The Eight-Point Algorithm

Carlo Tomasi

This note describes a method for computing estimates of the rigid transformation G and estimates of the coordinates of a set of n points $\mathcal{P}_1, \ldots, \mathcal{P}_n$ in the two camera reference frames from the n pairs $(\mathbf{x}_1, \mathbf{y}'_1), \ldots (\mathbf{x}_n, \mathbf{y}'_n)$ of noisy measurements of their corresponding images. The transformation G is called camera *motion*, and the point coordinates $\mathbf{X}_i$, $\mathbf{X}'_i$ of the world points in the two reference systems are collectively called the scene *structure*. This classic method is called the *eight-point* algorithm and is was invented by Hugh Christopher Longuet-Higgins in 1981 [3].

The points $\mathbf{x}_i$ and $\mathbf{y}'_i$ are in the canonical reference system of each camera, so their third coordinate is equal to 1. As such, they can be viewed as either the Euclidean coordinates of 3D points, or as homogeneous coordinates of 2D image points. However, we use Euclidean coordinates for other 3D points to describe the eight-point algorithm—a method devised before homogeneous coordinates became pervasive in computer vision. Nowadays, this algorithm is typically embedded in code that uses homogeneous coordinates everywhere else, for notational convenience. We also resort to homogeneous coordinates in Appendix B where we discuss triangulation, that is, the calculation of $\mathbf{X}$, $\mathbf{X}'$ from the image points and G.

Since cameras fundamentally measure angles, both structure and motion can be estimated only up to a common nonzero multiplicative scale factor. The resulting degree of freedom is eliminated by assuming that

$$\|\mathbf{s}'\| = \|\mathbf{t}\| = 1 \; . \tag{1}$$

An initial version of the method described below appeared in 1981 [3] and is often called the *eight-point algorithm*, because it requires a *minimum* of $n = 8$ pairs of corresponding image points.

The epipolar constraint described in a previous note can be rewritten in the following form:

$$\mathbf{a}^T \boldsymbol{\eta} = 0 \quad \text{where} \quad \mathbf{a} = \mathbf{x} \otimes \mathbf{y}' = \begin{bmatrix} x_1 \, \mathbf{y}' \\ x_2 \, \mathbf{y}' \\ x_3 \, \mathbf{y}' \end{bmatrix} \tag{2}$$

is the Kronecker product[1] of $\mathbf{x} = \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T$ and $\mathbf{y}'$, and

$$\boldsymbol{\eta} = E(:) = \begin{bmatrix} e_{11} & e_{21} & e_{31} & e_{12} & e_{22} & e_{32} & e_{13} & e_{23} & e_{33} \end{bmatrix}^T$$

is the stack of entries in E read by columns. Equation (2) can be replicated n times, one per image point pair, to yield a linear system

$$A\boldsymbol{\eta} = \mathbf{0} \quad \text{where} \quad A = \begin{bmatrix} \mathbf{a}_1 & \cdots & \mathbf{a}_n \end{bmatrix}^T$$

[1] More generally, the *Kronecker product* of two matrices B and C where B is $m \times n$ is defined as follows:

$$B \otimes C = \begin{bmatrix} b_{11}C & \ldots & b_{1n}C \\ \vdots & & \vdots \\ b_{m1}C & \ldots & b_{mn}C \end{bmatrix} .$$

is an $n \times 9$ matrix. The homogeneous nature of this system reflects the fact that translation $\mathbf{t}$ and therefore the essential matrix E are defined up to a nonzero multiplicative scale factor. As we know from a previous note, to prevent the trivial solution $\boldsymbol{\eta} = \mathbf{0}$ and at the same time solve the system above in the least-squares sense to account for measurement inaccuracies, one computes

$$\boldsymbol{\eta} = \arg \min_{\|\boldsymbol{\eta}\|=1} \|A\boldsymbol{\eta}\| = \mathbf{v}_9 \quad \text{where} \quad A = U_A \Sigma_A V_A^T$$

is the Singular Value Decomposition (SVD) of A and $\mathbf{v}_9$ is the last column of V_A. The resulting vector $\boldsymbol{\eta}$ is then reshaped into an estimate E of the essential matrix.[2]

As we know, the essential matrix has rank 2 and equal singular values. However, the solution just found may not satisfy these constraints exactly, because it comes from noisy data. It should come to no surprise that the best approximation of E that does satisfy these constraints can be computed by first computing the SVD of E and then forcing the singular values to be equal to $(1, 1, 0)$:

$$E = U\Sigma V^T \quad \text{and then} \quad E \leftarrow U \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} V^T \ .$$

We also know that the null space of E is the one-dimensional space spanned by $\mathbf{t}$, which also spans the null space of the skew matrix $[\mathbf{t}]_\times$. So an estimate of $\mathbf{t}$ is

$$\mathbf{t}_{1,2} = \pm \mathbf{v}_3$$

where $\mathbf{v}_3$ is the last column of V. The ambiguity in the sign of $\mathbf{t}$ will be resolved later.

Given $\mathbf{t}$, one can construct the skew matrix $[\mathbf{t}]_\times$, and then estimate R by solving the following *Procrustes problem* [1]:

$$E \approx R\,[\mathbf{t}]_\times \ . \tag{3}$$

where the approximation is in the Frobenius norm. That is,

$$R = \arg \min_R \|E - R\,[\mathbf{t}]_\times\|_F = \sqrt{\sum_{i,j} d_{ij}^2} \quad \text{where} \quad D = [d_{ij}] = E - R\,[\mathbf{t}]_\times \ .$$

Appendix A shows[3] that if E and $[\mathbf{t}]_\times$ were full rank, the solution to problem (3) would be

$$R = Q \det(Q) \quad \text{where} \quad Q = U_C V_C^T \quad \text{and} \quad C = U_C \Sigma_C V_C^T$$

is the SVD of the 3×3 matrix

$$C = E\,[\mathbf{t}]_\times^T \ ,$$

and where the multiplication by $\det(Q)$ ensures that the resulting orthogonal matrix is a rotation. This multiplication is allowed, because if E is an essential matrix then so is $-E$.

However, the two matrices E and $[\mathbf{t}]_\times$ have rank 2, and their third singular vectors (both left and right)—equal to the unit translation vectors $\mathbf{s}'$ and $\mathbf{t}$ from the discussion above—are defined up to a sign. Because of this, the Procrustes problem has two solutions:

$$R_{1,2} = W_{1,2} \det(W_{1,2}) \quad \text{where} \quad W_{1,2} = \boldsymbol{\alpha}_1 \boldsymbol{\beta}_1^T + \boldsymbol{\alpha}_2 \boldsymbol{\beta}_2^T \pm \mathbf{s}'\mathbf{t}^T$$

[2] As we found out in a previous note, the two nonzero singular values of the essential matrix are equal to each other, and the matrix $\tilde{E}$ that satisfies this constraint and is closest to E in the Frobenius norm is $\tilde{E} = U \mathrm{diag}([1, 1, 0]) V^T$ where $E = U\Sigma V^T$ is the SVD of E. However, the singular values of $\tilde{E}$ are not needed in the computation that follows, so this correction is unnecessary.

[3] Let $A = E$ and $B = [\mathbf{t}]_\times$ in that proof, so that $p = n = 3$.

where

$$U_B = \begin{bmatrix} \boldsymbol{\alpha}_1 & \boldsymbol{\alpha}_2 & -\mathbf{s}' \end{bmatrix} \quad , \quad V_B = \begin{bmatrix} \boldsymbol{\beta}_1 & \boldsymbol{\beta}_2 & \mathbf{t} \end{bmatrix} .$$

The minus sign for $\mathbf{t}$ comes from equation

$$\mathbf{s}' = -R\mathbf{t}$$

found in an earlier note for the transformation from the second camera frame back to the first. Combining the twofold ambiguity in $\mathbf{t}$ with that in R yields four solutions, each corresponding to a different essential matrix:

$$(\mathbf{t}, R_1) , \ (-\mathbf{t}, R_2) , \ (\mathbf{t}, R_2) , \ (-\mathbf{t}, R_1) .$$

Appendix B shows that only one of these solutions places all reconstructed world points in front of both cameras. The correct solution can then be identified by computing structure for all four cases by triangulation, and choosing the one solution that enforces structure to be in front of both cameras. Allowing for reconstruction errors, a safer approach is to chose the solution with a *majority* of points in front of the camera. Appendix B shows the details of this calculation and Figures 1 and 2 list the complete MATLAB code for 3D reconstruction with two cameras.

References

[1] G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins University Press, 3rd edition, 1996.

[2] R. I. Hartley. Chirality. *International Journal of Computer Vision*, 26(1):41–61, 1998.

[3] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, September 1981.

```
function [G, X, Y]  = longuetHiggins(x, y)

% Number of point correspondences
n = size(x, 2);

if size(y, 2) ~= n
    error('The number of points in the two images must be the same');
end

% Set up matrix A such that A*E(:) = 0, where E is the essential matrix.
% This system encodes the epipolar constraint
A = zeros(n, 9);
for i = 1:n
    A(i,:) = kron(x(:,i),y(:,i))';
end

if rank(A) < 8
    error('Measurement matrix rank deficient')
end;

% The singular vector corresponding to the smallest singular value of A
% is the arg min_{norm(e) = 1} A * e, and is the LSE estimate of E(:)
[~, ~, V] = svd(A);
E = reshape(V(:,9), 3, 3);

% The two possible translation vectors are t and -t, where t is a unit
% vector in the null space of E
[~, ~, VE] = svd(E);
t = VE(:, 3);

% Two rotation matrix choices are found by solving the Procrustes problem
% for the rows of E and skew(t), and allowing for the ambiguity resulting
% from the sign of the null-space vectors (both E and skew(t) are rank 2).
% These two choices are independent of the sign of t, because both E and -E
% are essential matrices
tx = skew(t);
[UR, ~, VR] = svd(E * tx);
R1 = UR * VR';
R1 = R1 * det(R1);
UR(:, 3) = -UR(:, 3);
R2 = UR * VR';
R2 = R2 * det(R2);

% Combine the two sign options for t with the two choices for R
t = [t, t, -t, -t];
R = cat(3, R1, R2, R1, R2);

% Pick the combination of t and R that yields the greatest number of
% positive depth (Z) values in the structure results in the frames of
% reference of both cameras. Ideally, all depth values should be positive
npd = zeros(4, 1);
X = zeros(4, n, 4);
Y = zeros(4, n, 4);
for k = 1:4
    G = [R(:, :, k), -R(:, :, k) * t(:, k); 0 0 0 1];
    [X(:, :, k), Y(:, :, k)] = triangulate(x, y, G);
    npd(k) = sum(X(3, :, k) > 0 & Y(3, :, k) > 0);
end
[~, best] = max(npd);
G = [R(:, :, best), -R(:, :, best) * t(:, best); 0 0 0 1];
X = X(:, :, best);
Y = Y(:, :, best);
```

Figure 1: Main function of the MATLAB code for 3D reconstruction with two cameras.

```
function [X, Y] = triangulate(x, y, G)

n = size(x, 2);

Pi = [eye(3), zeros(3, 1)];
Phi = Pi * G;

X = zeros(4, n);

for i=1:n
    A = [x(1, i) * Pi(3, :) - Pi(1, :);
         x(2, i) * Pi(3, :) - Pi(2, :);
         y(1, i) * Phi(3, :) - Phi(1, :);
         y(2, i) * Phi(3, :) - Phi(2, :)];


     [~, ~, v] = svd(A);
    X(:, i) = v(:,4);
end

Y = G * X;

% Normalize fourth coordinate
X = homogeneous(euclidean(X));
Y = homogeneous(euclidean(Y));



function T = skew(t)

T = [0 -t(3) t(2); t(3) 0 -t(1); -t(2) t(1) 0];



function h = homogeneous(e)

h = [e; ones(1, size(e, 2))];



function e = euclidean(h)

w = h(end, :);
d = size(h, 1) - 1;
e = h(1:d, :);

nz = w ~= 0;
e(:,  nz) = h(1:d, nz) ./ (ones(d, 1) * w(nz));
```

Figure 2: Auxiliary MATLAB functions for 3D reconstruction with two cameras.

Appendix A: Solving the Procrustes Problem

This proof is adapted from a classical text on matrix computations [1], and applies to any two matrices A and B of size $p \times n$ that encode two sets of n data points in p dimensions.

Theorem .1. *Let corresponding columns of the two matrices $A, B \in \mathbb{R}^{p \times n}$ encode n pairs of corresponding points in $\mathbb{R}^p$ with $p \leq n$. The following algorithm finds an orthogonal matrix $Q \in \mathbb{R}^{p \times p}$ that minimizes the Frobenius norm of $\|A - QB\|_F$.*

$$\begin{aligned} C &= AB^T \\ [U, \Sigma, V] &= \mathit{svd}(C) \\ Q &= UV^T \end{aligned}$$

Proof. The trace $\text{tr}(C)$ of a matrix C is the sum of its diagonal entries, and from the definition of Frobenius norm of a matrix C,

$$\|C\|_F^2 = \sum_{i,j} c_{ij}^2 = \text{tr}(CC^T) \ .$$

Then,

$$\|A - QB\|_F^2 = \text{tr}[(A - QB)(A - QB)^T] = \text{tr}(AA^T) + \text{tr}(BB^T) - 2\,\text{tr}(AB^TQ^T)$$

where we used the fact that Q is orthogonal and that the trace of the sum of several matrices is the sum of their traces.

The first two terms in the right-hand side of the equation above do not depend on Q, so minimizing $\|A - QB\|_F^2$ is the same as maximizing $\text{tr}(AB^TQ^T)$. If

$$AB^T = U\Sigma V^T$$

is the SVD of AB^T, then we want to find the maximum of

$$\text{tr}(U\Sigma V^TQ^T) = \text{tr}(U\Sigma V^TQ^TUU^T) = \text{tr}(U\Sigma ZU^T) \quad \text{where} \quad Z = V^TQ^TU$$

is an orthogonal matrix. It is easy to verify that the trace is a commutative operator, so that

$$\text{tr}(U\Sigma ZU^T) = \text{tr}(\Sigma ZU^TU) = \text{tr}(\Sigma Z) = \sum_{i=1}^{p} \sigma_i z_{ii} \ .$$

Since Z is the product of orthogonal matrices, it is itself orthogonal. The rows of orthogonal matrices have unit norm, so no entry in an orthogonal matrix can have magnitude greater than 1. So the sum in the last term above is maximized when

$$z_{11} = \ldots = z_{pp} = 1 \ ,$$

which occurs when Z is the $p \times p$ identity matrix I_p. So one solution is achieved when

$$Z = I_p \quad \text{that is,} \quad V^TQ^TU = I_p \quad \text{or} \quad Q^T = VU^T \ .$$

The last equation was obtained by multiplying the previous one by V on the left and by U^T on the right. Thus,

$$Q = UV^T$$

as promised.

If the matrix C is full rank (so that both A and B are full rank), then this is the only solution. Otherwise, this is just *a* solution, because some of the σ_i are zero, so the corresponding values z_{ii} do not matter. The case in which $\text{rank}(C) = p - 1$ is both simple and relevant to the eight-point algorithm. In that case, the null space of C has dimension 1, so the only ambiguity in U and V that pertains to the last singular value is the sign of its last singular vectors $\mathbf{u}_p$ and $\mathbf{v}_p$. Changing the sign of both vectors leaves the product UV^T unaltered, because

$$UV^T = \begin{bmatrix} \mathbf{u}_1 & \dots & \mathbf{u}_p \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 & \dots & \mathbf{v}_p \end{bmatrix}^T = \sum_{i=1}^{p} \mathbf{u}_i \mathbf{v}_i^T \ .$$

So if UV^T is one solution, then the other one is

$$\begin{bmatrix} \mathbf{u}_1 & \dots & -\mathbf{u}_p \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 & \dots & \mathbf{v}_p \end{bmatrix}^T$$

which is the same as

$$\begin{bmatrix} \mathbf{u}_1 & \dots & \mathbf{u}_p \end{bmatrix} \begin{bmatrix} \mathbf{v}_1 & \dots & -\mathbf{v}_p \end{bmatrix}^T \ .$$

Δ

Appendix B: Resolving the Sign Ambiguity

Because of the sign ambiguity in $\mathbf{s}'$ and $\mathbf{t}$, the Procrustes problem has two solutions:

$$R_{1,2} = W_{1,2} \det(W_{1,2}) \quad \text{where} \quad W_{1,2} = \boldsymbol{\alpha}_1 \boldsymbol{\beta}_1^T + \boldsymbol{\alpha}_2 \boldsymbol{\beta}_2^T \pm \mathbf{s}' \mathbf{t}^T$$

where

$$U_B = \begin{bmatrix} \boldsymbol{\alpha}_1 & \boldsymbol{\alpha}_2 & -\mathbf{s}' \end{bmatrix} \quad , \quad V_B = \begin{bmatrix} \boldsymbol{\beta}_1 & \boldsymbol{\beta}_2 & \mathbf{t} \end{bmatrix} \ .$$

Equivalently, if U_B and V_B are first replaced by their rotation versions $U_B \det(U_B)$ and $V_B \det(V_B)$ (so that their determinants are equal to 1), we have

$$R_1 = \boldsymbol{\alpha}_1 \boldsymbol{\beta}_1^T + \boldsymbol{\alpha}_2 \boldsymbol{\beta}_2^T - \mathbf{s}' \mathbf{t}^T \quad \text{and} \quad R_2 = -\boldsymbol{\alpha}_1 \boldsymbol{\beta}_1^T - \boldsymbol{\alpha}_2 \boldsymbol{\beta}_2^T - \mathbf{s}' \mathbf{t}^T \ . \tag{4}$$

These equations reveal that R_1 and R_2 relate to each other through a 180-degree rotation of either camera reference frame around the baseline. To see this, write the transformation between these two frames of reference as a transformation from frame 1 to the world frame composed with one from world frame to frame 2:

$$R_2 R_1^T = (-\boldsymbol{\alpha}_1 \boldsymbol{\beta}_1^T - \boldsymbol{\alpha}_2 \boldsymbol{\beta}_2^T - \mathbf{s}' \mathbf{t}^T)(\boldsymbol{\beta}_1 \boldsymbol{\alpha}_1^T + \boldsymbol{\beta}_2 \boldsymbol{\alpha}_2^T - \mathbf{s}' \mathbf{t}^T) = -\boldsymbol{\alpha}_1 \boldsymbol{\alpha}_1^T - \boldsymbol{\alpha}_2 \boldsymbol{\alpha}_2^T + \mathbf{s}' (\mathbf{s}')^T \ ,$$

and this rotation maps $\boldsymbol{\alpha}_1$ to $-\boldsymbol{\alpha}_1$, $\boldsymbol{\alpha}_2$ to $-\boldsymbol{\alpha}_2$, and $\mathbf{s}$ (or $\mathbf{t}$) to itself, as promised.

The transformation between the first and the last of the four solutions above places camera 2 on the opposite side of camera 1 along the baseline.[4] This transformation can equivalently described as leaving the cameras where they are, pointing in the same way, but replacing all structure vectors $\mathbf{X}_i$ and $\mathbf{X}'_i$ by their opposites $-\mathbf{X}_i$ and $-\mathbf{X}'_i$. This transformation is said to change the *chirality* of structure in the literature [2],

[4] Of course, the same transformation can be described as a displacement of camera 1 relative to camera 2.

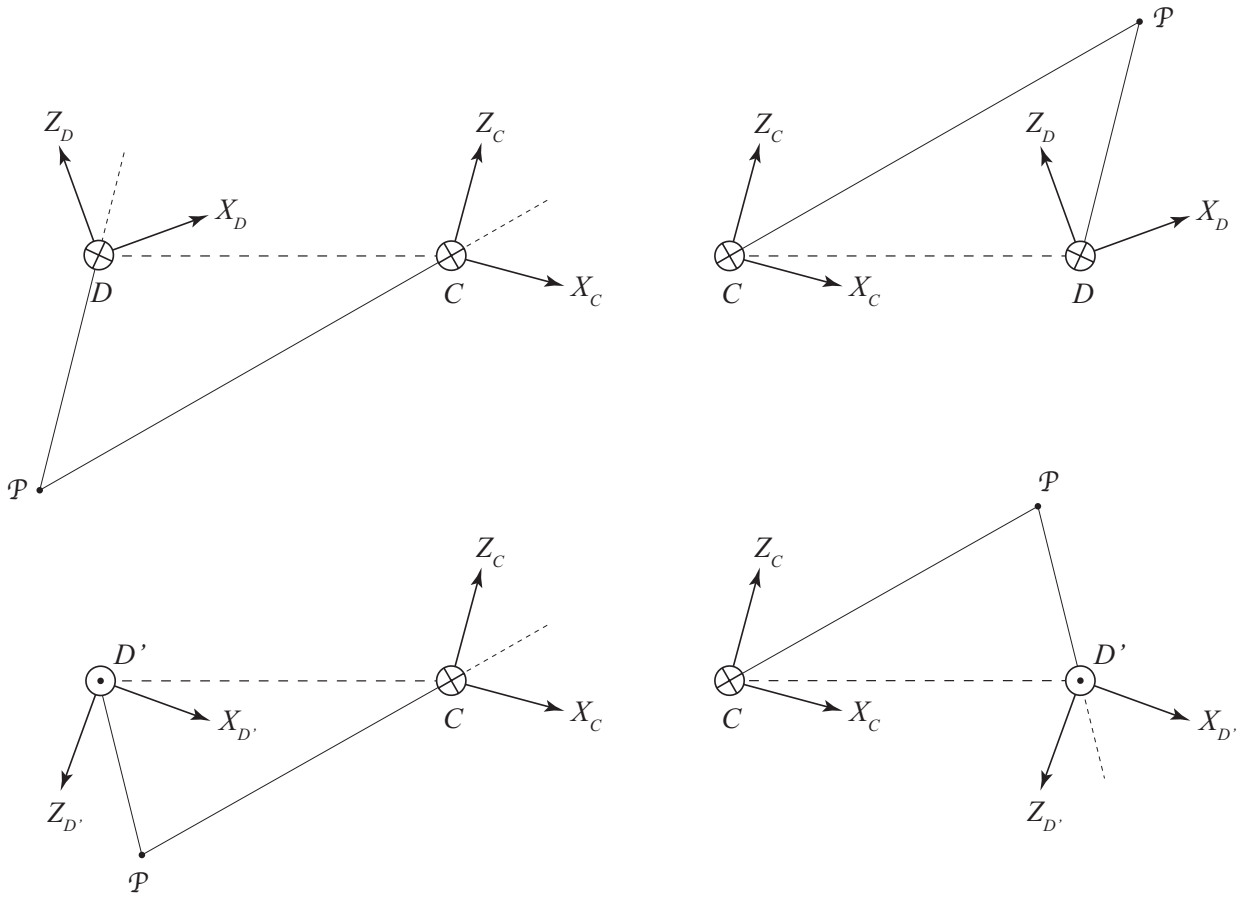

Figure 3: The fourfold ambiguity of reconstruction corresponds to the two ways to pick the sign of $\mathbf{t}$ (left or right diagrams) and the two ways to choose the rotation matrix R (top or bottom diagrams). A circle with a cross (a dot) denotes a Y axis pointing into (out of) the page. Only the arrangement in the top right has the scene structure (represented by the single point $\mathcal{P}$ and its two projection rays) in front of both cameras. [To keep the figure uncluttered we use X, Y, Z instead of X_1, X_2, X_3.]

because superposing the original structure with the transformed one requires a change of handedness of the reference system (that is, a mirror flip). This transformation has the effect of placing the scene *behind* the two cameras if it is in front of them to begin with. With some abuse of terminology, a change of chirality in computer vision means merely changing whether structure is in front or behind a camera. In this sense, structure has two values of chirality, one per camera. A 180-degree rotation around the baseline—obtained by replacing R_1 with R_2 or *vice versa*—changes chirality once more, but only for the camera being rotated.

The four motion solutions given earlier correspond to using top right, top left, bottom right, and bottom left camera pairs in Figure 3, in this order. The two top pairs in the figure are said to form a *twisted pair*, and so are the two bottom pairs.

Only one of these solutions puts the scene points in front of both cameras. So the correct solution can be identified by computing structure for all four cases by triangulation, and choosing the one solution that enforces most of the structure solution (allowing for a few reconstruction errors) to be in front of both

cameras:

$$\mathbf{k}^T\mathbf{X}_i^e > 0 \quad \text{and} \quad \mathbf{k}^T(\mathbf{X}')_i^e > 0 \quad \text{for} \quad i = 1, \ldots, n \quad \text{where} \quad \mathbf{k}^T = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}$$

and where $\mathbf{X}_i^e$ and $(\mathbf{X}')_i^e$ are the Euclidean coordinates corresponding to homogeneous coordinates $\mathbf{X}_i$ and $\mathbf{X}'_i$.

Since only the sign of the structure components is needed, a simple triangulation method will do. The projection equation for each point $\mathbf{X}$ can be written as follows:

$$\alpha\mathbf{x} = \Pi\mathbf{X} \quad \text{and} \quad \beta\mathbf{y}' = \Phi'\mathbf{X} \quad \text{where} \quad \Phi' = \Pi G$$

and vectors $\mathbf{x}$ and $\mathbf{y}'$ can be normalized so their third components x_3 and y'_3 are equal to 1. We use the canonical projection matrix in these equations because $\mathbf{x}$ and $\mathbf{y}'$ are the canonical coordinates of the image points. If we have the image coordinates $\boldsymbol{\xi}$ and $\boldsymbol{\eta}'$, we need to compute their canonical versions as

$$\mathbf{x} = K^{-1}\boldsymbol{\xi} \quad \text{and} \quad \mathbf{y}' = (K')^{-1}\boldsymbol{\eta}'$$

where K and K' are the two camera calibration matrices. These equations can then be spelled out into their separate rows as follows:

$$\alpha x_1 = \boldsymbol{\pi}_1^T\mathbf{X} \quad , \quad \alpha x_2 = \boldsymbol{\pi}_2^T\mathbf{X} \quad , \quad \alpha = \boldsymbol{\pi}_3^T\mathbf{X} \quad , \quad \beta y'_1 = (\boldsymbol{\varphi}'_1)^T\mathbf{X} \quad , \quad \beta y'_2 = (\boldsymbol{\varphi}'_2)^T\mathbf{X} \quad , \quad \beta = (\boldsymbol{\varphi}'_3)^T\mathbf{X} \quad .$$

The expressions for α and β given by the third and sixth equation above can be replaced into the other equations to obtain the following 4×4 homogeneous linear system in $\mathbf{X}$:

$$H\mathbf{X} = \begin{bmatrix} x_1\boldsymbol{\pi}_3 - \boldsymbol{\pi}_1 \ , & x_2\boldsymbol{\pi}_3 - \boldsymbol{\pi}_2 \ , & y'_1\boldsymbol{\varphi}'_3 - \boldsymbol{\varphi}'_1 \ , & y'_2\boldsymbol{\varphi}_3 - \boldsymbol{\varphi}'_2 \end{bmatrix}^T \mathbf{X} = \mathbf{0} \ .$$

The solution $\mathbf{X}$ is the last right singular vector of H, and the coordinates $\mathbf{X}'$ of the same point in the other camera can be found through the rigid transformation

$$\mathbf{X}' \sim G\mathbf{X} \quad \text{where} \quad G \sim \left[\begin{array}{c|c} R & -R\,\mathbf{t} \\ \hline \mathbf{0}^T & 1 \end{array}\right] \ .$$