Rigid Motion from Point Matches

In this tutorial we address the following problem: "Given two sets of points matched between two images find the rotation (given by rotation matrix R) and translation (given by translation vector **T**) which map S_1 into S_2".

There are many algorithms for this problem ([1, 2, 4, 6, 8, 9, 10, 11]) and we have classified them here in three types:

1. Essential matrix algorithms – linear algorithms that require at least eight points ([6, 7, 12]). Extensions of these algorithms when five to seven point matches are available are discussed in [2, 8, 11].
2. Direct algorithms – these algorithms solve for rotation and translation directly and require minimisation of five dimensional nonlinear cost functions ([3, 10, 12]).
3. Subspace algorithm – valid for small rotation between two frames. Requires nonlinear minimisation of two-dimensional cost function [4].

The coordinate system is fixed on the camera with the origin coinciding with the projection centre of the camera and the z-axes coinciding with the optical axes and pointing toward the scene. The focal length is assumed known, and without loss of generality set to unity. Therefore, the equation of the image plane is $z = 1$.

Let us consider an arbitrary point before and after the motion. The following notation is introduced and will be used throughout the tutorial.

$\mathbf{P} = (X, Y, Z)^T$	spatial vector of the point before the motion;
$\mathbf{P}' = (X', Y', Z')^T$	spatial vector of the point after the motion;
$\mathbf{p} = (x, y, 1)^T = \mathbf{P}/Z$	image vector of the point before the motion; and
$\mathbf{p}' = (x', y', 1)^T = \mathbf{P}'/Z'$	image vector of the point after the motion.

A mapping $[\bullet]_\times$ from a 3-dimensional vector to a 3×3 matrix is defined as follows:

$$\left[(x_1 \quad x_2 \quad x_3)^T\right]_\times = \begin{bmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{bmatrix}$$

Using this mapping, we can express the cross product operation of two vectors by the matrix vector multiplication: $\mathbf{a} \times \mathbf{b} = [\mathbf{a}]_\times \mathbf{b}$.

The motion model that is assumed in this paper is a rigid motion model which can be decomposed into rotation and translation and is described by the following equation:

$$\mathbf{P}' = R \cdot \mathbf{P} + \mathbf{T} \tag{1}$$

and after some rearrangement (see [19]) we obtain:

$$\mathbf{p}'^T \, (\mathbf{T_0} \times (R\mathbf{p})) = 0 \tag{2}$$

where $\mathbf{T_0}$ denotes the direction of translation. Equation (2) is often referred to as the epipolar constraint and its geometrical interpretation is that vectors $\mathbf{p}'$, $R\mathbf{p}$ and $\mathbf{T_0}$ must lie in the same plane which is called epipolar plane of the point. Its intersection with the image plane is called the epipolar line of the point.

Alternatively, equation (2) may be rewritten as:

$$\mathbf{p}'^T\, E\mathbf{p} = 0 \tag{3}$$

where $E = [\mathbf{T_0}]_\times R$. The matrix E is called the essential matrix and has been found to be very useful in Computer Vision. The essential matrix has five degrees of freedom, and hence, at least five point matches are necessary to estimate it. Other important properties of the essential matrix are given in Lemmas 1-3:

Lemma 1. $\mathbf{T_0}$ is a unit eigenvector of E^T corresponding to the zero eigenvalue.

Proof: $E^T\mathbf{T_0} = ([\mathbf{T_0}]_\times R)^T\,\mathbf{T_0} = -R^T[\mathbf{T_0}]_\times\mathbf{T_0} = -R^T\mathbf{T_0}\times\mathbf{T_0} = -R^T\,\mathbf{0} = 0\cdot\mathbf{T_0}$.

Lemma 2. $\mathrm{svd}(E) = (1\,1\,0)^T$ or equivalently, eigenvalues of EE^T are 1, 1 and 0. For a proof see [7, 19].

Lemma 3. $\|E\|^2 = \mathrm{trace}(E^T E) = 2$. See [18] for a proof.

Given n point correspondences, equation (3) can be rewritten as linear equations in the elements of E and we obtain

$$A\mathbf{E} = \mathbf{0}, \tag{4}$$

where

$$A = \begin{bmatrix} x_1x_1' & x_1y_1' & x_1 & y_1x_1' & y_1y_1' & y_1 & x_1' & y_1' & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_nx_n' & x_ny_n' & x_n & y_nx_n' & y_ny_n' & y_n & x_n' & y_n' & 1 \end{bmatrix}$$

and

$$E = [e_{11} \quad e_{21} \quad e_{31} \quad e_{12} \quad e_{22} \quad e_{32} \quad e_{13} \quad e_{23} \quad e_{33}], \text{ for } E = [e_{ij}]_{3\times 3}.$$

One method of solving for rotation and translation when more than seven point matches are available is the eight-point algorithm proposed by Longuet-Higgins [6]. The algorithm first computes the coefficients of E from the equation (3) and then determines R and $\mathbf{T_0}$.

The essential matrix is found by minimising the error in equation (3), i.e. by solving for vector $\mathbf{E}$ in

$$\min_{\mathbf{E}}\|A\mathbf{E}\|, \quad \text{subject to:} \quad \|\mathbf{E}\|^2 = 2. \tag{5}$$

The solution of (4) is the eigenvector of A^TA associated with the smallest eigenvalue.

Once E is found, $\mathbf{T_0}$ is determined as the unit eigenvector of E^T and R is found by solving the minimisation problem

$$\min_R \left\| [\mathbf{T_0}]_\times R - E \right\|, \quad \text{subject to: } \ R \text{ is a rotation matrix.}$$

This problem may be solved by using quaternion representation of rotation matrix, see [12] for details.

The drawbacks of this algorithm are:

- It needs at least eight points, although the solution exists when five or more point matches are available; and
- It is sensitive to noise.

In this tutorial we are not going to discuss solution of the problem when less than eight point matches are available. Two, a bit complicated methods, which are based on the solution of tenth order polynomial are given in [2,8]. Somehow simpler method, based on the minimisation of a two-dimensional cost function is given in [11].

As can be seen from the Lemma 2, the essential matrix must satisfy two constraints:

1) it has rank 2 (which may be represented as a polynomial of degree 3 in coefficients of E); and
2) the nonzero singular values of E must be equal (polynomial constraint of degree 4).

When solving for E in the first step, these constraints are simply ignored and that's why the eight-point algorithm is linear. The price that must be paid is that this algorithm is highly sensitive to noise and often performs unsatisfactory.

Therefore, it often pays to find the motion parameters directly by minimising the cost function:

$$\sum_{i=1}^{n} \left(\mathbf{p}'^T_i \, \mathbf{T_0} \times (R\mathbf{p}_i) \right)^2 \qquad (6).$$

which is derived directly from equation (2). Unfortunately, there is no closed form solution for the minimisation problem (6) and the nonlinear minimisation methods must be used. Better solution of the motion parameters have been reported by using geometrical criteria such as distance of the point to its epipolar line, or to its optimal reprojection. In general, the minimisation criteria is of the form:

$$\sum_{i=1}^{n} \alpha_i \left(\mathbf{p}'^T_i \, \mathbf{T_0} \times (R\mathbf{p}_i) \right)^2 \qquad (7)$$

where α_i is a weight that depends upon chosen criteria. The most common criteria is distance of the points in both images to the corresponding epipolar lines in which case the weight is given by:

$$\alpha_i = \frac{1}{\left\| R^T (\mathbf{T_0} \times \mathbf{p}'^T_i) \right\|^2 + \left\| R(\mathbf{T_0} \times \mathbf{p}^T_i) \right\|^2}.$$

When the motion (or more precisely, rotation) between two images is small, an approximate algorithm proposed by Heeger and Jepson [4] may be used. Essentially, they used a first order approximation of rotation matrix, which is valid when the angle of rotation is small

$$R = I + [\mathbf{\Omega}]_\times$$

where I denotes a 3×3 identity matrix, and $\mathbf{\Omega}$ denotes angles of rotation. The motion equation (1) now becomes:

$$\Delta\mathbf{P}_i = \mathbf{P}'_i - \mathbf{P}_i = \mathbf{\Omega}\times\mathbf{P}_i + \mathbf{T}$$

and, after short rearrangement, we obtain:

$$\mathbf{v}_i = \Delta\mathbf{p}_i = d_i A_i \mathbf{T_0} + B_i \mathbf{\Omega}$$

where $\mathbf{v}_i$ denotes image velocity, d_i denotes scaled inverse depth ($\|\mathbf{T}\|/Z_i$) and

$$A_i = \begin{bmatrix} 1 & 0 & -x_i \\ 0 & 1 & -y_i \end{bmatrix} \qquad B_i = \begin{bmatrix} -x_i y_i & 1 + x_i^2 & -y_i \\ -1 - y_i^2 & x_i y_i & x_i \end{bmatrix}.$$

If we assume that $\mathbf{T_0}$ is known and collect all the equations into one matrix, we obtain matrix equation:

$$\mathbf{v} = A(\mathbf{T})\mathbf{p} + B\mathbf{\Omega} = C(\mathbf{T})\mathbf{q}$$

where

$$A(\mathbf{T}) = \begin{bmatrix} A_1\mathbf{T} & \cdots & \mathbf{0} \\ \vdots & & \\ \mathbf{0} & & A_n\mathbf{T} \end{bmatrix}, \quad B = \begin{bmatrix} B_1 \\ 0 \\ B_n \end{bmatrix}, \quad C(\mathbf{T}) = \begin{bmatrix} A(\mathbf{T}) & B \end{bmatrix} \text{ and } \mathbf{q} = \begin{bmatrix} \mathbf{p} \\ \Omega \end{bmatrix}.$$

The cost function is defined as

$$E(\mathbf{T},\mathbf{q}) = \left\|\mathbf{v} - C(\mathbf{T})\mathbf{q}\right\|^2,$$

and it may be shown that minimising this function over $\mathbf{T}$ and $\mathbf{q}$ is equivalent to minimising following cost function over $\mathbf{T}$ alone:

$$E(\mathbf{T}) = \left\|(I - C(T)C(\mathbf{T})^*\right\|^2$$

where * denotes a pseudo-inverse of a matrix. So, our goal is to find translation vector $\mathbf{T_0}$ which yields minimum cost function $E(\mathbf{T})$. This problem has to be solved by using nonlinear optimisation, but it is inexpensive, as it requires only minimisation over two-dimensional space (surface of the unit sphere).

References

[1] Aggarwal J.K. and Nandhakumar N., "On the computation of Motion from Sequences of Images – A review*", Proceedings of the IEEE*, vol. 76, pp. 917-935, 1988.

[2] Faugeras O. and Maybank S., "Motion from Point Matches: Multiplicity of Solutions", *Intl. Journal Computer Vision,* vol. 4, pp. 225-246, 1990.

[3] Harris C. G., "Determination of ego-motion from matched points", In *Proceedings of the 3rd Alvey Vision Conference,* pp. 189-192, 1987.

[4] Heeger D. and Jepson A., "Subspace Methods for Recovering Rigid Motion I: Algorithm and Implementation", *Intl. Journal Computer Vision,* vol. 7, pp. 95-117, 1992.

[5] Huang T. and Faugeras O., "Some Properties of the *E* Matrix in Two-View Motion Estimation", *IEEE PAMI,* vol. 11, 1310-1312, December 1989.

[6] Longuet-Higgins, H.C., "A computer algorithm for reconstructing a scene from two projections", *Nature* 293, pp.133-135, 1981.

[7] Longuet-Higgins, H.C., The reconstruction of a scene from two projections – configurations that defeat the 8-point algorithm",*Proc. IEEE 1st Conf. On Artif. Intell. Applications*, pp. 395-397, 1984.

[8] Netravali A. *et. al*. "Algebraic Methods in 3-D Motion estimation from Two-View Point Correspondences", *Intl. Journal of Imaging Systems and Technology*, vol. 1., pp. 78-99, 1989.

[9] Philip J.,"Estimation of three-dimensional motion of rigid objects from noisy observations", *IEEE Trans. PAMI,* vol. 13, pp. 61-66, 1991.

[10] Szeliski R. and Kang S. B., "Recovering 3D shape and motion from image streams using nonlinear least squares", *Journal of Visual Communication and Image Representation*, vol. 5, pp.10-28, 1994.

[11] Trajković M. and Hedley M. "The eigenspace method for motion recovery from less than eight point correspondences", (manuscript in preparation).

[12] Weng J., Huang. T and Ahuja N., "Motion and Structure from Image Sequences", Springer-Verlag, 1993.