Introduction of EPnP

Yibin Wu, GNSS Research Center, WHU

EPnP[1] is a typical solution to the PnP (Perspetive-n-points) problem---the estimation of the pose of a calibrated camera from n 3D-to-2D point correspondences ,as shown in Figure 1. It is applicable for all $n \geq 4$ and handles properly for both planar and co-planar configurations. Its core idea is to express the n 3D points as a weighted sum of four virtual control points and compute the coordinates of points in camera-coordinate, since it is easy to estimate pose from n 3D-to-3D point correspondences. PnP is a basic algorithm for problems which need estimate the camera pose from consecutive frames, like SLAM, VIO, SFM, etc. A comparison of several PnP methods is shown in Table 1.

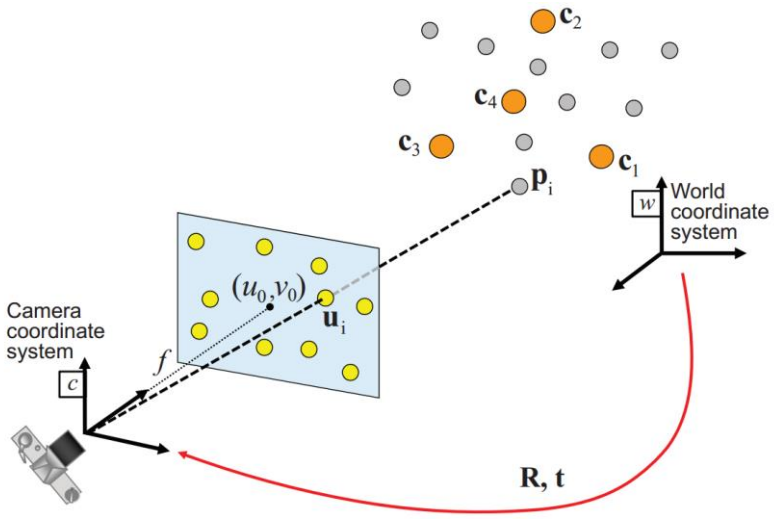

Figure 1 PnP problem

Table 1 Comparison of PnP methods

PnP method	Need Intrinsic Paras.?	No. of correspondences at least	Precision	Speed
P3P	Yes	3 (4)	Medium	Fast
EPnP	Yes	4	high	Fast
UPnP	No	4	Medium	Fast
DLT	No	6	Medium	Fast
MRE(LSI)	Yes	3	highest	Slow

The main process can be divided into 5 steps:

1). Select four control points c_j^w;

2). Compute the coefficient α;

3). Compute the c_j^c in the camera coordinate

3.1) $Mx = 0$ 3.2) $L\beta = \rho$ 3.3) Gauss-Newton;

4). Recover 3d points in the camera coordinate;

5). Compute the R and t.

At first, four control points are selected in the world coordinates using the points centroid and PCA (Principal Components Analysis) to ensure stability, then,

$$p_i^w = \sum_{j=1}^{4} \alpha_{ij} c_j^w \text{ , } with \sum_{j=1}^{4} \alpha_{ij} = 1$$

The α_{ij} can be computed as soon as the control points are selected since p_i^w are known. The same relation holds in the camera coordinate system and we can also write

$$p_i^c = \sum_{j=1}^{4} \alpha_{ij} c_j^c$$

From the pinhole projection model, we have

$$s_i \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} p_i^c = \begin{pmatrix} f_x & 0 & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \sum_{j=1}^{4} \alpha_{ij} \begin{pmatrix} x_j^c \\ y_j^c \\ z_j^c \end{pmatrix}$$

s_i are scalar projective parameters.$s_i = \sum_{j=1}^{4} \alpha_{ij} \ z_j^c$. Then

$$\begin{cases} \sum_{j=1}^{4} \alpha_{ij} \left[f_x X_j^c + Z_j^c (c_x - u_i) \right] = 0 \\ \sum_{j=1}^{4} \alpha_{ij} \left[f_y Y_j^c + Z_j^c (c_y - v_i) \right] = 0 \end{cases}$$

Hence, by concatenating them for all n reference points, we generate a linear system of the form

$$\boldsymbol{M}_{2n*12} * \boldsymbol{X}_{12*1} = 0$$

$$\begin{pmatrix} \alpha_{i1} f_x & 0 & \alpha_{i1}(c_x - u_i) & \cdots & \alpha_{i4} f_x & 0 & \alpha_{i4}(c_x - u_i) \\ 0 & \alpha_{i1} f_y & \alpha_{i1}(c_y - v_i) & \cdots & 0 & \alpha_{i4} f_y & \alpha_{i4}(c_y - v_i) \\ & & \cdots & & & \cdots & \end{pmatrix} \begin{pmatrix} X_1^c \\ Y_1^c \\ Z_1^c \\ \cdots \\ X_4^c \\ Y_4^c \\ Z_4^c \end{pmatrix} = 0$$

By SVD (Singular Value Decomposition), we have

$$\boldsymbol{M} = \boldsymbol{U \Sigma V^T}$$

$$\boldsymbol{M^T M}_{12*12} = \boldsymbol{V(\Sigma^T \Sigma) V^T}$$

$$\boldsymbol{X} = \sum_{k=1}^{N} \beta_k \boldsymbol{V}_k$$

The set V_k are the columns of the right-singular vectors of M corresponding to the N null singular of $\boldsymbol{M}$. As analyzed in the paper, we consider that the effective dimension N of the null space of $\boldsymbol{M^T M}$ can vary from 1 to 4 depending on the configuration of

the reference points, the focal length of the camera, and the amount of noise. (Actually, in the code implemented in OpenCV, it is calculated with them respectively and the best one with minimum re-projection error is selected.)

Now, we know $\boldsymbol{V}_k$ and we have to know β_k to calculate $\boldsymbol{X}$.

Since the distance between two points is constant no matter which coordinates they were expressed in, we have

$$\left\|\boldsymbol{c}_i^c - \boldsymbol{c}_j^c\right\|^2 = \left\|\boldsymbol{c}_i^w - \boldsymbol{c}_j^w\right\|^2$$

Here, i, j represent the index of all corresponding points. For example, when $N = 4$,

$$\left\|(\beta_1 \boldsymbol{V}_1^{[i]} + \beta_2 \boldsymbol{V}_2^{[i]} + \beta_3 \boldsymbol{V}_3^{[i]} + \beta_4 \boldsymbol{V}_4^{[i]}) - (\beta_1 \boldsymbol{V}_1^{[j]} + \beta_2 \boldsymbol{V}_2^{[j]} + \beta_3 \boldsymbol{V}_3^{[j]} + \beta_4 \boldsymbol{V}_4^{[j]})\right\|^2$$

$$= \left\|\boldsymbol{c}_i^w - \boldsymbol{c}_j^w\right\|^2$$

$$[(\boldsymbol{V}_1^{[i]} - \boldsymbol{V}_1^{[j]})^2 \quad (\boldsymbol{V}_1^{[i]} - \boldsymbol{V}_1^{[j]})(\boldsymbol{V}_2^{[i]} - \boldsymbol{V}_2^{[j]}) \quad (\boldsymbol{V}_2^{[i]} - \boldsymbol{V}_2^{[j]})^2 \quad (\boldsymbol{V}_1^{[i]} - \boldsymbol{V}_1^{[j]})(\boldsymbol{V}_3^{[i]} - \boldsymbol{V}_3^{[j]}) \quad (\boldsymbol{V}_2^{[i]} - \boldsymbol{V}_2^{[j]})(\boldsymbol{V}_3^{[i]} - \boldsymbol{V}_3^{[j]})$$
$$(\boldsymbol{V}_3^{[i]} - \boldsymbol{V}_3^{[j]})^2 \quad (\boldsymbol{V}_1^{[i]} - \boldsymbol{V}_1^{[j]})(\boldsymbol{V}_4^{[i]} - \boldsymbol{V}_4^{[j]}) \quad (\boldsymbol{V}_2^{[i]} - \boldsymbol{V}_2^{[j]})(\boldsymbol{V}_4^{[i]} - \boldsymbol{V}_4^{[j]}) \quad (\boldsymbol{V}_3^{[i]} - \boldsymbol{V}_3^{[j]})(\boldsymbol{V}_4^{[i]} - \boldsymbol{V}_4^{[j]}) \quad (\boldsymbol{V}_4^{[i]} - \boldsymbol{V}_4^{[j]})^2]$$

$$\cdot[\beta_{11}\ \beta_{12}\ \beta_{22}\ \beta_{13}\ \beta_{23}\ \beta_{33}\ \beta_{14}\ \beta_{24}\ \beta_{34}\ \beta_{44}]^T = \left\|c_i^w - c_j^w\right\|^2$$

$$\boldsymbol{L\beta} = \boldsymbol{\rho}$$

$\boldsymbol{V}_1^{[i]}$ presents the first vector of $\boldsymbol{V}$ that corresponds to the coordinates of the control point c_i^c, as shown in Figure 2. Although the linearization procedure treats all 10 products $\beta_{ab} = \beta_a \beta_b$ as unknown and there are not enough constriants any more, we can solve the problem using a relinearization technique[2].

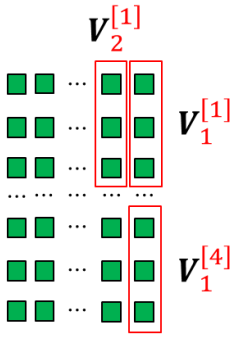

Figure 2 Diagrammatic drawing of $\boldsymbol{V}$

Then, we use the Guass-Newton algorithm to optimize our estimate of $\boldsymbol{\beta}$.

$$Error(\beta) = \sum_{(i,j)\ s.t.\ i<j} (\left\|c_i^c - c_j^c\right\|^2 - \left\|c_i^w - c_j^w\right\|^2)$$

$$Error(\beta) = \sum_{(i,j)\ s.t.\ i<j} (\sum_{k=1}^{4} \left\|\beta_k \boldsymbol{V}_k^{[i]} - \beta_k \boldsymbol{V}_k^{[j]}\right\|^2 - \left\|c_i^w - c_j^w\right\|^2)$$

Take the first order approximation

$$Error(\beta_0 + \Delta\beta) = Error(\beta_0) + Error'(\beta)\Delta\beta = 0$$
$$Error'(\beta)\Delta\beta = \rho - L\beta_0$$

Hence, we have

$$\boldsymbol{A}_{n*4} * \boldsymbol{x}_{4*1} = \boldsymbol{b}_{6*1}$$

$$A = [\frac{\partial Error(\beta)}{\partial \beta_1} \quad \frac{\partial Error(\beta)}{\partial \beta_2} \quad \frac{\partial Error(\beta)}{\partial \beta_3} \quad \frac{\partial Error(\beta)}{\partial \beta_4}]$$

$$= [2L_1\beta_1 + L_2\beta_2 + L_4\beta_3 + L_7\beta_4 \quad L_2\beta_1 + 2L_3\beta_2 + L_5\beta_3 + L_8\beta_4 \quad \cdots \quad \cdots]$$

With about 5 times iteration, we can get a comparatively accurate result.

Now we can calculate the coordinates of control points in camera coordinate and all the corresponding points in camera coordinate of world points.

$$c_j^c = \sum_{k=1}^{4} \beta_k \boldsymbol{V}_k^j$$

$$p_i^c = \sum_{j=1}^{4} \alpha_{ij} c_j^c$$

Therefore, we obtain all the p_i^c and p_i^w and we can solve R and T easily by the following steps,

1) Compute center: $p_c^c = \frac{\sum p_c^i}{N}, p_w^c = \frac{\sum p_w^i}{N}$

2) Remove center: $q_c^i = p_c^i - p_c^c, q_w^i = p_w^i - p_w^c$

3) Compute $H = \sum_{i=1}^{N} q_c^i {q_w^i}^T$

4) SVD: $H = U\Sigma V^T,\ R = VU^T,\ t = p_c^c - Rp_w^c$

$$(if \det(R) < 0,\ R(2,.) = -R(2,.))$$

Finally, we can use the R and T computed by formula above as initial value and use Non-Linear optimization algorithm (Guass-Newton or Levenberg-Marquardt) to optimize them by minimize the re-projection error.

Moreover, if the number of points correspondence is much more than 4, we can use RANSAC to get a more accurate and robust result. RANSAC can always find the optimal solution when the model is determined and the maximum number of iterations is allowed. In the RANSAC code implemented in OpenCV, five points are selected every time to build the model.

Bibliography

[1] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o (n) solution to the pnp problem," *International journal of computer vision,* vol. 81, p. 155, 2009.

[2] A. Kipnis and A. Shamir, "Cryptanalysis of the HFE Public Key Cryptosystem by Relinearization," Berlin, Heidelberg, 1999, pp. 19-30.