5.1.1. Elimination of Structure Unknowns

Since we assume rigidity we know that the motion can be represented by a rotation and a translation

$$P' = RP + T \tag{5.1}$$

where P and P' are the position vectors of a world point before and after the motion, R is the rotation matrix and T is the translation vector. In the relative orientation problem all of them are unknowns. From the projective equations we have

$$p' = \frac{P'}{Z'}$$

and

$$p = \frac{P}{Z}$$

where p and p' are the projections of the world points on the image plane and Z and Z' are the z-coordinates of points P and P' respectively. Vectors p and p' are the only knowns in this problem since we can measure them on the image. If we eliminate the world points then Eq. (5.1) becomes

$$Z'p' = ZRp + T. \tag{5.2}$$

Now we have one equation where p and p' are knowns and all the rest are unknowns. The unknowns are of two kinds: structure unknowns that are related to the structure of the scene, e.g. the position of the points viewed, and motion unknowns which describe the motion. Z and Z' belong to the first kind of unknowns and there are two of them for each point in the scene. The motion parameters belong to the second kind and are independent of the number of points in the scene.

To solve the problem we have to start eliminating unknowns. The strategy we follow is to eliminate all the per-point unknowns and get an equation independent of structure. Before we do this let us look at the balance of equations first. Eq. (5.2) is a vector equation that is equivalent to 3 scalar equations. If we eliminate the two structure unknowns Z and Z' we have only one equation left.

To eliminate the Z' we can use a property of the cross product that says that the cross product of a vector with itself is the zero vector:

$$a \times a = \vec{0}$$

and multiply both sides of Eq. (5.2) by p'

$$Z'p' \times p' = Z(Rp) \times p' + T \times p' = 0$$

and we still have a vector equation that is equavalent to 3 scalar ones (but they are not independent anymore) and one less unknown. We can eliminate the remaining unknown by using a property of the dot product that says that the dot product of two orthogonal vectors is equal to zero. We know that the cross product of T and p' is another vector that is orthogonal to both T and p'. So if we take the dot product of both sides with T we have

$$0 = Z\Big((Rp)\times p'\Big)\cdot T + \Big(T\times p'\Big)\cdot T = Z\Big((Rp)\times p'\Big)\cdot T.$$

Now assuming Z is not equal to zero (otherwise the object would be way too close to our camera) we get

$$\Big((Rp)\times p'\Big)\cdot T = 0. \tag{5.3}$$

We now have one scalar equation whose unknowns are the motion parameters. We can simplify things a bit more if we notice that Eq. (5.3) is a *triple scalar product*. Recall that one of the definitions of the cross product of two vectors

$$V_1 = \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix}$$

and

$$V_2 = \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix}$$

is equal to the following depterminant

$$V_1 \times V_2 = \begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \end{vmatrix} =$$

$$(b_1c_2 - b_2c_1)\hat{x} + (c_1a_2 - c_2a_1)\hat{y} + (a_1b_2 - a_2b_1)\hat{z} = \tag{5.4}$$

$$\begin{bmatrix} b_1c_2 - b_2c_1 \\ c_1a_2 - c_2a_1 \\ a_1b_2 - a_2b_1 \end{bmatrix}$$

where $\hat{x}$, $\hat{y}$, $\hat{z}$ are the unit vectors along the corresponding axes. If V_3 is

$$V_3 = \begin{bmatrix} a_3 \\ b_3 \\ c_3 \end{bmatrix} = a_3\hat{x} + b_3\hat{y} + c_3\hat{z}$$

then, the triple scalar product of V_1, V_2, V_3 can be written as

$$(V_1, V_2, V_3) = (V_1 \times V_2)\cdot V_3 = \begin{vmatrix} a_3 & b_3 & c_3 \\ a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \end{vmatrix}$$

allowing the well known properties of the determinants can be applied here. Every time we swap two rows the determinant changes sign and in our case if we swap two vectors then the left hand side of (5.3) should stay zero. If we apply this property a couple of times we get

$$p' \cdot (T \times (Rp)) = 0. \tag{5.5}$$

Time now for yet another representation of the cross product. Eq. (5.4) can be rewritten in matrix form as

$$V_1 \times V_2 = \begin{bmatrix} b_1c_2 - b_2c_1 \\ c_1a_2 - c_2a_1 \\ a_1b_2 - a_2b_1 \end{bmatrix} = \begin{bmatrix} 0 & -c_1 & b_1 \\ c_1 & 0 & -a_1 \\ -b_1 & a_1 & 0 \end{bmatrix} \begin{bmatrix} a_2 \\ b_2 \\ c_2 \end{bmatrix}$$

We can also write the dot product in matrix form:

$$V_1 \cdot V_2 = V_1{}^T V_2 = V_2{}^T V_1.$$

Using the above notation we can write (5.5) as

$$p'^T \tilde{T} R p = 0$$

where

$$\tilde{T} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

and

$$T = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix}$$

and if we do the substitution

$$E = \tilde{T} R \tag{5.6}$$

we finally arrive at

$$p'^T E p = 0 \tag{5.7}$$

which is the celebrated *epipolar constraint* and in various forms has been reinvented many times in the history of science and engineering.

The epipolar constraint Eq. (5.7) is a very convenient equation because it is linear in terms of the elements of E so we can compute it from a set of point correspondences.

5.1.2. Recovering the E-matrix

Since Eq. (5.7) is a linear equation on the elements of E we can apply it to several point pairs p_i and p'_i and get a best estimate of this matrix. But since image data are always corrupted by noise, we need to take many more than the minimum number of such points. The standard tool for doing that is the *Least Squares*.

It is obvious that we need to do a small transformation on Eq. (5.7) to look more like an equation. Matrix E

$$E = \begin{bmatrix} e_1 & e_2 & e_3 \\ e_4 & e_5 & e_6 \\ e_7 & e_8 & e_9 \end{bmatrix}$$

can be written as a vector

$$\mathbf{e} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \\ e_7 \\ e_8 \\ e_9 \end{bmatrix} .$$

If we define vector

$$A_i = \begin{bmatrix} x'_i x_i \\ x'_i y_i \\ x'_i z_i \\ y'_i x_i \\ y'_i y_i \\ y'_i z_i \\ z'_i x_i \\ z'_i y_i \\ z'_i z_i \end{bmatrix}$$

then Eq. (5.7) can be written as

$$A_i^T \mathbf{e} = 0 \tag{5.8}$$

This is a homogeneous equation which means that vector $\mathbf{e} = 0$ satisfies it. And if a vector $\mathbf{e}_0 \neq \vec{0}$ satisfies it, then $2\mathbf{e}_0$ satisfies it. In plain terms this means that we cannot say if we are looking at an object 1 ft large, 1 ft away and the camera moved 1 inch, or the object is two ft large, two ft away and the camera moved 2 inches. So we can only expect to recover structure and motion up to a scale factor. The proportions are preserved but we cannot tell if the object is small and close or far away and large. It is not that the information is there and we did not do the math right; it is inherently impossible to determine the scale factor from monocular views. Humans cannot either, and cinematographers use this to make little ships in swimming pools look like real life navies.

We will see how to handle this problem later in this section. Meanwhile we have to solve the following problem. Image data are notoriously unreliable and vector A_i contains image data. The only way to overcome this is to use more data than we need and the noise in the data will cancel out. The classic approach to this is *Least Squares*: we sum up the squares of all the equations and minimize this sum

$$\sum_i \left(A_i^T \mathbf{e}\right)^2 = \sum_i \left(\mathbf{e}^T A_i A_i^T \mathbf{e}\right) = \mathbf{e}^T \left(\sum_i A_i A_i^T\right) \mathbf{e} = \mathbf{e}^T A \mathbf{e}$$

We can try to minimize the above quantity but we already know the answer. The vector $\mathbf{e} = 0$ minimizes it, but it is not what we want. There are three ways to solve this problem. One that seems the simplest is to arbitrarily set one element of $\mathbf{e}$ to 1.0 and then solve a regular linear problem. While this has been used in practice, it it not the best because if we are unlucky and choose an element of $\mathbf{e}$ that is equal to zero, then we get an unstable solution. The second approach is to minimize it, subject to the condition that $|\mathbf{e}| = 1$. This is a perfectly fine solution but requires the use of *Lagrange Multipliers*. We opt for the third solution that is mathematicaly equivalent but a bit easier. We simply minimize

$$\lambda = \frac{\mathbf{e}^T A \mathbf{e}}{\mathbf{e}^T \mathbf{e}} \tag{5.9}$$

which is independent of the length of $\mathbf{e}$. We now take the derivatives of λ with respect to the elements e_j of $\mathbf{e}$

$$\frac{\partial \lambda}{\partial e_j} = \frac{2(\mathbf{e}^T \mathbf{e})\hat{b}_j^{\,T} A\mathbf{e} - 2\hat{b}_j \mathbf{e}\mathbf{e}^T A\mathbf{e}}{\left(\mathbf{e}^T \mathbf{e}\right)^2} = 2\,\frac{\hat{b}_j\left[(\mathbf{e}^T\mathbf{e})A\mathbf{e} - \mathbf{e}\mathbf{e}^T A\mathbf{e}\right]}{\left(\mathbf{e}^T \mathbf{e}\right)^2}$$

where $\hat{b}_j = \partial\mathbf{e}/\partial e_j$ and is a nine dimensional unit vector, all elements of which are zero except the j^{th} which is equal to one. We know that

$$\frac{\hat{b}_j\left[\mathbf{e}^T\mathbf{e}A\mathbf{e} - \mathbf{e}\mathbf{e}^T A\mathbf{e}\right]}{\left(\mathbf{e}^T \mathbf{e}\right)^2} = 0$$

for $1 \le j \le 9$, in other words $\mathbf{e}^T\mathbf{e}A\mathbf{e} - \mathbf{e}\mathbf{e}^T A\mathbf{e}$ has a zero projection to all nine unit vectors $\hat{b}_j$. From this we can infer that

$$\mathbf{e}^T\mathbf{e}A\mathbf{e} = \mathbf{e}\mathbf{e}^T A\mathbf{e}$$

or

$$\left(\mathbf{e}^T\mathbf{e}\right)A\mathbf{e} = \mathbf{e}\left(\mathbf{e}^T A\mathbf{e}\right)$$

and since the parenthesized quantities are scalars

$$A\mathbf{e} = \mathbf{e}\,\frac{\mathbf{e}^T A\mathbf{e}}{\mathbf{e}^T\mathbf{e}}$$

and if we use Eq. (5.9) we get

$$A\mathbf{e} = \mathbf{e}\lambda$$

in which case **e** is an eigenvector of A and λ is the corresponding eigenvalue. So the minimum of Eq. (5.8) is attained when **e** is the eigenvector of A that has the smallest eigenvalue (all eigenvalues of A are positive).

The procedure then to find **e** is to form matrix $A = \sum_i A_i A_i^T$, compute the smallest eigenvalue and find the correspoinding eigenvector. This eigenvector is **e**.

5.1.2.1. Solving for Translation

We can recover the translation T if we notice the

$$T^T\tilde{T} = (\tilde{T}T)^T = (T \times T)^T = 0$$

so we can infer that

$$T^T E = (T^T\tilde{T})R = 0 \tag{5.10}$$

which means that T is a vector that satisfies Eq. (5.10). Although there are simpler ways to find such a vector we use a technique called *Singular Value Decomposition* (SVD). Any square matrix M, symmetric or not, can be decomposed into a product of three matrices

$$M = USV^T$$

where U and V are orthonormal matrices (we can extend the definition when M is not square, but this is another issue). The columns U_i and V_i of matrices U and V are called left and right singular vectors. S is a diagonal matrix with all the diagonal elements σ_i non negative. This decomposition is *almost* always unique up to some trivial transformations. We can for instance exchange the left singular vectors U_i and U_j if we exchange V_i with V_j and σ_i with σ_j as well. Also we can replace U_i with $-U_i$ if we replace V_i with $-V_j$ as well. And finaly if one of the singular values σ_k is equal to zero then we can flip the sign of either U_k of V_k and get away with it. There is one more complication regarding the expression *almost* always unique. On an extremely rare occasion when two singular values are identical there are infinite number of decompositions. According to Murphy's Theorem matrix E always falls under this category. Nevertheless, despite the filthiest intentions of the Founding Fathers of mathematics, all these amount to a minor nuisance. The reason is the following. The non-uniqueness of the decomposition matters only if we decompose two matrices and we expect some simple relation between the respective components, which is exactly what we need to do here. But if we decompose one of the two matrices and using the expected relation between the components to decompose the other without calling the SVD function again, then we can make sure that we have the right pair of decompositions.

Another way to write the decomposition is

$$M = \sum_i U_i \sigma_i V_i^T \tag{5.11}$$

where U_i and V_i are the columns od U and V respectively and σ_i are the singular values. From Eq. (5.11) above it is easy to notice that if all singular values are non zero then $Mx \neq 0$ for all non-zero vectors x.

Too much philosophy. Let's do some work. We can decompose matrix E into

$$E = U_E S_E V_E{}^T$$

and since we know there is a T that satisfies Eq. (5.10), we know that one of the singular values is zero. The corresponding left singular vector is parallel to the translation T. We can "find" the magnitude of T if we notice that

$$EE^T = \tilde{T} R R^T \tilde{T}^T = \tilde{T}\tilde{T}^T = T_2 \mathbf{1} - TT^T \tag{5.12}$$

from which we can find the magnitude of T. We do not need though, because the computation of R works just fine if we do not know it but also the magnitude of this T has nothing to do with the length of the real T which is indeterminent and is always 1/2 if the vector **e** we computed in the previous section is a unit vector. So it does not matter if we know it. By the way: Eq. (5.12) provides an alternative way to compute T.

5.1.2.2. Solving for Rotation Matrix

The next step is to find the rotation matrix. We know T up to a sign, but it is not that easy to find R. Although $E = \tilde{T} R$, and we know $\tilde{T}$, $\tilde{T}$ is not an invertible matrix, therefore we cannot just eliminate $\tilde{T}$ to get R. The trick this time involves again the SVD of E and $\tilde{T}$.

We can decompose matrices E and $\tilde{T}$ into

$$E = U_E S_E V_E{}^T \tag{5.13}$$

$$\tilde{T} = U_T S_T V_T{}^T. \tag{5.14}$$

We also know that $E = \tilde{T} R$ so

$$E = U_T S_T V_T{}^T R$$

and the "uniqueness" of the SVD implies that

$$\begin{aligned} U_T &= U_E \\ S_T &= S_E \\ {V_T}^T R &= {V_E}^T \end{aligned} \tag{5.15}$$

or at least if E has an SVD as per Eq. (5.13) then $\tilde{T}$ has an SVD as per Eq. (5.14) and conditions of Eq. (5.15) hold. In this case we can solve for R and get

$$R = V_T {V_E}^T. \tag{5.16}$$

The task now is to reconstruct the SVD of Eq. (5.14) using the SVD of Eq. (5.13). The first thing to notice is that

$$U_E^T \tilde{T} = U_T^T U_T S_T {V_T}^T = S_T {V_T}^T$$

and since U_E and $\tilde{T}$ are known we now know $S_T {V_T}^T$. Unfortunately, we know that the first singular value of S_T is zero and so eliminating it is not as simple as multiplying with the inverse. But we know that V_T is an orthonormal matrix and every column (or row) of an orthonormal matrix has unit length. Let's have a closer look at $S_T {V_T}^T$ and the way to proceed will become clear.

$$S_T {V_T}^T = \begin{bmatrix} 0 \\ \cdots\cdots\cdots \\ \sigma_2 V_2^T \\ \cdots\cdots\cdots \\ \sigma_3 V_3^T \end{bmatrix}$$

where σ_2 and σ_3 are the singular values of $\tilde{T}$. We know $S_T {V_T}^T$, so we know $\sigma_2 V_2$, and we also know that sigma is positive, so

$$V_2 = \frac{\sigma_2 V_2}{|\sigma_2 V_2|}$$

and similarly for V_3. We also know that every column (or row) of an orthonormal matrix is orthogonal to every other column (or row). So the first column of V_T is

$$V_1 = s_1 V_2 \times V_3$$

where s_1 is a yet to be determined sign. And so now we have a complete V_T matrix, up to a sign ambiguity. We can plug it in Eq. (5.16) and get R. Not so fast. R is a rotation matrix, so its determinant is unity, whereas orthogonal matrices have determinants that are ± 1. Furthermore, if R satisfies our equations so does $-R$ because matrix E itself has sign ambiguity. So

$$R = s_2 V_T {V_E}^T$$

where s_2 is another yet to be determined sign. After brushing up our linear algebra we can determine that the determinant of R is

$$|R| = s_1 s_2 \, |V_E| = 1$$

since $|V_T| = s_1$. Now we can determine the sign s_1

$$s_1 = s_2 |V_T| \, |V_E|$$

which leads to two solutions for R, both with positive determinant (sometimes called right handed rotation matrices) depending on the value of s_2.[†]

This completes our quest for the decomposition of the matrix E. We get a three way sign ambiguity, one ambiguity for the translation vector T and two for the rotation matrix R, but we already saw that the two ambiguities for the rotation can be reduced to one since they lead to negative determinants (also called lefthanded rotation matrices). But we cannot do anything at this point for the rest of the ambiguities and just return two translations and two rotations. In the next stage we see how we can reject all but one of the 4 solutions.

5.1.2.3. Spurious Solutions The eight solutions that we get are not an artifact of the particular algorithm since they have a physical meaning. Four of them are easy to eliminate since they have lefthanded rotation matrices (unless of course we are doing structure from motion in front of mirror: vanity is the deadliest of the sins for Computer Visionaries!). Now let's see the

Assume that we have the correct solution. Since moving a rigid object in front of a stationary camera is equivalent to moving the camera in front of a stationary scene, we choose for convenience the later. We place the cameras to their correct relative position and we draw lines from the nodal point (center of the lens) to the image point and then extend them to the space. It is obvious that the extensions from an image point in the one camera and its corresponding point in the other camera will intersect in space. The intersection is on the 3-D point that gave rise to the images (Fig. 5.1). If they did not intersect then there is something wrong with the relative position of the cameras.

If we now rotate the second camera 180 degrees around the translation vector T, then the corresponding extensions will remain coplanar and thus intersect. But as we notice from Fig. 5.2, the intersection is behind the camera. So this is the one spurious solution and can be distinguished from the presence of behind the camera points.

The number of solutions doubles if we reverse the sign of the translation. Extensions that were coplanar before are still coplanar and as we see in Fig. 5.3 they still intersect behind the camera.

[†]one can define s_2 in a different way and have slightly (very slightly) simpler equations. If we define V_T as

$$V_T = \begin{bmatrix} & \vdots & & \vdots & \\ s_1 V_1 & \vdots & s_2 V_2 & \vdots & s_2 V_3 \\ & \vdots & & \vdots & \end{bmatrix}$$

where we set $s_1 = |V_T|$.

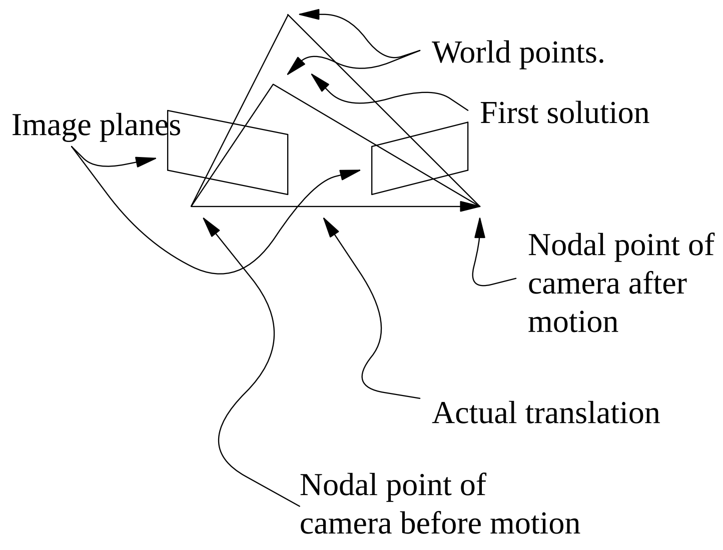
World points.
First solution
Image planes
Nodal point of camera after motion
Actual translation
Nodal point of camera before motion

Figure 5.1: Two points are “seen” by the camera before and after motion. It is obvious that if we draw the rays connecting the nodal point of a camera and an image and then extend this ray into space, it will pass through the 3-D point. These extensions from two cameras involving a single 3-D point, intersect each other at this point, so they are coplanar.

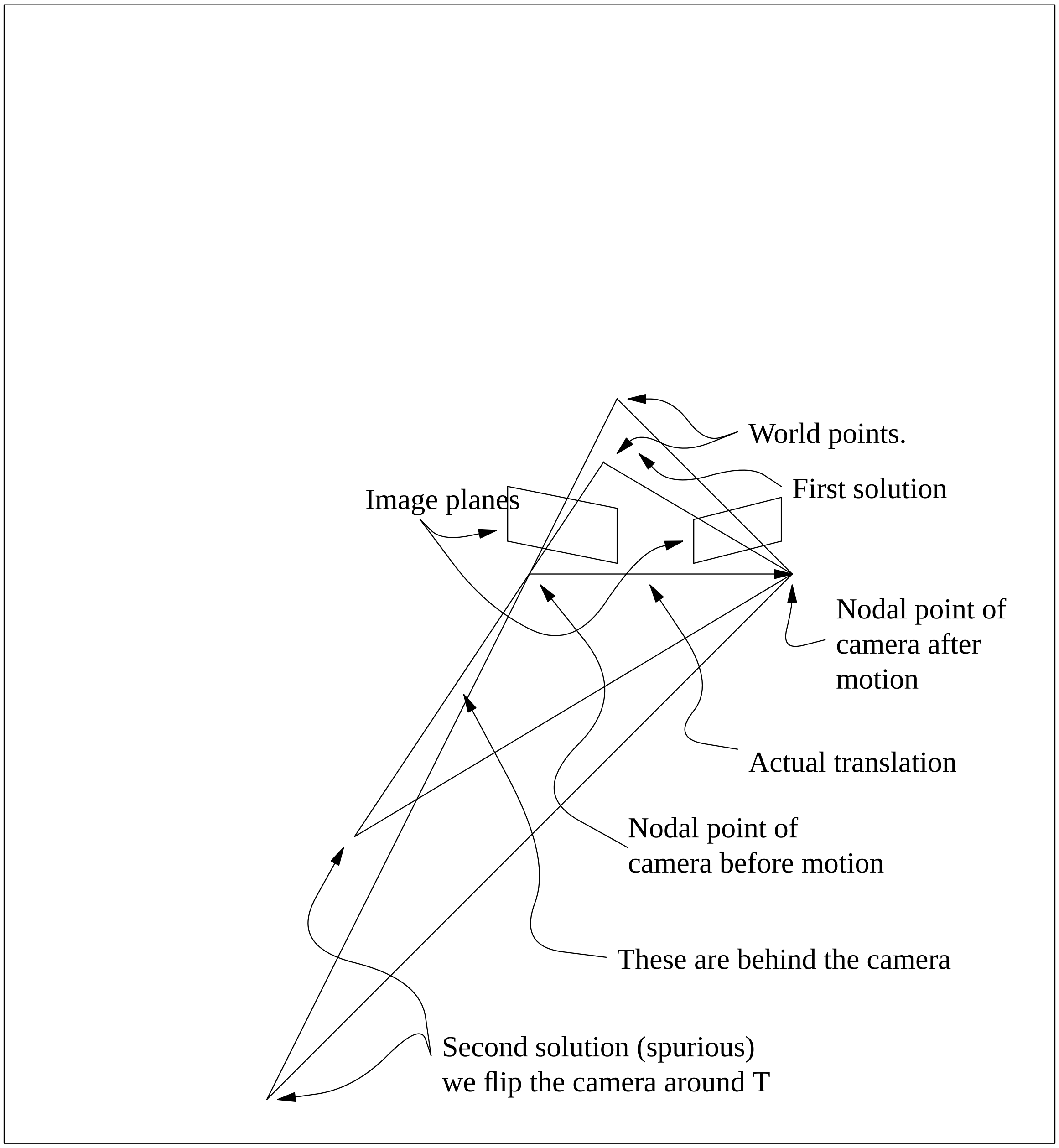
World points.
First solution
Image planes
Nodal point of camera after motion
Actual translation
Nodal point of camera before motion
These are behind the camera
Second solution (spurious) we flip the camera around T

Figure 5.2: If we rotate the right camera (the after-motion camera) 180 degrees around the translation vector, all the rays that were coplanar with the corresponding ray in the left camera before the rotation will be coplanar after the rotation and thus intersect. The intersection will be behind one or the other camera. This is an impossible configuration.

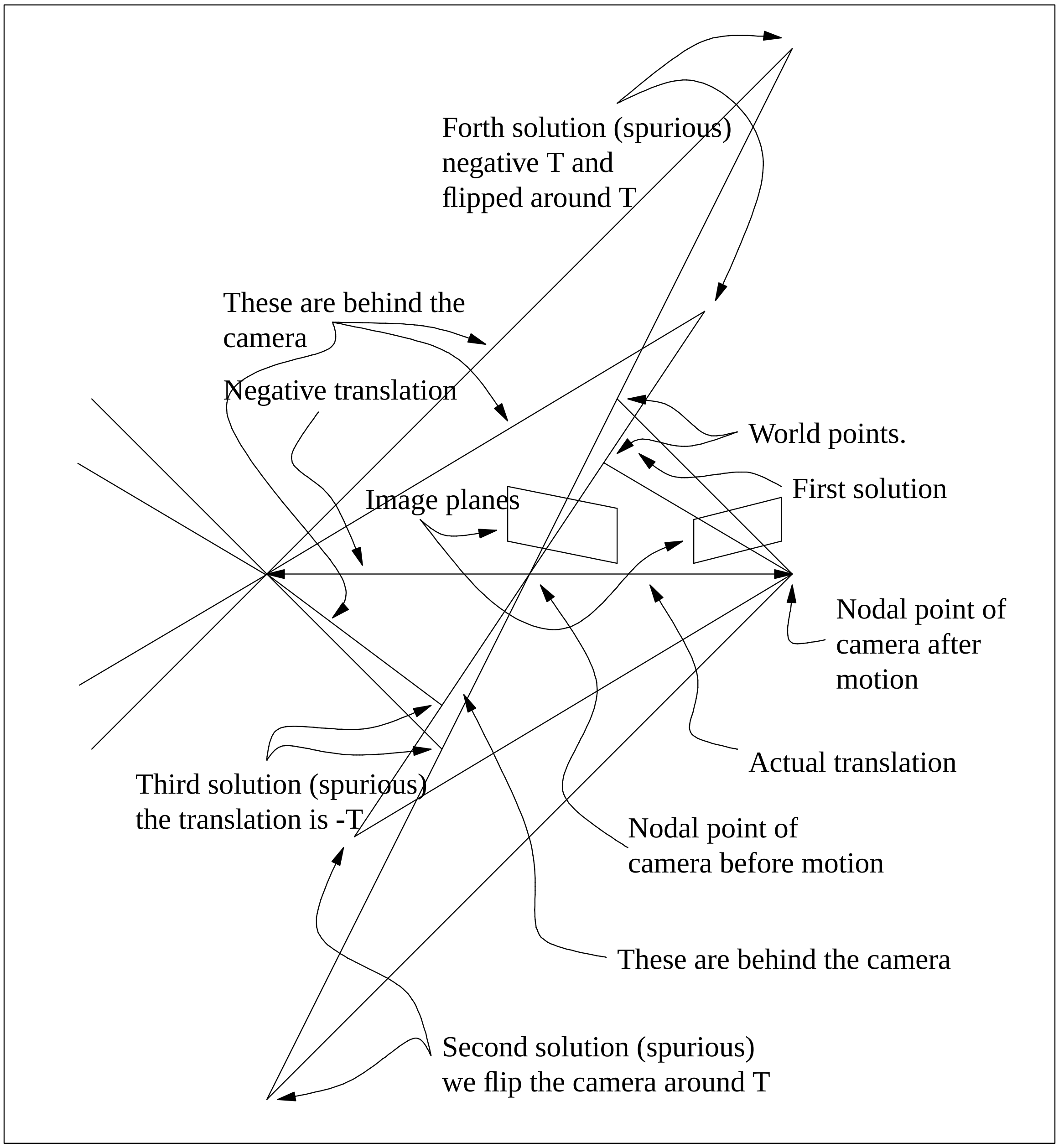

Figure 5.3: If we change the sign of T, then the after-motion camera moves to the other side and everything is scaled by -1 including the Z distances. This way we double the number of solutions to four. But the new ones will contain points behind the camera, so they are spurious.

The simple algorithm to reject the spurious solutions is as follows. Compute structure for all of the four combinations of rotations and translations and for each combination count the number of points behind the camera, both before and after the motion. The combination with the fewest behind the camera points wins.

6. Recovering Structure

Starting again from the rigidity Eq. (5.2)

$$Z' p' = ZRp + T.$$

we can assume that the motion parameters R and T are known and find the structure Z and Z', which are the depths of the image points. This is a vector equation equivalent to three scalar ones, but we have two unknowns. We can easily forget the one equation and solve for our two unknowns, but then we know this is not the best thing to do. A far better solution is to use all three equations and find the least squares solution. So we try to minimize

$$Q(Z', Z) = (Z' p' - ZRp - T)^2 = (Z' p' - ZRp - T)^T (Z' p' - ZRp - T)$$

from which after taking the derivatives with respect to the unknowns Z and Z' we get

$$\frac{\partial Q(Z', Z)}{\partial Z'} = 2 p'^T (Z' p' - ZRp - T) = 0$$
$$\frac{\partial Q(Z', Z)}{\partial Z} = 2 p^T R^T (Z' p' - ZRp - T) = 0$$

which leads to

$$\begin{bmatrix} p'^T p' & -p'^T Rp \\ -p'^T Rp & p^T p \end{bmatrix} \begin{bmatrix} Z' \\ Z \end{bmatrix} = \begin{bmatrix} T^T p' \\ -T^T Rp \end{bmatrix}$$

and simple matrix inversion gives us

$$\begin{bmatrix} Z' \\ Z \end{bmatrix} = \frac{1}{(p'^T p')(p^T p) - (p'^T Rp)^2} \begin{bmatrix} p^T p & p'^T Rp \\ p'^T Rp & p'^T p' \end{bmatrix} \begin{bmatrix} T^T p' \\ -T^T Rp \end{bmatrix}.$$

This gives the ability to discard the spurious solutions for the motion with the following very simple algorithm: Compute the depths Z and Z' for all four solutions and for each one count the points that have at least one negative depth. The solution with the fewest behind the camera points is the real one.