Some Notes on 3D Computer Vision

(last edited 04/20/2004)

1 Introduction

This short set of notes is intended to be a helpful guide for the final assignment. It complements the lectures, and other materials on 3D computer vision posted on the course web site at:

`http://mil.ufl.edu/~nechyba/eel6562/course_materials.html`

2 Camera calibration

In this section, we show how to compute the projection matrix P that maps 3D world coordinates onto 2D image coordinates.

2.1 Definitions

Let (x, y) denote a 2D image coordinate corresponding to a 3D world coordinate (X, Y, Z). Then, the *projection matrix* P,

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \tag{1}$$

defines the mapping from 3D world coordinates to 2D image coordinates, such that,

$$\begin{bmatrix} sx \\ sy \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2}$$

where s denotes an arbitrary homogeneous scale factor. Note that the projection matrix P is a function of both the *intrinsic* and *extrinsic* parameters of the camera.

2.2 Estimation of P

Here, we assume that we are given a set of n points for which we know both the 2D image coordinates (x_k, y_k) and 3D world coordinates (X_k, Y_k, Z_k), $k \in \{1, \dots, n\}$. From these, we would like to estimate P.

Let us first expand equation (2), to arrive at the following 3D to 2D mapping:

$$x = \frac{p_{11}X + p_{12}Y + p_{13}Z + p_{14}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}} \tag{3}$$

$$y = \frac{p_{21}X + p_{22}Y + p_{23}Z + p_{24}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}} \tag{4}$$

In equations (3) and (4), (x, y) and (X, Y, Z) are known, and the parameters of the projection matrix P, p_{ij},

$i \in \{1,2,3\}, j \in \{1,2,3,4\}$ are unknown. We can rewrite equations (3) and (4) in matrix-vector notation as:

$$\begin{bmatrix} -X & -Y & -Z & -1 & 0 & 0 & 0 & 0 & xX & xY & xZ & x \\ 0 & 0 & 0 & 0 & -X & -Y & -Z & -1 & yX & yY & yZ & y \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{5}$$

Thus, each pair of points, (x_k, y_k) and (X_k, Y_k, Z_k), gives us two linear constraints (equations) in terms of the 12 unknown parameters P. For n points we get $2n$ constraints:

$$\begin{bmatrix} -X_1 & -Y_1 & -Z_1 & -1 & 0 & 0 & 0 & 0 & x_1X_1 & x_1Y_1 & x_1Z_1 & x_1 \\ 0 & 0 & 0 & 0 & -X_1 & -Y_1 & -Z_1 & -1 & y_1X_1 & y_1Y_1 & y_1Z_1 & y_1 \\ -X_2 & -Y_2 & -Z_2 & -1 & 0 & 0 & 0 & 0 & x_2X_2 & x_2Y_2 & x_2Z_2 & x_2 \\ 0 & 0 & 0 & 0 & -X_2 & -Y_2 & -Z_2 & -1 & y_2X_2 & y_2Y_2 & y_2Z_2 & y_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -X_n & -Y_n & -Z_n & -1 & 0 & 0 & 0 & 0 & x_nX_n & x_nY_n & x_nZ_n & x_n \\ 0 & 0 & 0 & 0 & -X_n & -Y_n & -Z_n & -1 & y_nX_n & y_nY_n & y_nZ_n & y_n \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \\ p_{34} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{bmatrix} \tag{6}$$

$$\mathbf{Ap} = \mathbf{0} \tag{7}$$

There are two ways we can solve for the parameters $\mathbf{p}$ in (7) (which is short-hand notation for equation (6) above). We can arbitrarily set one of the parameters in $\mathbf{p}$ equal to 1 (e.g. $p_{34} = 1$), such that:

$$\begin{bmatrix} -X_1 & -Y_1 & -Z_1 & -1 & 0 & 0 & 0 & 0 & x_1X_1 & x_1Y_1 & x_1Z_1 \\ 0 & 0 & 0 & 0 & -X_1 & -Y_1 & -Z_1 & -1 & y_1X_1 & y_1Y_1 & y_1Z_1 \\ -X_2 & -Y_2 & -Z_2 & -1 & 0 & 0 & 0 & 0 & x_2X_2 & x_2Y_2 & x_2Z_2 \\ 0 & 0 & 0 & 0 & -X_2 & -Y_2 & -Z_2 & -1 & y_2X_2 & y_2Y_2 & y_2Z_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -X_n & -Y_n & -Z_n & -1 & 0 & 0 & 0 & 0 & x_nX_n & x_nY_n & x_nZ_n \\ 0 & 0 & 0 & 0 & -X_n & -Y_n & -Z_n & -1 & y_nX_n & y_nY_n & y_nZ_n \end{bmatrix} \begin{bmatrix} p_{11} \\ p_{12} \\ p_{13} \\ p_{14} \\ p_{21} \\ p_{22} \\ p_{23} \\ p_{24} \\ p_{31} \\ p_{32} \\ p_{33} \end{bmatrix} = \begin{bmatrix} -x_1 \\ -y_1 \\ -x_2 \\ -y_2 \\ \vdots \\ -x_n \\ -y_n \end{bmatrix} \tag{8}$$

$$\mathbf{Ap} = \mathbf{b} \tag{9}$$

Equation (9) can now be solved using linear least squares:

$$\mathbf{p} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{10}$$

Alternatively, we can minimize $\mathbf{Ap}$ subject to the constraint $||\mathbf{p}|| \neq 0$ (e.g. $||\mathbf{p}|| = 1$), such that $\mathbf{p}$ will be given by,

$$\min_{\mathbf{p}} ||\mathbf{Ap}||, ||\mathbf{p}|| = 1 \tag{11}$$

For the problem formulation in equation (11), the solution for $\mathbf{p}$ is given by the eigenvector $\mathbf{v}$ of $\mathbf{A}^T\mathbf{A}$ corresponding to the smallest eigenvalue. This eigenvector $\mathbf{v}$ can be computed through *singular value decomposition (SVD)* [2]. SVD is an extremely useful linear algebra tool that decomposes any $m \times n$ matrix $\mathbf{A}$, $m > n$ as follows:

$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{12}$$

where $\mathbf{U}$ is an $m \times n$ orthogonal matrix, $\mathbf{D}$ is an $n \times n$ diagonal matrix whose diagonal elements σ_i are the *singular values* of $\mathbf{A}$, arranged from largest to smallest, and $\mathbf{V}$ is an $n \times n$ orthogonal matrix of eigenvectors $\mathbf{v}_i$ corresponding to singular values σ_i. In this decomposition, the last column of $\mathbf{V}$ corresponds to the solution for $\mathbf{p}$.

Thus, to solve for $\mathbf{p}$ in equation (11), we first compute the SVD decomposition of $\mathbf{A}$ or $\mathbf{A}^T\mathbf{A}$, and then assign the last column of the resulting $\mathbf{V}$ matrix as our solution. Functions for doing SVD are readily available in most mathematical software packages, including *Matlab* and *Mathematica*.

2.3 Triangulation from multiple views

Here, we assume that we are given the 2D image coordinates of a 3D point in the world in two different views of the same scene; let us denote these 2D coordinates as (x, y) and (x', y'). Furthermore, we assume that we know the projection matrices P and P' corresponding to the two different views. Our goal here is to estimate the 3D coordinate $\mathbf{X} = (X, Y, Z)$ of the imaged point.

Rewriting equations (3) and (4), we get:

$$\begin{bmatrix} xp_{31} - p_{11} & xp_{32} - p_{12} & xp_{33} - p_{13} \\ yp_{31} - p_{21} & yp_{32} - p_{22} & yp_{33} - p_{23} \\ x'p'_{31} - p'_{11} & x'p'_{32} - p'_{12} & x'p'_{33} - p'_{13} \\ y'p'_{31} - p'_{21} & y'p'_{32} - p'_{22} & y'p'_{33} - p'_{23} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} -(xp_{34} - p_{14}) \\ -(yp_{34} - p_{24}) \\ -(x'p'_{34} - p'_{14}) \\ -(y'p'_{34} - p'_{24}) \end{bmatrix} \tag{13}$$

$$\mathbf{A}\mathbf{X} = \mathbf{b} \tag{14}$$

Now, equation (14) can be solved for $\mathbf{X}$ using equation (10).

3 Two-view epipolar geometry

In this section, we discuss several ways of computing the fundamental matrix F that defines the *epipolar geometry* relating two views of the same scene; we assume that some number of corresponding 2D image-point pairs (x, y) and (x', y') are known.

3.1 Definitions

Let,

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \mathbf{x}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \tag{15}$$

denote the homogeneous representation of corresponding 2D image coordinates (x, y) and (x', y'). Then, the *fundamental matrix* F,

$$F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \tag{16}$$

defines the epipolar geometry such that,

$$\mathbf{x}'^T F \mathbf{x} = 0 \tag{17}$$

In lecture, we showed that F can be represented as a function of the intrinsic parameters K and K' and the relative orientation and translation between the two views R and $\mathbf{t}$ as:

$$F = K'^{-T}[\mathbf{t}]_\times R K^{-1} \tag{18}$$

where for a vector $\mathbf{v} = \begin{bmatrix} v_1 & v_2 & v_3 \end{bmatrix}^T$,

$$[\mathbf{v}]_\times = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \tag{19}$$

As such, F is a matrix of rank 2 (i.e. $\det(F) = 0$) and is determined only up to a scale factor, as can be seen from equation (18).

3.2 Estimation of F

Here, we assume that n corresponding 2D image-point pairs, (x_k, y_k) and (x'_k, y'_k), $k \in \{1, \dots, n\}$ are known, and that their homogeneous representation is given by $\mathbf{x}_k$ and $\mathbf{x}'_k$, respectively.[1] From these, we would like to estimate F.

Let us expand equation (17):

$$\begin{bmatrix} x'x & x'y & x' & y'x & y'y & y' & x & y & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0 \tag{20}$$

Thus, each pair of 2D image points, (x_k, y_k) and (x'_k, y'_k), gives us one linear constraint (equation) in terms of the 9 unknown parameters F. For n points we get:

$$\begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ x'_2x_2 & x'_2y_2 & x'_2 & y'_2x_2 & y'_2y_2 & y'_2 & x_2 & y_2 & 1 \\ x'_3x_3 & x'_3y_3 & x'_3 & y'_3x_3 & y'_3y_3 & y'_3 & x_3 & y_3 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0 \tag{21}$$

$$\mathbf{A}\mathbf{f} = \mathbf{0} \tag{22}$$

In the subsections below, we describe several different algorithms for computing F, starting with equation (22). First, however, we discuss how to evaluate the quality of estimation for F.

[1] For the purposes of these notes, we assume that all correspondences are correct; that is, we don't consider the possibility of mismatched pairs. If such outliers do exist, the RANSAC algorithm should be applied to detect and remove these outliers.

3.3 Evaluation of F estimation

In the presence of noise (e.g. small errors in 2D image coordinates of corresponding pairs $\mathbf{x}_k$ and $\mathbf{x}'_k$),

$$\mathbf{x}'_k F \mathbf{x}_k \neq 0 \tag{23}$$

In other words, $\mathbf{x}'_k$ in one image will not, in general, fall exactly on the epipolar line $F\mathbf{x}_k$, and $\mathbf{x}_k$ will not, in general, fall exactly on the epipolar line $F^T\mathbf{x}'_k$. Therefore, we can measure the quality of the F estimation as a function of the distance between image points and epipolar lines. Consider a homogeneous point $\mathbf{x}$ and 2D-line ℓ (in homogeneous notation):

$$\mathbf{x} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad \ell = \begin{bmatrix} \lambda \\ \mu \\ \nu \end{bmatrix} \tag{24}$$

Then, the distance $d(\mathbf{x}, \ell)$ between $\mathbf{x}$ and ℓ is given by [1],

$$d(\mathbf{x}, \ell) = \frac{\mathbf{x}^T \ell}{\sqrt{\lambda^2 + \mu^2}} \tag{25}$$

Therefore, for a set of 2D image-point pairs,

$$J = \sum_k d(\mathbf{x}'_k, F\mathbf{x}_k)^2 + d(\mathbf{x}_k, F^T\mathbf{x}'_k)^2 \tag{26}$$

defines a cost function that measures the quality of the F estimation; the closer J is to zero, the better is the estimate of F.

3.4 Eight-point algorithm

As before, we can apply SVD, such that $\mathbf{f}$ is given by,

$$\min_{\mathbf{f}} ||\mathbf{Af}||, ||\mathbf{f}|| = 1 \tag{27}$$

However, this solution for F (i.e. $\mathbf{f}$) is not guaranteed to be of rank 2. We can enforce this constraint, by applying SVD once again. Let,

$$F = \mathbf{UDV}^T \tag{28}$$

where,

$$\mathbf{D} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & \sigma_3 \end{bmatrix} \tag{29}$$

Then, the closest singular matrix F' to F is given by,

$$F' = \mathbf{UD'V}^T \tag{30}$$

where,

$$\mathbf{D'} = \begin{bmatrix} \sigma_1 & 0 & 0 \\ 0 & \sigma_2 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{31}$$

Note: While this algorithm is known as the "eight-point algorithm", obviously more pairs of corresponding image points can and should be used for better results; "eight" simply refers to the smallest allowable number of points.

3.5 Normalized eight-point algorithm

This algorithm proceeds as the "eight-point" algorithm, except that homogeneous coordinates $\mathbf{x}_k$ and $\mathbf{x}'_k$ are first mapped through an affine transformation T and T' that seeks to mitigate poor conditioning of the matrix $\mathbf{A}^T\mathbf{A}$ in equation (22). Here's an outline of the algorithm:

1. Transform coordinates $\mathbf{x}_k$ and $\mathbf{x}'_k$ to $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{x}}'_k$, respectively, where,
$$\hat{\mathbf{x}}_k = T\mathbf{x}_k, \quad \hat{\mathbf{x}}'_k = T'\mathbf{x}'_k. \tag{32}$$
2. Find the fundamental matrix $\hat{F}$ corresponding to the transformed image coordinates $\hat{\mathbf{x}}_k$ and $\hat{\mathbf{x}}'_k$ using the eight-point algorithm from above (including enforcement of the rank 2 constraint).
3. Set $F = T'^T \hat{F} T$ as the fundamental matrix corresponding to the original, untransformed coordinates $\mathbf{x}_k$ and $\mathbf{x}'_k$.

See Section 5 in [1], for more information on appropriate mappings T, T'.

3.6 Seven-point algorithm

In the seven-point algorithm, the rank 2 constraint (i.e. $\det(F) = 0$) is explicitly enforced. Once again, SVD plays a hand. Let
$$\mathbf{A} = \mathbf{U}\mathbf{D}\mathbf{V}^T \tag{33}$$
where $\mathbf{A}$ refers to equation (22). Without proof of why this should be so, the solution for F (i.e. $\mathbf{f}$) is now parameterized as,
$$\mathbf{f} = \mathbf{f}_1 + \lambda\mathbf{f}_2 \tag{34}$$
or, alternatively,
$$F = F_1 + \lambda F_2 \tag{35}$$
where $\mathbf{f}_1$ and $\mathbf{f}_2$ are the two right-most columns of V in (33). The constraint $\det(F) = 0$ leads to a cubic polynomial in λ,
$$\lambda^3 + a_2\lambda^2 + a_1\lambda + a_0 = 0 \tag{36}$$
which will yield either one or three real-valued solutions in λ.

To find the correct value for λ in the case of multiple real-valued solutions, one can first perform the above computations on a subset (≥ 7) of all available correspondences, and then evaluate the cost function J in equation (26) over all correspondences for the three possible values of λ. The correct value of λ will yield the smallest value of J.

Note: While this algorithm is known as the "seven-point algorithm", obviously more pairs of corresponding image points can and should be used for better results; "seven" simply refers to the smallest allowable number of points.

3.7 Nonlinear minimization

The quality of the F estimation for the above summarized algorithms can typically be improved substantially through iterative, nonlinear optimization of criterion J in equation (26). Such minimization proceeds as follows:

1. Find an initial estimate F_0 through, for example, the eight-point algorithm.
2. Using F_0 as the initial estimate, minimize J through nonlinear optimization (e.g. Levenberg-Marquardt, conjugate-gradient, etc.)

References

[1] R. I. Hartley, "In Defense of the Eight-Point Algorithm," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, pp. 580-93, 1997.

[2] W. H. Press, *et. al*, *Numerical Recipes in C: the Art of Scientific Computing, 2nd ed.*, Section 2.6, pp. 59-70, Cambridge University Press, Cambridge, 1992.