Projective Geometry Considered Harmful

Berthold K.P. Horn

Introduction

Methods based on projective geometry have become popular in machine vision because they lead to elegant mathematics, and easy-to-solve linear equations [Longuett-Higgins 81, Hartley 97a, Quan & Lan 99]. It is often not realized that one pays a heavy price for this. Such method do not correctly model the physics of image formation and as a result require more correspondences and are considerably more sensitive to measurement error than methods based on true perspective projection.

Projective geometry based methods are rarely used in photogrammetry [Wolf 74, Slama 80]. There the cost of acquiring the data is high and every effort is made to extract information about the scene and about the image taking geometry that is as accurate as possible.

Since linear equations are easier to solve, there may appear to be an advantage in computational cost, but this advantage — if any — has been eroded by the reduced cost and increased speed of computation. Given the noisy nature of image measurements, one simply cannot afford to throw away accuracy.

The issue of the limitations of projective geometry when applied to photogrammetric problems has been raised before, particuarly in the context of the relative orientation problem that arises in binocular stereo [Hartley 97b]. But relative orientation is a relatively complex problem where it is hard to gain insight from simple geometric arguments or numerical experiments. As a result, not all researchers have been persuaded that methods based on projective geometry are in fact inferior. Still, it is hard to see the attraction of linear methods for relative orientation, since good methods for solving the least squares problem of relative orientation do exist [Horn 90, 91].

We revisit this topic here in the context of a simpler problem, that of exterior orientation with respect to a planar object. We examine the difference between the mapping from the object plane to the image plane defined by true perspective projection and that defined by projective geometry. We show that virtually none of the transformations allowed by projective geometry correspond to real camera image-taking situations.

We then compare the algorithms and study the sensitivity to noise using Monte Carlo methods and show that the error sensitivity of projective geometry based methods is much higher.

Projective Geometry versus Perspective Projection.

The true mapping from coordinates in the object coordinate system to image coordinates consists of two steps:

(i) Rigid body transformation of the object coordinate system into the camera coordinate system. Rigid body transformations are combinations of rotation and translation. If, for now, we use orthonormal matrices to represent rotation, we can write

$$\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = R \begin{pmatrix} x_t \\ y_t \\ z_t \end{pmatrix} + \begin{pmatrix} x_o \\ y_o \\ z_o \end{pmatrix} \tag{1}$$

where R is orthonormal and represents rotation, while $\mathbf{t} = (x_o, y_o, z_o)^T$ is the translation (position of the object coordinate system origin in the camera coordinate system). Orthonormality implies $R^T R = I$. This equation imposes six independent non-linear constraints on the 9 elements of the 3×3 matrix R. In addition, for R to represent a rotation (rather than a reflection), we also need $\det(R) = +1$,

(ii) Perspective projection

$$\begin{aligned} u &= f(x_c/z_c) + u_o \\ v &= f(y_c/z_c) + v_o \end{aligned} \tag{2}$$

where f is the principal distance (effective focal length) and $(u_o, v_o)^T$ is the principal point (base of perpendicular dropped from the center of projection to the image plane). The interior orientation of the camera is summarized in the vector $(u_o, v_o, f)^T$ from the center of projection to the principal point in the image plane.

For simplicity, we now consider a planar object, where we can for convenience arrange the object coordinate system such that $z_t = 0$ in the plane of the object. If we write out the elements of the rotation matrix we find:

$$\begin{pmatrix} x_c \\ y_c \\ z_c \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix} \begin{pmatrix} x_t \\ y_t \\ 0 \end{pmatrix} + \begin{pmatrix} x_o \\ y_o \\ z_o \end{pmatrix}. \tag{3}$$

Absorbing the translation $(x_o, y_o, z_o)^T$ into the 3×3 matrix and dividing the

third component by f we get:

$$\begin{pmatrix} x_c \\ y_c \\ z_c/f \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & x_o \\ r_{21} & r_{22} & y_o \\ r_{31}/f & r_{32}/f & z_o/f \end{pmatrix} \begin{pmatrix} x_t \\ y_t \\ 1 \end{pmatrix}. \tag{4}$$

Note that this equation involves a vector $(x_t, y_t, 1)^T$, unlike the previous equation using $(x_t, y_t, 0)^T$ (This makes it easier to match this result with the projective geometry formulation). Now perspective projection (eq. 2) gives us

$$\begin{pmatrix} k(u-u_o) \\ k(v-v_o) \\ k \end{pmatrix} = \begin{pmatrix} r_{11} & r_{12} & x_o \\ r_{21} & r_{22} & y_o \\ r_{31}/f & r_{32}/f & z_o/f \end{pmatrix} \begin{pmatrix} x_t \\ y_t \\ 1 \end{pmatrix}, \tag{5}$$

where $k = z_c/f$. We write the equation in this unusual form to make it easier to identify terms with those occuring in the equations that arise in projective geometry — to be discussed next.

The homogeneous representation of a point in a plane uses three numbers $(u, v, w)^T$ [Wylie 70]. The actual planar coordinates are obtained by dividing the first two elements by the third: $x = u/w$, and $y = v/w$. Naturally, this representation is not unique, since any non-zero multiple $(ku, kv, kw)^T$ corresponds to the same position in the plane. Homogeneous coordinates are used because they make it easier to apply the methods of projective geometry.

A 3×3 matrix T represents a homogeneous transformation from the object plane to the image plane. T multiplied by a 3-vector $(x_t, y_t, 1)^T$ representing position in the object plane yields a 3-vector $(ku, kv, k)^T$ that represents the corresponding position in the image plane — both in homogeneous coordinates:

$$\begin{pmatrix} ku \\ kv \\ k \end{pmatrix} = \begin{pmatrix} t_{11} & t_{12} & t_{13} \\ t_{21} & t_{22} & t_{23} \\ t_{31} & t_{32} & t_{33} \end{pmatrix} \begin{pmatrix} x_t \\ y_t \\ 1 \end{pmatrix} \tag{6}$$

This matches the equation for true perspective geometry *provided* we set up the elements of the matrix T as follows:

$$\begin{aligned} t_{11} &= r_{11}, \quad t_{12} = r_{12}, \quad t_{13} = x_o \\ t_{21} &= r_{21}, \quad t_{22} = r_{22}, \quad t_{23} = y_o \\ t_{31} &= r_{31}/f, \quad t_{32} = r_{32}/f, \quad t_{33} = z_o/f \end{aligned} \tag{7}$$

In *addition*, for this identification of transformation formulas to work, measurements in the image must be made in a coordinate system with the origin at the principal point (so that u_o, $v_o = 0$). Keep in mind that any non-zero multiple of the matrix T in the above equation describes the same projective geometric relationship (i.e. there is a scale factor ambiguity).

What is important to note from the above is that if T is to represent a real perspective projection of the object plane into the image plane then it *must* satisfy

two non-linear constraints

$$t_{11}t_{12} + t_{21}t_{22} + f^2 t_{31}t_{32} = 0 \tag{8}$$

and

$$t_{11}t_{11} + t_{21}t_{21} + f^2 t_{31}t_{31} = t_{12}t_{12} + t_{22}t_{22} + f^2 t_{32}t_{32} \tag{9}$$

which follow directly from the orthonormality of the first two columns of the rotation matrix R.

Given the scale factor ambiguity, we can arbitrarily pick $t_{33} = 1$ and choose the other eight elements of T independently. (We can do this unless the object coordinate system origin lies in the image plane, since if $t_{33} = 0$, the origin of the object coordinate system $(0, 0, 1)^T$ is transformed by T into a coordinate with zero third component, that is, a point at infinity in the image plane).

The above two non-linear constraints reduce the degrees of freedom further from eight to six, which is as it should be, since rotation has three degrees of freedom and translation has three. Hence there are really only six independent variables, not eight.

We can always find a matrix T corresponding to a real perspective projection (if we assume that the principal point is known), using the equations above, but in general it is *not* possible to go in the other direction, that is, to find a perspective projection that corresponds to an arbitrary matrix T. That is, almost all homogeneous transformations T have the property that they do *not* allow a physical interpretation in terms of rigid body motion and perspective projection.

Importantly, an arbitrary 3×3 matrix T will *not* satisfy the two non-linear constraint and hence can *not* represent a true perspective projection. Examples of mappings allowed by projective geometry but not by perspective projection are skewing and anisotropic scaling. If, for example, we distort a normal perspective image by the additional operations

$$\begin{pmatrix} 1 & s \\ 0 & 1 \end{pmatrix} \qquad \text{or} \qquad \begin{pmatrix} 1 & 0 \\ 0 & k \end{pmatrix}$$

applied to the image coordinates $(x_i, y_i)^T$ then in general we will have an "image" that could not have been obtained from *any* position with *any* camera orientation. Yet such distortions merely change the elements of T and thus *are* permitted by projective geometry.

One may wonder whether there is a physical imaging situation that *does* correspond to a homogeneous transformation by an unconstrained matrix T. There is:

The transformations of perspective geometry correspond to taking a perspective image of a perspective image.

In this case, we obtain an overall transformation that need *not* satisfy the two non-linear constraints. (Interestingly, the transformation is not further generalized

by taking a picture of a picture of a picture). But in any case we are typically interested in a "direct" image, *not* a picture taken of a picture.

If one finds a transformation T that best fits the data without imposing the nonlinear constraints, one is left with the problem of finding the rigid body transformation R and $\mathbf{t}$ "nearest" to the linear transformation T. Without imposing the two constraints, the "solution" can adjust to measurement errors in ways that increase the error in the "nearest" rigid body transformation.

Estimating Object Attitude and Position using Projective Geometry

There are two distinct steps:

(i) the determinination of a projective geometry transformation T that maps points in the object plane into points in the image plane; and

(ii) finding an orientation (rotation R) and position (translation $\mathbf{t}$) of the object coordinate system expressed in terms of the camera coordinate system that approximates the transformation T.

Both parts can be dealt with using homogeneous coordinate notation. See Appendix A for details of recovering T (part (i)).

Recovery of Orientation

A simple method for the recovery of the orientation and position of the object coordinate system in the camera coordinate system from T can be derived using the concept of vanishing points.

A straight line on the object maps into a straight line in the image under perspective projection. However, if we move along the line in the object at a constant rate, we do not move along the corresponding image line at a constant rate. The movement slows down and approaches a limit as we go off to infinity along the line in the object plane. This limiting point is called the vanishing point for that object line.

Consider the homogeneous 3-vector $(\alpha a, \alpha b, 1)^T$ as $\alpha \to \infty$. This is clearly just the same as $(a, b, 0)^T$. It follows that the vanishing point for a line with direction $(a, b)^T$ in the object plane is

$$\begin{aligned} u &= (t_{11}a + t_{12}b)/(t_{31}a + t_{32}b) \\ v &= (t_{21}a + t_{22}b)/(t_{31}a + t_{32}b) \end{aligned} \tag{10}$$

Importantly, if we construct a line from the center of projection ($(0, 0, 0)^T$) of the camera to the vanishing point in the image plane ($z = f$), we have a line that is parallel (in three dimensions) to the line on the object. We can apply this idea

to the x- and y-axes of the object, and from the two vanishing points find the directions of these two axes in the camera coordinate system.

The vanishing point for the x-axis is just $(1,0,0)^T$ in the object coordinate system. Multiplying the matrix T by this vector yields the homogeneous image coordinate $(t_{11}, t_{21}, t_{31})^T$. Similarly, we get $(t_{12}, t_{22}, t_{32})^T$ from $(0,1,0)^T$ for the y-axis. These two correspond to image coordinates $(t_{11}/t_{31}, t_{21}/t_{31})^T$ and $(t_{12}/t_{32}, t_{22}/t_{32})^T$ respectively (Points in the image, when written as vectors in the camera coordinate system, have the third component equal to f. For convenience we drop this constant component and write image positions using 2-vectors).

If we connect the center of projection to these points in the image plane we obtain direction vectors parallel to

$$\begin{aligned} \mathbf{x} &= (t_{11}, t_{21}, f t_{31})^T, \\ \mathbf{y} &= (t_{12}, t_{22}, f t_{32})^T. \end{aligned} \tag{11}$$

We can divide these two vectors by their magnitude to obtain unit vectors $\hat{\mathbf{x}}$ ad $\hat{\mathbf{y}}$ in the direction of the x- and y-axes of the object plane (expressed in the camera coordinate system). So it is easy to find the directions of the two object coordinate system axes (expressed in the camera coordinate system) directly from the first two columns of T (provided the principal distance f is known).

Since the z-axis — perpendicular to the object plane — has to be at right angles to any line in the object plane, we can find its direction simply by taking the cross-product of the directions of the x- and y-axes found above. A rotation matrix relating (3-d) object coordinates to (3-d) camera coordinates can now be constructed by adjoining the three unit column vectors in the directions of the coordinate axes:

$$R = \begin{pmatrix} \hat{\mathbf{x}} & \hat{\mathbf{y}} & \hat{\mathbf{z}} \end{pmatrix} \tag{12}$$

where $\hat{\mathbf{x}}$, $\hat{\mathbf{y}}$, $\hat{\mathbf{z}}$ are unit column vectors constructed from T, as described above.

Recovery of Translation

To complete the analysis, we find the position of the origin of the object coordinate system in the camera coordinate system. The homogeneous coordinates of the origin in the object plane are obviously just $(0,0,1)^T$. Multiplying T by this vector yields $(t_{13}, t_{23}, t_{33})^T$. The image of the origin of the object coordinate system then is at $(t_{13}/t_{33}, t_{23}/t_{33})^T$. Connecting the origin to this point in the image plane ($z = f$), yields a vector parallel to

$$\mathbf{t} = (t_{13}, t_{23}, f t_{33})^T. \tag{13}$$

So it is easy to find the *direction* of the translational vector to the object origin directly from the last column of T.

We can find the distance to the object origin from the center of projection if we can determine the magnification of a line parallel to the image plane at that distance (that is, the ratio of the length of the line in the image to the length of the line on the object). If the magnification is M (typically less than one), then the z-component of the translation vector must be f/M. We can use this value to scale the direction vector $\mathbf{t}$ found above.

A line in the object plane that is parallel to the image plane must be perpendicular to the camera coordinate system z-axis and also perpendicular to a normal of the object plane. Such a vector can be found by taking the cross-product of the camera system z-axis $(0, 0, 1)^T$, and a normal to the object plane (i.e. the z-axis of the object coordinate system found above). If the latter is $(z_1, z_2, z_3)^T$, then taking the cross-product with $(0, 0, 1)^T$ yields the vector $(-z_2, z_1, 0)^T$. The third component of this vector is zero, as it must be if it is to be parallel to the image plane. It is similarly easy to verify that it actually lies in the object plane.

We can use the matrix T to map the point with homogeneous coordinates $(-z_2, z_1, 1)^T$ into the image plane. The length of the line from the image of this point to the image $(t_{13}/t_{33}, t_{23}/t_{33})^T$ of the origin of the object coordinate system can be easily computed. The magnification M can be obtained by dividing this length by the length of the line on the object, which is obviously just

$$\sqrt{z_1^2 + z_2^2}.$$

Once we know the magnification M we can determine the translational offset of the object origin from the camera origin by multiplying $\mathbf{t} = (t_{13}, t_{23}, f t_{33})^T$ by $M/(f t_{33})$.

Analysis

Most homogeneous transform matrices T do not correspond to perspective projections of a (rotated and translated) plane. So one may wonder how we were able to compute a coordinate system transformation as above based on the model of perspective projection. The answer is that we selectively neglected some of the information in the matrix T. In fact, if we now construct a T' based on the recovered "rotation matrix" R, the translation $\mathbf{t}$ and perspective projection, we will find that in general T' is not equal to T. The reconstructed T' is only equal to the original T when there are no measurement errors.

A way this problem manifests itself is that the "rotation matrix" R constructed above is typically not orthonormal. This is because when we estimate the directions of the x- and y-axes above, there is no guarantee that they be orthogonal. We construct them from the first two columns of the matrix T and so should find that the dot-product of $(t_{11}, t_{21}, f t_{31})^T$ and $(t_{12}, t_{22}, f t_{32})^T$ be zero (eq. 8). There is, however, nothing in the method used to determine the matrix T

(see Appendix A) that enforces this constraint. The same goes for the non-linear constraint on the magnitude of the vectors derived from the first two columns of T by multiplying the third component by f (eq. 9).

The fact that the directions of the x- and y-axes of the object do not end up being orthogonal presents a practical problem for this method. One can attempt to get around this problem by approximation. That is, one finds a real transformation that is "near" the physically unrealizable transformation represented by the matrix T. In the case of non-orthogonal axis, it is possible to make adjustments to the two axes in order to make them orthogonal. The smallest adjustments that makes the vectors orthogonal are those obtained by a suitable choice of α in

$$\mathbf{x}' = \mathbf{x} + \alpha\mathbf{y} \quad \text{and} \quad \mathbf{y}' = \mathbf{y} + \alpha\mathbf{x}. \tag{14}$$

Finding the appropriate multiple involves solving

$$\alpha^2(\mathbf{x} \cdot \mathbf{y}) + \alpha(\mathbf{x} \cdot \mathbf{x} + \mathbf{y} \cdot \mathbf{y}) + \mathbf{x} \cdot \mathbf{y} = 0 \tag{15}$$

(which happens to be numerically badly conditioned because $\mathbf{x} \cdot \mathbf{y}$ is small). The need for this "work around" clearly illustrates one disadavantage of using the projective geometry approach to solving this problem. The same can be said about the second non-linear constraint on the elements of T.

Of even more serious practical concern is the related disadvantage of higher sensitivity to noise. If the positions of corresponding object and image points were known exactly, then the matrix T would satisfy the two non-linear constraints and the above analysis would yield the correct solution. In practice, however, there are always small errors in measurement of image positions. Using the corrupted measurements of four points, we can still always find a matrix T that *exactly* maps the four object points into the four image measurements.

However, there will in general be no rotation and translation plus perspective projection that does this. When we follow the procedure above to determine the real attitude and position of the object plane we find a perspective transformation that will not map the four object points into the image points exactly (because no such transformation exists). Perhaps more importantly, the rotation and translation estimated this way are more seriously affected by measurement error than they would be if the true perspective projection had been modeled, as we see next.

Minimizing the Image Projection Error

An alternative to the projective geometry approach is one based on true perspective projection. This leads to non-linear equations, but correctly models a real camera. We are given the coordinates of a set of object points and the corresponding image coordinates. The task is to find the rotation R and translation $\mathbf{t}$

of the object that allows perspective projection to image the object points where they are actually observed.

With more than three correspondences between object points and image points, there will generally be no rotation R and translation $\mathbf{t}$ that map the object points exactly into the corresponding image points. This leads to an error minimization problem.

As a measure of the "mismatch" we can use the sum of the squares of the differences between predicted positions of images of object points and the corresponding measured positions. Such a method is optimal if the measurement errors in image position are independent and have a Gaussian distribution, and if we assume that we know the coordinates of the object points accurately.

We would like to find a transformation that makes the total error as small as possible. The *ad hoc* method based on homogeneous transform described above does not yield this minimum. That is, in general a rigid body transformation can be found that has lower overall error in predicting where object points will be imaged — typically a lot lower.

One can apply the Newton-Raphson method to find the rotation and translation that minimizes the sum of squares of the total image projection errors. The first and second derivatives of the error with respect to the unknown parameters can be estimated numerically. This brings up the question of what parameters to use to describe the translation and rotation. Naturally, the nine elements of the orthonormal rotation matrix cannot be used as parameters since they are not independent. In our view, Euler angles are also unsatisfactory because of the singularities that arise [Horn 87, 90, 91]. So are several other commonly used representation for rotation.

As demonstrated elsewhere [Horn 87, 90, 91], unit quaternions have several advantages when used to representation rotations. The unit quaternion notation is redundant, since it depends on four numbers to represent rotation. However, the single constraint that a quaternion be a unit quaternion is much easier to handle than the six constraints required to ensure that a matrix is orthonormal (and the one condition that ensures it represents a rotation instead of a reflection).

For purposes of the numerical optimization using the Newton-Raphson method, it is convenient to pick the three components of the "vector" part $\mathbf{q}$ of the quaternion as the parameters to vary — computing the "scalar" part q_0 always to maintain the unit quaternion constraint $\mathring{\mathrm{q}} \cdot \mathring{\mathrm{q}} = 1$, where $\mathring{\mathrm{q}} = (q_0, \mathbf{q})$.

The gradient $\mathbf{g}$ of the total error with respect to the parameters $\mathbf{p}$ has six components (three for rotation parameters and three for translation parameters), The Hessian H (matrix of second derivatives of the total error) is a 6×6 matrix. The new guess for the parameters is given by

$$\mathbf{p}' = \mathbf{p} - H^{-1}\mathbf{g}. \tag{16}$$

With a reasonable first guess this process converges to machine precision with a very small number of steps. A reasonable first guess can be obtained, for example, using the projective geometry method described above. An alternative is to sample the space of rotations systematically (using e.g. the rotation group of the icosahedron) or sample the rotation space randomnly for starting guesses.

Monte Carlo simulation experiments

Simulation experiments show that the solution based on minimizing the total image error can reduce the error by over an order of magnitude in typical situations compared to the approximation obtained from the projective geometry based method discussed earlier (details depend on the parameters of the camera and the object distance).

One may question whether minimizing the error in predicting where object points appear in the image improves the accuracy with which one can determine the object coordinate system. More significant from the point of view of some applications in robotics or industrial automation is the error in the rotation matrix R which has all the information on the orientation of the object. Similarly one may want to know what the error in the translation is.

One issue is how to summarize the error in the recovered object system attitude. Let the true rotation of the object coordinate system (with respect to the camera coordinate system) be R_1 and the estimated rotation be R_2. Then a good measure of dissimilarity is how far one would have to rotate one to bring it into alignment with the other. That is, the 'size' of the rotation given by $dR = R_1 R_2^T$. Any rotation — including dR — can be expressed as a rotation about some axis $\boldsymbol{\omega}$ through an angle $\delta\theta$. The axis and angle can be determined from the rotation matrix — most easily by first recovering the unit quaternion (Note also that $\text{Trace}(dR) = 1 + 2\cos\delta\theta$). A simple measure of the dissimilarly in attitude is the angle $\delta\theta$ computed from dR.

Simulation experiments show that the solution based on minimizing the image error in perspective projection (as described above) reduces the error in orientation (defined as in the last paragraph) by over an order of magnitude in typical situations. The same goes for the error in translation.

In a particular simulation, we chose an object plane with four points at the corners of a square with 168 mm edges, located 1600 mm from a camera with focal length 18 mm. The image pixels are 8.4 μm $\times$ 8.4 μm. The object plane is rotated such that its normal makes an angle of 60 degrees with the optical axis. It is assumed that the error in image position determination is 1/5th of a pixel The average error in object attitude using the projective geometry method is 4.3

degrees, while the average error using the true perspective projection method is 0.18 degrees.

Note also that the projective geometry method needs a minimum of four points, while the perspective projection method can work with a minimum of three (non-colinear) points. If there are three points, we can use Church's method (see Appendix B) to solve the exterior orientation problem [Church 45, Fischler & Bolles 80]. Even with *three* points, the method has an error of only 0.23 degrees in attitude — still very much less than the error found when using the projective geometry based method with *four* points.

When $N > 4$ points are used, the attitude error is reduced in both cases, but the overwhelming advantage of correctly modeling image projection is retained. The key point is that under all conditions tested, the average error in the projective geometry based solution was over an order of magnitude larger than that in the perspective projection based method.

Summary

(i) Methods based on projective geometry are fundamentally different from methods based on perspective projection;
(ii) Methods based on projective geometry yield a transformation matrix T that in general does not correspond to a physical imaging situation — that is, a rotation, translation and perspective projection;
(iii) Optimization methods based on the real physical imaging equations (true perspective projection) produce considerably more accurate results.

References

1. Church, E. (1945) "Revised Geometry of the Aerial Photograph," Bulletin of Aerial Photogrammetry, No. 15, Syracuse University.

2. Dijkstra, E.W. (1968) "Go To Statement Considered Harmful," *Communications of the ACM*, Vol. 11, No. 3, pp. 147–148, March.

3. Fischler, M.A. and R.C. Bolles (1980) "Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography," DARPA Image Understanding Workshop, April 1980, pp. 71—88.

4. Hartley, R.I. (1997a) "Kruppa's Equations Derived from the Fundamental Matrix," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 2, February.

5. Hartley, R.I. (1997b) "In Defense of the Eight-Point Algorithm," *IEEE Transations of Pattern Analysis and Machine Intelligence*, Vol. 19, No. 6, June.

6. Horn, B.K.P. (1986) *Robot Vision*, MIT Press and McGraw-Hill

7. Horn, B.K.P. (1987) "Closed Form Solution of Absolute Orientation using Unit Quaternions," *Journal of the Optical Society A*, Vol. 4, No. 4, pp. 629–642, April.

8. Horn, B.K.P. (1990) "Relative Orientation," *International Journal of Computer Vision*, Vol. 4, No. 1, pp. 59–78, January.

9. Horn, B.K.P. (1991) "Relative Orientation Revisited," *Journal of the Optical Society of America, A*, Vol. 8, pp. 1630–1638. October.

10. Longuett-Higgins, H.C. (1981) "A Computer Program to Reconstruct a Scene from Two Projections," *Nature*, Vol. 293, pp. 133–135, September

11. Quan, L. and Z. Lan (1999) "Linear N-Point Camera Pose Determination," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 21, No. 8, August.

12. Slama, C.C. (ed.) (1980) *Manuals of Photogrammetry*, 4th edition, American Society of Photogrammetry

13. Wolf, P.R. (1974) *Elements of Photogrammetry*, McGraw-Hill

14. Wylie, C.R. Jr. (1970) *Introduction to Projective Geometry*, McGraw-Hill

Appendix A: Finding the matrix T from correspondences

From each correspondence between a point in the object plane at $(x_i, y_i, 1)^T$ and a point in the image plane at $(ku_i, kv_i, k)^T$ we get two equations by eliminating k (section 13.7 in [Horn 86]):

$$\begin{aligned} x_i t_{11} + y_i t_{12} + t_{13} - x_i u_i t_{31} - y_i u_i t_{32} - u_i t_{33} = 0 \\ x_i t_{21} + y_i t_{22} + t_{23} - x_i v_i t_{31} - y_i v_i t_{32} - v_i t_{33} = 0 \end{aligned} \tag{17}$$

Four correspondences yield a system of eight homogeneous equations:

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1u_1 & -y_1u_1 & -u_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2u_2 & -y_2u_2 & -u_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3u_3 & -y_3u_3 & -u_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4u_4 & -y_4u_4 & -u_4 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1v_1 & -y_1v_1 & -v_1 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2v_2 & -y_2v_2 & -v_2 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3v_3 & -y_3v_3 & -v_3 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4v_4 & -y_4v_4 & -v_4 \end{pmatrix} \begin{pmatrix} t_{11} \\ t_{12} \\ t_{13} \\ t_{21} \\ t_{22} \\ t_{23} \\ t_{31} \\ t_{32} \\ t_{33} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \tag{20}$$

While the matrix T has nine elements, there are only eight degrees of freedom since any non-zero multiple of T describes the same mapping between the two planes. So we can arbitrarily pick a value for one element of T, say $t_{33} = 1$. This moves the last column of the matrix above to the right-hand side (with change of sign) and leaves us with eight (non-homogeneous) equations in eight unknown.

The matrix has a distinctive structure, with two 4×3 blocks of zeros, which can be exploited to partition the matrix.

The transformation matrix T can also be found using the fundamental theorem of projective geometry — which states that four 3-vectors must be linearly dependent. This reduces the computation to solving three sets of three equations instead of a single set of eight equations in eight unknowns [Appendix B of Fischler & Bolles 80].

If there are $N > 4$ correspondences, we end up with more equations than unknowns. We can find a solution that minimizes the sum of squares of errors in these equations using the pseudo-inverse of the $2n \times 8$ coefficient matrix. Note however, that what is minimized is *not* the sum of squares of errors in image position, but the error in homogeneous coordinates u_i and v_i, that is, some arbitrary multiples of image position — different multiples for different i. Which is obviously not the best "error" to minimize.

Appendix B: Church's Method for Three Correspondences

Exterior orientation can be recovered using just three correspondences (the projective geometry method requires four correspondences because it ignores two non-linear constraints that arise in true perspective projection). One way to solve the problem using three correspondences is the "tripod method" of Church [Church 45, Appendix A in Fischler & Bolles 80]. We first solve for the lengths of the three legs r_1, r_2, r_3 measured from the center of projection to the three points on the object using the equations:

$$\begin{aligned} r_1^2 + r_2^2 - 2r_1 r_2 \cos\theta_{12} - d_{12}^2 &= 0 \\ r_2^2 + r_3^2 - 2r_2 r_3 \cos\theta_{23} - d_{23}^2 &= 0 \\ r_3^2 + r_1^2 - 2r_3 r_1 \cos\theta_{31} - d_{31}^2 &= 0 \end{aligned} \tag{21}$$

Here d_{12}, d_{23}, and d_{31} are the known pairwise distances between the three points on the object, while θ_{12}, θ_{23}, and θ_{31} are the known angles between rays from the center of projection to these points. The cosines of these angles can be readily found by taking dot-products of vectors from the center of projection $(0, 0, 0)^T$ to the corresponding points in the image plane ($z = f$) after normalizing their lengths.

The three unknown lengths r_1, r_2, r_3 can be recovered using a simple iterative scheme to solve the non-linear equations above. Let an error vector $\mathbf{e} = (e_1, e_2, e_3)^T$ be defined where e_1, e_2 and e_3 are the left hand sides of the three equations above. Let D be the 3×3 matrix of first derivatives of components of $\mathbf{e}$ with respect to the unkown r_1, r_2, and r_3. Then given a guess $\mathbf{r} = (r_1, r_2, r_3)^T$, we compute a new value using

$$\mathbf{r}' = \mathbf{r} - D^{-1}\mathbf{e} \tag{22}$$

This process converges quickly given reasonable initial guesses. However, up to four different positive solutions may exist.

Once the lengths of the 'legs' are known, the position of the three object points in the camera coordinate system can be found simply by multiplying unit vectors in the directions to the corresponding image points by r_1, r_2, and r_3.

At this point the coordinates of three points are known both in the object coordinate system and in the camera coordinate system. This makes it possible to recover the rotation and translation that maps from the object coordinate system to the camera coordinate system. We have

$$\mathbf{c}_i = R\mathbf{o}_i + \mathbf{t} \tag{23}$$

for $i = 1$, 2, and 3. We can eliminate the translation $\mathbf{t}$ by subtracting coordinates to obtain

$$\begin{aligned} (\mathbf{c}_2 - \mathbf{c}_1) &= R(\mathbf{o}_2 - \mathbf{o}_1) \\ (\mathbf{c}_3 - \mathbf{c}_2) &= R(\mathbf{o}_3 - \mathbf{o}_2) \end{aligned} \tag{24}$$

We also have

$$\big((\mathbf{c}_3 - \mathbf{c}_2) \times (\mathbf{c}_2 - \mathbf{c}_1)\big) = R\big((\mathbf{o}_3 - \mathbf{o}_2) \times (\mathbf{o}_2 - \mathbf{o}_1)\big) \tag{25}$$

We can combine these equalities into one equation $M_c = RM_o$ where

$$\begin{aligned} M_c &= \big(\mathbf{c}_2 - \mathbf{c}_1 \quad \mathbf{c}_3 - \mathbf{c}_2 \quad (\mathbf{c}_3 - \mathbf{c}_2) \times (\mathbf{c}_2 - \mathbf{c}_1)\big) \\ M_o &= \big(\mathbf{o}_2 - \mathbf{o}_1 \quad \mathbf{o}_3 - \mathbf{o}_2 \quad (\mathbf{o}_3 - \mathbf{o}_2) \times (\mathbf{o}_2 - \mathbf{o}_1)\big) \end{aligned} \tag{26}$$

and so $R = M_c M_o^{-1}$. Finally we can recover the translation using

$$\mathbf{t} = \overline{\mathbf{c}} - R\overline{\mathbf{o}} \tag{27}$$

where $\overline{\mathbf{o}}$ and $\overline{\mathbf{c}}$ are the averages of the three coordinates in the object and the camera coordinate system respectively.