ENEE731 Project
Shape and motion from image streams: A factorization method

Naotoshi Seo, sonots@umd.edu

December 20, 2006

1 Introduction

The structure from motion - recovering scene geometry and camera motion from a sequence of images - is an important task and has wide applicability in many tasks, such as navigation and robot manipulation. Tomasi and Kanade [1] first developed a factorization method to recover shape and motion under an orthographic projection model, and obtained robust and accurate results. Poelman and Kanade [2] have extended the factorization method to paraperspective projection. Triggs [3] further extended the factorization method to fully perspective projection. This method recovers a consistent set of projective depths (projective scale factors) for the image points.

In this project, we implemented these three factorization methods, and comparisons are shown.

2 Algorithm: The Orthographic Factorization

The orthographic factorization method [1] was implemented.

Given an image stream, suppose that we have tracked P feature points over F frames. We then trajectories of image coordinates $\{(u_{fp}, v_{fp}) | f = 1, ..., F, p = 1, ..., P\}$. We write the horizontal feature coordinates u_{fp} into an $F \times P$ matrix $\mathbf{U}$. Similarly, an $F \times P$ matrix $\mathbf{V}$ is built from the vertical coordinates v_{fp}.

Define the *measurement matrix* of size $2F \times P$

$$\mathbf{W} = \begin{bmatrix} \mathbf{U} \\ \hline \mathbf{V} \end{bmatrix}.$$

Compute the $2F \times 1$ translation vector $\mathbf{t}$ whose each entry is the mean of the entries in the same row of the measurement matrix $\mathbf{W}$. Define the *registered measument matrix*

$$\tilde{\mathbf{W}} = \mathbf{W} - \mathbf{t}[1..1]$$

The $\tilde{\mathbf{W}}$ can be expressed in a matrix form:

$$\tilde{\mathbf{W}} = \mathbf{RS},$$

where the matrix of size $2F \times 3$

$$\mathbf{R} = \begin{bmatrix} \mathbf{i}_1^T \\ \vdots \\ \mathbf{i}_F^T \\ \mathbf{j}_1^T \\ \vdots \\ \mathbf{j}_F^T \end{bmatrix} \tag{1}$$

represents the camera rotation, and the matrix of size $3 \times P$

$$\mathbf{S} = [\mathbf{s}_1 ... \mathbf{s}_P]$$

is the shape matrix. The rows of $\mathbf{R}$ represent the orientations of the horizontal and vertical camera reference axes throughout the stream, while the colums of $\mathbf{S}$ are the coordinates of the P feature points with respect to their centroid. The goal of the factorization method is to compute the matrices $\mathbf{R}$ and $\mathbf{S}$.

Step 1

Compute the singular-value decomposition

$$\tilde{\mathbf{W}} = \mathbf{O_1 \Sigma O_2}.$$

Step 2

Define $\mathbf{O_1}'$ which is the first three columns of $\mathbf{O_1}$, $\mathbf{\Sigma}'$ which is the first 3×3 submatrix of $\mathbf{\Sigma}$, and $\mathbf{O_2}'$ which is the first three rows of $\mathbf{O_2}$. Define $\hat{R} = \mathbf{O_1}'(\mathbf{\Sigma}')^{1/2}$ and $\hat{S} = (\mathbf{\Sigma}')^{1/2}\mathbf{O_2}'$.

Step 3 - Metric Constraints

Impose the metric constraints,

$$\hat{\mathbf{i}}_f^T Q Q^T \hat{\mathbf{i}}_f = 1, \quad \hat{\mathbf{j}}_f^T Q Q^T \hat{\mathbf{j}}_f = 1, \quad \hat{\mathbf{i}}_f^T Q Q^T \hat{\mathbf{j}}_f = 0$$

where Q is a 3×3 matrix, and $\hat{\mathbf{i}}_f^T$, $\hat{\mathbf{j}}_f^T$ are elements of $\hat{\mathbf{R}}$. To solve Q, we can

1. use Newton's method, or
2. define $L = QQ^T$ and solve the linear system of equations for L and use Cholesky decomposition to get Q.

Step 3.1 - Solving the linear system

We used the 2nd way. Morita and Kanade[6] explains the steps to compute Q. By denoting

$$L = \begin{bmatrix} I_1 & I_2 & I_3 \\ I_2 & I_4 & I_5 \\ I_3 & I_5 & I_6 \end{bmatrix},$$

the quadratic system can be rewritten as

$$GI = c,$$

where the $3F \times 6$ matrix G, the 6×1 vector I, and the $3F \times 1$ vector c are defined by

$$G = \begin{bmatrix} \mathbf{g}^T(\mathbf{i}_1, \mathbf{i}_1) \\ \vdots \\ \mathbf{g}^T(\mathbf{i}_F, \mathbf{i}_F) \\ \mathbf{g}^T(\mathbf{j}_1, \mathbf{j}_1) \\ \vdots \\ \mathbf{g}^T(\mathbf{j}_F, \mathbf{j}_F) \\ \mathbf{g}^T(\mathbf{i}_1, \mathbf{j}_1) \\ \vdots \\ \mathbf{g}^T(\mathbf{i}_F, \mathbf{j}_F) \end{bmatrix}, I = \begin{bmatrix} I_1 \\ \vdots \\ I_6 \end{bmatrix}, c = \begin{bmatrix} 1 \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}. \tag{2}$$

where c has $2F$ ones and F zeros, and

$$\mathbf{g}(\mathbf{a},\mathbf{b}) = \begin{bmatrix} a_1b_1 \\ a_1b_2 + a_2b_1 \\ a_1b_3 + a_3b_1 \\ a_2b_2 \\ a_2b_3 + a_3b_2 \\ a_3b_3 \end{bmatrix}. \tag{3}$$

The simplest solution of the system is given by the pseudo inverse method, such that

$$I = G^*c.$$

The vector I determines the symmetric matrix L. The Cholesky decomposition or eigen decomposition or square root of a matrix of L gives Q. as long as the L is positive definite.

Step 3.2 - Enforcing positive definiteness

We might obtain a matrix that is not positive definite when we estimeated L using the linear method. Higham [7] introduced a method computing a 'nearest' symmetric positive semidefinite matrix from a matrix. The lecture note [8] gives the easy notation. First, we compute the symmetrix matrix as

$$L = \frac{L + L^T}{2}.$$

In our case, the matrix L is already a symmetric matrix. Now we eigen decompose L:

$$L = \mathbf{U\Sigma U}^T$$

and form the matrix $\mathbf{\Sigma}_+$ by setting any negative eigenvalues to zero. The positive semidefinite matrix that is closed to L is then given by

$$L_{psd} = \mathbf{U\Sigma}_+\mathbf{U}^T.$$

However, we want a positive definite matrix. In this case, we set any nagative eigenvalues to $\epsilon > 0$. The reason is apparent from the definition of the positive definite matrix. Furthermore, we want the matrix Q, it can be simply obtained as $\mathbf{U\Sigma}_+{}^{1/2}$ without computing L. As a result,

1. Eigen decompose $L = \mathbf{U\Sigma U}^T$
2. Form a matrix $\mathbf{\Sigma}_+$ by setting any negative values to $\epsilon > 0$,
3. Compute $Q = \mathbf{U\Sigma}_+{}^{1/2}$

Step 4

Compute the rotation matrix $\mathbf{R}$ and the shape matrix $\mathbf{S}$ as

$$\mathbf{R} = \hat{\mathbf{R}}Q, \quad \mathbf{S} = Q^{-1}\hat{\mathbf{S}}.$$

Step 5

Align the first camera reference system with world reference system by forming the products

$$\mathbf{R} = \mathbf{R}\mathbf{R_0}, \quad \mathbf{S} = \mathbf{R_0}^T\mathbf{S}, \tag{4}$$

where the orthonomal matrix $\mathbf{R_0} = [\mathbf{i}_0\ \mathbf{j}_0\ \mathbf{k}_0]$ rotates the first camera reference system $\mathbf{R}_1 = [\mathbf{i}_1\ \mathbf{j}_1\ \mathbf{k}_1]^T$ into the indentity matrix $\mathbf{I}$. Now let be $\mathbf{R}'_0 = [\mathbf{i}_1\ \mathbf{j}_1\ \mathbf{k}_1]$. Then,

$$\mathbf{R}_1\mathbf{R}'_0 = \begin{bmatrix} ||\mathbf{i}_1|| & 0 & 0 \\ 0 & ||\mathbf{j}_1|| & 0 \\ 0 & 0 & ||\mathbf{k}_1|| \end{bmatrix}.$$

because axis are perpendicular each other. From this,

$$\mathbf{R}_0 = \left[\frac{\mathbf{i}_1}{||\mathbf{i}_1||} \frac{\mathbf{j}_1}{||\mathbf{j}_1||} \frac{\mathbf{k}_1}{||\mathbf{k}_1||} \right].$$

3 Algorithm: The Paraperspective Factorization

The paraperspective factorization method [2] was implemented. At the implementation phase, the difference with the orthographic case is only the metric constraints.

Let $\mathbf{M}$ be the estimated motion matrix

$$\mathbf{M} = \begin{bmatrix} \mathbf{m}_1^T \\ \vdots \\ \mathbf{m}_F^T \\ \mathbf{n}_1^T \\ \vdots \\ \mathbf{n}_F^T \end{bmatrix} \tag{5}$$

Let T be the translation vector

$$\mathbf{T} = \begin{bmatrix} x_1 \\ \vdots \\ x_F \\ y_1 \\ \vdots \\ y_F \end{bmatrix} \tag{6}$$

The metric constraints are as followings:

$$\begin{array}{cl} \frac{|\mathbf{m}_f|^2}{1+x_f^2} - \frac{|\mathbf{n}_f|^2}{1+y_f^2} & = 0 \\ \mathbf{m}_f \cdot \mathbf{n}_f - x_f y_f \frac{1}{2}\left(\frac{|\mathbf{m}_f|^2}{1+x_f^2} + \frac{|\mathbf{n}_f|^2}{1+y_f^2}\right) & = 0 \\ |\mathbf{m}_1| & = 1 \end{array} \tag{7}$$

We impose

$$|\mathbf{m}_f|^2 = \hat{\mathbf{m}}_f^T Q Q^T \hat{\mathbf{m}}_f, \quad |\mathbf{n}_f|^2 = \hat{\mathbf{n}}_f^T Q Q^T \hat{\mathbf{n}}_f, \quad \mathbf{m}_f \cdot \mathbf{n}_f = \hat{\mathbf{m}}_f^T Q Q^T \hat{\mathbf{n}}_f$$

where Q is a 3×3 matrix, and $\hat{\mathbf{m}}_f, \hat{\mathbf{n}}_f$ are elements of the estimated motion matrix $\hat{\mathbf{M}}$. The matrix Q is solvable by the same way with the orthographic case.

4 Algorithm: The Projective Factorization

The projective factorization method [3] was implemented. The perspective projection equation models the projection of a 3D point $\mathbf{X}_j = [x_j, y_j, z_j, 1]^T$ on an image as

$$\lambda_{ij}\mathbf{x}_{ij} = \mathbf{P}_i\mathbf{X}_j$$

where $\mathbf{x}_{ij} = [u_{ij}, v_{ij}, 1]^T$ are the image coordinates of point j in the i^{th} view, λ_{ij} is the *projective depth* on the point and $textbfP_i$ is a 3×4 projection matrix. The complete set of image projections can be gathered into a single $(3m \times n)$ rescaled measurement matrix equation:

$$\mathbf{W} = \left[\begin{array}{ccc} \lambda_{11}\mathbf{x}_{11} & \cdots & \lambda_{1n}\mathbf{x}_{1n} \\ \vdots & & \vdots \\ \lambda_{m1}\mathbf{x}_{m1} & \cdots & \lambda_{mn}\mathbf{x}_{mn} \end{array} \right] = \left(\begin{array}{c} \mathbf{P}_1 \\ \vdots \\ \mathbf{P}_m \end{array} \right) (\mathbf{X}_1 \cdots \mathbf{X}_n) \tag{8}$$

where $\mathbf{P}_i$ is the 3×4 projection matrix.

Step 1

Normalize the image coordinates, by applying transformations $\mathbf{T}_i$ to each image as:

1. Translating so that their centroid is at the origin
2. Scaling so that the average distance from the origin is $\sqrt{2}$.

The matrix $\mathbf{T}_i$ is constructed as

$$\mathbf{T}_i = \left[\begin{array}{ccc} s_i & 0 & -s_i c_{ix} \\ 0 & s_i & -s_i c_{iy} \\ 0 & 0 & 1 \end{array} \right] \tag{9}$$

where c_{ix} and c_{iy} are the mean in the horizontal and vertical coordinates of image i respectively and s_i is the scaling factor which is determined as

$$s_i = \frac{\sqrt{2}}{d_i}$$

where d_i is the mean distance from the centerized origin.

Step 2 - Fundamental Matrix

Estimate the fundamental matrices $\mathbf{F}_{ij}$ and epipoles e_{ij} with the 8-point method [4][5]. The fundamental matrix is defined by the equation

$$\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0$$

for any pair of matching points $\mathbf{x}'$ and $\mathbf{x}$ in two images. The algorithm to estimate the fundamental matrix is as followings:

1. Normalize the image coordinates by a combined translation and scale transformation. (This is the above step 1)
2. Solve a set of linear equations derived from the Longuet-Higgins equation. Solution is either by singular value decomposition or an eigensystem method.
3. The constraint that the fundamental matrix should be of rank two is enforced. This is done by by computing the matrix closest to the fundamental matrix in Frobenius norm.
4. The fundamental matrix corresponding to the original matched point coordinates is recovered by applying an "un-normalising" transform.
5. Perform numerical analysis of the estimated fundamental matrix

Step 3

Determine the scale factors λ_{ip}

$$\lambda_{ip} = \frac{(e_{ij} \wedge \mathbf{x}_{ip}) \cdot (\mathbf{F}_{ij}\mathbf{x}_{ip})}{||e_{ij} \wedge \mathbf{x}_{ip}||^2} \lambda_{jp}$$

by recursively chaining together to give estimates for the complete set of depths for point p, starting from some arbitrary initial value such as $\lambda_{1p} = 1$.

Step 4

Build the rescaled measurement matrix $\mathbf{W}$ with $\{\lambda_{ij}\}$ and $\{\mathbf{x}_{ij}\}$.

Step 5

Balance $\mathbf{W}$ by the following iterative scheme:

1. Rescale each column l so that $\sum_{r=1}^{3m}(w_{rl})^2 = 1$.
2. Rescale each triplet of rows $(3k-2, 3k-1, 3k)$ so that $\sum_{l=1}^{n}\sum_{i=3k-2}^{3k} w_{il}^2 = 1$.
3. If the entries of $\mathbf{W}$ changed significantly, repeat 1 and 2.

Step 6

Compute the SVD of the balanced matrix $\mathbf{W}$

Step 7

From the SVD, recover projective motion and shape. Note that the projective factorization has no metric constraints.

Step 8

Adapt projective motion, to account for the normalization transformations $\mathbf{T}_i$ of step 1. $\mathbf{P}_i$ is recovered as

$$\mathbf{P}_i = \mathbf{T}_i^{-1}\hat{\mathbf{P}}_i.$$

5 Experiments and Results

5.1 Datasets

The dataset of project requirements are the castle image sequences [11] and the medusa image sequences [12]. KLT Tracker library [9] was used to obtain feature tracking points. However, the KLT software did not track feature points well for castle image sequences. The reason is probably why the number of frames per second in the castle sequences are pretty low (image sequences look not continuous well.) Because the feature tracking part is not essential with this project, we used the hotel image sequences [10] which were taken in laboratory carefully, instead of the castle sequences. Furthermore, a synthetic data was used in order to make sure behaviors of programs.

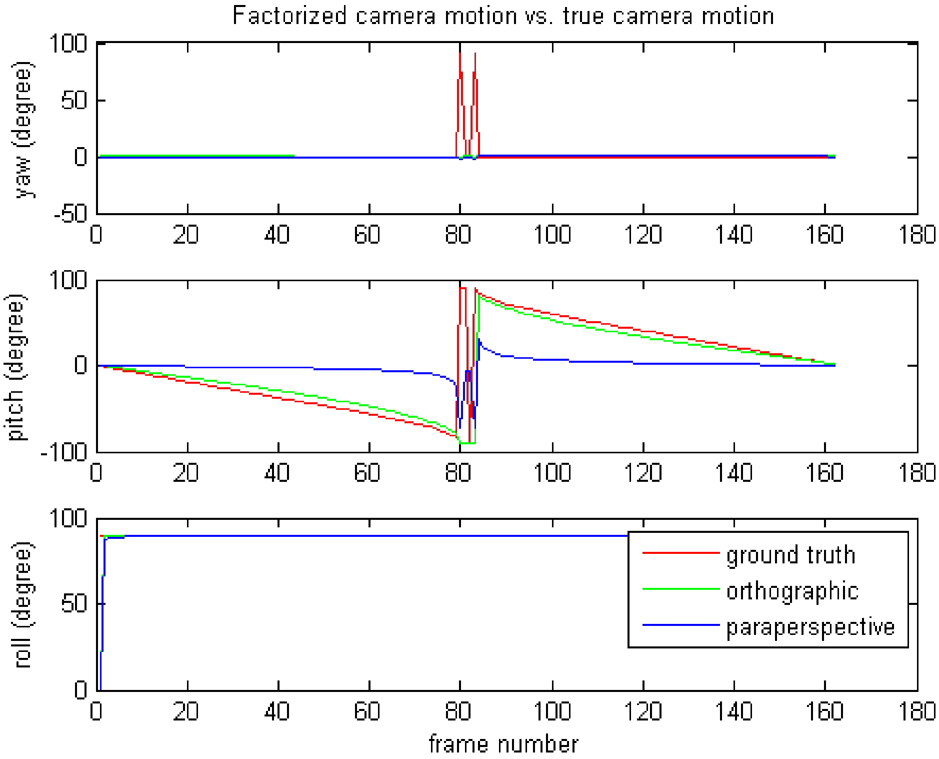
5.2 Synthetic Data

The synthetic data was created as follows. The camera motion goes around the randomly generated 80 points in half an elliptical path. The size of 80 points range from -2 to -2 in their x and y coordinates. The lengths of the main axis of the ellipse are 16 and 12. A Gaussian noise source having variance of 0.5 was added to the data in the image frame. A total of 162 frames were generated on this configuration.

This data was used for testing the orthographic factorization and the paraperspective factorization. The camera motion was recovered and plotted where yaw is the angle toward an axe $\mathbf{k}_1 = (0, 0, 1)$, roll is the angle toward $\mathbf{i}_1 = (1, 0, 0)$, and pitch is the angle toward $\mathbf{j}_1 = (0, 1, 0)$ given by

$$\text{yaw} = atan\left(\frac{\mathbf{i}_{f2}}{\mathbf{i}_{f1}}\right), \quad \text{roll} = atan\left(\frac{\mathbf{i}_{f3}}{\mathbf{i}_{f2}}\right), \quad \text{pitch} = atan\left(\frac{\mathbf{i}_{f3}}{\mathbf{i}_{f1}}\right).$$

Note that these angles handle only 180 degrees.

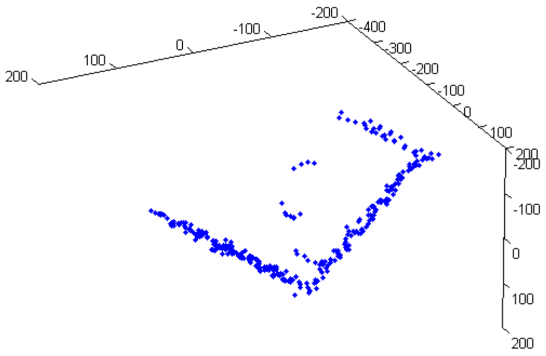


The paraperspective factorization is very sensitive.

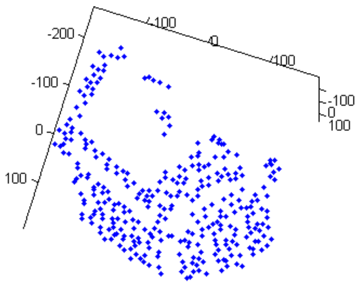
5.3 Hotel Sequences

The number of frames of the hotel sequences is 101 and 430 feature points were tracked and finally 382 feature points were used.

Orthographic



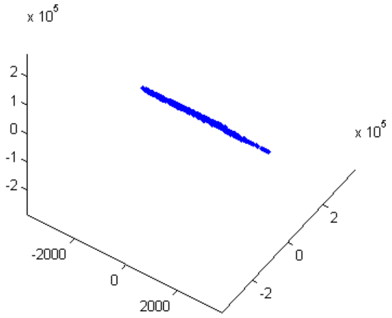
(a) top view



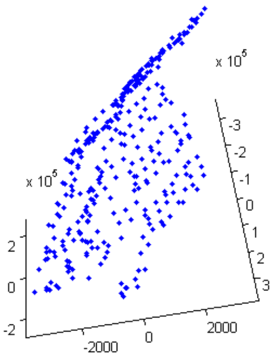
(b) upper view

Paraperspective

The metric constraints (7) turned out a matrix to be non-positive definite, thus we applied an algorithm to create a positive definite matrix which approximates the original non-positive definite matrix [7][8]. Because of this approximation, the result of the paraperspective factorization resulted worse than the orthographic factorization.

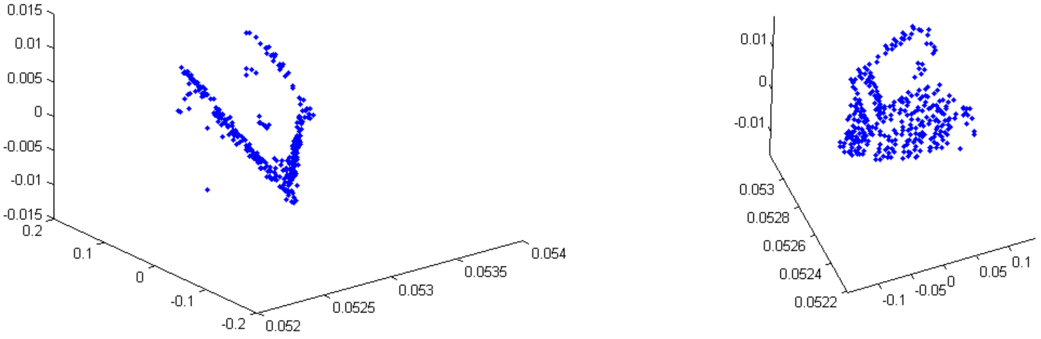


(a) top view (resulted in no thickness)



(b) side view

Projective

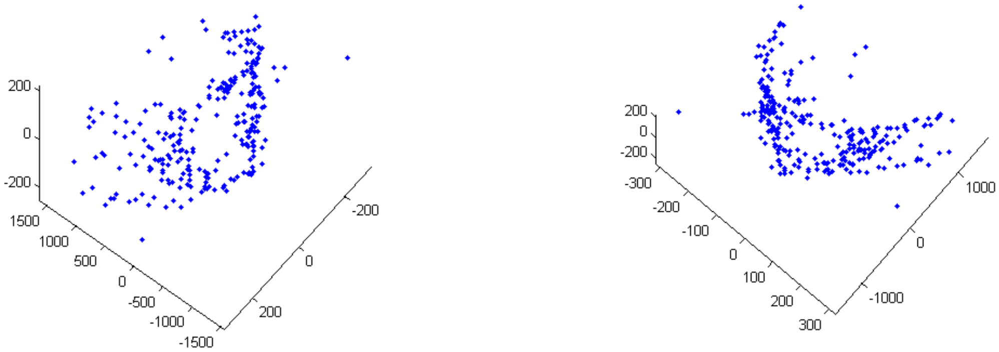


(a) top view (b) upper view

5.4 Medusa Sequences

For the medusa sequences, the middle sequences of the medusa sequences which her frontal faces are shown were cropped. The number of frames is 70 and 260 feature tracking points were used.

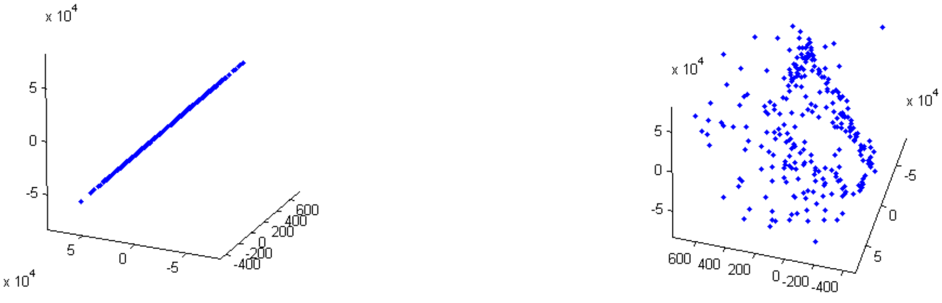
Orthographic



(a) side view (b) upper view

Parapespective

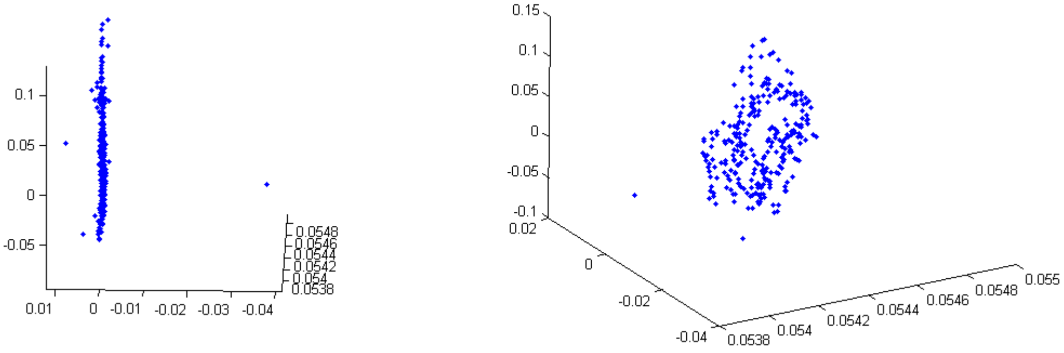
As the case for hotel sequences, the metric constraints (7) turned out a matrix to be non-positive definite again.



(a) top view

(b) side view

Projective



(a) top view

(b) side view

6 Observation

Three factorization methods, the orthographic factorization [1], the paraperspective factorization [2], and the projective factorization [3], were implemented.

The orthographic factorization recovered the 3D motions and structures satisfactory, however, the paraperspective factorization which is more dependent on good data could not recover the 3D motions and structures for given real scene image sequences. If desired data is obtained, the paraperspective assumption should be more robust for image sequences in which the object translates significantly toward or away from the camera. We solved the metric transformation problem by solving linear systems, and it impoverished the estimation of motion and structure. This problem might be resolved by using another method such as Newton method to solve the metric transformation instead.

The projective factorization method also resulted poorer in terms of estimation of structures. The projective factorization requires more estimation steps including the estimation of the fundamental matrix, and this could cause the poor results.

In conclusion, the simplest orthographic factorization was most robust in our experiments. However, this does not mean that the orthographic factorization is always best because the paraspective and projective facorization enables to estimate the camera motion in the direction of optical axis, too.

The program source codes are available at `http://note.sonots.com/?SciSoftware%2FFactorization`.

Bibliography

[1] C. Tomasi and T. Kanade, *Shape and motion from image streams – a factorization method,* International Journal of Computer Vision, 9(2):137–154, 1992. `http://www.pnas.org/cgi/reprint/90/21/9795.pdf` `http://www.hpl.hp.com/personal/Donald_Tanguay/cvrg/tomasiTr92Text.pdf`

[2] C. J. Poelman and T. Kanade, *A paraperspective factorization method for shape and motion recovery,* Pattern Analysis and Machine Intelligence, IEEE Transactions on , vol.19, no.3, pp.206-218, Mar 1997

[3] Bill Triggs, *Factorization methods for projective structure and motion.* In Proceeding of 1996 Computer Society Conference on Computer Vision and Pattern Recognition, pages 845–51, San Francisco, CA, USA, 1996. IEEE Comput. Soc. Press. `http://citeseer.ist.psu.edu/article/triggs96factorization.html`

[4] R. Hartley. In defence of the 8-point algorithm. In *Proceedings of the 5th International Conference on Computer Vision,* pages 1064-1070, June 1995.

[5] S. Butterfield and D. Hogg. An Evaluation of the Normalised 8-Point Algorithm, 1998. `http://citeseer.comp.nus.edu.sg/butterfield95evaluation.html`

[6] T. Morita and T. Kanade, *A Sequential Factorization Method for Recovering Shape and Motion from Image Streams,* Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 19, no.8, pp.858-867, Aug 1997

[7] Nicholas J. Higham, *Computing a nearest symmetric positive semidefinite matrix.* Linear Algebra and its Applications, 103. pp. 103-118. 1998 `http://www.maths.man.ac.uk/~nareports/narep126.pdf`

[8] *CSE 252B: Computer Vision II Lecture 16 Structure from Motion from Tracked Points, UCSD*, pp.7. `http://www-cse.ucsd.edu/classes/sp04/cse252b/notes/lec16/lec16.pdf`

[9] KLT: An Implementation of the Kanade-Lucas-Tomasi Feature Tracker `http://www.ces.clemson.edu/~stb/klt/`

[10] CMU VASC Image Database: hotel http://vasc.ri.cmu.edu//idb/html/motion/hotel/index.html

[11] Leuven castle image sequence http://www.cs.unc.edu/~marc/data/castlejpg.zip

[12] Sagalassos medusa head sequence http://www.cs.unc.edu/~marc/medusa.dv