Chapter 1

Imaging Geometry

1.1 Introduction

In[1] the first chapter of this book we will learn about imaging geometry. We will discuss various transformations which will be used in many problems in Computer Vision. These transformations are also used in Computer Graphics. We will learn about coordinate systems and how to relate one coordinate system with another coordinate system. We will discuss translation, rotation, scaling, perspective transformation, camera modeling, and camera calibration. These transformations will help us to know the location and orientation of the object after it is translated by some amount, rotated around a given axis, and scaled by some factor. We will derive a matrix for each of these transformations.

The world we live in is three dimensional, that is any point in space can be specified by three coordinates (X, Y, Z). The image is a 2-D plane, so we need two coordinates (x, y) to represent a point in the image. One dimension is lost in the projection process. One of the important goals in Computer Vision is to recover this lost dimension. The methods for recovering 3-D information from 2-D images are called *shape from 'X'*, where 'X' can be stereo, motion, shading, texture, etc. We will discuss these methods in great detail in this book. However, in this chapter we will focus on the derivation of the camera matrix and camera calibration.

1.2 Translation and Scaling

Consider a point on an object with coordinates $(X1, Y1, Z1)$. Assume that the object is translated by dx, dy, and dz respectively in the X, Y, and Z directions. The new coordinates of the point are given by:

$$X2 = X1 + dx \tag{1.1}$$
$$Y2 = Y1 + dy \tag{1.2}$$
$$Z2 = Z1 + dz \tag{1.3}$$

These equations can be written in matrix form as follows:

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \tag{1.4}$$

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = T \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \tag{1.5}$$

where $T = \begin{bmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$ is called translation matrix. The inverse translation matrix is given by $T^{-1} = \begin{bmatrix} 1 & 0 & 0 & -dx \\ 0 & 1 & 0 & -dy \\ 0 & 0 & 1 & -dz \\ 0 & 0 & 0 & 1 \end{bmatrix}$. You can verify that $TT^{-1} = T^{-1}T = I$, where I is the identity matrix.

Similarly if the object is scaled by Sx, Sy, and Sz respectively in the X, Y, and Z directions, the new coordinates of the point are given by:

$$X2 = X1 \times Sx \tag{1.6}$$

$$Y2 = Y1 \times Sy \tag{1.7}$$

$$Z2 = Z1 \times Sz \tag{1.8}$$

These equations can be written in matrix form as follows:

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix}, \tag{1.9}$$

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = S \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix}, \tag{1.10}$$

$$\tag{1.11}$$

where $S = \begin{bmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ is called the scaling matrix. The inverse of scaling matrix is given by $S^{-1} = \begin{bmatrix} 1/Sx & 0 & 0 & 0 \\ 0 & 1/Sy & 0 & 0 \\ 0 & 0 & 1/Sz & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

1.3 Rotation

Consider a vector whose endpoint is at $(X1, Y1, Z1)$, as shown in Figure 1.1.a. Let R denote the length of vector, and ϕ denote the angle the vector makes with $X-$axis. Assume that the vector is rotated by the angle θ around the $Z-$axis in the counterclockwise direction. The new coordinates $(X2, Y2, Z2)$ of the point after rotation can be computed from the old coordinates and the angle of rotation. Consider a triangle in Figure 1.1.a whose side makes an angle ϕ with the $X-$axis. Using the standard trigonometric relations we can write:

$$X1 = R\cos\phi \tag{1.12}$$
$$Y1 = R\sin\phi. \tag{1.13}$$

Similarly, for the triangle in Figure 1.1.a whose side makes an angle $\theta + \phi$ with the $X-$axis we can write:

$$X2 = R\cos(\theta+\phi) = R\cos\theta\cos\phi - R\sin\theta\sin\phi \tag{1.14}$$
$$Y2 = R\sin(\theta+\phi) = R\sin\theta\cos\phi + R\cos\theta\sin\phi. \tag{1.15}$$

Substituting first two equations into the last two equations, we get:

$$X2 = X1\cos\theta - Y1\sin\theta \tag{1.16}$$
$$Y2 = X1\sin\theta + Y1\cos\theta. \tag{1.17}$$

These equations can also be written in matrix form as follows:

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \tag{1.18}$$

$$\begin{bmatrix} X2 \\ Y2 \\ Z2 \\ 1 \end{bmatrix} = R_\theta^Z \begin{bmatrix} X1 \\ Y1 \\ Z1 \\ 1 \end{bmatrix} \tag{1.19}$$

where $R_\theta^Z = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$ is the rotation matrix. We will use a superscript to denote the axis of rotation, a subscript to denote the angle, and we will assume rotation in the counterclockwise direction. The inverse rotation matrix is given by $(R_\theta^Z)^{-1} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$. The inverse rotation by θ is equivalent to the rotation by $-\theta$ around the same axis. Therefore, $(R_\theta^Z)^{-1} = R_{-\theta}^Z$. Moreover, the rotation matrix is an orthonormal matrix, that is, $(R_\theta^Z)^T R_\theta^Z = R_\theta^Z (R_\theta^Z)^T = I$. Therefore, the inverse of a rotation matrix is just its transpose.

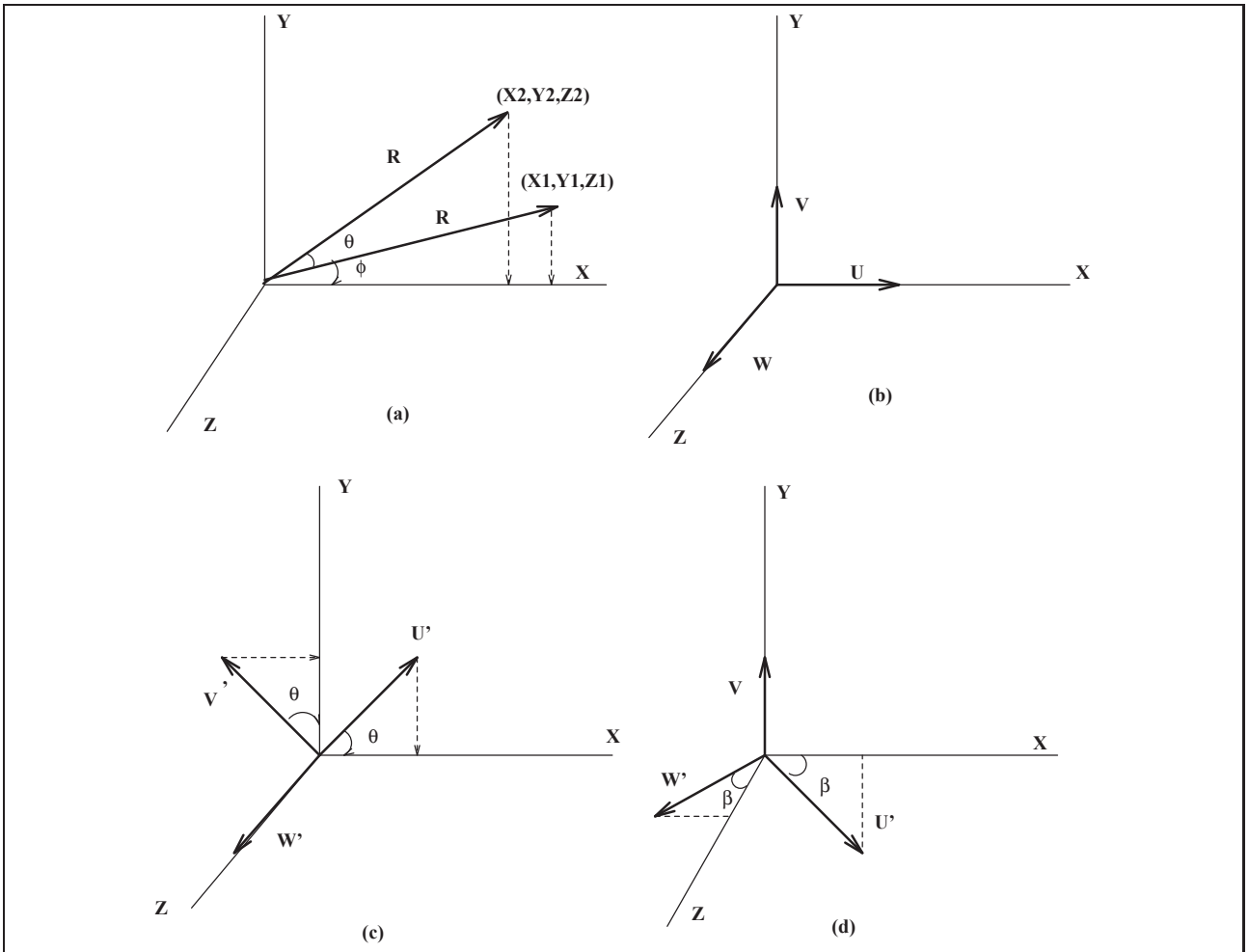

Figure 1.1: Rotation. (a) Rotation around Z axis. (b) Two coordinate systems, UVW superposed on XYZ. (c) UVW rotated by θ around Z axis in the counterclockwise direction. (d)UVW rotated by β around Y axis in the clockwise direction.

An alternate way to derive the rotation matrix is to consider two coordinate systems (X, Y, Z) and (U, V, W) as shown in Figure 1.1.b. Assume that U, V, W are the unit vectors. Let us rotate the UVW coordinate system around the Z-axis by an angle θ as shown in Figure 1.1.c. Note that the $W-$axis remains fixed with the $Z-$axis. However, the $U-$ and V-axes are changed. The new coordinates U', V', and W' can be written in terms of the old coordinate system. The U'-vector has two components, one along $X-$axis, and one along Y-axis. Note that it does not have a Z component. The new U' vector is given by $(\cos\theta, \sin\theta, 0)$. Similarly, the V'-vector is given by $(-\sin\theta, \cos\theta, 0)$. Since the W vector did not change, it remains as $(0, 0, 1)$. The rotation matrix can now be formed by putting the coordinates of the U', V', and W' vectors respectively in the first, second and third columns, and $(0, 0, 0, 1)$ in the fourth column. The resultant matrix is exactly the same as the matrix R^Z_θ.

This technique can be applied to derive the rotation matrix around any axis. For instance, the rotation matrix around the $Y-$axis in the clockwise direction is given by (see Figure 1.1.d):

$$R^Y_{-\beta} = \begin{bmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

1.4 Perspective Transform

A simplified model of image formation is shown in Figure 1.2.a. In this so called pinhole camera model, the lens is assumed to be a single point. The world coordinates (X, Y, Z) of a point are transformed to the image coordinates (x, y) under perspective projection. Assume that the origin of the coordinate system is at the lens center (shown by O in Figure 1.2.a), and f is the focal length of the camera, the distance from the lens to the image plane. Then using the two equivalent triangles we can write:

$$\frac{-y}{Y} = \frac{f}{Z}. \tag{1.20}$$

Since the image is always formed upside down, it is customary to use a negative sign in front of the y in the above equation. From the above equation, the y image coordinates are given by:

$$y = -\frac{fY}{Z}. \tag{1.21}$$

Similarly, the $x-$coordinates of the image are given by:

$$x = -\frac{fX}{Z}. \tag{1.22}$$

The above equations represent the perspective transform with the origin at the lens. If the origin is moved to the image, as shown in Figure 1.2.b, the perspective projection is defined by the following equations:

$$x = \frac{fX}{f - Z} \tag{1.23}$$

$$y = \frac{fY}{f - Z}. \tag{1.24}$$

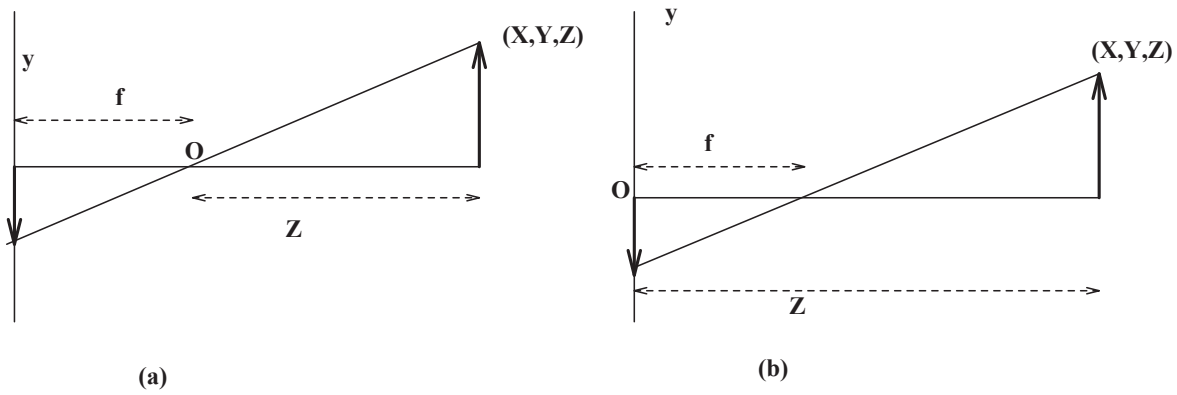

Figure 1.2: Pinhole camera model. (a) Origin at the lens. (b) Origin at the image plane.

It is not possible to derive a perspective matrix similar to the translation, scaling, and rotation matrices due to the nature of equations 1.23 and 1.24. We need to introduce another transformation called *homogeneous transformation*. The homogeneous transformation converts the *Cartesian world coordinates* (X, Y, Z) into the *homogeneous world coordinates* (kX, kY, kZ, k). In this transformation each X, Y, Z coordinate is multiplied by a constant k, and k is appended as the fourth component of the vector. Similarly, the inverse homogeneous transform converts the homogeneous image coordinates $(C_{h_1}, C_{h_2}, C_{h_3}, C_{h_4})$ into the Cartesian image coordinates $(\frac{C_{h_1}}{C_{h_4}}, \frac{C_{h_2}}{C_{h_4}}, \frac{C_{h_3}}{C_{h_4}})$, by dividing each of the three components with the fourth component.

The perspective matrix can now be defined as:

$$P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-1}{f} & 1 \end{bmatrix}.$$

The perspective transform, which relates the homogeneous world coordinates with the homogeneous image coordinates is defined as:

$$\begin{bmatrix} kX \\ kY \\ kZ \\ -\frac{kZ}{f} + k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-1}{f} & 1 \end{bmatrix} \begin{bmatrix} kX \\ kY \\ kZ \\ k \end{bmatrix}$$

We can easily derive the Cartesian image coordinates from the above equation as:

$$x = \frac{fX}{f - Z}, \tag{1.25}$$

$$y = \frac{fY}{f - Z}, \tag{1.26}$$

which are exactly the same equations as equations 1.23 and 1.24. The inverse perspective matrix is given by: $P^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{1}{f} & 1 \end{bmatrix}$.

1.5 Camera Model

The perspective transform relates the world coordinates with the image coordinates when the camera is located at the origin of the world coordinates. However, in real life we need to translate and rotate the camera in order to bring the object of interest in the field of view. Therefore, the complete camera model involves several other transformations besides the perspective transform.

In one simple model, assume that the camera is first at the origin of the world coordinates, is then translated by (X_0, Y_0, Z_0) (matrix G), then rotated around the Z-axis by θ in the counterclockwise direction (matrix $R^Z_{-\theta}$), rotated again around the X-axis by ϕ in the counterclockwise direction(matrix $R^X_{-\phi}$), and then translated by (r_1, r_2, r_3) (matrix C). In this case, the world homogeneous coordinates (W_h) are related to the camera image coordinates (C_h) as follows:

$$C_h = PC\,R^X_{-\phi}R^Z_{-\theta}GW_h \tag{1.27}$$

where $P = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & \frac{-1}{f} & 1 \end{bmatrix}$, $R^Z_{-\theta} = \begin{bmatrix} \cos\theta & \sin\theta & 0 & 0 \\ -\sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$,

$R^X_{-\phi} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & \sin\phi & 0 \\ 0 & -\sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, $G = \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$, $C = \begin{bmatrix} 1 & 0 & 0 & -r_1 \\ 0 & 1 & 0 & -r_2 \\ 0 & 0 & 1 & -r_3 \\ 0 & 0 & 0 & 1 \end{bmatrix}$.

Note that the inverse transforms for the translation and rotations are used here. This is because it is the coordinate system attached to the camera that is being translated and rotated, rather than the object. Now, the Cartesian image coordinates can be computed from equations:

$$x = f\frac{(X - X_0)\cos\theta + (Y - Y_0)\sin\theta - r_1}{-(X - X_0)\sin\theta\sin\phi + (Y - Y_0)\cos\theta\sin\phi - (Z - Z_0)\cos\phi + r_3 + f}, \tag{1.28}$$

$$y = f\frac{-(X - X_0)\sin\theta\cos\phi + (Y - Y_0)\cos\theta\cos\phi + (Z - Z_0)\sin\phi - r_2}{-(X - X_0)\sin\theta\sin\phi + (Y - Y_0)\cos\theta\sin\phi - (Z - Z_0)\cos\phi + r_3 + f}. \tag{1.29}$$

1.6 Camera Calibration

In the previous section we discussed the camera model, which relates the world coordinates with the image coordinates. It was assumed that the camera focal length, rotation angles, and translation displacements are known. An alternative method for computing the camera model is to use the camera as a measuring device to determine the unknowns in the camera model. This process is called camera calibration. In this method, a few points in 3-D with known coordinates and their corresponding image coordinates are used to calibrate the camera.

The projection of 3-D homogeneous coordinates onto the image plane (from equation 1.27) can be summarized as:

$$C_h = AW_h, \tag{1.30}$$

$$\begin{bmatrix} C_{h_1} \\ C_{h_2} \\ C_{h_3} \\ C_{h_4} \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{1.31}$$

where C_h and W_h respectively denote the camera and world homogeneous coordinates, and $A = PCR^X_{-\phi}R^Z_{-\theta}G$, denote the camera matrix. The aim in the camera calibration is to determine the matrix A using the known 3-D points and their corresponding image coordinates. The matrix A is 4×4, which has 16 unknowns. However, C_{h_3} is not meaningful in the image coordinates. Because the image is only two dimensional, the image coordinates can be determined by C_{h_1}, C_{h_2}, C_{h_4}. Therefore, we do not need to determine the third row of the matrix, which is used for C_{h_3}. Now, we are only left with 12 unknowns in the matrix A. We will find out later in this section that we can arbitrary fix one of the 12 unknowns and determine the remaining 11 unknowns.

As we know, the Cartesian image coordinates (x, y) are given by

$$x = \frac{C_{h_1}}{C_{h_4}}, \tag{1.32}$$

$$y = \frac{C_{h_2}}{C_{h_4}}. \tag{1.33}$$

From the equations 1.31-1.33 we get:

$$C_{h_1} = a_{11}X + a_{12}Y + a_{13}Z + a_{14} = C_{h_4}x \tag{1.34}$$

$$C_{h_2} = a_{21}X + a_{22}Y + a_{23}Z + a_{24} = C_{h_4}y \tag{1.35}$$

$$C_{h_4} = a_{41}X + a_{42}Y + a_{43}Z + a_{44} \tag{1.36}$$

substituting C_{h_1}, C_{h_2}, and C_{h_4} in equations 1.32-1.33 and rearranging we get:

$$a_{11}X + a_{12}Y + a_{13}Z + a_{14} - a_{41}Xx - xa_{42}Y - xa_{43}Z - xa_{44} = 0 \tag{1.37}$$

$$a_{21}X + a_{22}Y + a_{23}Z + a_{24} - a_{41}Xy - ya_{42}Y - ya_{43}Z - ya_{44} = 0. \tag{1.38}$$

In the above equations there are 12 unknowns $(a_{11}, \ldots\ldots, a_{44})$, and five knowns (X, Y, Z, x, y). One point in 3-D with coordinates (X, Y, Z), and corresponding image coordinates (x, y) gives two such equations with 12 unknowns. n such points will give $2n$ equations, which can be solved for 12 unknowns.

$$a_{11}X_1 + a_{12}Y_1 + a_{13}Z_1 + a_{14} - x_1a_{41}X_1 - x_1a_{42}Y_1 - x_1a_{43}Z_1 - x_1a_{44} = 0 \tag{1.39}$$

$$a_{11}X_2 + a_{12}Y_2 + a_{13}Z_2 + a_{14} - x_2a_{41}X_2 - x_2a_{42}Y_2 - x_2a_{43}Z_2 - x_2a_{44} = 0 \tag{1.40}$$

$$\vdots$$

$$a_{11}X_n + a_{12}Y_n + a_{13}Z_n + a_{14} - x_na_{41}X_n - x_na_{42}Y_n - x_na_{43}Z_n - x_na_{44} = 0 \tag{1.41}$$

$$a_{21}X_1 + a_{22}Y_1 + a_{23}Z_1 + a_{24} - y_1a_{41}X_1 - y_1a_{42}Y_1 - y_1a_{43}Z_1 - y_1a_{44} = 0 \tag{1.42}$$

$$a_{21}X_2 + a_{22}Y_2 + a_{23}Z_2 + a_{24} - y_2a_{41}X_2 - y_2a_{42}Y_2 - y_2a_{43}Z_2 - y_2a_{44} = 0 \tag{1.43}$$

$$\vdots$$

$$a_{21}X_n + a_{22}Y_n + a_{23}Z_n + a_{24} - y_na_{41}X_n - y_na_{42}Y_n - y_na_{43}Z_n - y_na_{44} = 0 \tag{1.44}$$

In these equations the subscripts on (X, Y, Z) and (x, y) denote the point number. The above system can be written in matrix form as follows:

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 & -x_1Z_1 & -x_1 \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 & -x_2Z_2 & -x_2 \\ \vdots & & & & & & & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -x_nX_n & -x_nY_n & -x_nZ_n & -x_n \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1X_1 & -y_1Y_1 & -y_1Z_1 & -y_1 \\ 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 & -y_2X_2 & -y_2Y_2 & -y_2Z_2 & -y_2 \\ \vdots & & & & & & & & & & & \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -y_nX_n & -y_nY_n & -y_nZ_n & -y_n \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{14} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{24} \\ a_{41} \\ a_{42} \\ a_{43} \\ a_{44} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \quad (1.45)$$

or

$$CP = 0, \quad (1.46)$$

where C is a $2n \times 12$ matrix, P a is 12×1 vector, and 0 is also a 12×1 vector. This is a homogeneous system which has multiple solutions. Therefore, we can arbitrarily pick one of the unknowns, and determine the remaining unknowns. Let $a_{44} = 1$, then the above system can be rewritten as:

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -X_1x_1 & -x_1Y_1 & -x_1Z_1 \\ X_2 & Y_2 & Z_2 & 1 & 0 & 0 & 0 & 0 & -X_2x_2 & -x_2Y_2 & -x_2Z_2 \\ \vdots & & & & & & & & & & \\ X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -X_nx_n & -x_nY_n & -x_nZ_n \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -X_1y_1 & -y_1Y_1 & -y_1Z_1 \\ 0 & 0 & 0 & 0 & X_2 & Y_2 & Z_2 & 1 & -X_2y_2 & -y_2Y_2 & -y_2Z_2 \\ \vdots & & & & & & & & & & \\ 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -X_ny_n & -y_nY_n & -y_nZ_n \end{bmatrix} \begin{bmatrix} a_{11} \\ a_{12} \\ a_{13} \\ a_{14} \\ a_{21} \\ a_{22} \\ a_{23} \\ a_{24} \\ a_{41} \\ a_{42} \\ a_{43} \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} \quad (1.47)$$

$$DQ = R \quad (1.48)$$

In the above system, the matrix D and vector R are known. We can determine the 11 unknowns in vector Q by using the pseudo inverse, if at least 5.5 or more 3-D points and their corresponding image coordinates are known. Multiplying the above equation by D^T on both sides and rearranging we get:

$$D^T DQ = D^T R \quad (1.49)$$

$$Q = (D^T D)^{-1} D^T R. \quad (1.50)$$

Once Q is determined, the matrix A is defined, which completes the camera calibration.

1.7 Recovering Camera Parameters

An interesting inverse problem deals with the recovery of camera parameters from the camera matrix. For example, given a photograph taken by an unknown camera from an unknown

location which, has been cropped and/or enlarged, how can we determine the camera's position and orientation, and determine the extent to which the picture was cropped or enlarged? Applications of this occur in several areas. For instance, in autonomous navigation, a cruise missile could obtain the camera transformation matrix from a terrain model stored on-board and then compute the camera parameters that define the vehicle's location and heading. Also, a stationary camera viewing a robot arm workspace could determine the position of the arm. The markings of several points on part of the manipulator would allow their easy extraction from an image and provide ground truth for the camera calibration. The camera parameters can then be derived from the camera transformation matrix, and the location and orientation of the manipulator can be obtained relative to the stationary camera.

1.7.1 Camera Location

Consider a 3-D point **X1** with coordinates $(X1, Y1, Z1, 1)$ (see Figure 1.4). If camera matrix A is known we can determine the image coordinates of a point by using equation 1.30, that is $\mathbf{U1} = A\mathbf{X1}$, where $\mathbf{U1} = (C_{h_1}, C_{h_2}, C_{h_3}, C_{h_4})$. If we multiply **U1** with A^{-1}, we should get the original **X1** back. Let us set $C_{h_3} = 0$, then $\mathbf{U1'} = (C_{h_1}, C_{h_2}, 0, C_{h_4})$. Now, backproject $\mathbf{U1'}$ by $A^{-1}\mathbf{U1'} = \mathbf{X11}$. We will get a new 3-D point **X11**, which should lie on the same line connecting **X1** and the center of projection L (see Figure 1.4). Similarly, we can take another 3-D point $\mathbf{X2} = (X2, Y2, Z2, 1)$, and generate **X21**, which will lie on the line passing through **X2** and the center of projection. Now, we can easily determine L by the intersection of these two lines.

1.7.2 Camera Orientation

The orientation of a camera is defined as the orientation of the image plane. It is clear from Figure 1.5 that as the object moves closer to the lens its image moves farther away from the image center along the $Y-$axis. When the object is at the lens, the image will be formed at infinity. The only way the image of a finite world point can be formed at infinity is if the fourth component of the homogeneous image coordinates is zero. From equation 1.30 we have

$$\begin{bmatrix} C_{h_1} \\ C_{h_2} \\ C_{h_3} \\ 0 \end{bmatrix} = A \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tag{1.51}$$

which gives the following constraint on the camera orientation:

$$a_{41}X + a_{42}Y + a_{43}Z + a_{44} = 0 \tag{1.52}$$

This is the equation of a plane passing through lens L parallel to the image plane. From this equation it is clear that (a_{41}, a_{42}, a_{43}) is the normal to the image plane, and parallel to the camera.

The orientation in terms of clockwise rotation, θ, around the $X-$axis, followed by the clockwise rotation, ϕ, around the $Y-$axis(as shown in figure 1.3), is given by:

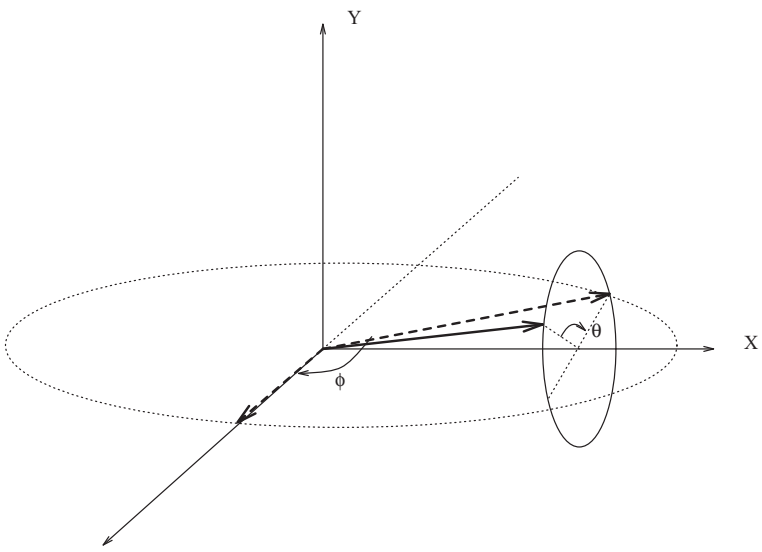

Figure 1.3: Camera Orientation.

$$\theta = \arctan \frac{a_{42}}{-a_{43}}, \tag{1.53}$$

$$\phi = \arcsin \frac{-a_{41}}{\sqrt{{a_{41}}^2 + {a_{42}}^2 + {a_{43}}^2}}. \tag{1.54}$$

It is also possible to compute the focal length, scaling and other camera parameters from the camera matrix. However, the derivations are fairly involved, therefore we do not cover them in this introductory book. The interested reader should look at [23] in the bibliography for the details.

1.8 Rodrigue's Formula

Assume that we want to rotate vector v around vector n by angle θ. We can decompose vector v into two parts, one collinear to n, and other perpendicular to n as follows, (see Figure 1.6):

$$v = (v.n)n + (v - (v.n)n). \tag{1.55}$$

The first component is invariant under rotation, whereas the second component, undergoes a planar rotation through angle θ which may be written:

$$\cos\theta(v - (v.n)n) + \sin\theta(n \times (v - (v.n)n)). \tag{1.56}$$

This gives:

$$v' = \cos\theta\ v + \sin\theta\ n \times v + (1 - \cos\theta)(v.n)n, \tag{1.57}$$

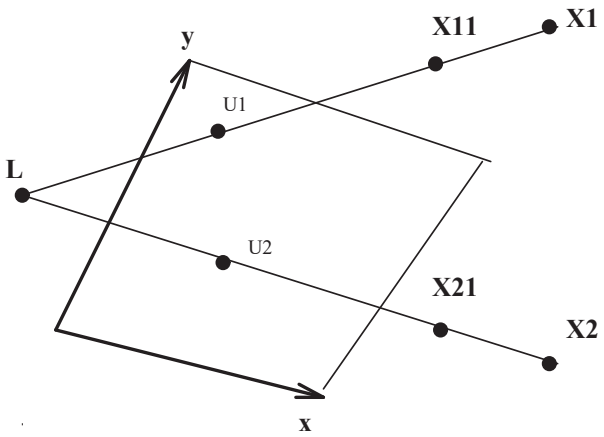

Figure 1.4: Determining the location of camera. The image of 3-D points $X1$ and $X11$ is formed at $U1$, and image of points $X2$ and $X21$ is formed at $U2$. The intersection of a line passing through $X1$ and $X11$ with the line passing through $X2$ and $X21$ is L, which is the location of camera.

or

$$v' = v + \sin\theta\ n \times v + (1 - \cos\theta) n \times (n \times v), \tag{1.58}$$

where $n \times (n \times v) = (v.n)n - v$. Now, from the above equation the rotation matrix R^n_θ can be written as:

$$R^n_\theta = I + \sin\theta\ X(n) + (1 - \cos\theta) X(n)^2, \tag{1.59}$$

where $X(n)$ is operator which is the vector product by n, that is the antisymmetric matrix formed from the components of n as follows:

$$X(n) = \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix}. \tag{1.60}$$

Therefore to represent the rotation, we need vector n, and the angle θ. We can further simplify this, and represent the rotation by only a vector, r, such that:

$$r = ||r|| \frac{r}{||r||} = \theta\ n. \tag{1.61}$$

The magnitude of r is the amount of rotation, and the direction of of r is the axis of rotation. Finally, the rotation matrix can be written as:

$$R^r = R^r_{||r||} = I + \sin\theta \frac{X(r)}{||r||} + (1 - \cos\theta) \frac{X(r)^2}{||r||^2}, \tag{1.62}$$

where $X(r)$ is given by:

$$X(r) = \begin{bmatrix} 0 & -r_z & r_y \\ r_z & 0 & -r_x \\ -r_y & r_x & 0 \end{bmatrix}. \tag{1.63}$$

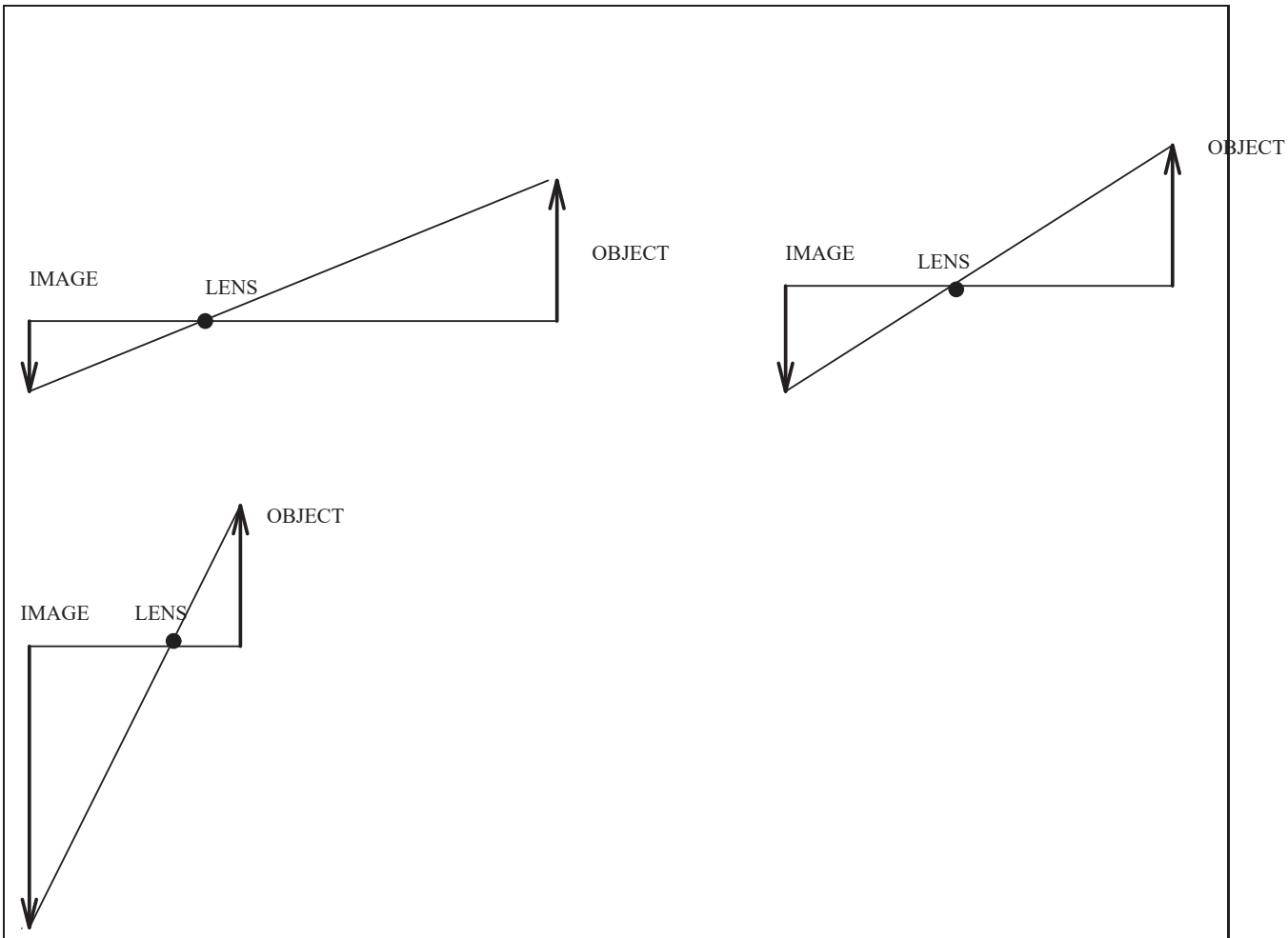

Figure 1.5: Determining the orientation of camera. As the object moves closer to the lens its image moves farther away from the image center along the $Y-$axis. When the object is at the lens, the image will be formed at infinity.

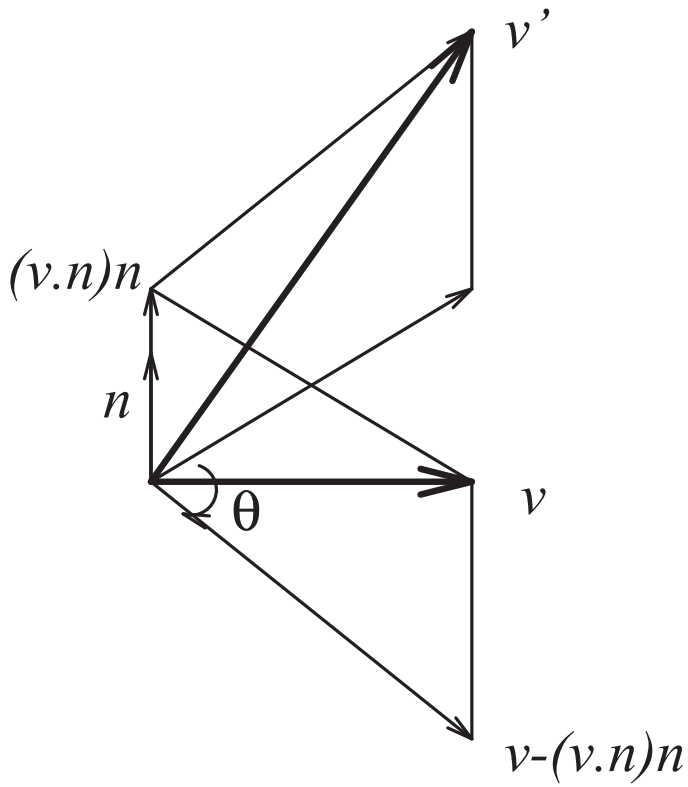

Figure 1.6: Rotation of vector r around vector n by angle θ.

1.9 Quaternions

A quaternion is a pair $q = (s, w)$, where s is a scalar and is commonly called real part of q, and w is a 3-D vector, and is called imaginary part of q. Two quaternions can be added like any 4-D vectors. However, multiplication of two quaternions is given by:

$$(s, w) * (s', w') = (ss' - w.w', w \times w' + sw' + s'w), \tag{1.64}$$

where "$\times$" and "." denote the usual vector and dot products. The conjugate and the norm of a quaternion are defined as:

$$\bar{q} = (s, -w), \tag{1.65}$$

$$|q|^2 = \bar{q} * q = ||w||^2 + s^2. \tag{1.66}$$

For any rotation R of a 3-D vector, p, there exist two opposite quaternions q and $-q$ which satisfy $Rp = q * p * \bar{q}$ (p is converted to a quaternion by assuming zero real part, and similarly the real part of the resultant quaternion on the right is set to zero to match with the rotated vector on the left side). The quaternion q is computed from the unit direction vector n of the axis of rotation and its angle θ by the formula:

$$q = (\cos(\frac{\theta}{2}), \sin(\frac{\theta}{2})n). \tag{1.67}$$

Similarly, to each pair of unit quaternions $(q, -q)$ there is a associated unique rotation matrix

R,

$$R = \begin{bmatrix} a^2+b^2-c^2-d^2 & 2(bc-ad) & 2(bd+ac) \\ 2(bc+ad) & a^2+c^2-b^2-d^2 & 2(cd-ab) \\ 2(bd-ac) & 2(cd+ab) & a^2+d^2-b^2-c^2 \end{bmatrix} \quad (1.68)$$

where $q = (a, b, c, d)$.

1.10 Pose Estimation

In pose estimation, the problem is to determine the orientation and position of an object (camera) which would result in the projection of a given set of three-dimensional points into a given set of image points. One of the important applications of pose estimation is in the model based object recognition. In 2-D to 3-D model based object recognition a 3-D model of the object, and its 2-D projection from a particular vantage point of the camera is given, the aim is to determine three rotations and three translations relative to some base coordinate system.

Let (X', Y', Z') be the 3-D model coordinates of a point, and (x', y') be the projected image coordinates. Assume that the object (camera) is rotated first by (ϕ_X, ϕ_Y, ϕ_Z), that maps (X', Y', Z') to (X, Y, Z). Next, the object is translated by (T_X, T_Y, T_Z), and finally the image is produced using perspective transformation using lens centered coordinate system, and taking positive value of f. The image coordinates after perspective projection are given by:

$$(x', y') = (\frac{f(X+T_X)}{Z+T_Z}, \frac{f(Y+T_Y)}{Z+T_Z}). \quad (1.69)$$

In order to simplify these equations, instead of using translation (T_X, T_Y), we will use (D_X, D_Y), which are the projected image displacement in x and y direction, and D_Z is 3-D translation in Z direction. Now, the image coordinates are given by:

$$(x', y') = (\frac{fX}{Z+D_Z} + D_X, \frac{fY}{Z+D_Z} + D_Y) = (fXc + D_X, fYc + D_Y), \quad (1.70)$$

where $c = \frac{1}{Z+D_Z}$.

The pose estimation can be formulated as an optimization problem which minimizes the error between the model and image coordinates [12]. The error between the components of model and the image can be expressed in terms of small changes in the six unknowns $(D_X, D_Y, D_Z, \phi_X, \phi_Y, \phi_Z)$, and their derivatives with respect to the image coordinates. Since errors are known, the small changes in the parameters can iteratively be computed from these equations, and the next estimate can be obtained by adding the changes to the previous estimate. The equation for error, $E_{x'}$, in x' image coordinate as the sum of the products of its partial derivatives times the error correction values (using the first order Taylor series approximation) can be written as:

$$E_{x'} = \frac{\partial x'}{\partial D_X}\Delta D_X + \frac{\partial x'}{\partial D_Y}\Delta D_Y + \frac{\partial x'}{\partial D_Z}\Delta D_Z + \frac{\partial x'}{\partial \phi_X}\Delta\phi_X + \frac{\partial x'}{\partial \phi_Y}\Delta\phi_Y + \frac{\partial x'}{\partial \phi_Z}\Delta\phi_Z \quad (1.71)$$

The partial derivatives in the above equation can be easily computed from equation 1.70. For example, $\frac{\partial x'}{\partial D_X} = 1$, and $\frac{\partial x'}{\partial \phi_Y} = f\frac{\partial X}{\partial \phi_Y}\frac{1}{Z+D_Z} - \frac{fX}{(Z+D_Z)^2}\frac{\partial Z}{\partial \phi_Y}$. We can compute $\frac{\partial X}{\partial \phi_Y}$ and $\frac{\partial Z}{\partial \phi_Y}$ from the rotation matrix, R^Y_ϕ. As we know if (X', Y', Z') is rotated around Y-axis by angle ϕ_Y, the new coordinates (X, Y, Z) are given by:

$$X = X'\cos\phi_Y + Z\sin\phi_Y, \tag{1.72}$$
$$Y = Y', \tag{1.73}$$
$$Z = -X'\sin\phi_Y + Z'\cos\phi_Y. \tag{1.74}$$

Then,

$$\frac{\partial X}{\partial \phi_Y} = -X'\sin\phi_Y + Z'\sin\phi_Y = Z, \tag{1.75}$$
$$\frac{\partial Y}{\partial \phi_Y} = 0, \tag{1.76}$$
$$\frac{\partial Z}{\partial \phi_Y} = -X'\cos\phi_Y - Z'\sin\phi_Y = -X. \tag{1.77}$$

We can compute the other partial derivatives similarly. We can also write a second equation for y' image coordinates similar to equation 1.71, and compute the partial derivatives. The partial derivatives of x' and y' with respect to each camera parameter are given in the Figure 1.7.

For m image points we can write $2 \times m$ equations with six unknowns. This linear systems of equations can be written as follows:

$$A\Delta = E, \tag{1.78}$$

where A is the derivative matrix , Δ is the vector of six unknowns parameters, (D_X, D_Y, D_Z, ϕ_X, ϕ_Y, ϕ_Z), for determining orientation and location of camera, E is the vector of error terms. Now, the problem is to compute Δ, which can be done using the least squares fit as follows:

$$\Delta = (A^T A)^{-1} A^T E. \tag{1.79}$$

Figure 1.9 shows a demonstration of this method for pose estimation.

We can also use lines instead of points in the pose estimation. One possible approach is to measure errors as the perpendicular distance of each endpoint of the model line from the corresponding line in the image, and then take the derivatives in terms of this distance rather than in terms of x' and y'. The equation of line can be written as:

$$\frac{-m}{\sqrt{m^2+1}}x + \frac{1}{\sqrt{m^2+1}}y = d, \tag{1.80}$$

where m is the slope, and d is the perpendicular distance of the line from the origin. The equation of a line through point (x', y') can be obtained from equation 1.80 by substituting (x', y') instead of (x, y) and d' instead of d. Then the perpendicular distance of point (x', y') from the line is simply $d - d'$. It is easy to calculate the derivatives of d' for use in the iterative scheme, because the derivatives of d' are just linear combination of x' and y', which we already know.

	x'	y'
D_X	1	0
D_Y	0	1
D_Z	$-fc^2X$	$-fc^2Y$
ϕ_X	$-fc^2XY$	$-fc(Z+cY^2)$
ϕ_Y	$fc(Z+cX^2)$	fc^2XY
ϕ_Z	$-fcY$	fcX

Figure 1.7: The partial derivatives of image coordinates with respect to each of the camera parameters.

1. Start with an initial estimate of six parameters, $(D_X^0 D_Y^0, D_Z^0, \phi_X^0, \phi_Y^0, \phi_Z^0)$. If you do not know, assume all parameters to be zero.
2. Apply transformation to the model, and project the model on the image plane by computing (x', y').
3. Compute the errors $E_{x'}$, and $E_{y'}$. If the errors are acceptable, quit.
4. Find change in six parameters,$(\Delta D_X, \Delta D_Y, \Delta D_Z, \Delta\phi_X, \Delta\phi_Y, \Delta\phi_Z^0)$ of transformation, by using least squares fit. Goto step (2).

Figure 1.8: Pose estimation algorithm.

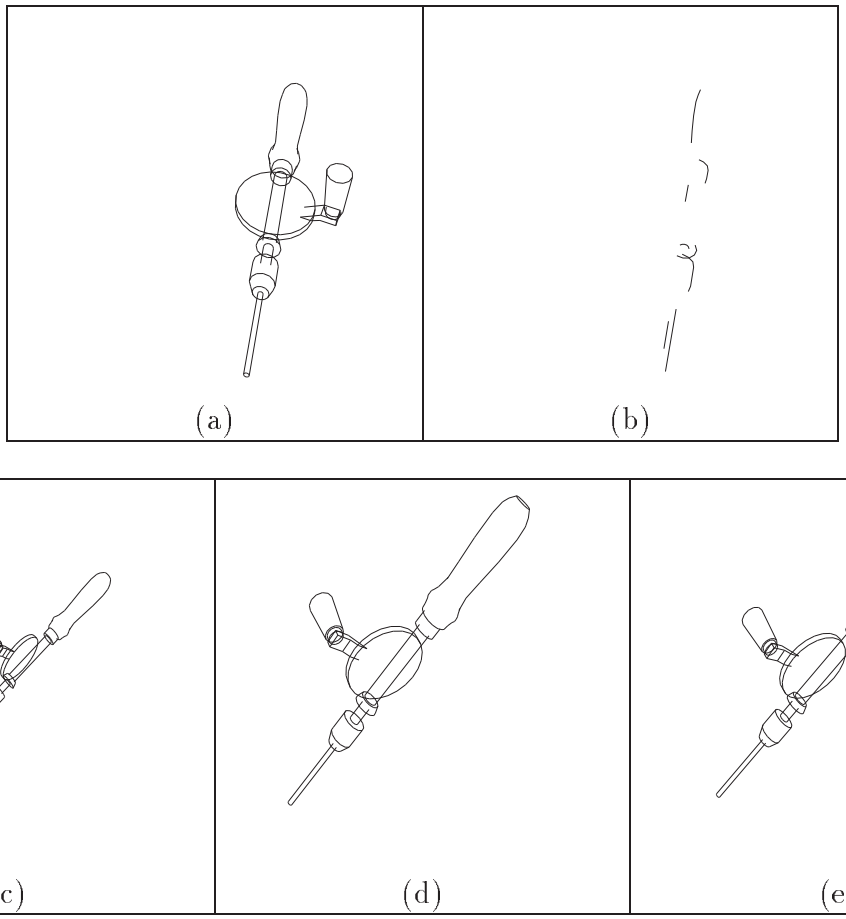

Figure 1.9: (a) The $3-D$ model of a drill. (b) The projection of the edges, when the drill is in unknown pose. (c) The position of drill after one iteration. (d) The position of drill after two iterations. (e) The position of drill after three iterations, which is the correct pose of the drill.

1.11 Exercises

1. What is the rotation matrix for an object rotation of 30^0 around the **Z** axis, followed by 60^0 around the **X** axis, and followed by a rotation of 90^0 around the **Y** axis. All rotations are counter clockwise.

2. What is the rotation matrix for an object rotation of Φ around the **X** axis, followed by Ψ around the **X** axis, and followed by a rotation of Θ around the **Z** axis. All rotations are counterclockwise.

3. Consider a vector $(7, 3, 2)$ which is rotated around the **Z** axis by 90^0, and then rotated around the **Y** axis by 90^0 and finally translated by $(4, -3, 7)$. Find the new coordinates of the vector. All rotations are clockwise.

4. Show that following identities are true.
   $R^Y_{90}R^X_{90} = R^Z_{270}R^Y_{90}$, $R^Y_{90}R^X_{180} = R^Z_{180}R^Y_{90}$
   $R^y_{180}R^X_{90} = R^Z_{180}R^X_{270}$, $R^Y_{180}R^X_{270} = R^Z_{180}R^X_{90}$.

5. Derive equations 1.28 and 1.29.

6. Consider the following camera model. Assume that first the camera gimbal center is at the origin of the world coordinates. Next, the camera is translated by amount $(0, 2, 2)$, and it is rotated by angle 90° around Z axis , followed by rotation of 135° around X axis. Both rotations are in the clockwise directions. After that the camera (image plane) is translated by amount $(.02, .01, .03)$ with respect to gimbal center. Finally, image is formed using perspective transform. Assume that focal length of the camera is .030. Find the image coordinates of the point which has world coordinates $(1, 1, 0.2)$.

7. Show how any image point (x_0, y_0) is mapped to the world coordinates using the inverse perspective matrix, verify that the equations 1.25 and 1.26 are satisfied.

8. Derive equations 1.53 and 1.54.

9. Verify equations 1.56, 1.60 and 1.62.

10. Given 3×3 rotation matrix, R, compute r in the Rodrigues' formula.

11. Verify equation 1.68.

12. Given 3×3 rotation matrix, R, compute quaternion q.

13. Suppose we want to minimize the squared sum of distances between a set of points p_i and their rotated versions $p'_i = Rp_i$. Show that minimizing criterion

$$C = \sum_i ||{p'}_i - Rp_i||^2, \tag{1.81}$$

can be written as

$$C = \sum_i [A_i q]^t [A_i q] = q^t [\sum_i A_i^t A_i] q, \tag{1.82}$$

where A_i is a 4×4 matrix depending on p_i and ${p'}_i$. The solution is given by the unit eigen vector associated with the smallest eigen value of the matrix $B = \sum_i A_i^t A_i$.

14. Assume p, q and r be quaternions. Show that:

$$\overline{(p * q)} = -(\bar{q} * \bar{p}), \tag{1.83}$$
$$(p * q).(p * q) = (p.p).(q.q), \tag{1.84}$$
$$(p * q).(p * r) = (p.p).(q.r), \tag{1.85}$$

where . is the dot product.

15. A rotation matrix R for rotation around an arbitrary vector by some angle can be expressed as the product of three rotations (Euler's angles): rotation around X axis by angle γ, followed by the rotation around Y axis by angle β and followed by the rotation around Z axis by angle α. All the rotations are counter clockwise. Show that:

$$R = \begin{bmatrix} \cos\alpha\cos\beta & \cos\alpha\sin\beta\sin\gamma - \sin\alpha\cos\gamma & \cos\alpha\sin\beta\cos\gamma + \sin\alpha\sin\gamma \\ \sin\alpha\cos\beta & \sin\alpha\sin\beta\sin\gamma + \cos\alpha\cos\gamma & \sin\alpha\sin\beta\cos\gamma - \cos\alpha\sin\gamma \\ -\sin\beta & \cos\beta\sin\gamma & \cos\beta\cos\gamma \end{bmatrix}.$$

16. For small rotations we can assume $\sin\theta = \theta$, and $\cos\theta = 1$. Simplify the above matrix assuming small rotations. Is the rotation matrix now linear in α, β and γ?

17. Given 3×3 rotation matrix, R, compute Euler's angles α, β and γ.

18. Derive the expressions for the partial derivatives given in the Table shown in Figure 1.7.

19. By using equation 1.80 compute the perpendicular distance of a point (x', y') from a line.