General Camera Transform Solver

Owen Lu
Electroimpact Inc.
owenl@electroimpact.com

***Abstract*— Assuming an intrinsic camera matrix, the pose of the camera can be estimated by knowing n 3D coordinates and their projections onto the image plane. The previous calibration model assumed that points were within a plane in 3D in order to calculate its relative pose. This meant that for each picture taken in a different orientation, the two transforms must be composed in order to get a relative transform. As this new algorithm allows arbitrary 3D points to be used in pose calibration, a set of images of points within multiple non-coplanar checkboards can be used to create a set of points forming a volume in their native tool point coordinate system. That is, no coordinate convention is assumed, and coordinate transforms are direct. This paper details an exact method to calculating the pose of a camera using the Efficient Perspective n Point (EPnP) method proposed by Lepetit et. Al.**

I. Introduction

The previous implementation of the camera calibration algorithm implicitly assumes that the world coordinates are intrinsically defined by the calibration checkerboard pattern. Because of this, closed form solutions for the camera intrinsic parameters can easily be written after the homography and intrinsic constraint equations are solved.

In other words, the world coordinates are fixed and all assumed to exist in a plane which makes the calculation of the pose easy relative to this fixed coordinate system. However, transforms relative to other coordinates can only be calculated via composition, which compounds the error.

The problem is that within moving coordinate system C_1 the transform defined by $\mathbf{R}, \mathbf{t}$, which describes the camera pose, is assumed to be a constant. However, calibration points expressed in C_1 contained in $\mathbf{X}$, an $n \times 3$ matrix, are non-coplanar as C_1 moves relative to the plane to take m unique pictures. This cluster of points cannot be used to calibrate the pose of the camera if the old algorithm is used.

II. General approach

Central to the EPnP approach is the homogenous barycentric coordinate system in three dimensions which is defined by 4 non-coplanar points dubbed "control points" by Lepetit et. Al. The basic principle is that if all 3D points can be calculated as a weighted sum of 4 non-coplanar points within world coordinates, only these 4 control points points need to be solved in the camera coordinate system to efficiently recover all n 3D points within the camera coordinate system. At this point, any algorithm to determine the rigid transform between n points in world coordinates mapped to n points in camera coordinates can be used. The result is the camera pose. In principle there are only a few key equations, and how they are solved is completely up to the person implementing the algorithm. The authors suggest some methods such as linearization and relinearization, but in this paper Gauss-Newton gradient solvers are used in order to get a general solution form regardless of varying numbers of right hand singular vectors generated by the constraint equations.

III. Algorithm Summary

The algorithm is broken down into 8 detailed steps.

1. Define control points in world coordinates for the homogeneous barycentric system
2. Calculate the $\alpha_{j\in[1,2,3,4]}$ weights based on the control points for each n data points
3. For each of the n data points, generate 2 constraints on the control points described in the camera coordinates. Concatenate all equations and form $\mathbf{M}$ such that $\mathbf{M}x = 0$ where x is a 12 vector containing the 4 control points and their components within the camera system.
4. Solve for the right singular vectors of $\mathbf{M}$ to describe the general solution space of x.
5. Use Gauss-Newton optimization method to generate the best linear combination x_1, x_2, x_3, x_4 which are the right singular vectors associated with the 4 smallest singular values of $\mathbf{M}$. Let x be this best linear combination from a least squares optimization.
6. Recover the control points in camera space from x.
7. Use control points in camera space and the previously calculated weights in step 2 to find all n points in camera space.
8. Solve the rigid body rotation problem $\mathbf{R}, \mathbf{t}$ which defines the pose of the camera.

These steps are described in detail in the following sections.

IV. Implementation

In general, since the model of the system is a linear model we attempt to describe as much of the model as possible as matrix operations.

First, the control points which are arbitrary are selected. Note they must be non-coplanar. We store these inside the 4×3 matrix C_w describing the control points in the world frame.

1. Define the control points

We simply pick the orthogonal basis vectors of the rectangular world system and add the origin. The choice is arbitrary, but this choice is well conditioned which results in numerical stability.

$$\mathbf{C}^w = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \\ \mathbf{c}_4 \end{bmatrix}$$

2. Calculate the weights for each point

Let $\mathbf{X}^w$ be the points in the world coordinate system where the ith in $\mathbf{X}^w$ is a 1×3 vector describing the x, y, z coordinates.

To solve the weights of the barycentric system we can augment the matrices $\mathbf{C^w}^T$ and $\mathbf{X}^T$ by adding a row of ones.

Let:
The augmented $\mathbf{C^w}^T$ be $\mathbf{C}_\star^{wT}$
The augmented $\mathbf{X}^{wT}$ be $\mathbf{X}_\star^{wT}$

$$\mathbf{C}_\star^{wT}\boldsymbol{\alpha} = \mathbf{X}_\star^{wT}$$

Where $\boldsymbol{\alpha}$ is a $4 \times n$ matrix where each column contains a vector of the control points weights for the ith point.

$$\boldsymbol{\alpha} = \begin{bmatrix} \alpha_{1_i} & \cdots & \alpha_{1_n} \\ \alpha_{2_i} & \cdots & \alpha_{2_n} \\ \alpha_{3_i} & \cdots & \alpha_{3_n} \\ \alpha_{4_i} & \cdots & \alpha_{4_n} \end{bmatrix}$$

Since $\mathbf{C}_\star^{wT}$ is square we can simply invert it.

$$\boldsymbol{\alpha} = inv\big(\mathbf{C}_\star^{wT}\big)\mathbf{X}_\star^T$$

At this point, $\boldsymbol{\alpha}$ has been calculated.

3. Generate the constraint matrix

Matrix $\mathbf{M}$ is derived from concatenating 2 constraint equations from each of the n points. We assume that for the ith point

$$s_i \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = \mathbf{A} \sum_{j=1}^{4} \alpha_{j_i} \mathbf{c}_j^T \tag{1}$$

It is assumed that matrix $\mathbf{A}$ describing the camera is skew-less.

$$\mathbf{A} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

This gives rise to two constraints, formed from the u_i, v_i pixel equations. This means that

$$\sum_{j=1}^{4} \alpha_{j_i} f_u x_j + \alpha_{j_i}\ u_0 - u_i\ z_j = 0$$
$$\sum_{j=1}^{4} \alpha_{j_i} f_v y_j + \alpha_{j_i}\ v_0 - v_i\ z_j = 0$$

Where $c_j = [x_j, y_j, z_j]$

The resultant vector is a $2n \times 12$ matrix which is solved for its right singular values. This is the same as solving for x in the following equation.

$$\mathbf{M}x = 0$$

4. Solve for the right singular values of $\boldsymbol{M}$

In practice, finding the 4 eigenvectors of $\mathbf{M}^T\mathbf{M}$ that are associated with the smallest eigenvalues is more numerically stable than finding the null space of $\mathbf{M}$ since noise can make it such that the null space is empty for example.

The maximum number of null vectors is 4 which is the reason we find the 4 eigenvectors associated with the smallest eigenvalues. This basically means that the vectors are sent very close to the origin. Roughly satisfying our above equation.

At this point, we have four vectors x_1, x_2, x_3, x_4.

Assume that the corresponding eigenvalues have the following relationship: $\lambda_1 < \lambda_2 < \lambda_3 < \lambda_4$.

5. Gauss-Newton algorithm to linear combinations

Therefore, x_1 is the best single null vector approximation, to $\mathbf{M}x = 0$. It follows that x_2 is the second best approximation and so forth.

Thus, it makes sense to attempt to create a solution x which is a linear combination of $x_{i\in[1,2,3,4]}$.

We wish to solve the weights β_i that gives x such that the norms between corresponding points in world space and camera space are equal. This creates an equivalent scale between the two coordinate systems and also refines their directions.

In practice, it is helpful to use the 2-norm squared and compare the values since the expressions are simpler.

For any pair of points in camera space, the norms squared should be equal to that of the pair of points in world space.

$$\|p_a^c - p_b^c\|^2 = \|p_a^w - p_b^w\|^2$$

Since we have defined every point as a weighted sum of control points, there are only 4 points that we need to consider. Since there are only 6 ways to pair 2 points within a set of 4, we have only 6 equations to be summed for overall squared error.

We define each the 6 differences as such

Let

$$\begin{aligned}\delta_1^w &= \mathbf{c}_1 - \mathbf{c}_2\\ \delta_2^w &= \mathbf{c}_1 - \mathbf{c}_3\\ \delta_3^w &= \mathbf{c}_1 - \mathbf{c}_4\\ \delta_4^w &= \mathbf{c}_2 - \mathbf{c}_3\\ \delta_5^w &= \mathbf{c}_2 - \mathbf{c}_4\\ \delta_6^w &= \mathbf{c}_3 - \mathbf{c}_4\end{aligned}$$

$$\sigma^2 = \sum_i^6 \left(\left\|\beta_1 \delta_i^c{}_1 + \beta_2 \delta_i^c{}_2 + \beta_3 \delta_i^c{}_3 + \beta_4 \delta_i^c{}_4\right\|^2 - \|\delta_i^w\|^2\right)^2$$

We notice that squaring the norm squared yields quartic equations, which creates negative solutions. For example, if β was the vector of weights that minimized σ^2, then $-\beta$ would also be a solution to the minimum.

In practice, this means that the rotation and translation are negative, which results in the points within the camera system having a negative z coordinate. This is impossible due to camera coordinate conventions, therefore, the points are then multiplied by (-1) and the proper transform is found. This problem is avoided if the 2-norm squared residuals are used instead of the 2-norm squared-squared residuals. However, the derivative computation is elegantly expressed using the σ^2 expression from earlier which is helpful in using Gauss-Newton methods.

The problem is then state below:

Minimize σ^2 with respect to $(\beta_1, \beta_2, \beta_3, \beta_4)$

Since Gauss-Newton requires the derivatives of the residuals to be computed for each data point i

Let

$$r_i = \left\|\beta_1 \delta_i^c{}_1 + \beta_2 \delta_i^c{}_2 + \beta_3 \delta_i^c{}_3 + \beta_4 \delta_i^c{}_4\right\|^2 - \|\delta_i^w\|^2$$

We formulate the Jacobian, which holds the derivative of r_i with respect to $\beta_{i\in[1,2,3,4]}$ in the columns, evaluated at (β) in its ith row. β is the vector containing all β_i. This is basically a derivative matrix calculated for each data point. Note that superscript s denotes the iteration count. This means that we must initialize β^0.

Let

$$\beta = \begin{bmatrix}\beta_1\\ \beta_2\\ \beta_3\\ \beta_4\end{bmatrix}$$

$$r = \begin{bmatrix}r_1\\ \vdots\\ r_6\end{bmatrix}$$

$$\mathbf{\Delta}_{ij} = \delta_i \cdot \delta_j$$

$$\mathbf{\Delta} = [\Delta_1, \Delta_2, \Delta_3, \Delta_4]$$

$$\mathbf{J}_{ij} = \frac{\partial r_i(\beta^s)}{\partial \beta_j} = 2\beta^T \Delta_j$$

$$\beta^{s+1} = \beta^s - inv\ \mathbf{J}^T\mathbf{J}\ \mathbf{J}^T r(\beta^s)$$

This results in a general method to solve for β regardless of the number of approximate singular vectors x. For example, if we are only considering using vector x_1 to solve the system. Then β_1 is initialized and $\beta_{i\in[2,3,4]} = 0$.

Calculating the linear combination to find x is then the linear combination with the β values

$$x = \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3 + \beta_4 x_4$$

Then the transformed control points in camera space can be calculated.

6. Recover the camera control points

Let

$$x = \begin{bmatrix}x_1\\ \vdots\\ x_{12}\end{bmatrix}$$

Then the transformed control points can be recovered in camera space.

$$\begin{aligned}\mathbf{c}_1^c &= [x_1, x_2, x_3]\\ \mathbf{c}_2^c &= [x_4, x_5, x_6]\\ \mathbf{c}_3^c &= [x_7, x_8, x_9]\\ \mathbf{c}_4^c &= [x_{10}, x_{11}, x_{12}]\end{aligned}$$

$$\mathbf{C}^c = \begin{bmatrix}\mathbf{c}_1^c\\ \mathbf{c}_2^c\\ \mathbf{c}_3^c\\ \mathbf{c}_4^c\end{bmatrix}$$

7. *Calculate the coordinates in camera space*

At this point since the weights in $\boldsymbol{\alpha}$ can be used to calculate all the points $\mathbf{X}^c$

$$\mathbf{X}^c = \boldsymbol{\alpha}\mathbf{C}^c$$

Although scales should be very close at this point, we compute the scale factor to equate the norms again.

Let

$$d_i^c = \|\mathbf{X}_i^c\|$$

$$d_i^w = \|\mathbf{X}_i^w\|$$

Then we compute λ such that the squared error in norms r^2

$$r^2 = \sum_{i=1}^{n} \lambda d_i^c - d_i^w \ ^2$$

Taking the derivative of r^2 with respect to lambda

$$\frac{dr^2}{d\lambda} = \sum_{i=1}^{n} 2 \ \lambda d_i^c - d_i^w \ d_i^c$$

Setting the derivative to zero allows us to express λ in closed form by using dot products, or equivalently, inner products.

$$\lambda = \frac{d^w \cdot d^c}{d^c \cdot d^c} = \frac{d^{w^T} d^c}{d^{c^T} d^c}$$

At this point, the scales are the closest possible using a single scaling factor with respect to a squared residual error.

At this point, we check if the z values contained in $\mathbf{X}^c$ are negative. If they are, then they are all negative, so we multiply $\mathbf{X}^c$ by (-1) to get the correct points.

8. *Solve the rigid body transformation problem*

Now we have two matrices with the 3D points in camera space and world space. The last step is to solve the rigid rotation problem. Start by calculating the centroids.

$$x^c = mean \ \mathbf{X}^c$$
$$x^w = mean(\mathbf{X}^w)$$

The by abuse of notation matrix $\mathbf{X}$ subtracted by the centroid x denotes a point by point subtraction.

$$\delta\mathbf{X}^c = \mathbf{X}^c - x^c$$

$$\delta\mathbf{X}^w = \mathbf{X}^w - x^w$$

Let

$$\mathbf{S} = \delta\mathbf{X}^w diag \ 1, \ldots, 1_n \ \delta\mathbf{X}^{c^T}$$

Compute the Singular Value Decomposition (SVD) of $\mathbf{S}$

$$\mathbf{S} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^T$$

Finally, the rotation and translation of the camera with respect to the reference coordinate frame is calculated.

$$\mathbf{R} = \mathbf{V}\mathbf{U}^T$$

$$\mathbf{t} = x^c - \mathbf{R}x^w$$

In practice, the process is completed 4 times to solve 4 different transforms $\mathbf{R}, \mathbf{t}$ using different numbers of null vectors. For example, we compute the linear weights β for 1 vector, 2 vectors, 3 vectors, and finally 4 vectors. We always keep the smallest null vectors first, and add the next smallest to compute the transformed control points. Finally, we compare the reprojection error of all 4 sets in order to pick out the best transform.

V. Results

Using an example where a matrix $\mathbf{A}$ calibrated with small amounts of skew has randomly generated transform $\mathbf{R}$ and $\mathbf{t}$. This allows us to project random points onto the image plane that form a volume.

1. *Gauss-Newton Iteration Performance*

Notably the numerical method to generate linear combinations uses approximately 10 iterations to converge on a solution. The error at first iteration tends to grow, and then quickly reduces. The seed value for the weights in β must be non-zero as the derivative is linear with respect to components in β.

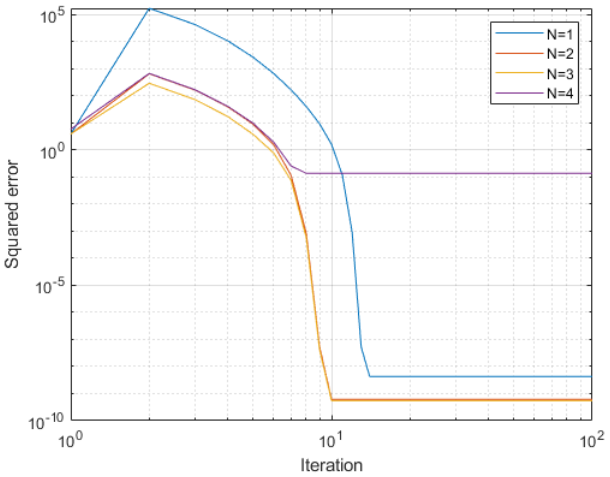

Figure 1 - Convergence of Gauss-Newton when solving for β

The numerically generated rotation matrix and translation vector are given below.

$$\mathbf{R} = \begin{bmatrix} 0.9819 & -0.0125 & 0.1888 \\ 0.0025 & 0.9986 & 0.0531 \\ -0.1892 & -0.0517 & 0.9806 \end{bmatrix}$$

$$\mathbf{t} = [-148.4549 \quad -91.4726 \quad 320.5354]^T$$

Using the method described, the Frobenius Norm of the difference between the $\mathbf{R}$ matrix results in values around 10^{-10}. The norm of the difference in $\mathbf{t}$ is on the order of 10^{-8}. This shows very good agreement which should be expected given that the pixel locations were generated.

From previous papers, the pixel locations of reprojection had a standard error of approximately 2 pixels in each axis. We will add random Gaussian noise of to each of the pixel coordinates and calculate the transforms again.

Suppose that the matrix $\mathbf{D}$ is an $n \times 2$ matrix with rows containing pixel coordinate we add the Gaussian noise matrix $\mathbf{G}$ component-wise to $\mathbf{D}$ with $\mu = 0$, $\sigma = 2$.

The result is $\|\Delta\mathbf{R}\| \approx 10^{-3}$ and $\|\Delta\mathbf{t}\| \approx 1$. As the camera matrix $\mathbf{A}$ was generated using previous calibration models, the norm of $\Delta\mathbf{t}$ is in millimeters. This means that the position of the camera was found to 1mm accuracy.

The reprojection error can then also be calculate and plotted on a histogram.

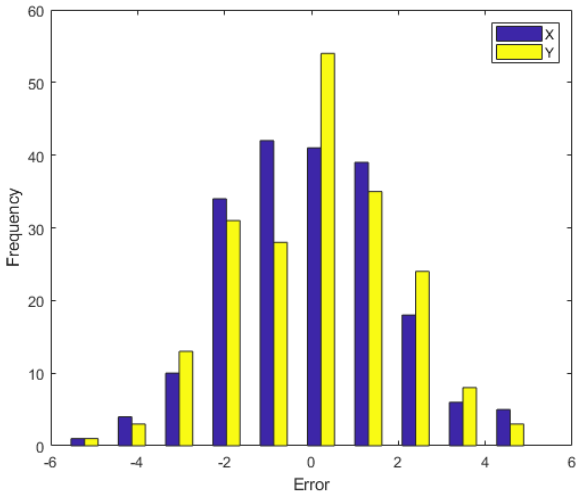

Figure 2-Histogram of reprojection errors after transform solve

Clearly, we can see the error is roughly normal since Gaussian noise was added to the pixels. By the binning, we can see that the standard deviation is approximately 2 which shows that the errors were not increased noticeably due to the transform.