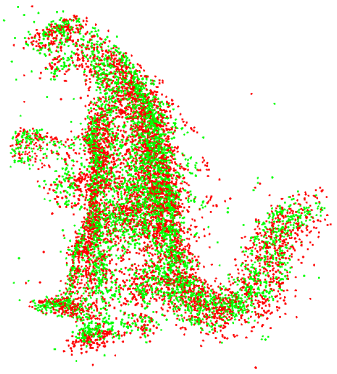
(b)



(c)

	reprojection error		reprojection error
0	3.277965703463469	⋮	⋮
1	2.037807322757024	140	1.626138870635717
2	1.767180606187605	141	1.626109073343624
3	1.721032319350261	142	1.626079434501709
4	1.698429496315309	143	1.626049951753774
5	1.684614811452468	144	1.626020622805242
6	1.675366012050569	145	1.625991445421568
7	1.668829491793228	146	1.625962417425169
8	1.664028486785132	147	1.625933536694230
9	1.660393246948761	148	1.625904801160639
10	1.657569357560945	149	1.625876208807785

(d)



(e)

Figure 2: (a) One of the 36 frames of the test image sequence. (b) The sparsity pattern of the Hessian for decimated 100 points; nonzero elements are indicated in black. (c) The reprojection error e vs. the number of iteration. (d) The numerical value of e at each iteration. (b) 3-D reconstruction. Red: initial positions. Green: final positions.

rotations $\boldsymbol{R}_\kappa$ from the projection matrices $\boldsymbol{P}_\kappa$ provided in the database (Appendix A) and computed the 3-D coordinates by least squares (Appendix B). Then, we started bundle adjustment. The total number $n = \sum_{\alpha=1}^{N} \sum_{\kappa=1}^{M} I_{\alpha\kappa}$ of points visible in the images is 16432. The reprojection error per point in pixel corresponding to Eq. (32) for two views is

$$e = f_0 \sqrt{\frac{E}{2n - (3N + 9M - 7)}}. \tag{33}$$

The initial reprojection error of the least squares reconstruction is was $e = 3.27797$ pixels, which reduced to $e = 1.625876$ pixels after 149 iterations. Figure 2(c) plots the decrease of e for the number of iterations, and Fig. 2(d) lists their numerical values. The number of iterations was 149. The execution time was 21 minutes and 51 seconds. The program was implemented in the C++ language, using Intel Core2Duo E6750, 2.66GHz for CPU with main memory 4GB and Windows Vista for the OS. Figure 2(e) shows the reconstructed 3-D points viewed from some angle: the red points are initial reconstruction, and the green points are the final reconstruction.

8. CONCLUSIONS

We have described in detail the algorithm of bundle adjustment for 3-D reconstruction from multiple images based on our latest research results. The main focus of this paper is on the treatment of camera rotations in a mathematically sound manner and the efficiency of computation and memory usage when the number of points and image frames is very large. As an example, we computed the fundamental matrix from two-view point correspondences and observed that the same solution is obtained as the method of Kanatani and Sugaya [9], confirming that their method is indeed optimal. However, bundle adjustment is less efficient than their method. As another example, we reconstructed 3-D using a real video database provided by the University of Oxford. It has a very large number of points, so that it is difficult to implement bundle adjustment directly. However, we have shown that we can reconstruct 3-D using our techniques for efficient computation and efficient memory usage.

Theoretically, bundle adjustment is a universal tool for 3-D reconstruction that can be used in any situations. However, it requires a good initial guess to ensure convergence within a practically reasonable time. A typical method for approximate 3-D reconstruction is the Tomasi-Kanade factorization [7, 14] using the affine model to approximate the camera imaging geometry, and more accurate reconstruction can be done by the technique of self-calibration [1, 6] using the perspective camera model. In any case, bundle adjustment should be regarded as a means of not reconstructing 3-D from scratch but refining the 3-D structure already reconstructed by other means.

Acknowledgments. The authors thank Takayuki Okatani of Tohoku University for helpful suggestions on this research. 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 21500172).

References

[1] H. Ackermann and K. Kanatani, Fast projective reconstruction: Toward ultimate efficiency *IPSJ Trans. Comp. Vis. Image Media*, **49**-SIG 6 (2008-3), 68–78.

[2] R. I. Hartley, In defense of the eight-point algorithm, *IEEE Trans. Patt. Anal. Mach. Intell.*, **19**-6 (1997-6), 580–593.

[3] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed., Cambridge University Press, Cambridge, U.K., 2004.

[4] K. Kanatani, *Group-Theoretical Methods in Image Understanding*, Springer, Berlin, Germany, 1990.

[5] K. Kanatani, *Statistical Optimization for Geometric Computation: Theory and Practice* Elsevier, Amsterdam, the Netherlands, 1996; reprinted, Dover, York, NY, U.S.A., 2005.

[6] K. Kanatani, Latest progress of 3-D reconstruction from multiple camera images, in Xing P. Guo (ed.), *Robotics Research Trends*, Nova Science, Hauppauge, NY, U.S.A., pp. 33–75.

[7] K. Kanatani, Y. Sugaya and H. Ackermann, Uncalibrated factorization using a variable symmetric affine camera, *IEICE Tran. Inf. & Syst.*, **E90-D**-5 (2007-5), 851–858.

[8] K. Kanatani and Y. Sugaya, Extended FNS for constrained parameter estimation, *Proc. Meeting on Image Recognition and Understanding, 2007*, Hiroshima, Japan, July/August 1, 2007, pp. 219–226.

[9] K. Kanatani and Y. Sugaya, Compact fundamental matrix computation, *IPSJ Trans. Comput. Vis. Appl.* **2** (2010-3), pp. 59–70.

[10] K. Kanatani, Y. Sugaya and Y. Kanazawa, Latest algorithms for 3-D reconstruction form two views, in C. H. Chen (ed.) *Handbook of Pattern Recognition and Computer Vision*, 4th ed., World Scientific Publishing, Singapore, 2009, pp. 201–234.

[11] M. I. A. Lourakis and A. A. Argyros, Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment?, *Proc. 10th Int. Conf. Comput. Vis.*, Vol. 2, October 2005, Beijing, China, pp. 1526–1531.

[12] M. I. A. Lourakis and A. A. Argyros, SBA: A software package for generic sparse bundle adjustment, *ACM Trans. Math. Software*, **36**-1 (2009-3), 2:1–30.

[13] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C: The Art of Scientific Computing*, 2nd ed., Cambridge University Press, Cambridge, U.K., 1992.

[14] C. Tomasi and T. Kanade, Shape and motion from image streams under orthography: A factorization method, *Int. J. Comput. Vis.*, **9**-2 (1992-10), 137–154. (1992).

[15] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. Fitzgibbon, Bundle adjustment—A modern synthesis, in B. Triggs, A. Zisserman, and R. Szeliski, (eds.), *Vision Algorithms: Theory and Practice*, Springer, Berlin, 2000, pp. 298–375.

APPENDIX

A. Decomposing the Projection Matrix

Write the projection matrix as $\boldsymbol{P} = \begin{pmatrix} \boldsymbol{Q} & \boldsymbol{q} \end{pmatrix}$ by letting $\boldsymbol{Q}$ be the first 3×3 submatrix of $\boldsymbol{P}$ and $\boldsymbol{q}$ its fourth column. From Eq. (1), we see that the sign of $\boldsymbol{P}$ is indeterminate, so choose the sign so that $\det \boldsymbol{Q} > 0$; if $\det \boldsymbol{Q} < 0$, we change the signs of both $\boldsymbol{Q}$ and $\boldsymbol{q}$. Since $\boldsymbol{P}$ still has scale indeterminacy, we have

$$\boldsymbol{Q} = c\boldsymbol{K}\boldsymbol{R}^\top, \qquad \boldsymbol{q} = -c\boldsymbol{K}\boldsymbol{R}^\top\boldsymbol{t}, \tag{34}$$

where c is an unknown positive constant. From these, we see that the translation $\boldsymbol{t}$ is given by

$$\boldsymbol{t} = -\boldsymbol{Q}^{-1}\boldsymbol{q}. \tag{35}$$

Since $\boldsymbol{R}$ is a rotation matrix, satisfying $\boldsymbol{R}^\top\boldsymbol{R} = \boldsymbol{I}$, we obtain from the first of Eqs.(34)

$$\boldsymbol{Q}\boldsymbol{Q}^\top = c^2\boldsymbol{K}\boldsymbol{R}^\top\boldsymbol{R}\boldsymbol{K}^\top = c^2\boldsymbol{K}\boldsymbol{K}^\top. \tag{36}$$

Its inverse is

$$(\boldsymbol{Q}\boldsymbol{Q}^\top)^{-1} = \frac{1}{c^2}(\boldsymbol{K}^{-1})^\top(\boldsymbol{K}^{-1}). \tag{37}$$

By the Choleski decomposition, we can express this in terms of an upper triangular matrix $\boldsymbol{C}$ in the form

$$(\boldsymbol{Q}\boldsymbol{Q}^\top)^{-1} = \boldsymbol{C}^\top\boldsymbol{C}. \tag{38}$$

Since the inverse of an upper triangular matrix is also upper triangular, we obtain from Eqs. (37) and (38)

$$\boldsymbol{C} = \frac{1}{c}\boldsymbol{K}^{-1} \quad \text{i.e.,} \quad \boldsymbol{C}^{-1} = c\boldsymbol{K}. \tag{39}$$

From the first of Eqs. (34) and Eq. (39), we obtain

$$\boldsymbol{Q} = \boldsymbol{C}^{-1}\boldsymbol{R}^\top, \tag{40}$$

from which $\boldsymbol{R}$ is given by

$$\boldsymbol{R} = (\boldsymbol{C}\boldsymbol{Q})^\top. \tag{41}$$

The matrix $\boldsymbol{K}$ of intrinsic parameters is obtained by multiplying $\boldsymbol{C}^{-1}$ in Eq. (39) by a constant that makes its (3,3) element 1.

B. 3-D Reconstruction by Least Squares

Clearing the fractions in Eqs. (3), we obtain

$$\begin{aligned} &xP^{31}X + xP^{32}Y + xP^{33}Z + xP^{34} \\ &\quad = f_0P^{11}X + f_0P^{12}Y + f_0P^{13}Z + f_0P^{14}, \\ &yP^{31}X + yP^{32}Y + yP^{33}Z + yP^{34} \\ &\quad = f_0P^{21}X + f_0P^{22}Y + f_0P^{23}Z + f_0P^{24}. \end{aligned} \tag{42}$$

Collect, for each point p_α, equations of this form for all the n_α $(= \sum_{\kappa=1}^{M} I_{\alpha\kappa})$ frames where the αth point appears. Then, we obtain the following $2n_\alpha$ linear equation of the 3-D coordinates $(X_\alpha, Y_\alpha, Z_\alpha)$ of p_α:

$$\begin{pmatrix} \vdots & \vdots & \vdots \\ x_{\alpha\kappa}P_\kappa^{31} - f_0P_\kappa^{11} & x_{\alpha\kappa}P_\kappa^{32} - f_0P_\kappa^{12} & x_{\alpha\kappa}P_\kappa^{33} - f_0P_\kappa^{13} \\ y_{\alpha\kappa}P_\kappa^{31} - f_0P_\kappa^{21} & y_{\alpha\kappa}P_\kappa^{32} - f_0P_\kappa^{22} & y_{\alpha\kappa}P_\kappa^{33} - f_0P_\kappa^{23} \\ \vdots & \vdots & \vdots \end{pmatrix}$$

$$\begin{pmatrix} \vdots \\ x_{\alpha\kappa}P_\kappa^{33} - f_0P_\kappa^{13} \\ y_{\alpha\kappa}P_\kappa^{33} - f_0P_\kappa^{23} \\ \vdots \end{pmatrix} \begin{pmatrix} X_\alpha \\ Y_\alpha \\ Z_\alpha \end{pmatrix} = -\begin{pmatrix} \vdots \\ x_{\alpha\kappa}P_\kappa^{34} - f_0P_\kappa^{14} \\ y_{\alpha\kappa}P_\kappa^{34} - f_0P_\kappa^{24} \\ \vdots \end{pmatrix}. \tag{43}$$

Solving this by least squares, we can obtain an initial solution.