The Eight-Point Algorithm

Carlo Tomasi

November 20, 2017

This note describes a method for computing estimates of the rigid transformation ${}^aT_b = ({}^aR_b, {}^a\mathbf{t}_b)$ between two cameras a and b and estimates of the coordinates ${}^a\mathbf{P}_1, \ldots, {}^a\mathbf{P}_n$ of a set of n points in the reference system of one of the two cameras from the n pairs $({}^a\mathbf{p}_{a,1}, {}^b\mathbf{p}_{b,1}), \ldots ({}^a\mathbf{p}_{a,n}, {}^b\mathbf{p}_{b,n})$ of noisy measurements of their corresponding images. The transformation ${}^aT_b$ is called camera *motion*, and the point coordinates ${}^a\mathbf{P}_1, \ldots, {}^a\mathbf{P}_n$ are collectively called the scene *structure*. The image points ${}^a\mathbf{p}_{a,i}$ and ${}^b\mathbf{p}_{b,i}$ are regarded as 3D points with their third coordinate equal to 1, the standard focal distance. The classic method described below is called the *eight-point* algorithm and is was invented by Hugh Christopher Longuet-Higgins in 1981 [3]. Its main goal is to find ${}^aT_b$. Triangulation, that is, the calculation of structure from the image points and ${}^aT_b$, is outlined in Appendix B.

To simplify notation in the manipulations that follow, we again let

$$\mathbf{a} = {}^a\mathbf{p}_a \quad , \quad \mathbf{b} = {}^b\mathbf{p}_b \quad , \quad \mathbf{A} = {}^a\mathbf{P} \quad , \quad \mathbf{B} = {}^b\mathbf{P} \quad , \quad R = {}^aR_b \quad , \quad \mathbf{t} = {}^a\mathbf{t}_b \, ,$$

adding a subscript to $\mathbf{a}$, $\mathbf{b}$, or $\mathbf{A}$ when necessary to distinguish different points.

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

$$\|\mathbf{t}\| = 1 \, . \tag{1}$$

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:

$$\mathbf{c}^T \boldsymbol{\eta} = 0 \quad \text{where} \quad \mathbf{c} = \mathbf{a} \otimes \mathbf{b} = \begin{bmatrix} a_1 \, \mathbf{b} \\ a_2 \, \mathbf{b} \\ a_3 \, \mathbf{b} \end{bmatrix} \tag{2}$$

is the Kronecker product[1] of $\mathbf{a} = \begin{bmatrix} a_1 & a_2 & a_3 \end{bmatrix}^T$ and $\mathbf{b}$, and

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

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

$$F \otimes G = \begin{bmatrix} f_{11}G & \ldots & f_{1n}G \\ \vdots & & \vdots \\ f_{m1}G & \ldots & f_{mn}G \end{bmatrix} .$$

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

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

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

$$\boldsymbol{\eta} = \arg\min_{\|\boldsymbol{\eta}\|=1} \|C\boldsymbol{\eta}\| = \mathbf{v}_9 \quad \text{where} \quad C = U_C \Sigma_C V_C^T$$

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

As we know, 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 in the SVD $E = U\Sigma V^T$ of E. 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

$$R = Q\det(Q) \quad \text{where} \quad Q = U_F V_F^T \quad \text{and} \quad F = U_F \Sigma_F V_F^T$$

is the SVD of the 3×3 matrix

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

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. Since their third singular value is therefore zero, the third singular vectors (both left and right) of these two matrices are defined up to a sign. Recall that the third right singular vector is the direction of the translation $\mathbf{t}$ from camera a to camera b in the reference frame of a. Similarly, the third left singular vector is the direction of the translation $\mathbf{s} = -R^T\mathbf{t}$ from camera b to

[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 $p = n = 3$.

camera a in the reference frame of b. Because of this sign ambiguity in the solution, the Procrustes problem has two solutions:

$$R_{1,2} = Q_{1,2}\det(Q_{1,2}) \quad \text{where} \quad Q_{1,2} = \boldsymbol{\alpha}_1\boldsymbol{\beta}_1^T + \boldsymbol{\alpha}_2\boldsymbol{\beta}_2^T \pm \boldsymbol{\alpha}_3\boldsymbol{\beta}_3^T$$

where

$$U_F = \begin{bmatrix} \boldsymbol{\alpha}_1 & \boldsymbol{\alpha}_2 & \boldsymbol{\alpha}_3 \end{bmatrix} \quad , \quad V_F = \begin{bmatrix} \boldsymbol{\beta}_1 & \boldsymbol{\beta}_2 & \boldsymbol{\beta}_3 \end{bmatrix} .$$

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 C 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. Appendices B and C show 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 [T, 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

% Transform images from 2D to 3D in the standard reference frame
o = ones(1, n);
x = [x; o];
y = [y; o];

% Set up matrix A such that A*E(:) = 0, where E is the essential matrix.
% This system encodes the epipolar constraint y' * E * x = 0 for each of
% the points x and y
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. The vector t (or -t) is also the
% second epipole of the camera pair
[~, ~, 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);
[UF, ~, VF] = svd(E * tx);
R1 = UF * VF';
R1 = R1 * det(R1);
UF(:, 3) = -UF(:, 3);
R2 = UF * VF';
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 for the frames of
% reference of both cameras. Ideally, all depth values should be positive
npd = zeros(3, 1);
X = zeros(3, n, 4);
Y = zeros(3, n, 4);
for k = 1:4
    T.R = R(:, :, k);
    T.t = t(:, k);
    [X(:, :, k), Y(:, :, k)] = triangulate(x, y, T);
    npd(k) = sum(X(3, :, k) > 0 & Y(3, :, k) > 0);
end
[~, best] = max(npd);
T.R = R(:, :, best);
T.t = t(:, best);
X = X(:, :, best);
Y = Y(:, :, best);
```


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

```
function [A, B] = triangulate(a, b, T)

n = size(a, 2);

A = zeros(3, n);
iT = T.R(1, :);
jT = T.R(2, :);
kT = T.R(3, :);
kTt = kT * T.t;
iTjT = [iT; jT];
iTjTt = iTjT * T.t;

C = [eye(2), zeros(2, 1); zeros(2, 3)];
c = zeros(4, 1);

for m = 1:n
    C(1:2, 3) = -a(1:2, m);
    C(3:4, :) = b(1:2, m) * kT - iTjT;
    c(3:4, 1) = kTt * b(1:2, m) - iTjTt;
    A(:, m) = C \ c;
end

B = T.R * (A - T.t * ones(1, size(A, 2)));
```

```
function T = skew(t)

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

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 $\|A - QB\|_F$.*

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

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 if matrix G has the same size as matrix F^T then

$$\text{tr}(FG) = \text{tr}(GF) \ ,$$

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: Approximate Triangulation

Triangulation is the process of computing the coordinates $\mathbf{A}$ of each point in space from its projections in the two images, given that the transformation $(R, \mathbf{t})$ between the two cameras is known. This Appendix shows a simple triangulation method obtained by solving the two projection equations for $\mathbf{A}$. In this derivation, the two image projections are represented by vectors $\mathbf{a}_2$ and $\mathbf{b}_2$, which are the coordinates of the two projections of $\mathbf{A}$ in the canonical *image* reference system. If the focal distances of the two cameras are f_a and f_b, then $\mathbf{a}_2$ and $\mathbf{b}_2$ relate to the coordinates $\mathbf{a}$ and $\mathbf{b}$ (which are measured in the canonical *camera* reference system) by

$$\mathbf{a} = f_a \begin{bmatrix} \mathbf{a}_2 \\ 1 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = f_b \begin{bmatrix} \mathbf{b}_2 \\ 1 \end{bmatrix} \ .$$

There are four scalar projection equations (one for each point coordinate in the two images) in three unknowns (the coordinates of $\mathbf{A}$), so the resulting linear system in $\mathbf{A}$ is over-constrained. In this Appendix, this system is solved in the sense of least squares, by minimizing the norm of the discrepancy between the left-hand side and the right-hand side of this system. The least-squares solution is optimal when this discrepancy, called the *algebraic error*, is Gaussian and isotropic. However, this is typically not the case: What is likely Gaussian and sometimes isotropic is the *image reprojection error*, that is, the norm of the difference between the measured image point coordinates $\mathbf{a}_2$ and $\mathbf{b}_2$ and the coordinates obtained by projecting the solution $\mathbf{A}$ onto the two images.

Because of this, the solution to triangulation given here is not optimal. However, the solution found by using Longuet-Higgins's algorithm is typically used to initialize *bundle adjustment*, a computation that refines both motion $(R, \mathbf{t})$ and structure $(\mathbf{A}_1, \dots, \mathbf{A}_n)$ to minimize the image reprojection error—a nonlinear function of the unknowns. As a consequence, the approximate triangulation method described here is typically adequate, both as an initializer for bundle adjustment and to resolve the sign ambiguity discussed in Appendix C.

The projection equations for each point $\mathbf{A}$ can be written as follows for the two cameras:

$$\mathbf{a}_2 = \frac{1}{Z}\begin{bmatrix} X \\ Y \end{bmatrix} \quad \text{and} \quad \mathbf{b}_2 = \frac{1}{\mathbf{k}^T(\mathbf{A}-\mathbf{t})} R_2(\mathbf{A}-\mathbf{t}) \quad \text{where} \quad R_2 = \begin{bmatrix} \mathbf{i}^T \\ \mathbf{j}^T \end{bmatrix}$$

and where $\mathbf{i}^T$, $\mathbf{j}^T$, $\mathbf{k}^T$ are the rows of the rotation matrix R. The vector $\mathbf{A} = (X, Y, Z)^T$ collects the unknown coordinates of the point in space. Multiplying each equation by the denominator in its right-hand side and rearranging terms yield the following over-constrained 4×3 system of linear equations in $\mathbf{A}$:

$$\left[\begin{array}{c|c} I & -\mathbf{a}_2 \\ \hline \mathbf{b}_2\,\mathbf{k}^T - R_2 & \end{array}\right] \mathbf{A} = \left[\begin{array}{c} \mathbf{0} \\ \hline (\mathbf{b}_2\mathbf{k}^T - R_2)\mathbf{t} \end{array}\right]$$

where I is the 2×2 identity matrix and $\mathbf{0}$ is a column vector with two zeros. The solution $\mathbf{A}$ to this system can be found by the Least Squares method, and $\mathbf{B}$ can be computed by transforming $\mathbf{A}$ to the reference system of camera b:

$$\mathbf{B} = R\,(\mathbf{A} - \mathbf{t})\ .$$

This procedure is to be repeated for each of the image-point pairs.

Appendix C: 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 system 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:

$$R_2R_1^T = (-\boldsymbol{\alpha}_1\boldsymbol{\beta}_1^T - \boldsymbol{\alpha}_2\boldsymbol{\beta}_2^T - \mathbf{s}\mathbf{t}^T)(\boldsymbol{\beta}_1\boldsymbol{\alpha}_1^T + \boldsymbol{\beta}_2\boldsymbol{\alpha}_2^T - \mathbf{s}\mathbf{t}^T) = -\boldsymbol{\alpha}_1\boldsymbol{\alpha}_1^T - \boldsymbol{\alpha}_2\boldsymbol{\alpha}_2^T + \mathbf{s}(\mathbf{s})^T\ ,$$

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 $\mathbf{A}_i$ and $\mathbf{B}_i$ by their opposites $-\mathbf{A}_i$ and $-\mathbf{B}_i$. This transformation is said to change the *chirality* of structure in the literature [2], because superposing the original structure with the transformed one requires a change of handedness of the

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

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, as shown in Appendix B, 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:

$$\mathbf{e}_3^T \mathbf{A}_i > 0 \quad \text{and} \quad \mathbf{e}_3^T \mathbf{B}_i > 0 \quad \text{for} \quad i = 1, \ldots, n \quad \text{where} \quad \mathbf{e}_3^T = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} .$$

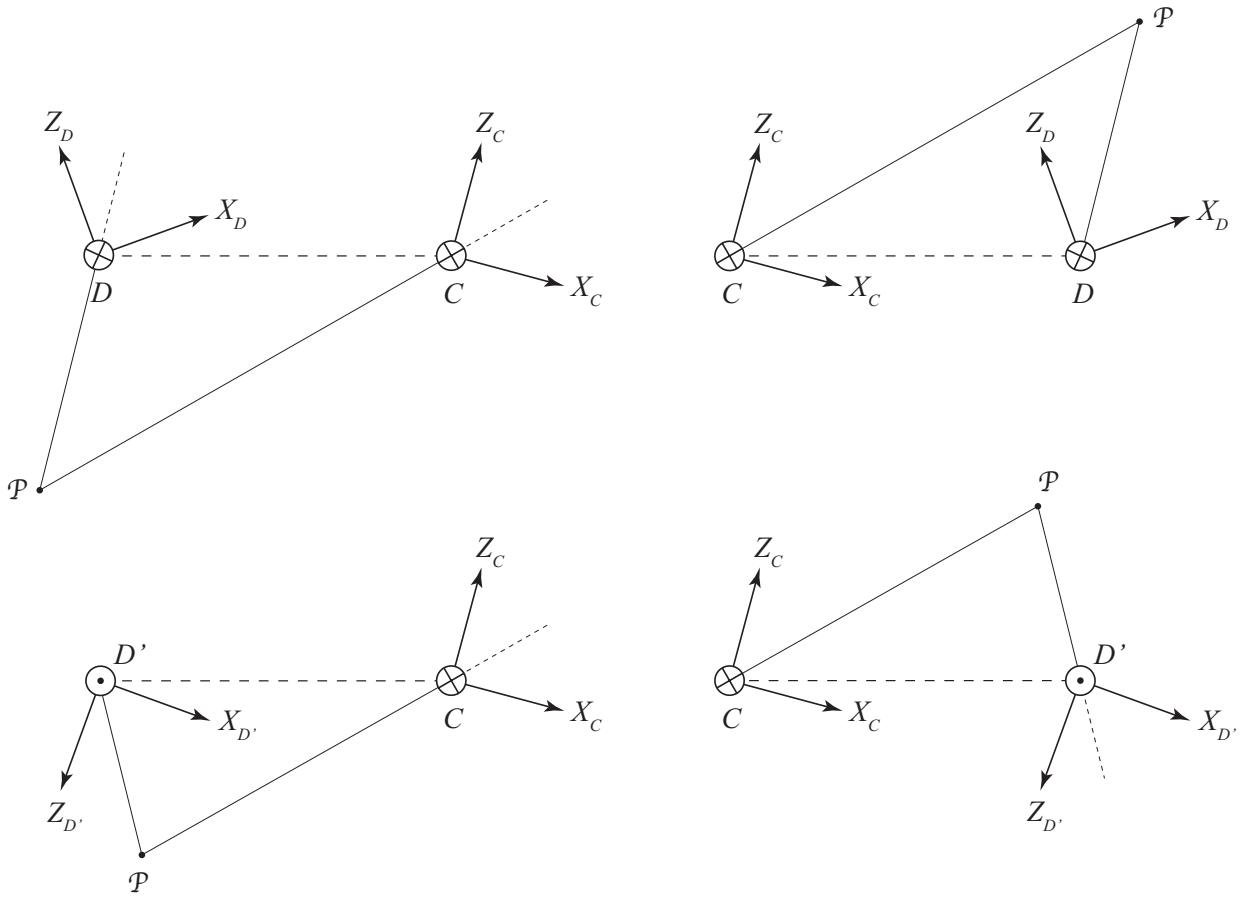

Figure 3: The fourfold ambiguity of reconstruction corresponds to the two ways to pick the sign of $\mathbf{t}$ (left or right diagrams) and the two ways to choose the rotation matrix R (top or bottom diagrams). A circle with a cross (a dot) denotes a Y axis pointing into (out of) the page. Only the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