Chapter 13

3D Sensing and Object Pose Computation

The main concern of this chapter is the quantitative relationship between 2D image structures and their corresponding 3D real world structures. The previous chapter investigated relationships between image and world by primarily studying the qualitative phenomena. Here, we show how to make measurements needed for recognition and inspection of 3D objects and for robotic manipulation or navigation.

Figure 13.1: Two images of the driver of a car taken from two of four onboard cameras. A multiple camera measurement system is used to compute the 3D location of certain body points (identified by the bright ellipses) which are then used to compute posture. (Images courtesy of Herbert Reynolds, Michigan State University Ergonomics Lab.)

Consider Figure 13.1, for example. The problem is to measure the body posture of a driver operating a car: the ultimate purpose is to design a better driving environment. In another application, shown in Figure 13.2, the camera system must recognize 3D objects and their pose so that a parts handling robot can grasp them. For this application, the camera system and the robot arm communicate in terms of 3D world coordinates.

This chapter treats some of the engineering and mathematics for sensing in 3D. Problems

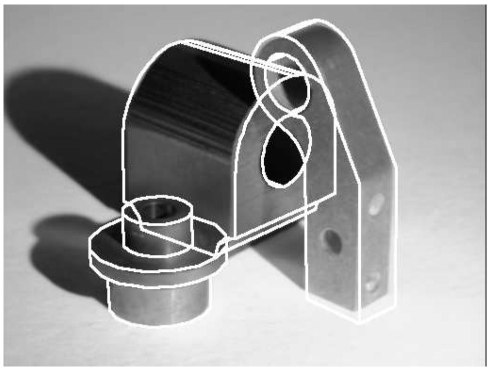


Figure 13.2: The overlay of graphics on the image of three 3D objects has been used to recognize and localize the objects. In order to do this, the recognition system has matched 2D image parts to 3D model parts and has computed the geometrical 3D transformation needed to create the observed image from the model objects. The robot controller can then be told of the identity and pose of each part. (Image courtesy of Mauro Costa.)

are formulated in terms of the intuitive geometry and then the mathematical models are developed. The algebra of transformations in 3D is central to the mathematics. The role of 3D object models is described and different sensor configurations and their calibration procedures are discussed.

13.1 General Stereo Configuration

Figure 13.3 shows a general configuration of two cameras viewing the same 3D workspace. Often in computer graphics, a right-handed coordinate system is used with the -z axis extending out from the camera so that points farther from the camera have more negative depth cordinates. We keep depth positive in most of the models in this chapter but sometimes use another system to be consistent with a published derivation. Figure 13.3 shows a general stereo configuration that does not have the special alignment of the cameras as assumed in Chapter 12. The cameras both view the same workpiece on a worktable: the worktable is the entire 3D world in this situation and it has its own global coordinate system **W** attached to it. Intuitively, we see that the location of 3D point $^{\mathbf{w}}\mathbf{P} = [^wP_x, {}^wP_y, {}^wP_z]^t$ in the workspace can be obtained by simply determining the intersection of the two imaging rays $^{\mathbf{w}}\mathbf{P}^{\mathbf{1}}\mathbf{O}$ and $^{\mathbf{w}}\mathbf{P}^{\mathbf{2}}\mathbf{O}$. We shall derive the algebra for this computation below: it is straighforward but there are some complications due to measurement error.

In order to perform the general stereo computation illustrated in Figure 13.3, the following items must be known.

- We must know the pose of camera $\mathbf{C_1}$ in the workspace **W** and some camera internals, such as focal length. All this information will be represented by a *camera matrix*, which algebraically defines a ray in 3D space for every image point $^{\mathbf{1}}\mathbf{P}$. Sections 13.3 and 13.7 describe camera calibration procedures by which this information can be obtained.

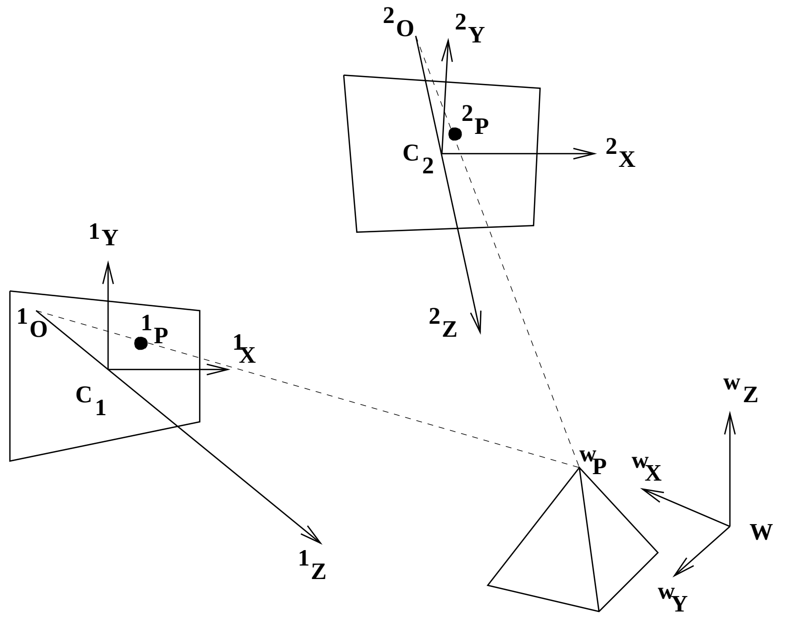


Figure 13.3: Two cameras $\mathbf{C_1}$ and $\mathbf{C_2}$ view the same 3D workspace. Point $\mathbf{P}$ on a workpiece is imaged at point $\mathbf{^1P}$ on the first image plane and at point $\mathbf{^2P}$ on the second image plane.

- Similarly, we must know the pose of camera $\mathbf{C_2}$ in the workspace $\mathbf{W}$ and its internal parameters; equivalently, we need its camera matrix.
- We need to identify the correspondence of the 3D point to the two 2D image points $(\mathbf{^wP}, \mathbf{^1P}, \mathbf{^2P})$.
- We need a formula for computing $\mathbf{^wP}$ from the two imaging rays $\mathbf{^wP^1O}$ and $\mathbf{^wP^2O}$.

Before addressing these items, we take the opportunity to describe three important variations on the configuration shown in Figure 13.3.

- **The configuration shown in Figure 13.3 consists of two cameras calibrated to the world coordinate space**. Coordinates of 3D point features are computed by intersecting the two imaging rays from the corresponding image points.
- **One of the cameras can be replaced by a projector** which illuminates one or more surface points using a beam of light or a special pattern such as a crosshair. This is shown in Figure 13.4. As we shall see below, the projector can be calibrated in much the same manner as a camera: the projected ray of light has the same algebraic representation as the ray imaging to a camera. Using a projector has advantages when surface point measurements are needed on a surface that has no distinguishing features.
- **Knowledge of the model object can replace one of the cameras**. Assume that the height of the pyramid in Figure 13.3 is known; thus, we already know coordinate ^wP_z, which means that point $\mathbf{P}$ is constrained to lie on the plane $z = {}^wP_z$. The

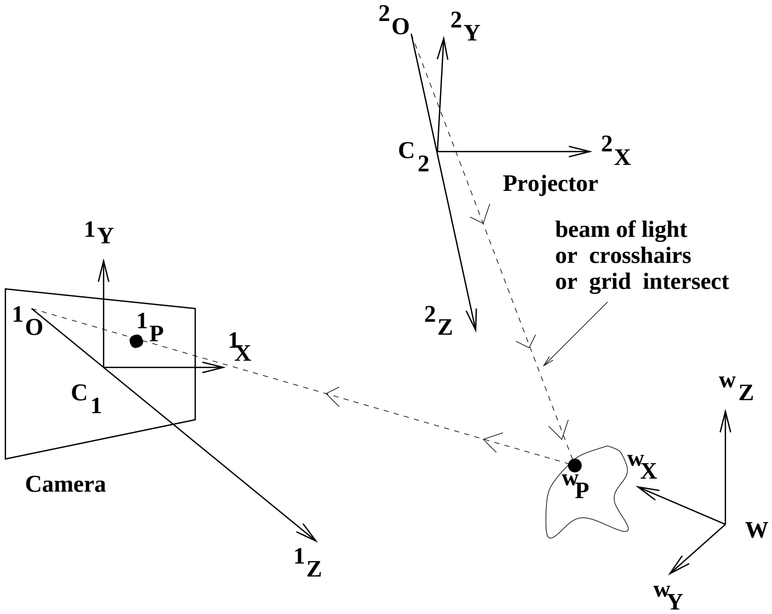


Figure 13.4: A projector can replace one camera in the general stereo configuration. The same geometric and algebraic constraints hold as in Figure 13.3; however, the projector can add surface features to an otherwise featureless surface.

other two coordinates are easily found by intersecting the imaging ray from the single camera $\mathbf{C_1}$ with that plane. In many cases model information adds enough constraint so that a single camera is sufficient.

13.2 3D Affine Transformations

Affine transformations of 2D spaces were treated in Chapter 11. In this chapter, we make the extension to 3D. These transformations are very important not only for 3D machine vision, but also for robotics and virtual reality. Basic transformations are translation, rotation, scaling and shear. These primitive transformations extend in a straightforward manner; however, some are more difficult to visualize. Once again we use the convenience of homogeneous coordinates, which extends a 3D point $[P_x, P_y, P_z]$ to have the 4 coordinates $[sP_x, sP_y, sP_z, s]$, where s is a nonzero scale factor. (As before, points are column vectors but we will sometimes omit the transpose symbol as there is no ambiguity caused by this.) In this chapter, we often use superscripts on point names because we need to name more coordinate systems than we did in Chapter 11. Below, we add perspective, orthographic, and weak perspective projections from 3D space to 2D space to our set of transformations.

13.2.1 Coordinate Frames

Coordinate frames or *coordinate systems* are needed in order to quantitatively locate points in space. Figure 13.5 shows a scene with four different relevant coordinate systems. Point

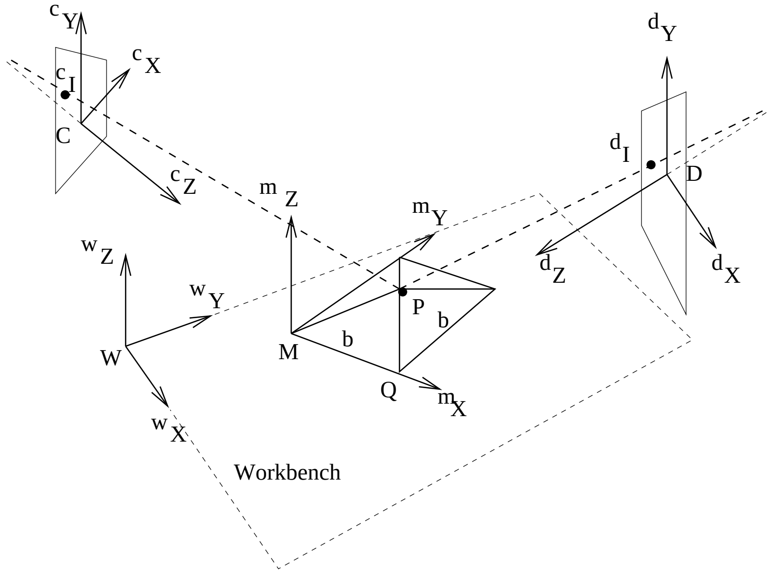


Figure 13.5: Point P can be represented in terms of coordinates relative to four distinct coordinate frames — (1) the model coordinate system **M**, (2) the world or workbench coordinate system **W**, (3) sensor **C**, and sensor **D**. Coordinates change with the coordinate frame; for example, point P appears to be left of Q to sensor **C**, but to the right of Q to **D**.

P, the apex of a pyramid, has four different coordinate representations. First, the point is represented in a CAD model as ${}^{\mathbf{M}}\mathbf{P} = [{}^{M}P_x, {}^{M}P_y, {}^{M}P_z] = [b/2, b/2, \frac{\sqrt{2}}{2}b]$, where b is the size of its base. Second, an instance of this CAD model is posed on the workbench as shown. The representation of the pyramid apex relative to the workbench frame is

$$ {}^{\mathbf{W}}\mathbf{P} = = [{}^{W}P_x, {}^{W}P_y, {}^{W}P_z] = \mathbf{TR}\,[b/2, b/2, \frac{\sqrt{2}}{2}b], \tag{13.1} $$

where **TR** is the combined rotation and translation of coordinate frame **M** relative to coordinate frame **W**. Finally, if two sensors **C** and **D** (or persons) view the pyramid from opposite sides of the workbench, the left-right relationship between points **P** and **Q** is reversed: the coordinate representations are different. ${}^{\mathbf{C}}\mathbf{P} = [{}^{C}P_x, {}^{C}P_y, {}^{C}P_z] \neq {}^{\mathbf{D}}\mathbf{P} = [{}^{D}P_x, {}^{D}P_y, {}^{D}P_z]$.

In order to relate sensors to each other, to 3D objects in the scene, and to manipulators that operate on them, we will develop mathematical methods to relate their coordinate frames. These same methods also enable us to model the motion of an object in space. Sometimes we will use a convenient notation to emphasize the coordinate frame in which a point is represented and the "direction" of the coordinate transformation. We denote the transformation ${}^{\mathbf{W}}_{\mathbf{M}}\mathbf{T}$ of a model point ${}^{\mathbf{M}}\mathbf{P}$ from model coordinates to workbench coordinates ${}^{\mathbf{W}}\mathbf{P}$ as follows.

$$ {}^{\mathbf{W}}\mathbf{P} = {}^{\mathbf{W}}_{\mathbf{M}}\mathbf{T}\ {}^{\mathbf{M}}\mathbf{P} \tag{13.2} $$

This notation is developed in the robotics text by Craig (1986): it can be very helpful when reasoning about motions of objects or matching of objects. In simple situations where the coordinate frame is obvious, we use simpler notation. We now proceed with the study of transformations.

13.2.2 Translation

Translation adds a translation vector of three coordinates x_0, y_0, z_0 to point $\mathbf{^1P}$ in coordinate frame **1** to get point $\mathbf{^2P}$ in coordinate frame **2**. In the example in Figure 13.5 some translation (and rotation) is necessary in order to relate a point in model coordinates to its pose on the workbench.

$$\begin{aligned}
\mathbf{^2P} &= \mathbf{T}(x_0, y_0, z_0)\ \mathbf{^1P} \\
\mathbf{^2P} &= \begin{bmatrix} ^2P_x \\ ^2P_y \\ ^2P_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & x_0 \\ 0 & 1 & 0 & y_0 \\ 0 & 0 & 1 & z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ^1P_x \\ ^1P_y \\ ^1P_z \\ 1 \end{bmatrix}
\end{aligned} \tag{13.3}$$

13.2.3 Scaling

A 3D scaling matrix can apply individual scale factors to each of the coordinates. Sometimes, all scale factors will be the same, as in the case of a change of measurement units or a uniform scaling in instantiating a model to a certain size.

$$\begin{aligned}
\mathbf{^2P} &= \mathbf{S}\ \mathbf{^1P} = \mathbf{S}(s_x, s_y, s_z)\mathbf{^1P} \\
\begin{bmatrix} ^2P_x \\ ^2P_y \\ ^2P_z \\ 1 \end{bmatrix} &= \begin{bmatrix} s_x\ ^2P_x \\ s_y\ ^2P_y \\ s_z\ ^2P_z \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} ^1P_x \\ ^1P_y \\ ^1P_z \\ 1 \end{bmatrix}
\end{aligned} \tag{13.4}$$

13.2.4 Rotation

Creation of a matrix representing a primitive rotation about a coordinate axis is especially easy; all we need do is write down the column vectors of the matrix to be the transformed values of the unit vectors under the rotation. (Recall that any 3D linear transformation is completely characterized by how that transformation transforms the three basis vectors.) The transformation about the z-axis is actually the same as the 2D transformation done in Chapter 11, except that it now carries along a copy of the z-coordinate of the 3D point. Figure 13.6 shows how the basis vectors are transformed by the primitive rotations.

Rotation of θ about the X axis:

$$\mathbf{^2P} = \mathbf{R}(^1X, \theta)\ \mathbf{^1P}$$

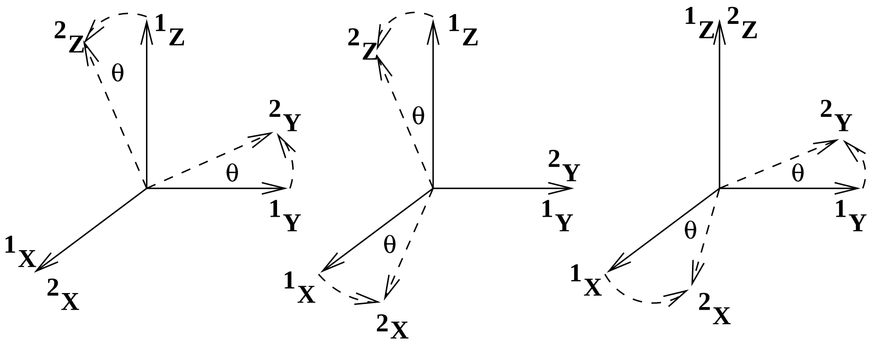


Figure 13.6: Rotations by angle θ about the (left) x-axis, (center) y-axis, and (right) z-axis.

$$\begin{bmatrix} {}^2P_x \\ {}^2P_y \\ {}^2P_z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta & 0 \\ 0 & \sin\theta & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^1P_x \\ {}^1P_y \\ {}^1P_z \\ 1 \end{bmatrix} \tag{13.5}$$

Rotation of θ about the Y axis:

$$\mathbf{{}^2P} = \mathbf{R}({}^1Y, \theta)\ \mathbf{{}^1P}$$

$$\begin{bmatrix} {}^2P_x \\ {}^2P_y \\ {}^2P_z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & 0 & \sin\theta & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\theta & 0 & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^1P_x \\ {}^1P_y \\ {}^1P_z \\ 1 \end{bmatrix} \tag{13.6}$$

Rotation of θ about the Z axis:

$$\mathbf{{}^2P} = \mathbf{R}({}^1Z, \theta)\ \mathbf{{}^1P}$$

$$\begin{bmatrix} {}^2P_x \\ {}^2P_y \\ {}^2P_z \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^1P_x \\ {}^1P_y \\ {}^1P_z \\ 1 \end{bmatrix} \tag{13.7}$$

Exercise 1

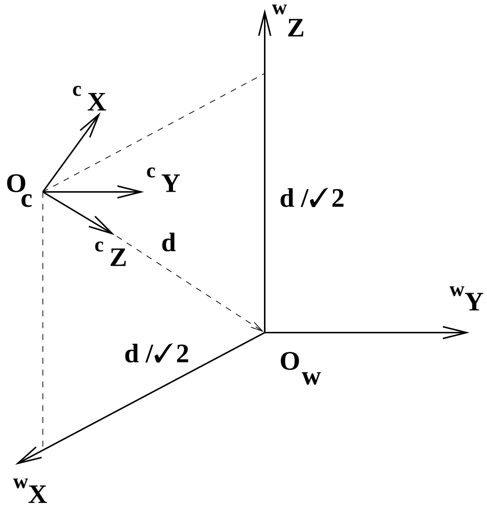
Verify that the columns of the matrices representing the three primitive rotations are orthonormal. Same question for the rows.

Exercise 2

Construct the rotation matrix for a counterclockwise rotation of $\Pi/4$ about the axis defined by the origin and the point $[1, 1, 0]^t$.

Exercise 3

Show how to make the general construction of the rotation matrix given a rotation angle of θ radians and the direction cosines $[c_x, c_y, c_z]^t$ of the axis.



Example: derive the combined rotation and translation needed to transform world coordinates **W** into camera coordinates **C**.

To construct the rotation matrix **R**, we write the coordinates of the basis vectors of **W** in terms of those of **C** so that any point in **W** coordinates can be transformed into **C** coordinates.

$$\begin{aligned} {}^{\mathbf{W}}\mathbf{X} &= \frac{-\sqrt{2}}{2}\,{}^{\mathbf{C}}\mathbf{X} + 0\,{}^{\mathbf{C}}\mathbf{Y} + \frac{-\sqrt{2}}{2}\,{}^{\mathbf{C}}\mathbf{Z} \\ {}^{\mathbf{W}}\mathbf{Y} &= 0\,{}^{\mathbf{C}}\mathbf{X} + 1\,{}^{\mathbf{C}}\mathbf{Y} + 0\,{}^{\mathbf{C}}\mathbf{Z} \\ {}^{\mathbf{W}}\mathbf{Z} &= \frac{\sqrt{2}}{2}\,{}^{\mathbf{C}}\mathbf{X} + 0\,{}^{\mathbf{C}}\mathbf{Y} + \frac{-\sqrt{2}}{2}\,{}^{\mathbf{C}}\mathbf{Z} \end{aligned} \tag{13.8}$$

These three vectors will be the three columns of the rotation matrix encoding the orientation of camera frame **C** relative to the world frame **W**. Once the camera "is rotated", world points must be translated by d along the z-axis so that the world origin will be located at coordinate $[0, 0, d]^t$ relative to **C**. The final change of coordinate transformation is as follows.

$${}^{\mathbf{C}}_{\mathbf{W}}\mathbf{TR} = \begin{bmatrix} \frac{-\sqrt{2}}{2} & 0 & \frac{\sqrt{2}}{2} & 0 \\ 0 & 1 & 0 & 0 \\ \frac{-\sqrt{2}}{2} & 0 & \frac{-\sqrt{2}}{2} & d \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13.9}$$

Check that ${}^{\mathbf{C}}_{\mathbf{W}}\mathbf{TR}\,{}^{\mathbf{W}}\mathbf{O_w} = {}^{\mathbf{C}}_{\mathbf{W}}\mathbf{TR}\,[0, 0, 0, 1]^t = [0, 0, d, 1]^t = {}^{\mathbf{C}}\mathbf{O_w}$, and ${}^{\mathbf{C}}_{\mathbf{W}}\mathbf{TR}\,{}^{\mathbf{W}}\mathbf{O_c} = {}^{\mathbf{C}}_{\mathbf{W}}\mathbf{TR}\,[d\frac{\sqrt{2}}{2}, 0, d\frac{\sqrt{2}}{2}, 1]^t = [0, 0, 0, 1]^t = {}^{\mathbf{C}}\mathbf{O_c}$.

Exercise 4

Consider the environment of the previous example. Place a unit cube at the world origin $\mathbf{O_W}$. Transform its corners K_j into camera coordinates. Verify that 4 of the edges have unit length by computing $\| K_i - K_j \|$ using camera coordinates.

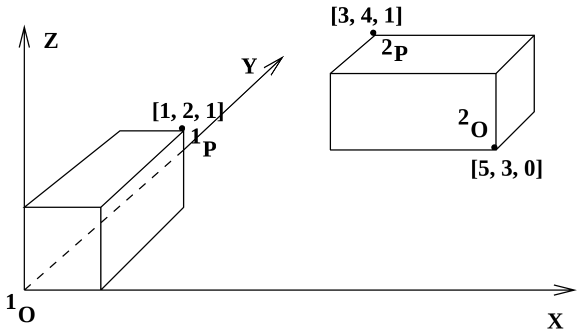


Figure 13.7: Two instances of the same block model.

13.2.5 Arbitrary Rotation

Any rotation can be expressed in the form shown in Equation 13.10. The matrix of coefficients r_{ij} is an orthonormal matrix: all the columns (rows) are unit vectors that are mutually orthogonal. All of the primitive rotation matrices given above have this property. Any rigid rotation of 3D space can be represented as a rotation of some angle θ about a single axis $\mathbf{A}$. $\mathbf{A}$ need not be one of the coordinate axes; it can be any axis in 3D space. To see this, suppose that the basis vector $^{\mathbf{1}}\mathbf{X}$ is transformed into a different vector $^{\mathbf{2}}\mathbf{X}$. The axis of rotation $\mathbf{A}$ can be found by taking the cross product of $^{\mathbf{1}}\mathbf{X}$ and $^{\mathbf{2}}\mathbf{X}$. In case $^{\mathbf{1}}\mathbf{X}$ is invariant under the rotation, then it is itself the axis of rotation.

$$
\begin{aligned}
{}^{\mathbf{2}}\mathbf{P} &= \mathbf{R}(A,\theta)\ {}^{\mathbf{1}}\mathbf{P} \\
\begin{bmatrix} {}^2P_x \\ {}^2P_y \\ {}^2P_z \\ 1 \end{bmatrix} &= \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^1P_x \\ {}^1P_y \\ {}^1P_z \\ 1 \end{bmatrix}
\end{aligned}
\tag{13.10}
$$

As a consequence, *the result of the motion of a rigid object moving from time t_1 to time t_2 can be represented by a single translation vector and a single rotation matrix, regardless of its actual trajectory during this time period.* Only one homogeneous matrix is needed to store both the translation and rotation: there are six parameters, three for rotation and three for translation.

$$
\begin{bmatrix} {}^2P_x \\ {}^2P_y \\ {}^2P_z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^1P_x \\ {}^1P_y \\ {}^1P_z \\ 1 \end{bmatrix}
\tag{13.11}
$$

Exercise 5

Refer to Figure 13.7. Give the homogeneous transformation matrix which maps all corners of the block at the origin onto corresponding corners of the other block. Assume that corner $^{\mathbf{1}}\mathbf{O}$ maps onto $^{\mathbf{2}}\mathbf{O}$ and corner $^{\mathbf{1}}\mathbf{P}$ maps onto $^{\mathbf{2}}\mathbf{P}$ and that the transformation is a rigid transformation.

Exercise 6 inverse of a rotation matrix

Argue that a rotation matrix always has an inverse. What is the inverse of the rotation $\mathbf{R}(A, \theta)$ of Equation 13.10?

13.2.6 Alignment via Transformation Calculus

Here we show how to align a model triangle with a sensed triangle. This should be a convincing example of the value of doing careful calculus with transformations. More important, it provides a basis for aligning any rigid model via correspondences between three model points with three sensed points. The development is done only with algebra using the basic transformation units already studied. Figures 13.8 and 13.9 illustrate the steps with a sketch of the geometry.

The problem is to derive the transformation ${}^{\mathbf{W}}_{\mathbf{M}}\mathbf{T}$ that maps the vertices A,B,C of a model triangle onto the vertices D,E,F of a congruent triangle in the workspace. In order to achieve this, we transform both triangles so that side AB lies along the ^{W}X-axis and so that the entire triangle lies in the XY-plane of $\mathbf{W}$. In such a configuration, it must be that coordinates of A and D are the same; also B and E, C and F. The transformation equation derived to do this can be rearranged to produce the desired transformation ${}^{\mathbf{W}}_{\mathbf{M}}\mathbf{T}$ mapping each point $^{\mathbf{M}}\mathbf{P_i}$ onto the corresponding $^{\mathbf{W}}\mathbf{P_i}$. The operations are given in Algorithm 1. It should be clear that each of these operations can be done and that the inverse of each exists. The family of operations requires careful programming and data structure design, however. This is a very important operation; in theory, any two congruent rigid objects can be aligned by aligning a subset of three corresponding points. In practice, measurement and computational errors are likely to produce significant error in aligning points far away from the $\triangle ABC$ used. An optimization procedure using many more points is usually needed.

13.3 Camera Model

Our goal in this section is to show that the *camera model* $\mathbf{C}$ in Equation 13.14 below is the appropriate algebraic model for perspective imaging and then to show how to determine the matrix elements from a fixed camera set up. The matrix elements are then used in computer programs that perform 3D measurements.

$$
\begin{aligned}
{}^{\mathbf{I}}\mathbf{P} &= {}^{\mathbf{I}}_{\mathbf{W}}\mathbf{C}\ {}^{\mathbf{W}}\mathbf{P} \\
\begin{bmatrix} s\,{}^{I}P_r \\ s\,{}^{I}P_c \\ s \end{bmatrix} &= {}^{\mathbf{I}}_{\mathbf{W}}\mathbf{C} \begin{bmatrix} {}^{W}P_x \\ {}^{W}P_y \\ {}^{W}P_z \\ 1 \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & 1 \end{bmatrix} \begin{bmatrix} {}^{W}P_x \\ {}^{W}P_y \\ {}^{W}P_z \\ 1 \end{bmatrix}
\end{aligned}
\qquad (13.14)
$$

Exercise 7 inverse of a translation matrix

Argue that a translation matrix always has an inverse. What is the inverse of the translation $\mathbf{T}(t_x, t_y, t_z)$ of Equation 13.3?

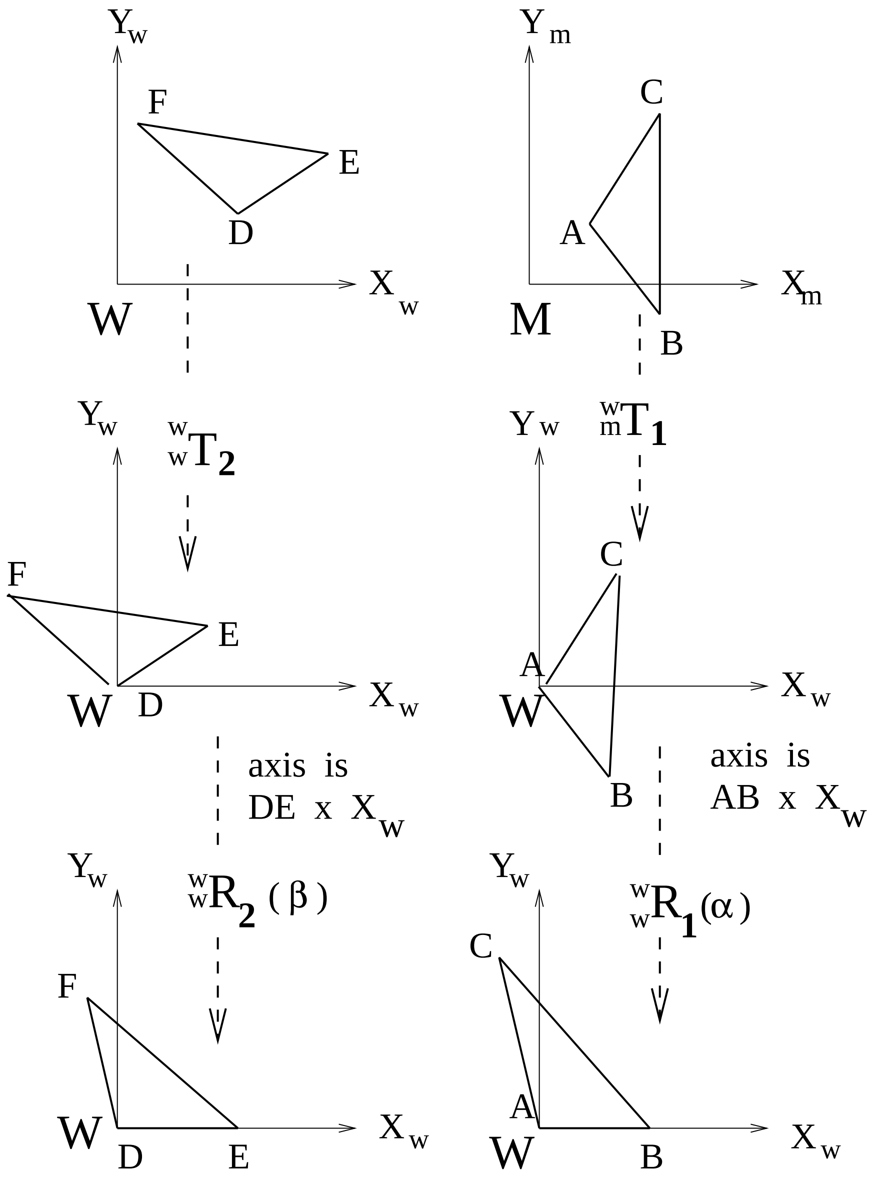
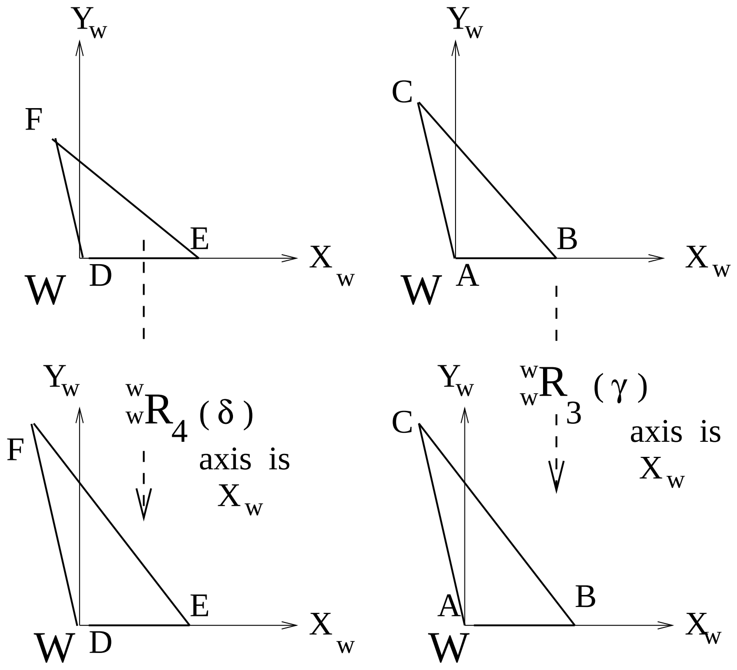


Figure 13.8: **Part I** Alignment of two congruent triangles. $\triangle ABC$ is the *model* triangle while $\triangle DEF$ is the sensed triangle. First, object points are translated such that A and D move to the origin. Second, each triangle is rotated so that rays AB and DE lie along the X-axis.



$$ {}^{w}_{w}R_4 \, {}^{w}_{w}R_2 \, {}^{w}_{w}T_2 \, {}^{w}P_i \; = \; {}^{w}_{w}R_3 \, {}^{w}_{w}R_1 \, {}^{w}_{m}T_1 \, {}^{m}P_i \; = \; {}^{w}P_i^* $$

Figure 13.9: **Part II** Alignment of two congruent triangles continued. Each triangle is rotated into the XY-plane. The axis of rotation is the X-axis; the angles of rotation are determined by rays AC and DF relative to the XY-plane.

Compute rigid transformation ${}^{W}_{M}\mathbf{T}$ that aligns model points A,B,C with world points D,E,F

1. Input the three 3D model points A,B,C and the corresponding three 3D world points D,E,F.
2. Construct translation ${}^{W}_{M}\mathbf{T_1}$ to shift points so that model point A maps to the world origin. Construct translation ${}^{W}_{W}\mathbf{T_2}$ to shift world points so that D maps to the world origin. This will align only points A and D in **W**.
3. Construct rotation ${}^{W}_{W}\mathbf{R_1}$ so that side AB is along the X-axis. Construct rotation ${}^{W}_{W}\mathbf{R_2}$ so that side DE is along the X-axis. This will align sides AB and DE in **W**.
4. Construct rotation ${}^{W}_{W}\mathbf{R_3}$ about the X-axis so that point C maps into the XY-plane. Construct rotation ${}^{W}_{W}\mathbf{R_4}$ about the X-axis so that point F maps into the XY-plane. Now all three points are aligned in **W**.
5. The model triangle and world triangle are now aligned, as represented by the following equation.

$$ {}^{W}_{W}\mathbf{R_3}\ {}^{W}_{W}\mathbf{R_1}\ {}^{W}_{M}\mathbf{T_1}\ {}^{M}\mathbf{P_i} \ = \ {}^{W}_{W}\mathbf{R_4}\ {}^{W}_{W}\mathbf{R_2}\ {}^{W}_{W}\mathbf{T_2}\ {}^{W}\mathbf{P_i} \tag{13.12}$$

$$ {}^{W}\mathbf{P_i} \ = \ (\ \mathbf{T_2^{-1}}\ \mathbf{R_2^{-1}}\ \mathbf{R_4^{-1}}\ \mathbf{R_3}\ \mathbf{R_1}\ \mathbf{T_1})\ {}^{M}\mathbf{P_i} \tag{13.13}$$

6. Return ${}^{W}_{M}\mathbf{T} \ = \ (\mathbf{T_2^{-1}}\ \mathbf{R_2^{-1}}\ \mathbf{R_4^{-1}}\ \mathbf{R_3}\ \mathbf{R_1}\ \mathbf{T_1})$

Algorithm 1: Derive the rigid transformation needed to align a model triangle with a congruent world triangle.

Exercise 8

Clearly, Algorithm 1 must fail if $\| A - B \| \neq \| A - B \|$. Insert the appropriate tests and error returns to handle the case when the triangles are not congruent.

Exercise 9 * program the triangle alignment

Using the method sketched in Algorithm 1, write and test a program to align a model triangle with a congruent triangle in world coordinates. Write separate functions to perform each basic operation.

Exercise 10

(a) Argue that the transformation returned by Algorithm 1 does in fact map model point A to world point D, model point B to world E, and model C to world F. (b) Argue that any other model point will maintain the same distances to A,B,C when transformed. (c) Argue that if a rigid n-vertex polyhedral model can be aligned with an object in the world using a rigid transformation, then the transformation can be obtained by aligning just two triangles using Algorithm 1.

$$
\begin{aligned}
{}^{I}P_r &= \frac{[c_{11}\ c_{12}\ c_{13}\ c_{14}] \circ [{}^{W}P_x\ {}^{W}P_y\ {}^{W}P_z\ 1]}{[c_{31}\ c_{32}\ c_{33}\ 1] \circ [{}^{W}P_x\ {}^{W}P_y\ {}^{W}P_z\ 1]} \\
{}^{I}P_c &= \frac{[c_{21}\ c_{22}\ c_{23}\ c_{24}] \circ [{}^{W}P_x\ {}^{W}P_y\ {}^{W}P_z\ 1]}{[c_{31}\ c_{32}\ c_{33}\ 1] \circ [{}^{W}P_x\ {}^{W}P_y\ {}^{W}P_z\ 1]}
\end{aligned}
$$

We will justify below that the perspective imaging transformation is represented by this 3 x 4 camera matrix ${}^{\mathbf{I}}_{\mathbf{W}}\mathbf{C}_{3\times 4}$, which projects 3D world point ${}^{\mathbf{W}}P = [{}^{W}P_x, {}^{W}P_y, {}^{W}P_z]^t$ to image point ${}^{\mathbf{I}}\mathbf{P} = [{}^{I}P_r, {}^{I}P_c]^t$. Using this matrix, there are enough parameters to model the change of coordinates between world $\mathbf{W}$ and camera $\mathbf{C}$ and to model all the scaling needed for both perspective transformation and the scaling of the real image coordinates into row and column of the image array. The matrix equation uses homogeneous coordinates: removal of the scale factor s using the dot product is also shown in Equation 13.14. We will now derive the parameters of the camera matrix ${}^{\mathbf{I}}_{\mathbf{W}}\mathbf{C}$.

13.3.1 Perspective Transformation Matrix

The algebra for the perspective tranformation was given in Chapter 12: the resulting equations are rephrased in Equation 13.15 below. Recall that these equations are derived from the simple case where the world and camera coordinates are identical. Moreover, the image coordinates $[{}^{F}P_x, {}^{F}P_y]$ are in the same (real) units as the coordinates in the 3D space and **not** in pixel coorinates. (Think of the superscript F as denoting a "floating point number" and not focal length, which is the parameter f of the perspective transformation.)

$$
\begin{aligned}
{}^{F}P_x/f &= {}^{C}P_x/{}^{W}P_z \quad or \quad {}^{F}P_x = (f/{}^{C}P_z)\ {}^{C}P_x \\
{}^{F}P_y/f &= {}^{C}P_y/{}^{W}P_z \quad or \quad {}^{F}P_y = (f/{}^{C}P_z)\ {}^{C}P_y
\end{aligned}
\tag{13.15}
$$

A pure perspective transformation shown in Figure 13.10 is defined in terms of the single parameter f, the focal length. The matrix ${}^{\mathbf{F}}_{\mathbf{C}}\mathbf{\Pi}(f)$ is shown in Equation 13.16 in its 4×4 form so that it can be combined with other transformation matrices; however, the third row producing ${}^{F}P_z = f$ is not actually needed and is ultimately ignored and often not given. Note that the matrix is of rank 3, not 4, and hence an inverse does not exist.

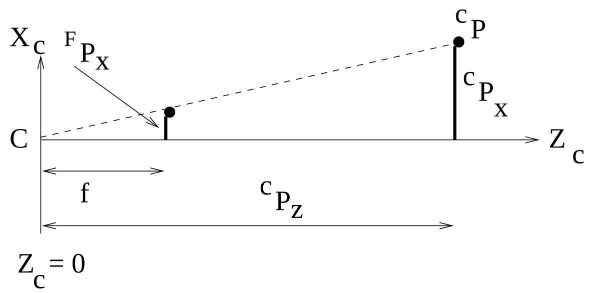


Figure 13.10: With the camera frame origin at the center of projection, ${}^{F}P_z = f$ always.

$$
{}^{\mathbf{F}}\mathbf{P} \quad = \quad {}^{\mathbf{F}}_{\mathbf{C}}\mathbf{\Pi}(f)\ \ {}^{\mathbf{C}}\mathbf{P}
$$

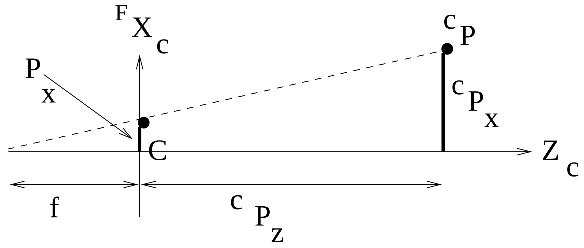


Figure 13.11: With the camera frame origin at the center of the image, ${}^F P_z = 0$ always.

$$\begin{bmatrix} s\,{}^F P_x \\ s\,{}^F P_y \\ s\,{}^F P_z \\ s \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 0 \end{bmatrix} \begin{bmatrix} {}^C P_x \\ {}^C P_y \\ {}^C P_z \\ 1 \end{bmatrix} \tag{13.16}$$

An alternative perspective transformation can be defined by placing the camera origin at the center of the image so that ${}^F P_z = 0$ as shown in Figure 13.11. The projection matrix would be as in Equation 13.17. (An advantage of this formulation is that it is obvious that we get orthographic projection as $f \rightarrow \infty$.)

$$\begin{aligned} {}^{\mathbf{F}}\mathbf{P} &= {}^{\mathbf{F}}_{\mathbf{C}}\mathbf{\Pi}(f)\ {}^{\mathbf{C}}\mathbf{P} \\ \begin{bmatrix} s\,{}^F P_x \\ s\,{}^F P_y \\ s\,{}^F P_z \\ s \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1/f & 1 \end{bmatrix} \begin{bmatrix} {}^C P_x \\ {}^C P_y \\ {}^C P_z \\ 1 \end{bmatrix} \end{aligned} \tag{13.17}$$

In the general case, as in Figure 13.3, the world coordinate system $\mathbf{W}$ is different from the camera coordinate system $\mathbf{C}$. A rotation and translation are needed to convert world point ${}^{\mathbf{w}}\mathbf{P}$ into camera coordinates ${}^{\mathbf{c}}\mathbf{P}$. Three rotation parameters and three translation parameters are needed to do this, but they are combined in complex ways to yield transformation matrix elements as given in the previous sections.

$$\begin{aligned} {}^{\mathbf{c}}\mathbf{P} &= \mathbf{T}(t_x, t_y, t_z)\ \mathbf{R}(\alpha, \beta, \gamma)\ {}^{\mathbf{w}}\mathbf{P} \\ {}^{\mathbf{c}}\mathbf{P} &= {}^{\mathbf{c}}_{\mathbf{w}}\mathbf{TR}(\alpha, \beta, \gamma, t_x, t_y, t_z)\ {}^{\mathbf{w}}\mathbf{P} \\ \begin{bmatrix} {}^c P_x \\ {}^c P_y \\ {}^c P_z \\ 1 \end{bmatrix} &= \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^w P_x \\ {}^w P_y \\ {}^w P_z \\ 1 \end{bmatrix} \end{aligned} \tag{13.18}$$

We can compose transformations in order to model the change of coordinates from $\mathbf{W}$ to $\mathbf{C}$ followed by the perspective transformation of ${}^{\mathbf{C}}\mathbf{P}$ onto the real image plane yielding ${}^{\mathbf{F}}\mathbf{P}$. The third row of the matrix is dropped because it just yields the constant value for

${}^F P_z$. $\mathbf{^F P}$ is on the *real image plane* and a scaling transformation is needed to convert to the pixel row and column coordinates of $\mathbf{^I P}$. Recall that matrix multiplication representing composition of linear transformations is associative.

$$\begin{aligned}
\mathbf{^F P} &= {}^{\mathbf{F}}_{\mathbf{C}}\mathbf{\Pi}(f)\ \ \mathbf{^C P} \\
&= {}^{\mathbf{F}}_{\mathbf{C}}\mathbf{\Pi}(f)\ \ ({}^{\mathbf{C}}_{\mathbf{W}}\mathbf{TR}(\alpha,\beta,\gamma,t_x,t_y,t_z)\ \ \mathbf{^W P}) \\
&= (\ {}^{\mathbf{F}}_{\mathbf{C}}\mathbf{\Pi}(f)\ \ {}^{\mathbf{C}}_{\mathbf{W}}\mathbf{TR}(\alpha,\beta,\gamma,t_x,t_y,t_z)\ )\ \ \mathbf{^W P}
\end{aligned}$$

$$\begin{bmatrix} s\ {}^F P_x \\ s\ {}^F P_y \\ s \end{bmatrix} = \begin{bmatrix} d_{11} & d_{12} & d_{13} & d_{14} \\ d_{21} & d_{22} & d_{23} & d_{24} \\ d_{31} & d_{32} & d_{33} & 1 \end{bmatrix} \begin{bmatrix} {}^W P_x \\ {}^W P_y \\ {}^W P_z \\ 1 \end{bmatrix} \tag{13.19}$$

The matrix Equation 13.19 uses elements d_{ij} and not c_{ij} because it is not quite the camera matrix, whose derivation was our goal. This is because our derivation thus far has used the same units of real space, say *mm* or *inches*, and has not included the scaling of image points into pixel rows and columns. Scale factors converting *mm* to pixels for the image rows and columns are easily combined with Equation 13.19 to obtain the full camera matrix **C**. Suppose that d_x is the horizontal size and d_y is the vertical size of a pixel in real-valued units. Instead of the real-valued coordinates $[{}^F P_x\ {}^F P_y]$ whose reference coordinate system has [0.0,0.0] at the lower left of the image, we want to go one step further to the integer-valued coordinates $[r, c]$ that refer to the row and column coordinates of a pixel of the image array whose reference coordinate system has [0,0] at the top left pixel of the image. The transformation from the real numbers to pixels, including the reversal of direction of the vertical axis is given by

$$\mathbf{^I P} = \begin{bmatrix} s\ r \\ s\ c \\ s \end{bmatrix} = {}^{\mathbf{I}}_{\mathbf{F}}\mathbf{S} \begin{bmatrix} s\ {}^F P_x \\ s\ {}^F P_y \\ s \end{bmatrix} \tag{13.20}$$

where ${}^{\mathbf{I}}_{\mathbf{F}}\mathbf{S}$ is defined by

$${}^{\mathbf{I}}_{\mathbf{F}}\mathbf{S} = \begin{bmatrix} 0 & -\frac{1}{d_y} & 0 \\ \frac{1}{d_x} & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{13.21}$$

The final result for the full camera matrix that transforms 3D points in the real world to pixels in an image is given by

$$\mathbf{^I p} = (\ {}^{\mathbf{I}}_{\mathbf{F}}\mathbf{S}\ \ {}^{\mathbf{F}}_{\mathbf{C}}\mathbf{\Pi}(f)\ \ {}^{\mathbf{C}}_{\mathbf{W}}\mathbf{TR}(\alpha,\beta,\gamma,t_x,t_y,t_z)\ )\ \ \mathbf{^W P}$$

$$\begin{bmatrix} s\ {}^I P_r \\ s\ {}^I P_c \\ s \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & 1 \end{bmatrix} \begin{bmatrix} {}^W P_x \\ {}^W P_y \\ {}^W P_z \\ 1 \end{bmatrix} \tag{13.22}$$

which is the full camera matrix of Equation 13.14.

Let us review intuitively what we have just done to model the viewing of 3D world points by a camera sensor. First, we have placed the camera so that its coordinate system is aligned completely with the world coordinate system. Next, we rotated the camera (${}^{\mathbf{W}'}_{\mathbf{W}}\mathbf{R}$) into its final orientation relative to $\mathbf{W}$. Then, we translated the camera (${}^{\mathbf{C}}_{\mathbf{W}'}\mathbf{T}$) to view the workspace from the appropriate position. Now all 3D points can be projected to the image plane of the camera using the model of the perspective projection (${}^{\mathbf{F}}_{\mathbf{C}}\mathbf{\Pi}(\mathbf{f})$). Finally, we need to scale the real image coordinates $[{}^F P_x, {}^F P_y]$ and reverse the direction of the vertical axis to get the pixel coordinates $[{}^i P_r, {}^i P_c]$. We have used our transformation notation fully to account for all the steps. It is usually difficult to actually execute this procedure with enough precision in practice using distance and angle measurements to obtain a sufficiently accurate camera matrix $\mathbf{C}$ to use in computations. Instead, a *camera calibration* procedure is used to obtain the camera matrix. While the *form* of the camera matrix is justified by the above reasoning, the actual values of its parameters are obtained by fitting to control points as described below. Before treating calibration, we show some important uses of the camera matrix obtained by it.

Exercise 11

From the arguments of this section, it is easy to see that the form of the camera matrix is as follows.

$$\begin{bmatrix} s\ {}^I P_r \\ s\ {}^I P_c \\ s \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & c_{34} \end{bmatrix} \begin{bmatrix} {}^W P_x \\ {}^W P_y \\ {}^W P_z \\ 1 \end{bmatrix} \tag{13.23}$$

Show that we can derive the 11 parameter form of Equation 13.22 from this 12 parameter form just by scaling the camera matrix by $1/c_{34}$. Verify that the 11 parameter form performs the same mapping of 3D scene points to 2D image points.

13.3.2 Orthographic and Weak Perspective Projections

The orthographic projection of ${}^{\mathbf{C}}\mathbf{P}$ just drops the z-coordinate of the world point: this is equivalent to projecting each world point parallel to the optical axis and onto the image plane. Figure 13.12 compares perspective and orthographic projections. Orthographic projection can be viewed as a perspective projection where the focal length f has gone to infinity as shown in Equation 13.24. Orthographic projection is often used in computer graphics to convey true scale for the cross section of an object. It is also used for development of computer vision theory, because it is simpler than perspective, yet in many cases a good enough approximation to it to test the theory.

$$\begin{aligned} {}^{\mathbf{F}}\mathbf{P} &= {}^{\mathbf{F}}_{\mathbf{C}}\mathbf{\Pi}(\infty)\ {}^{\mathbf{C}}\mathbf{P} \\ \begin{bmatrix} {}^F P_x \\ {}^F P_y \end{bmatrix} &= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} {}^C P_x \\ {}^C P_y \\ {}^C P_z \\ 1 \end{bmatrix} \end{aligned} \tag{13.24}$$

Often, perspective transformation can be nicely approximated by an orthographic projection followed by a uniform scaling in the real image plane. Projecting away the z-coordinate

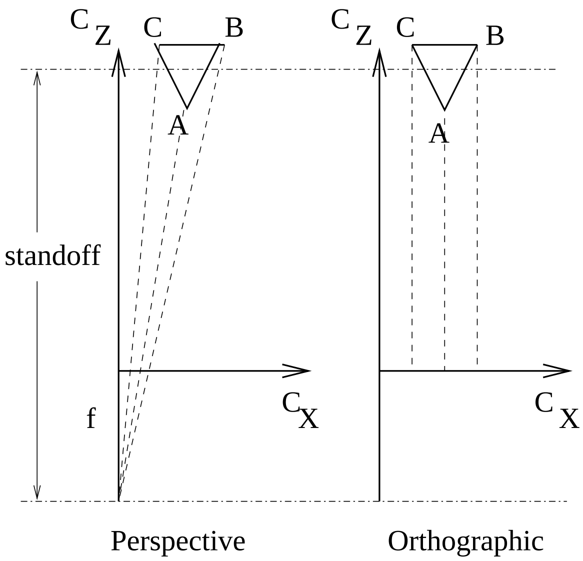


Figure 13.12: Perspective projection (left) versus orthographic (right).

of a point and applying uniform scaling has been called *weak perspective*. A good scale factor is the ratio of the standoff of the object to the focal length of the camera ($s = f/d$ in Figure 13.12).

$$
\begin{aligned}
{}^{\mathbf{F}}\mathbf{P} &= {}^{\mathbf{F}}_{\mathbf{C}}\mathbf{\Pi}(s)\ {}^{\mathbf{C}}\mathbf{P} \\
\begin{bmatrix} {}^{F}P_x \\ {}^{F}P_y \end{bmatrix} &= \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \end{bmatrix} \begin{bmatrix} {}^{C}P_x \\ {}^{C}P_y \\ {}^{C}P_z \\ 1 \end{bmatrix}
\end{aligned} \tag{13.25}
$$

A rule of thumb is that the approximation will be acceptable if the standoff of the object is 20 times the size of the object. The approximation also depends upon how far the object is off the optical axis; the closer the better. As the triangular object of Figure 13.12 moves farther to the right of the optical axis, the perspective images of points A and B will crowd closer together until B will be occluded by A; however, the orthographic images of A and B will maintain their distance. Most robotic or industrial vision systems will attempt to center the object of attention in the field of view. This might also be true of aerial imaging platforms. In these cases, weak perspective might be an appropriate model.

Exercise 12

Derive Equation 13.24 from Equation 13.16 using $f \rightarrow \infty$.

Mathematical derivations and algorithms are usually more easily developed using weak perspective rather than true perspective. The approximation will usually be good enough

Table 13.1: Weak Perspective versus True Perspective

${}^wP_x$	$f=5mm$	$s=5/1000$	$f=20mm$	$s=20/1000$	$f=50$	$s=50/1000$
0	0.000	0.000	0.000	0.000	0.000	0.000
10	0.051	0.050	0.204	0.200	0.510	0.500
20	0.102	0.100	0.408	0.400	1.020	1.000
50	0.255	0.250	1.020	1.000	2.551	2.500
100	0.510	0.500	2.041	2.000	5.102	5.000
200	1.020	1.000	4.082	4.000	10.204	10.000
500	2.551	2.500	10.204	10.000	25.510	25.000
1000	5.102	5.000	20.408	20.000	51.020	50.000

Comparison of values ${}^iP_x$ computed by the perspective transformation ${}^{\mathbf{F}}_{\mathbf{C}}\mathbf{\Pi}_{\mathbf{pers}}(\mathbf{f})$ versus transformation using weak perspective ${}^{\mathbf{F}}_{\mathbf{C}}\mathbf{\Pi}_{\mathbf{weak}}(\mathbf{s})$ with scale $s = f/1000$ for 3D points $[{}^cP_x, 0, 980]^t$. Focal lengths for perspective transformation are $5, 20$ and $50mm$. Weak perspective scale is set at $f/1000$ for a nominal standoff of $1000mm$.

to do the matching required by recognition algorithms. Moreover, a closed form weak perspective solution might be a good starting point for a more complex iterative algorithm using true perspective. Huttenlocher and Ullman have published some fundamental work on this issue. Table 13.1 gives some numerical comparisons between true perspective and weak perspective: the data shows a good approximation within the range of the table.

By substituting into the definition given above, a weak perspective transformation is defined by eight parameters as follows.

$$
\begin{aligned}
{}^{\mathbf{F}}\mathbf{P} &= {}^{\mathbf{F}}_{\mathbf{C}}\pi_{\mathbf{weak}} \; {}^{\mathbf{C}}_{\mathbf{W}}\mathbf{TR} \; {}^{\mathbf{W}}\mathbf{P} \\
\begin{bmatrix} {}^FP_x \\ {}^FP_y \end{bmatrix} &= \begin{bmatrix} s & 0 & 0 & 0 \\ 0 & s & 0 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_x \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} {}^WP_x \\ {}^WP_y \\ {}^WP_z \\ 1 \end{bmatrix} \qquad (13.26) \\
&= \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \end{bmatrix} \begin{bmatrix} {}^WP_x \\ {}^WP_y \\ {}^WP_z \\ 1 \end{bmatrix} \qquad (13.27)
\end{aligned}
$$

Exercise 13

While Equation 13.27 shows eight parameters in the weak perspective transformation, there are only seven independent parameters. What are these seven?

13.3.3 Computing 3D Points Using Multiple Cameras

We show how to use two camera models to compute the unknown 3D point $[x, y, z]$ from its two images $[r_1, c_1]$ and $[r_2, c_2]$ from two calibrated cameras. Since the coordinate system for our points is now clear, we drop the superscripts from our point notation. Figure 13.3 sketches the environment and Equation 13.14 gives the model for each of the cameras. The following imaging equations yield four linear equations in the three unknowns x, y, z.

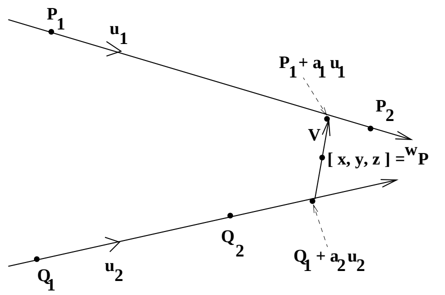
$$
\begin{bmatrix} s\ r_1 \\ s\ c_1 \\ s \end{bmatrix} = \begin{bmatrix} b_{11} & b_{12} & b_{13} & b_{14} \\ b_{21} & b_{22} & b_{23} & b_{24} \\ b_{31} & b_{32} & b_{33} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}
$$

$$
\begin{bmatrix} t\ r_2 \\ t\ c_2 \\ t \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} & c_{14} \\ c_{21} & c_{22} & c_{23} & c_{24} \\ c_{31} & c_{32} & c_{33} & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \quad (13.28)
$$

Eliminating the homogeneous coordinate s and t from Equations 13.28, we obtain the following four linear equations in the three unknowns.

$$
\begin{aligned}
r_1 &= (b_{11} - b_{31} r_1)x + (b_{12} - b_{32} r_1)y + (b_{13} - b_{33} r_1)z + b_{14} \\
c_1 &= (b_{21} - b_{31} c_1)x + (b_{22} - b_{32} c_1)y + (b_{23} - b_{33} c_1)z + b_{24} \\
r_2 &= (c_{11} - c_{31} r_2)x + (c_{12} - c_{32} r_2)y + (c_{13} - c_{33} r_2)z + c_{14} \\
c_2 &= (c_{21} - c_{31} c_2)x + (c_{22} - c_{32} c_2)y + (c_{23} - c_{33} c_2)z + c_{24} \quad (13.29)
\end{aligned}
$$

Any three of these four equations could be solved to obtain the point $[x, y, z]$; however, each subset of three equations would yield slightly different coordinates. Together, all four simultaneous equations are typically inconsistent, because due to approximation errors in the camera model and image points, the two camera rays will not actually intersect in mathematical 3D space. The better solution is to compute the closest approach of the two skew rays, or equivalently, the shortest line segment connecting them. If the length of this segment is suitably short, then we assign its midpoint as the intersection of the rays, which is the point $[x, y, z]$ as shown in Figure 13.13. If the shortest connecting segment is too long, then we assume that there was some mistake in corresponding the image points $[r_1, c_1]$ and $[r_2, c_2]$.



$\mathbf{P_1}$ and $\mathbf{P_2}$ are points on one line, while $\mathbf{Q_1}$ and $\mathbf{Q_2}$ are points on the other line. $\mathbf{u_1}$ and $\mathbf{u_2}$ are unit vectors along these lines. Vector $\mathbf{V} = \mathbf{P_1} + a_1\mathbf{u_1} - (\mathbf{Q_1} + a_2\mathbf{u_2})$ is the shortest (free) vector connecting the two lines, where a_1, a_2 are two scalars to be determined. a_1, a_2 can be determined using calculus to minimize the length of $\mathbf{V}$; however, it is easy to compute them using the constraints that $\mathbf{V}$ must be orthogonal to both $\mathbf{u_1}$ and $\mathbf{u_2}$.

Figure 13.13: The shortest distance between two skew lines is measured along a connecting line segment that is orthogonal to both lines.

Applying the orthogonality constraints to the shortest connecting segment yields the following two linear equations in the two unknowns a_1, a_2.

$$
\begin{aligned}
((\mathbf{P_1} + a_1\mathbf{u_1}) &- (\mathbf{Q_1} + a_2\mathbf{u_2})) \circ \mathbf{u_1} = 0 \\
((\mathbf{P_1} + a_1\mathbf{u_1}) &- (\mathbf{Q_1} + a_2\mathbf{u_2})) \circ \mathbf{u_2} = 0 \quad (13.30)
\end{aligned}
$$

$$\begin{aligned} 1\ a_1 - (\mathbf{u_1} \circ \mathbf{u_2})\ a_2 &= (\mathbf{Q_1} - \mathbf{P_1})\ \circ\ \mathbf{u_1} \\ (\mathbf{u_1} \circ \mathbf{u_2})\ a_1 - 1\ a_2 &= (\mathbf{Q_1} - \mathbf{P_1})\ \circ\ \mathbf{u_2} \end{aligned} \tag{13.31}$$

These equations are easily solved for a_1, a_2 by elimination of variables or by the method of determinants to obtain these solutions.

$$\begin{aligned} a_1 &= \frac{(\mathbf{Q_1} - \mathbf{P_1})\ \circ\ \mathbf{u_1}\ -\ ((\mathbf{Q_1} - \mathbf{P_1})\ \circ\ \mathbf{u_2})\ \circ\ (\mathbf{u_1} \circ \mathbf{u_2})}{1 - (\mathbf{u_1} \circ \mathbf{u_2})^2} \\ a_2 &= \frac{((\mathbf{Q_1} - \mathbf{P_1})\ \circ\ \mathbf{u_1})(\mathbf{u_1} \circ \mathbf{u_2})\ -\ (\mathbf{Q_1} - \mathbf{P_1})\ \circ\ \mathbf{u_2}}{1 - (\mathbf{u_1} \circ \mathbf{u_2})^2} \end{aligned} \tag{13.32}$$

Provided that $\| \mathbf{V} \|$ is less than some threshold, we report the "intersection" of the two lines as $[x, y, z]^t = (1/2)[(\mathbf{P_1} + a_1\mathbf{u_1}) + (\mathbf{Q_1} + a_2\mathbf{u_2})]$. It's important to go back to the beginning and realize that all the computations depend upon being able to define each of the lines by a pair of points (e.g. $\mathbf{P_1}, \mathbf{P_2}$) on the line. Often, each ray is determined by the optical center of the camera and the image point. If the optical center is not known, a point on the first camera ray can be found by choosing some value $z = z_1$ and then solving the two Equations 13.29 for coordinates x and y. If the ray is nearly parallel with the z-axis, then $x = x_0$ should be chosen. In this manner, the four needed points can be chosen.

Exercise 14

Implement and test a function in your favorite programming language to determine both the distance between two skew lines and the midpoint of the shortest connecting line segment. The function should take four 3D points as input and execute the mathematical formulas in this section.

Exercise 15

Use of Cramer's rule to solve Equations 13.31 for a_1, a_2 requires that the determinant of the matrix of coefficients be nonzero. Argue why this must be the case when two cameras view a single point.

We can use one camera and one projector to sense 3D surfaces. The geometry and mathematics is the same as in the two camera case. The biggest advantage is that a projector can artificially create surface texture on smooth surfaces so that feature points may be defined and put into correspondence. Use of structured light is treated below, after we show how to obtain camera models or projector models via calibration.

13.4 Best Affine Calibration Matrix

The problem of camera calibration is to relate the locations of the pixels in the image array of a given camera to the real-valued points in the 3D scene being imaged. This process generally precedes any image analysis procedures that involve the computation of 3D location and orientation of an object or measurements of its dimensions. It is also needed for the

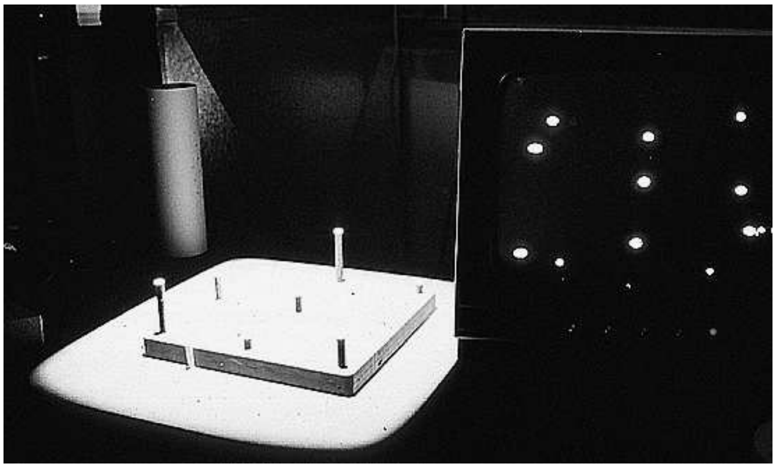


Figure 13.14: The calibration jig at the left has 9 dowels of random height (it can be rotated 3 times to obtain 25 unique calibration points). The image of the jig is on the display at the right.

stereo triangulation procedure described earlier.

It was shown in Section 13.3 that the eleven parameter camera matrix of Equation 13.14 was an appropriate mathematical model. We now show how to derive the values of the eleven parameters using least squares fitting. The camera view and focus are fixed and a calibration object, or *jig*, with known measurements is placed in the scene as in Figure 13.14. A set of n data tuples $< {}^{\mathbf{I}}\mathbf{P}_j,\ {}^{\mathbf{W}}\mathbf{P}_j >$ are then taken: ${}^{\mathbf{I}}\mathbf{P}_j = [{}^IP_r, {}^IP_c]$ is the pixel in the image where 3D point ${}^{\mathbf{W}}\mathbf{P}_j = [{}^WP_x, {}^WP_y, {}^WP_z]$ is observed. The number of points n must be at least 6, but results are better with $n = 25$ or more.

13.4.1 Calibration Jig

The purpose of a *calibration jig* is to conveniently establish some well-defined 3D points. Figures 13.14, 13.18 and 13.22 show three different jigs. The jig is carefully located in the world coordinate system $\mathbf{W}$, or perhaps defines the world coordinate system itself, such that the 3D coordinates $[{}^WP_x, {}^WP_y, {}^WP_z]$ of its features are readily known. The camera then views these features and obtains their 2D coordinates $[{}^IP_r, {}^IP_c]$. Other common types of jigs are rigid frames with wires and beads or rigid boards with special markings.

13.4.2 Defining the Least-Squares Problem

Equation 13.33 was obtained by eliminating the homogeneous scale factor s from the imaging model. We thus have two linear equations modeling the geometry of each imaging ray, and we have one ray for each calibration point. To simplify notation and also to avoid confusion of symbols we use $[x_j, y_j, z_j]$ for the world point ${}^WP_j = [{}^WP_x, {}^WP_y, {}^WP_z]$, and $[u_j, v_j]$ for the image point ${}^IP_j = [{}^IP_r, {}^IP_c]$. For each calibration point, the following two equations

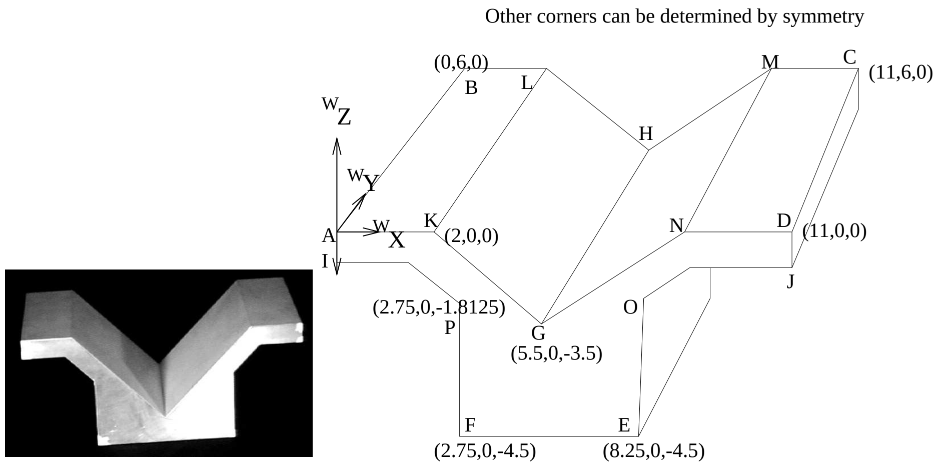


Figure 13.15: Precisely machined jig with many corners; the jig is 11 inches long, 6 inches wide and 4.5 inches high. All 3D corner coordinates are given in Figure 13.18.

are obtained.

$$\begin{aligned} u_j &= (c_{11} - c_{31}u_j)x_j + (c_{12} - c_{32}u_j)y_j + (c_{13} - c_{33}u_j)z_j + c_{14} \\ v_j &= (c_{21} - c_{31}v_j)x_j + (c_{22} - c_{32}v_j)y_j + (c_{23} - c_{33}v_j)z_j + c_{24} \end{aligned} \tag{13.33}$$

We rearrange the equations separating the knowns from the unknowns into vectors. All the entities on the left are known from the calibration tuples, while all the c_{km} on the right are unknowns to be determined.

$$\begin{bmatrix} x_j, y_j, z_j, 1, 0, 0, 0, 0, -x_j u_j, -y_j u_j, -z_j u_j \\ 0, 0, 0, 0, x_j, y_j, z_j, 1, -x_j v_j, -y_j v_j, -z_j v_j \end{bmatrix} \begin{bmatrix} c_{11} \\ c_{12} \\ c_{13} \\ c_{14} \\ c_{21} \\ c_{22} \\ c_{23} \\ c_{24} \\ c_{31} \\ c_{32} \\ c_{33} \end{bmatrix} = \begin{bmatrix} u_j \\ v_j \end{bmatrix} \tag{13.34}$$

Since each imaging ray gives two such equations, we obtain $2n$ linear equations from n calibration points, which can be compactly represented in the classic matrix form where $\mathbf{x}$ is the column vector of unknowns and $\mathbf{b}$ is the column vector of image coordinates.

$$\mathbf{A}_{2n\times 11}\ \mathbf{x}_{11\times 1}\ \approx\ \mathbf{b}_{2n\times 1} \tag{13.35}$$

Since there are 11 unknowns and 12 or more equations, this is an overdetermined system. There is no vector of parameters $\mathbf{x}$ for which all the equations hold: a least squares solution is appropriate. We want the set of parameters such that the sum (over all equations)

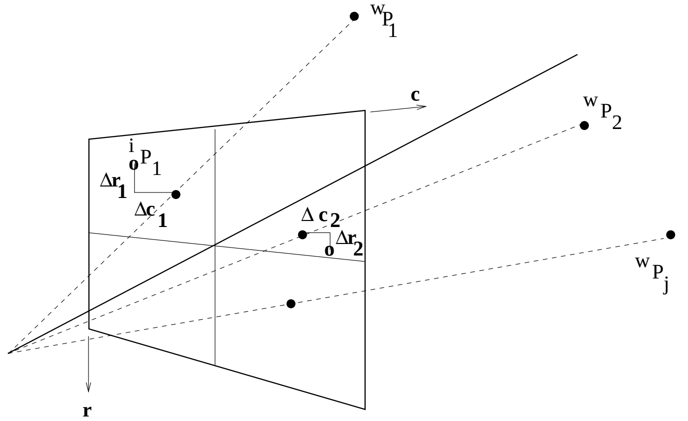


Figure 13.16: Image plane residuals are the differences between the actual observed image points (open dots) and the points computed using the camera matrix of Equation 13.14 (filled dots).

of the squared differences between the observed coordinate and the coordinate predicted by the camera matrix is minimized. Figure 13.16 shows four of these differences for two calibration points. These differences are called *residuals* as in Chapter 11. Figure 13.17 gives an abstract representation of the least squares solution: we want to compute the c_{km} that give a linear combination of the columns of $\mathbf{A}$ closest to $\mathbf{b}$. The key to solving this problem is the observation that the residual vector $\mathbf{r} = \mathbf{b} - \mathbf{A}\mathbf{x}$ is orthogonal to the column space of $\mathbf{A}$ yielding $\mathbf{A}^{\mathbf{t}}\mathbf{r} = \mathbf{0}$. Substituting $\mathbf{b} - \mathbf{A}\mathbf{x}$ for $\mathbf{r}$, we obtain $\mathbf{A}^{\mathbf{t}}\mathbf{A}\mathbf{x} = \mathbf{A}^{\mathbf{t}}\mathbf{b}$. $\mathbf{A}^{\mathbf{t}}\mathbf{A}$ is symmetric and positive definite, so it has an inverse, which can be used to solve for $\mathbf{x} = (\mathbf{A}^{\mathbf{t}}\mathbf{A})^{-1}\mathbf{A}^{\mathbf{t}}\mathbf{b}$. Several common libraries of numerical methods are available to solve this. (Using MATLAB, the solution is invoked using the simple statement $\mathsf{X} = \mathsf{A} \setminus \mathsf{B}$. Once the least squares solution X is found, the vector R of residuals is computed as $\mathsf{R} = \mathsf{B} - \mathsf{AX}$.) Consult the Heath (1997) reference or the users manual for your local linear algebra library.)

Figure 13.18 shows example results of computing the camera matrix for a camera viewing the calibration jig shown in Figure 13.15. The corners of the jig are labeled 'A' to 'P' and their world coordinates [X,Y,Z] are given in Figure 13.18 alongside the image coordinates [U,V] where the camera images them. Corner points 'B', 'C' and 'M' are occluded in this view, so no image coordinates are available. The camera matrix obtained by fitting the 13 correspondences is given at the bottom of the figure and the residuals are shown at the right. 16 of the 26 coordinates computed by the camera matrix applied to the 3D points are within one pixel of the observed image coordinate, 10 are more than one pixel different, but only 2 are off by two pixels. This example supports the validity of the affine camera model, but also exhibits the errors due to corner location and the distortion of a short focal length lens.

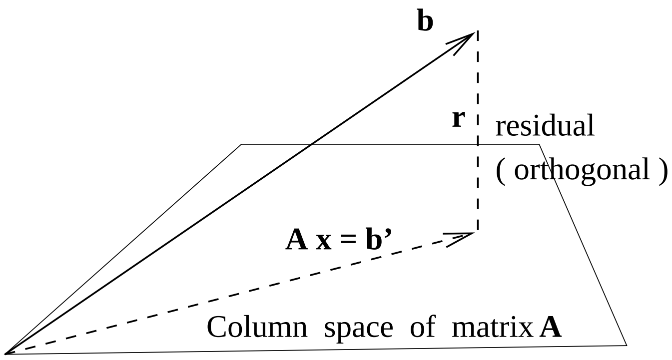


Figure 13.17: Least squares solution of the system $\mathbf{Ax} \approx \mathbf{b}$. The plane represents the 11-dimensional column space of matrix $\mathbf{A}_{2n\ x\ 11}$. All linear combinations $\mathbf{Ax}$ must be in this space, but $\mathbf{B}_{2n\ x\ 1}$ will not be in it. The least squares solution computes $\mathbf{b}'$ as the projection of $\mathbf{b}$ onto the 11D space, making $\mathbf{b}'$ the closest point in the space to $\mathbf{b}$.

```
#
#  IMAGE: g1view1.ras
#
#                 INPUT DATA                         |              OUTPUT DATA
#                                                    |
Point  Image 2-D (U,V)     3-D Coordinates (X,Y,Z)   |    2-D Fit Data       Residuals X Y
                                                     |
 A      95.00   336.00     0.00    0.00    0.00      |   94.53   337.89  |     0.47   -1.89
 B                         0.00    6.00    0.00      |                   |
 C                        11.00    6.00    0.00      |                   |
 D     592.00   368.00    11.00    0.00    0.00      |  592.21   368.36  |    -0.21   -0.36
 E     472.00   168.00     8.25    0.00   -4.50      |  470.14   168.30  |     1.86   -0.30
 F     232.00   155.00     2.75    0.00   -4.50      |  232.30   154.43  |    -0.30    0.57
 G     350.00   205.00     5.50    0.00   -3.50      |  349.17   202.47  |     0.83    2.53
 H     362.00   323.00     5.00    6.00   -3.50      |  363.44   324.32  |    -1.44   -1.32
 I      97.00   305.00     0.00    0.00   -0.75      |   97.90   304.96  |    -0.90    0.04
 J     592.00   336.00    11.00    0.00   -0.75      |  591.78   334.94  |     0.22    1.06
 K     184.00   344.00     2.00    0.00    0.00      |  184.46   343.40  |    -0.46    0.60
 L     263.00   431.00     2.00    6.00    0.00      |  261.52   429.65  |     1.48    1.35
 M                         9.00    6.00    0.00      |                   |
 N     501.00   363.00     9.00    0.00    0.00      |  501.16   362.78  |    -0.16    0.22
 O     467.00   279.00     8.25    0.00   -1.81      |  468.35   281.09  |    -1.35   -2.09
 P     224.00   266.00     2.75    0.00   -1.81      |  224.06   266.43  |    -0.06   -0.43


#               CALIBRATION MATRIX

    44.84          29.80        -5.504      94.53

     2.518         42.24         40.79     337.9

    -0.0006832      0.06489     -0.01027     1.000
```

Figure 13.18: Camera calibration output using the jig in Figure 13.15.

Exercise 16 replication of camera model derivation

(a) Locate software to perform least-squares fitting. Enter the point correspondences from Figure 13.18 and compute the camera matrix: compare it to the matrix shown in Figure 13.18. (b) Delete three points with the largest residuals and derive a new camera matrix. Are any of the new residuals worse than 2 pixels? (c) Define the 3D coordinates of a 1 x 1 x 1 cube that would rest on one of the top horizontal surfaces of the jig shown in Figure 13.15. Transform each of the eight corners of the cube into image coordinates by using the derived camera matrix. Also transform the four points defining the surface on which the cube rests. Plot the images of the transformed points and draw the connecting edges. Does the image of the cube make sense?

Exercise 17 subpixel accuracy

Refer to Figure 13.14. The center of the dowels can be computed to subpixel accuracy using the methods of Chapter 3. How? Can we use these noninteger coordinates for $[{}^I P_r, {}^I P_c]$ for the calibration data? How?

Exercise 18 best weak perspective camera model

Find the best weak perspective camera matrix to fit the data at the left in Figure 13.18. Carefully go back over the derivation of the simpler imaging equations to develop a new system of equations $\mathbf{A}\ \mathbf{x} = \mathbf{b}$. After obtaining the best camera matrix parameters, compare its residuals with those at the right in Figure 13.18.

Table 13.2: **3D feature points of jig imaged with two cameras**. 3D world coordinates ${}^{\mathbf{w}}\mathbf{x}, {}^{\mathbf{w}}\mathbf{y}, {}^{\mathbf{w}}\mathbf{z}$ are in inches. Coordinates of image 1 are ${}^{\mathbf{1}}\mathbf{u}$ and ${}^{\mathbf{1}}\mathbf{v}$ and are in row and column units. Coordinates of image 2 are ${}^{\mathbf{2}}\mathbf{u}$ and ${}^{\mathbf{2}}\mathbf{v}$.

Point	${}^{\mathbf{w}}\mathbf{x}$	${}^{\mathbf{w}}\mathbf{y}$	${}^{\mathbf{w}}\mathbf{z}$	${}^{\mathbf{1}}\mathbf{u}$	${}^{\mathbf{1}}\mathbf{v}$	${}^{\mathbf{2}}\mathbf{u}$	${}^{\mathbf{2}}\mathbf{v}$
A	0.0	0.0	0.0	167	65	274	168
B	0.0	6.0	0.0	96	127	196	42
C	11.0	6.0	0.0	97	545	96	431
D	11.0	0.0	0.0	171	517	154	577
E	8.25	0.0	-4.5	352	406	366	488
F	2.75	0.0	-4.5	347	186	430	291
G	5.5	0.0	-3.5	311	294	358	387
H	5.5	6.0	-3.5	226	337	NA	NA
I	0.0	0.0	-0.75	198	65	303	169
J	11.0	0.0	-0.75	203	518	186	577
K	2.0	0.0	0.0	170	143	248	248
L	2.0	6.0	0.0	96	198	176	116
M	9.0	6.0	0.0	97	465	114	363
N	9.0	0.0	0.0	173	432	176	507
O	8.25	0.0	-1.81	245	403	259	482
P	2.75	0.0	-1.81	242	181	318	283

Exercise 19 camera calibration

Table 13.2 shows the image points recorded for 16 3D corner points of the jig from Figure 13.15. In fact, coordinates in two separate images are recorded. Using the affine calibration procedure, compute a camera matrix using the 5-tuples from table columns 2-6.

Exercise 20 stereo computation

(a) Using the data in Table 13.2, compute two calibration matrices, one from columns 2-6 and one from columns 2-4 and 7-8. (b) Using the method of Section 13.3.3, compute the 3D coordinates of point **A** using only the two camera matrices and the image coordinates in columns 5-8 of the Table. Compare your result to the coordinates in Columns 2-4 of the table. (c) Consider the obtuse corner point between corner points **I** and **P** of the jig; call it point **Q**. Suppose point **Q** images at [196,135] and [281,237] respectively. Use the stereo method to compute the 3D coordinates of world point **Q** and verify that your results are reasonable.

13.4.3 Discussion of the Affine Method

The major issue relates to whether or not there really are 11 camera model parameters to estimate. As we have seen, there are only 3 independent parameters of a rotation matrix and 3 translational parameters defining the camera pose in world coordinates. There are 2 scaling factors relating real image coordinates to pixel rows and columns and focal length f. Thus, not all 11 parameters are independent. Treating them as independent means that the rotation parameters will not define an orthonormal rotation matrix. In case of a precisely built camera, we waste constraints; however, if the image plane might not be perpendicular to the optical axis, the increased parameters can produce a better model. We need more calibration points to estimate more free parameters, and the parameters that are derived do not explicitly yield the intrinsic properties of the camera. The affine method has several advantages, however — it works well even with skew between image rows and columns or between image plane and optical axis, it works with either pixel coordinates or real image coordinates, and the solution can be quickly computed without iteration. In section 13.7, we will introduce a different calibration method that uses more constraints and overcomes some of the problems mentioned.

Exercise 21 Calibrate your home camera

If you don't have a simple film camera, borrow one or buy a cheap disposable one. Obtain a rigid box to use as a jig: draw a few 'X's on each face. Measure the [x,y,z] coordinates of each corner and each 'X' relative to a RH coordinate system with origin at one corner of the box and axes along its edges. Take and develop a picture of the box. Identify and mark 15 corners and 'X's in the picture. Measure the coordinates of these points in inches using a rule. Following the example of this section, derive the camera matrix and residuals and report on these results. Check how well the camera matrix works for some points that were not used in deriving it. Repeat the experiment using *mm* units for the picture coordinates.

Exercise 22 "Texture map" your box

"Texture map" your picture onto the box of the previous exercise. (a) First, create a .pgm image file of your box as above. (b) Using the methods of Chapter 11, create a mapping from one face of the box (in 2D coordinates) to an image array containing your own picture. (c) Update the .pgm image file of your box by writing pixels from your picture into it. Hint: mapping two triangles rather than one parallelogram will produce a better result. Why?

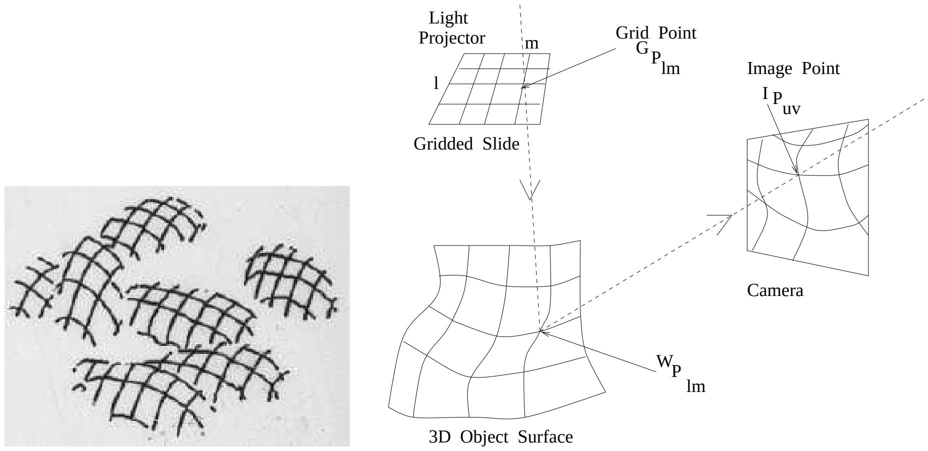


Figure 13.19: (Left) Potatoes illuminated by a slide containing a grid of lines. (Right) Structured light concept: if the imaging system can deduce which ray ${}^G P_{lm}$ created a certain bright feature imaged at ${}^I P_{uv}$, then four equations are available to solve for 3D surface point ${}^W P_{lm}$. Camera and projector matrices must both be available.

13.5 Using Structured Light

Sensing via *structured light* was motivated in the first section and illustrated in Figure 13.4. We now have all the mathematical tools to implement it. Figure 13.19 gives a more detailed view. Object surfaces are illuminated by a pattern of light: in this case a slide projector projects a regular grid of bright lines on surfaces. The camera then senses the result as a grid distorted by the surface structure and its pose. Since the grid of light is highly structured, the imaging system has strong information about which projector rays created the intersections that it sensed. Assume for the moment that the imaging system knows that grid intersect ${}^G P_{lm}$ is imaged at ${}^I P_{uv}$. There are then four linear equations to use to solve for the 3D surface point ${}^W P_{lm}$ being illuminated by the special light pattern. We must have both the camera calibration matrix $\mathbf{C}$ and the projector calibration matrix $\mathbf{D}$. The solution of the system $\mathbf{D}^W P_{lm} =^G P_{lm}$ and $\mathbf{C}^W P_{lm} =^I P_{uv}$ is the same as previously given for the general case of two camera stereo in Section 13.3.3.

A projector can be calibrated in much the same way as a camera. The projector is turned on so that it illuminates the planer surface of the worktable. Precise blocks are located on the table so that one corner exactly intercepts one of the projected grid intersects. A calibration tuple $< [{}^G P_l, {}^G P_m], [{}^W P_x, {}^W P_y, {}^W P_z] >$ is obtained, where ${}^G P_l, {}^G P_m$ are the ordinals (integers) of the grid lines defining the intersection and ${}^W P_x, {}^W P_y, {}^W P_z$ are the

Compute mesh of 3D points from image of scene with light stripes.

Offline Procedure:

1. Calibrate the camera to obtain camera matrix $\mathbf{C}$.
2. Calibrate the projector to obtain projector matrix $\mathbf{D}$.

Online Procedure:

1. Input camera and projector matrices $\mathbf{C}$, $\mathbf{D}$.
2. Input image of scene surface with light stripes.
3. Extract grid of bright stripes and intersections.
4. Determine the labels l, m of all grid intersections.
5. For each projected point P_{lm} imaged at P_{uv} compute 3D surface point P using $\mathbf{C}, \mathbf{D}$ and stereo equations.
6. Output mesh as a graph with vertices as 3D points and edges as their connections by bright stripes.

Algorithm 2: Computing 3D surface coordinates using calibrated camera and projector.

measured world coordinates of the corner of the block. We note in passing that by using the affine calibration procedure above, we can simply order the grid lines of the slide as $m = 1, 2, \cdots$ or $l = 1, 2, \cdots$ because the procedure can adapt to any scale factor.

Exercise 23

Create and calibrate a structured light projector in the following manner. Using your favorite image editing tool, create a digital image of a bright grid on a dark background; or, draw a pattern on paper and digitize it on a scanner. Connect a laptop to a projector and display your digital image; this should project your grid into space. Pose the projector so that it illuminates a tabletop workspace. Place precise blocks on the table to obtain calibration points and perform the affine calibration procedure. Report on the results.

The correspondence problem is still present as in two-camera stereo, although not quite as severe. Referring to the image of the potatoes, it is clear that there is a problem for the imaging system to determine the exact grid intersections seen. If only surface *shape* is needed and not location, it usually only matters that grid intersections are identified consistently relative to each other. See Figure 13.20 and the references by Hu and Stockman and by Shrikhande and Stockman (1989). Various engineering solutions have been implemented; for example, coding grid lines using variations of shape or color. Another solution is to change the grid pattern rapidly over time with the imaging system taking multiple images from which grid patterns can be uniquely determined. White light projectors have a limted depth-of-field where the grid pattern is sharp. Laser light projectors do not suffer

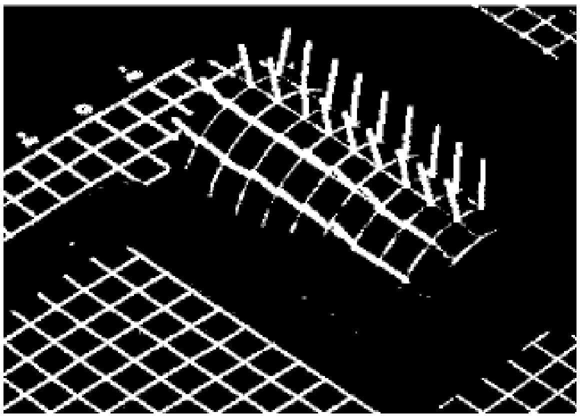


Figure 13.20: Surface normals computed using a projected grid: normals can be computed by assuming the grid lines are in sequence but exact grid lines are not needed. (Weak perspective models were used for both projector and camera.) See Shrikhande 1999.

from this limitation but the reflection off certain objects will be poor due to the low power and limited spectrum of a typical laser. In many controlled sensing environments structured light sensors are highly effective. Off-the-shelf sensor units can be purchased from several companies; some may have only one ray, one stripe, or perhaps two orthogonal stripes.

13.6 A Simple Pose Estimation Procedure

We want to use cameras to compute object geometry and pose. In this section we study a simple method of computing the pose of an object from three points of the image of that object. It is assumed that a geometric model of the object is known and that the focal length f of the camera is known. We study this simple method because it not only gives a practical means of computing pose, but because it also introduces some important concepts in a simple context. One of these is the idea of *inverse perspective* — computing 3D properties from properties in the 2D image plane of a perspective transformation. Another is the idea of using optimization to find a best set of parameters that will match 3D object points to 2D image points. Our most important simplifying assumption is that the world coordinate system is the same as the camera coordinate system, so we omit superscripts of points because they are not needed to indicate coordinate system (${}^{\mathbf{W}}\mathbf{P}_j = {}^{\mathbf{C}}\mathbf{P}_j \equiv P_j$). Another simplification is that we will work only in real geometrical space — we use no image quantization, or pixel coordinates.

The environment for the *Perspective 3 Point Problem* (**P3P**) is shown in Figure 13.21. Three 3D scene points P_i are observed in the u-v image plane as Q_i. The coordinates of the points P_i are the unknowns for which we want to solve. Since we assume that we know what points of an object model we are observing (a big assumption), we do know the distances between pairs of points: for a rigid object, these three distances do not vary as the object moves in space. In an application of human-computer interaction (HCI), the points P_i could be facial features of a specific person, such as the eyes and tip of the

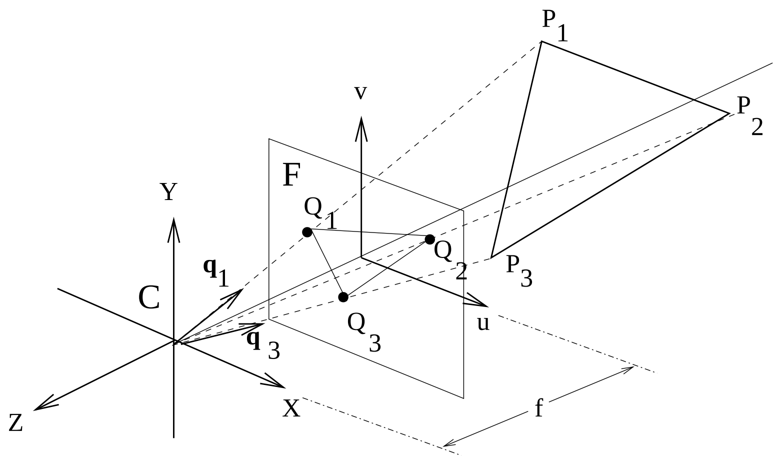


Figure 13.21: A simple case for pose estimation: a triangle $P_1P_2P_3$ with sides of known length in 3D is observed in the u-v image plane as triangle $Q_1Q_2Q_3$. The 3D positions of points P_j can be computed from the observed image points Q_j. The pose of an object containing the P_j can then be determined relative to the camera coordinate system. The focal length f is the distance from the center of projection to the image plane. Coordinates of the image points are $Q_j = [u_j, v_j, -f]$ relative to the camera coordinate system **C**.

nose. The face can be measured for the required distances. Computing the pose of the face will reveal where the person is looking. In a navigation application, the three points P_i could be geographic landmarks whose locations are known in a map. A navigating robot or drone can compute its own position relative to the landmarks using the following method.

The image locations of the observed points Q_i are known. Let $\mathbf{q}_i$ be the unit vector from the origin in the direction of Q_i. The 3D point P_i is also in this same direction. Therefore, we can solve for the locations of P_i from Q_i if we can compute the three scalars a_i such that

$$P_i = a_i \ \mathbf{q}_i \tag{13.36}$$

From the three equations represented in Equation 13.36 we derive three others relating the distances between points, which are known from the model.

$$d_{mn} = \|P_m - P_n\| \ \ (m \neq n) \tag{13.37}$$

We rewrite the P_i in terms of the observed Q_i and compute the 3D distances, using the dot product and the fact that $\mathbf{q_i} \circ \mathbf{q_i} = \mathbf{1}$.

$$\begin{aligned} {d_{mn}}^2 &= \|a_m \mathbf{q_m} - a_n \mathbf{q_n}\|^2 \\ &= (a_m \mathbf{q_m} - a_n \mathbf{q_n}) \circ (a_m \mathbf{q_m} - a_n \mathbf{q_n}) \\ &= {a_m}^2 - 2a_m a_n(\mathbf{q_m} \circ \mathbf{q_n}) + {a_n}^2 \end{aligned} \tag{13.38}$$

We now have three quadratic equations in the three unknowns a_i. The three left sides ${d_{mn}}^2$ are known from the model and the three $\mathbf{q_m} \circ \mathbf{q_n}$ are known from the image points

Q_i. Our P3P problem of computing the positions of the three points P_i is now reduced to solving three quadratic equations in three unknowns. In theory, there can be 8 different triples $[a_1, a_2, a_3]$ that will satisfy Equations 13.38. Extending Figure 13.21, it is easy to see that for each pose of the three points on one side of the coordinate system, there is a mirror set on the other side with parameters $[-a_1, -a_2, -a_3]$, and clearly if one triple satifies the equations, then so must the other. Thus, we can have at most four actual poses possible in a real situation because the object must be on one side of the camera. In Fischler and Bolles (1981) it is shown that four poses are possible in special cases; however, the common case is for two solutions.

We now show how to solve for the unknowns a_i (and hence P_i) using nonlinear optimization. This will give insight to other optimizations used in later sections. Mathematically, we want to find three roots of the following functions of the a_i.

$$\begin{aligned}
f(a_1, a_2, a_3) &= a_1{}^2 - 2a_1a_2(\mathbf{q_1} \circ \mathbf{q_2}) + a_2{}^2 - d_{12}{}^2 \\
g(a_1, a_2, a_3) &= a_2{}^2 - 2a_2a_3(\mathbf{q_2} \circ \mathbf{q_3}) + a_3{}^2 - d_{23}{}^2 \\
h(a_1, a_2, a_3) &= a_1{}^2 - 2a_1a_3(\mathbf{q_1} \circ \mathbf{q_3}) + a_3{}^2 - d_{13}{}^2
\end{aligned} \tag{13.39}$$

Suppose that we are near a root $[a_1, a_2, a_3]$ but that $f(a_1, a_2, a_3) \neq 0$. We want to compute some changes $[\Delta_1, \Delta_2, \Delta_3]$ so that, ideally, $f(a_1 + \Delta_1, a_2 + \Delta_2, a_3 + \Delta_3) = 0$, and in reality, it moves toward 0. We can linearize f in the neighborhood of $[a_1, a_2, a_3]$ and then compute the changes $[\Delta_1, \Delta_2, \Delta_3]$ needed to produce 0.

$$f(a_1 + \Delta_1, a_2 + \Delta_2, a_3 + \Delta_3) = f(a_1, a_2, a_3) + \begin{bmatrix} \frac{\partial f}{\partial a_1} & \frac{\partial f}{\partial a_2} & \frac{\partial f}{\partial a_3} \end{bmatrix} \begin{bmatrix} \Delta_1 \\ \Delta_2 \\ \Delta_3 \end{bmatrix} + \mathbf{h.o.t.} \tag{13.40}$$

By ignoring the higher order terms in Equation 13.40 and setting the left side to zero, we obtain one linear equation in the unknowns $[\Delta_1, \Delta_2, \Delta_3]$. Using the same concept for the functions g and h, we arrive at the following matrix equation.

$$\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} f(a_1, a_2, a_3) \\ g(a_1, a_2, a_3) \\ h(a_1, a_2, a_3) \end{bmatrix} + \begin{bmatrix} \frac{\partial f}{\partial a_1} & \frac{\partial f}{\partial a_2} & \frac{\partial f}{\partial a_3} \\ \frac{\partial g}{\partial a_1} & \frac{\partial g}{\partial a_2} & \frac{\partial g}{\partial a_3} \\ \frac{\partial h}{\partial a_1} & \frac{\partial h}{\partial a_2} & \frac{\partial h}{\partial a_3} \end{bmatrix} \begin{bmatrix} \Delta_1 \\ \Delta_2 \\ \Delta_3 \end{bmatrix} \tag{13.41}$$

The matrix of partial derivatives is the Jacobian matrix $\mathbf{J}$: if it is invertible at the point $[a_1, a_2, a_3]$ of our search then we obtain the following solution for the changes to these parameters.

$$\begin{bmatrix} \Delta_1 \\ \Delta_2 \\ \Delta_3 \end{bmatrix} = -\mathbf{J}^{-1}(a_1, a_2, a_3) \begin{bmatrix} f(a_1, a_2, a_3) \\ g(a_1, a_2, a_3) \\ h(a_1, a_2, a_3) \end{bmatrix} \tag{13.42}$$

We can compute a new vector of parameters by adding these changes to the previous parameters. We use A^k to denote the parameters $[a_1, a_2, a_3]$ at the k-th iteration and arrive at a familiar form of Newton's method. $\mathbf{f}$ represents the vector of values computed using functions f, g, and h.

$$A^{k+1} = A^k - \mathbf{J}^{-1}(A^k)\mathbf{f}(A^k) \tag{13.43}$$

Exercise 24 defining the Jacobian

Show that the Jacobian for the function $\mathbf{f}(a_1, a_2, a_3)$ is as follows, where t_{mn} denotes the dot product $\mathbf{q}_m \circ \mathbf{q}_n$.

$$\mathbf{J}(a_1, a_2, a_3) \equiv \begin{bmatrix} J_{11} & J_{12} & J_{13} \\ J_{21} & J_{22} & J_{23} \\ J_{31} & J_{32} & J_{33} \end{bmatrix}$$

$$= \begin{bmatrix} (2a_1 - 2t_{12}a_2) & (2a_2 - 2t_{12}a_1) & 0 \\ 0 & (2a_2 - 2t_{23}a_3) & (2a_3 - 2t_{23}a_2) \\ (2a_1 - 2t_{31}a_3) & 0 & (2a_3 - 2t_{31}a_1) \end{bmatrix}$$

Exercise 25 computing the inverse of the Jacobian

Using the symbols J_{ij} from the previous exercise, derive a representation for the inverse Jacobian $\mathbf{J}^{-1}$.

Algorithm 3 summarizes the method for computing the positions of the three 3D model points in terms of the camera coordinate system. Experience has shown that the algorithm converges within 5 to 10 iterations under typical situations. However, it is not clear how to control the algorithm so that multiple solutions can be obtained. Nonlinear optimization is sometimes as much art as algorithm and the reader should consult the references for its many nuances. Table 13.3 shows critical data during the iterations of a P3P solution. Simulated P_i were projected using focal length $f = 30$ to obtain simulated image coordinates Q_i. Beginning parameters were set well away from the true values. After taking some crude steps, the algorithm homes into the neighborhood of the true solution and outputs the same P_i as input, accurate to within two decimal places. As shown in columns 3-5 of Table 13.3, after the 9-th iteration, the difference between the model side length and the computed side length is less than 0.2 units. Starting with each $a_i \approx 100$ halves the number of iterations. If the standoff of an object from the camera is approximately known, then this is a good starting value for each parameter.

Ohmura *et al* (1988) built a system that could compute the position of a person's head 10 times per second. For the model feature points ${}^{\mathbf{M}}\mathbf{P}_j$, blue dots were placed on the face just left of the left eye, just right of the right eye, and just under the nose. (These points will not deform much with various facial expressions.) With the blue makeup the image points ${}^{\mathbf{F}}\mathbf{Q}_j$ could be detected rapidly and robustly. The pose of the face could then be defined by the computed points ${}^{\mathbf{C}}\mathbf{P}_j$ and the transformation mapping the ${}^{\mathbf{M}}\mathbf{P}_j$ to the ${}^{\mathbf{C}}\mathbf{P}_j$ (using Algorithm 1). Ballard and Stockman (1995) developed a system that located the eyes and nose on a face without any makeup, but performance was much slower due to the processing needed to identify the eyes and nose. Both groups reported that the error in the normal vector of the plane formed by the three points is of the order of a few degrees. If

Compute the position of 3 3D points in space from 3 image points.

Input three pairs $({}^M P_i, {}^F Q_i)$ of corresponding points from 3D and 2D.
Each ${}^M P_i$ is in model coordinates; ${}^F Q_i$ is in real image coordinates.
Input the focal length f of camera and tolerance Δ on distance.
Output the positions ${}^C P_i$ of the three model points relative to the camera.

1. **initialize:**
   - (a) From model points ${}^M P_i$ compute squared distances ${d_{mn}}^2$
   - (b) From image points ${}^F Q_i$ compute unit vectors $\mathbf{q_i}$ and dot products $2\mathbf{q_m} \circ \mathbf{q_n}$
   - (c) Choose a starting parameter vector $A^1 = [a_1, a_2, a_3]$ (How?)
2. **iterate:** until $\mathbf{f}(A^k) \approx 0$
   - (a) $A^{k+1} = A^k - \mathbf{J}^{-1}(A^k)\mathbf{f}(A^k)$
     - i. $A^k = A^{k+1}$
     - ii. compute $\mathbf{J}^{-1}(A^k)$ if $\mathbf{J}^{-1}$ exists
     - iii. evaluate $\mathbf{f}(A^k) = [f(a_1^k, a_2^k, a_3^k), g(a_1^k, a_2^k, a_3^k), h(a_1^k, a_2^k, a_3^k)]^t$
   - (b) stop when $\mathbf{f}(A^{k+1})$ is within error tolerance Δ of 0
     or stop if number of iterations exceeds limit
3. **compute pose:** From A^{k+1} compute each ${}^C P_i = {a_i}^{k+1}\mathbf{q_i}$

Algorithm 3: Iterative P3P Solution

Table 13.3: **Iterations of P3P solution.** Using focal length $f = 30$, simulated image coordinates Q_j were computed from projecting $P_1 = [-19.05, -30.16, 76.20], P_2 = [-19.05, -7.94, 88.90], P_3 = [0.00, -7.94, 88.90]$. Beginning parameters were set to $A^0 = [300, 300, 300]$ and $\Delta = 0.2$. In nine iterations the P3P program converged to the initial P_j accurate to two decimal places.

It. k	$\|f(A^k)\|$	$\|g(A^k)\|$	$\|h(A^k)\|$	a_1	a_2	a_3
1	6.43e+03	3.60e+03	1.09e+04	1.63e+02	1.65e+02	1.63e+02
2	1.46e+03	8.22e+02	2.48e+03	1.06e+02	1.08e+02	1.04e+02
3	2.53e+02	1.51e+02	4.44e+02	8.19e+01	9.64e+01	1.03e+02
...		...			...	
8	2.68e+00	6.45e-01	5.78e+00	8.414e+01	9.127e+01	8.926e+01
9	5.00e-02	3.87e-02	1.71e-01	8.414e+01	9.126e+01	8.925e+01

It. k	P_{1x}	P_{1y}	P_{1z}	P_{2x}	P_{2y}	P_{2z}	P_{3x}	P_{3y}	P_{3z}
1	-36.9	-58.4	147.6	-34.4	-14.4	160.7	0.0	-14.5	162.4
2	-24.0	-38.0	96.0	-22.5	-9.3	105.2	0.0	-9.3	103.6
...	...			...			...		
8	-19.1	-30.2	76.2	-19.1	-7.9	88.9	0.0	-7.9	88.9
9	-19.1	-30.2	76.2	-19.1	-7.9	88.9	0.0	-7.9	88.9

the plane of the three points ${}^{\mathbf{C}}\mathbf{P}_j$ is nearly normal to the image plane, then small errors in the observed image coordinates ${}^{\mathbf{F}}\mathbf{Q}_j$ can produce large errors in the computed orientation of the plane in 3D. In order to counter this effect, Ohmura *et al* oriented the camera axis about 20 degrees off to the side of the general direction of the face.

Equations 13.38 should always have a solution; thus we can compute the pose of an airplane from three points in the image of a frog! Choosing a good candidate model is important: this might be done by knowing that airplanes are not green or cannot be present, etc. *Verification* of a model is also important: this can be done by projecting more object model points and verifying them in the image. For example, to distinguish between two possible poses of a face, we might look for an ear, chin, and eyebrow after computing pose from the eyes and nose. We examine verification more in the following sections. Also, we will take into consideration that digital image points are identified in pixel coordinates and that radial lens distortion must be modeled along with the perspective projection.

Exercise 26

Discuss how the **P5P** problem can be solved using the **P3P** solution procedure.

13.7 * An Improved Camera Calibration Method

We now describe a calibration method, developed by Roger Tsai (1987), that has been widely used in industrial vision applications. It has been reported that with careful implementation this procedure can support 3D measurements with a precision of 1 part in 4000, which is quite good. Since the general idea of calibration has been carefully developed in Section 13.3,

Exercise 27 * WP3P problem

Acquire a copy of the 1988 paper by Huttenlocher and Ullman, which describes a solution for computing the pose of a rigid configuration of three points from a single weak perspective projection. The solution method is closed form and explicitly produces two solutions, unlike the P3P solution in this chapter. Program the solution method and test it on simulated data obtained by mathematically projecting 3-point configurations to obtain your three correspondences $< P_j, Q_j >$.

we proceed with a simpler notation.

- $\mathbf{P} = [x, y, z]$ is a point in the 3D world coordinate system.
- $\mathbf{p} = [u, v]$ is a point in the real image plane. (One can think of the u axis as horizontal and pointing to the right and the v axis as vertical and pointing upward.)
- $\mathbf{a} = [r, c]$ is a pixel in the image array expressed by two integers; r is the row coordinate and c is the column coordinate of the pixel. (Relative to convention for u and v above, the r axis is vertical and points downward. The c axis is horizontal and points to the right.)

Camera calibration is conceived to be parameter recovery; we are solving for the "camera parameters" that characterize camera geometry and pose. There are two different types of parameters to be recovered:

1. intrinsic parameters
2. extrinsic parameters

13.7.1 Intrinsic Camera Parameters

The *intrinsic* parameters are truly camera parameters, as they depend on the particular device being used. They include the following parameters:

- principal point $[u_0, v_0]$: the intersection of the optical axis with the image plane.
- scale factors $\{d_x, d_y\}$ for the x and y pixel dimensions.
- aspect distortion factor τ_1: a scale factor used to model distortion in the aspect ratio of the camera.
- focal length f: the distance from the optical center to the image plane.
- lens distortion factor (κ_1): a scale factor used to model radial lens distortion.

These definitions refer to the optical center of the camera lens. The camera origin is at this point. The optical axis is the perpendicular to the image plane going through the optical center. The principal point is often, but not always, in the center pixel of the image. The scale factors d_x and d_y represent the horizontal size and vertical size of a single pixel in real world units, such as millimeters. We will assume that u_0, v_0, d_x, d_y, and the aspect distortion factor τ_1 are known for a particular camera. Thus only the focal length f and the lens distortion factor κ_1 will be computed during the calibration process.

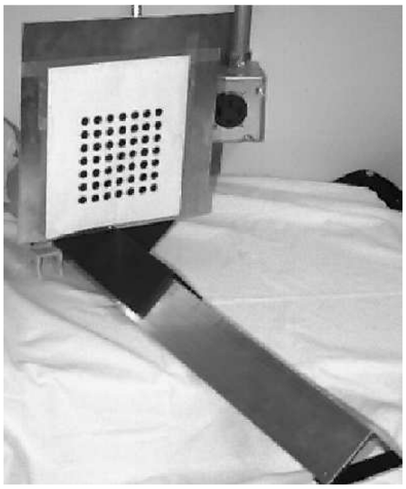


Figure 13.22: A calibration object that uses a moving 2D pattern to create many feature points in 3D space.

13.7.2 Extrinsic Camera Parameters

The *extrinsic* parameters describe the position and orientation (pose) of the camera system in the 3D world. They include:

- translation:

$$\mathbf{t} = [t_x \quad t_y \quad t_z]^T \tag{13.44}$$

- rotation:

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13.45}$$

The translation parameters describe the position of the camera in the world coordinate system, and the rotation parameters describe its orientation. We emphasize at the outset that **there are only three independent rotation parameters and not nine**.

The calibration method developed below is autonomous, reasonably accurate, efficient, and flexible (see Tsai (1987)). Furthermore, it can be used with any off-the-shelf camera and lens, although it may not perfectly model the specific lens chosen. Figure 13.22 shows a calibration object, which was also used in a 3D object reconstruction system to be discussed below. The device is a metal plate onto which have been painted a 7×7 array of black circles. The centers of the circles are used as the distinguished points. The object is mounted on a horizontal rail. It is perpendicular to the rail and can be moved along it in 10mm steps. The position of the rail defines the 3D world coordinate system.

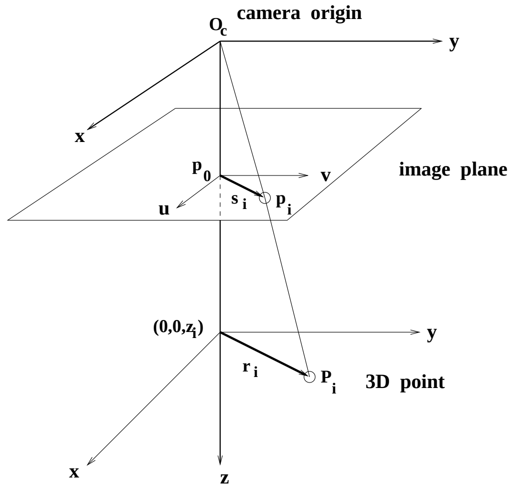


Figure 13.23: The geometric model for the Tsai calibration procedure. Point $\mathbf{p_i} = [u_i, v_i]$ on the image corresponds to point $\mathbf{P_i} = [x_i, y_i, z_i]$ on the calibration object. Point $\mathbf{p_0} = [u_0, v_0]$ is the principal point. Any radial distortion must displace image point $\mathbf{p_i}$ along direction $\mathbf{p_0} - \mathbf{p_i}$ in the image.

In the system shown in Figure 13.22, several images are taken at different positions along the rail corresponding to different distances from the camera. The camera must not be moved nor focused during the time it is being calibrated or used. In each image, the circles are detected and their centers computed. The result of the image processing is a set of correspondences between known points in the 3D world and corresponding points in the 2D images. $n > 5$ correspondences are required; we refer to them as

$$\{(\ [x_i, y_i, z_i],\ [u_i, v_i]\ )\ |\ i = 1, \ldots, n\}.$$

The real-valued image coordinates $[u, v]$ are computed from their pixel position $[r, c]$ by the formulas

$$u = \tau_1 d_x (c - u_0) \tag{13.46}$$

$$v = -d_y (r - v_0) \tag{13.47}$$

where d_x and d_y are the center-to-center distances between pixels in the horizontal and vertical directions, and τ_1 is the scale factor for distortions in the aspect ratio of the camera.

Figure 13.23 illustrates the geometry assumed by the procedure. The point $\mathbf{P_i} = [x_i, y_i, z_i]$ is an arbitrary point in the 3D world. The corresponding point on the image plane is labeled $\mathbf{p_i}$. Vector $\mathbf{r_i}$ is from the point $[0, 0, z_i]$ on the optical axis to the 3D point $\mathbf{P_i}$. Vector $\mathbf{s_i}$ is from the principal point $\mathbf{p_0}$ to the image point $\mathbf{p_i}$. Vector $\mathbf{s_i}$ is parallel to vector $\mathbf{r_i}$. Any radial distortion due to the lens is along $\mathbf{s_i}$.

Tsai made the following observation, which allows most of the extrinsic parameters to be computed in a first step. Since the radial distortion is along vector $\mathbf{s_i}$, the rotation matrix can be determined without considering it. Also, t_x and t_y can be computed without knowing κ_1. Computation of t_z must be done in a second step since change in t_z has an image effect similar to κ_1.

Instead of directly solving for all unknowns, we first solve for a set μ of computed parameters from which the extrinsic parameters can be derived. Given the n point correspondences between $[x_i, y_i, z_i]$ and $[u_i, v_i]$ for $i = 1$ to n, $n > 5$, a matrix $\mathbf{A}$ is formed with rows a_i;

$$a_i = [v_i x_i,\ v_i y_i,\ -u_i x_i,\ -u_i y_i,\ v_i]. \tag{13.48}$$

Let $\mu = [\mu_1, \mu_2, \mu_3, \mu_4, \mu_5]$ be a vector of unknown (computed) parameters defined with respect to the rotation parameters r_{11}, r_{12}, r_{21}, and r_{22} and the translation parameters t_x and t_y as:

$$\mu_1 = \frac{r_{11}}{t_y} \tag{13.49}$$

$$\mu_2 = \frac{r_{12}}{t_y} \tag{13.50}$$

$$\mu_3 = \frac{r_{21}}{t_y} \tag{13.51}$$

$$\mu_4 = \frac{r_{22}}{t_y} \tag{13.52}$$

$$\mu_5 = \frac{t_x}{t_y} \tag{13.53}$$

Let the vector $\mathbf{b} = [u_1, u_2, \ldots, u_n]$ contain the u_i image coordinates of the n correspondences. Since $\mathbf{A}$ and $\mathbf{b}$ are known, the system of linear equations

$$\mathbf{A}\mu = \mathbf{b} \tag{13.54}$$

can be solved for the unknown parameter vector μ. (See *Introduction to Linear Algebra, 2nd edition* by Johnson, Riess, and Arnold for techniques on solving linear systems of equations.) Now μ can be used to compute the rotation and translation parameters as follows.

1. Let $U = \mu_1^2 + \mu_2^2 + \mu_3^2 + \mu_4^2$. Calculate the square of the y component of translation t_y as:

$$t_y^2 = \begin{cases} \frac{U - [U^2 - 4(\mu_1\mu_4 - \mu_2\mu_3)^2]^{1/2}}{2(\mu_1\mu_4 - \mu_2\mu_3)^2} & \text{if } (\mu_1\mu_4 - \mu_2\mu_3) \neq 0 \\ \frac{1}{\mu_1^2 + \mu_2^2} & \text{if } (\mu_1^2 + \mu_2^2) \neq 0 \\ \frac{1}{\mu_3^2 + \mu_4^2} & \text{if } (\mu_3^2 + \mu_4^2) \neq 0 \end{cases} \tag{13.55}$$

2. Let $t_y = (t_y^2)^{1/2}$ (the positive square root) and compute four of the rotation parameters and the x-translation t_x from the known computed value of μ:

$$r_{11} = \mu_1 t_y \tag{13.56}$$

$$r_{12} = \mu_2 t_y \quad (13.57)$$
$$r_{21} = \mu_3 t_y \quad (13.58)$$
$$r_{22} = \mu_4 t_y \quad (13.59)$$
$$t_x = \mu_5 t_y \quad (13.60)$$

3. To determine the true sign of t_y, select an object point $\mathbf{P}$ whose image coordinates $[u, v]$ are far from the image center (to avoid numerical problems). Let $\mathbf{P} = [x, y, z]$, and compute

$$\xi_x = r_{11}x + r_{12}y + t_x \quad (13.61)$$
$$\xi_y = r_{21}x + r_{22}y + t_y \quad (13.62)$$

This is like applying the computed rotation parameters to the x and y coordinates of $\mathbf{P}$. If ξ_x has the same sign as u and ξ_y has the same sign as v, then t_y already has the correct sign, else negate it.

4. Now the remaining rotation parameters can be computed as follows:

$$r_{13} = (1 - r_{11}^2 - r_{12}^2)^{1/2} \quad (13.63)$$
$$r_{23} = (1 - r_{21}^2 - r_{22}^2)^{1/2} \quad (13.64)$$
$$r_{31} = \frac{1 - r_{11}^2 - r_{12}r_{21}}{r_{13}} \quad (13.65)$$
$$r_{32} = \frac{1 - r_{21}r_{12} - r_{22}^2}{r_{23}} \quad (13.66)$$
$$r_{33} = (1 - r_{31}r_{13} - r_{32}r_{23})^{1/2} \quad (13.67)$$

The orthonormality constraints of the rotation matrix $\mathbf{R}$ have been used in deriving these equations. The signs of r_{23}, r_{31}, and r_{32} may not be correct due to the ambiguity of the square root operation. At this step, r_{23} should be negated if the sign of

$$r_{11}r_{21} + r_{12}r_{22}$$

is positive, so that the orthogonality of the rotation matrix is preserved. The other two may need to be adjusted after computing the focal length.

5. The focal length f and the z component of translation t_z are now computed from a second system of linear equations. First a matrix $\mathbf{A}'$ is formed whose rows are given by

$$a_i' = (r_{21}x_i + r_{22}y_i + t_y,\ v_i) \quad (13.68)$$

where d_y is the center-to-center distance between pixels in the y direction.

Next a vector $\mathbf{b}'$ is constructed with rows defined by

$$b_i' = (r_{31}x_i + r_{32}y_i)vi. \quad (13.69)$$

We solve the linear system

$$\mathbf{A}'\mathbf{v} = \mathbf{b}' \quad (13.70)$$

for $\mathbf{v} = (f, t_z)^T$. We obtain only estimates for f and t_z at this point.

6. If $f < 0$ then change the signs of r_{13}, r_{23}, r_{31}, r_{32}, f, and t_z. This is to force the use of a right-handed coordinate system.

7. The estimates for f and t_z can be used to compute the lens distortion factor κ_1 and to derive improved values for f and t_z. The simple distortion model used here is that the true image coordinates $[\hat{u}, \hat{v}]$ are obtained from the measured ones by the following equations:

$$\hat{u} = u(1 + \kappa_1 r^2) \tag{13.71}$$
$$\hat{v} = v(1 + \kappa_1 r^2) \tag{13.72}$$

where the radius of distortion r is given by

$$r = (u^2 + v^2))^{1/2} \tag{13.73}$$

Using the perspective projection equations, modified to include the distortion, we derive a set of nonlinear equations of the form

$$\left\{ v_i(1 + \kappa_1 r^2) = f\frac{r_{21}x_i + r_{22}y_i + r_{23}z_i + t_y}{r_{31}x_i + r_{32}y_i + r_{33}z_i + t_z} \right\} \quad i = 1, \ldots, n \tag{13.74}$$

Solving this system by nonlinear regression gives the values for f, t_z, and κ_1.

13.7.3 Calibration Example

To see how this camera calibration procedure works, we will go through an example. The following table gives five point correspondences input to the calibration system. The units for both the world coordinate system and the **u-v** image coordinate system are centimeters.

i	x_i	y_i	z_i	u_i	v_i
1	0.00	5.00	0.00	-0.58	0.00
2	10.00	7.50	0.00	1.73	1.00
3	10.00	5.00	0.00	1.73	0.00
4	5.00	10.00	0.00	0.00	1.00
5	5.00	0.00	0.00	0.00	-1.00

Figure 13.24 shows the five calibration points in the 3D world and approximately how they will look on the image plane as viewed by the camera, whose position, orientation, and focal length are unknown and to be computed. Figure 13.25 shows the image points in the continuous **u-v** coordinate system.

Using the five correspondences, the matrix **A** and vector **b** of equation 13.54 are given by

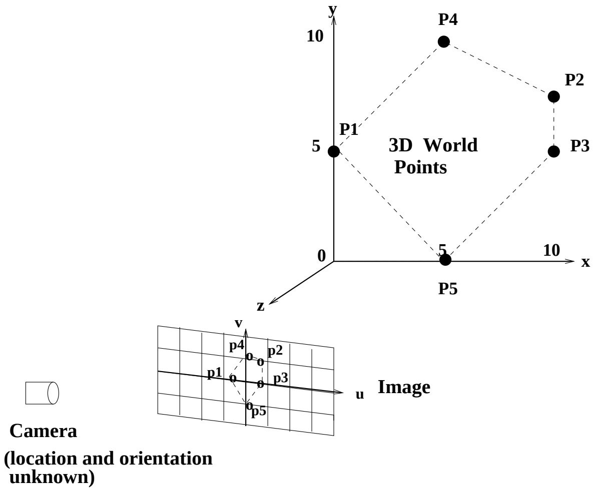


Figure 13.24: The 3D world points and corresponding 2D image points that are input to the calibration procedure, which will compute the camera parameters including position, orientation, and focal length.

$$
\begin{array}{cc}
 & \begin{array}{ccccc} v_i x_i & v_i y_i & -u_i x_i & -u_i y_i & v_i \end{array} \\
\mathbf{A} = & \begin{bmatrix}
0.00 & 0.00 & 0.00 & 2.89 & 0.00 \\
10.00 & 7.50 & -17.32 & -12.99 & 1.00 \\
0.00 & 0.00 & -17.32 & -8.66 & 0.00 \\
5.00 & 10.00 & 0.00 & 0.00 & 1.00 \\
-5.00 & 0.00 & 0.00 & 0.00 & -1.00
\end{bmatrix}
\end{array}
$$

and

$$
\begin{array}{cc}
 & u_i \\
\mathbf{b} = & \begin{bmatrix} -.58 \\ 1.73 \\ 1.73 \\ 0.00 \\ 0.00 \end{bmatrix}
\end{array}
$$

Solving $\mathbf{Au} = \mathbf{b}$ yields the vector $\mathbf{u}$ given by

$$
\begin{array}{cc}
 & \mu_i \\
\mathbf{u} = & \begin{bmatrix} -0.17 \\ 0.00 \\ 0.00 \\ -0.20 \\ 0.87 \end{bmatrix}
\end{array}
$$

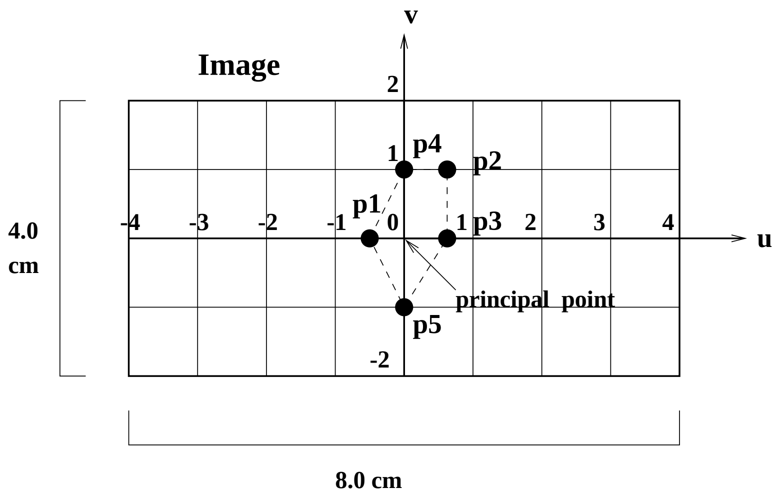


Figure 13.25: The image points in the **u-v** image coordinate system.

The next step is to calculate U and use it to solve for t_y^2 in equation 13.55. We have

$$U = \mu_1^2 + \mu_2^2 + \mu_3^2 + \mu_4^2 = .07$$

Using the first formula in 13.55, we have

$$t_y^2 = \frac{U - [U^2 - 4(\mu_1\mu_4 - \mu_2\mu_3)^2]^{1/2}}{2(\mu_1\mu_4 - \mu_2\mu_3)^2} = 25$$

When t_y is set to the positive square root (5), we have

$$\begin{aligned} r_{11} &= \mu_1 t_y = -.87 \\ r_{12} &= \mu_2 t_y = 0 \\ r_{21} &= \mu_3 t_y = 0 \\ r_{22} &= \mu_4 t_y = -1.0 \\ t_x &= \mu_5 t_y = 4.33 \end{aligned}$$

Next we compute ξ_x and ξ_y for the point $P_2 = (10.0,7.5,0.0)$ and corresponding image point $\mathbf{p_2} = (1.73,1.0)$, which is far from the image center, to check the sign of t_y.

$$\begin{aligned} \xi_x &= r_{11}x + r_{12}y + t_x = (-.87)(10) + 0 + 4.33 = -4.33 \\ \xi_y &= r_{21}x + r_{22}y + t_y = 0 + (-1.0)(7.5) + 5 = -2.5 \end{aligned}$$

Since the signs of ξ_x and ξ_y do not agree with those of $\mathbf{p_2}$, the sign of t_y is wrong, and it is negated. This gives us

$$
\begin{aligned}
t_y &= -5 \\
r_{11} &= .87 \\
r_{12} &= 0 \\
r_{21} &= 0 \\
r_{22} &= 1.0 \\
t_x &= -4.33
\end{aligned}
$$

Continuing, we compute the remaining rotation parameters:

$$
\begin{aligned}
r_{13} &= (1 - r_{11}^2 - r_{12}^2)^{1/2} = 0.5 \\
r_{23} &= (1 - r_{21}^2 - r_{22}^2)^{1/2} = 0. \\
r_{31} &= \frac{1 - r_{11}^2 - r_{12}r_{21}}{r_{13}} = 0.5 \\
r_{32} &= \frac{1 - r_{21}r_{12} - r_{22}^2}{r_{23}} = 0. \\
r_{33} &= (1 - r_{31}r_{13} - r_{32}r_{23})^{1/2} = .87
\end{aligned}
$$

Checking the sign of $r_{11}r_{21} + r_{12}r_{22} = 0$ shows that it is not positive, and therefore, r_{23} does not need to change.

We now form the second system of linear equations as follows.

$$
\begin{array}{cc}
 & \begin{array}{cc} r_{21}x_i + r_{22}y_i + t_y & v_i \end{array} \\
\mathbf{A}' = & \begin{bmatrix} 0.00 & 0.00 \\ 2.500 & -1.00 \\ 0.00 & 0.00 \\ 5.00 & -1.00 \\ -5.00 & 1.00 \end{bmatrix}
\end{array}
$$

and

$$
\begin{array}{cc}
 & (r_{31}x_i + r_{32}y_i)v_i \\
\mathbf{b}' = & \begin{bmatrix} 0.0 \\ 5.0 \\ 0.0 \\ 2.5 \\ -2.5 \end{bmatrix}
\end{array}
$$

Solving $\mathbf{A}'\mathbf{v} = \mathbf{b}'$ yields the vector $\mathbf{v} = [f, t_z]$ given by

$$
\begin{aligned}
f &= -1.0 \\
t_z &= -7.5
\end{aligned}
$$

Since f is negative, our coordinate system is not right-handed. To flip the z-axis, we negate r_{13}, r_{23}, r_{31}, r_{32}, f, and t_z. The results are:

$$\mathbf{R} = \begin{bmatrix} 0.87 & 0.00 & -0.50 \\ 0.00 & -1.00 & 0.00 \\ -0.50 & 0.00 & 0.87 \end{bmatrix}$$

and

$$\mathbf{T} = \begin{bmatrix} -4.33 \\ -5.00 \\ 7.50 \end{bmatrix}$$

and $f = 1$.

Since our example includes no distortion, these are the final results of the calibration procedure. Figure 13.26 illustrates the results of the calibration procedure shown from two different views.

Exercise 28

Verify that the rotation matrix $\mathbf{R}$ derived in the above example is orthonormal.

Exercise 29 Using camera parameter f

Using 3D Euclidean geometry and Figure 13.26a, find the projection of P_1 and P_3 on the image plane. (All the lines in Figure 13.26a are coplanar.) Verify that the results agree with the image coordinates p_1 and p_3 given in the example.

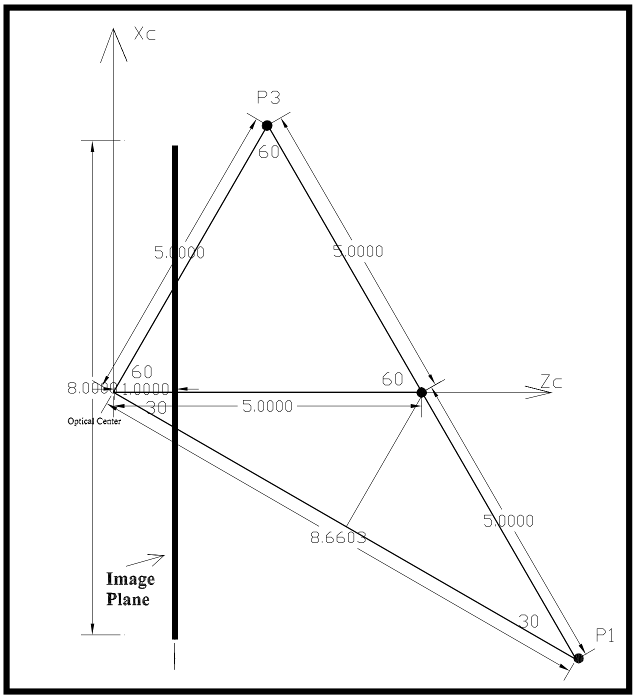
Exercise 30 Using camera parameters $\mathbf{R}$ and $\mathbf{T}$

P_1 and P_3 in the example were given in the world coordinate system. Find their coordinates in the camera coordinate system

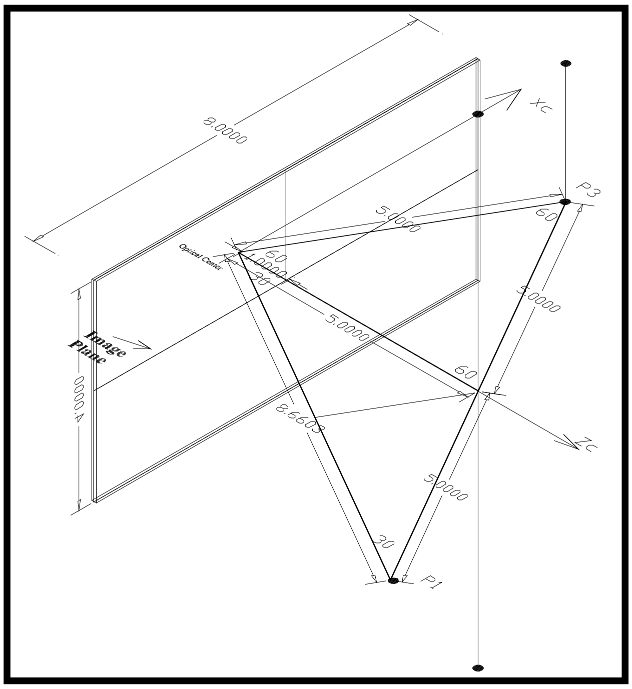
1. using Euclidean geometry and Figure 13.26a.
2. using the camera parameters determined by the calibration procedure.

13.8 * Pose Estimation

In industrial vision, especially for robot guidance tasks, it is important to obtain the pose of a 3D object in workspace coordinates. Since the pose of the camera in the workspace can be computed by calibration, the problem is reduced to that of determining the pose of the object with respect to the camera. The method for determining object pose given in this section should have accuracy superior to the simple method given in a previous section. Using point-correspondences is the most basic and most-often-used method for pose computation. For use of correspondences between 2D and 3D line segments, between 2D ellipses and 3D circles, and between any combination of point pairs, line-segment pairs, and ellipse-circle pairs, see the work of Ji and Costa (1998).



a) top view (the thick black line is the image plane)



b) perspective view

Figure 13.26: Two views of the camera and image plane in the world coordinate system as computed in the example via the Tsai calibration procedure.

13.8.1 Pose from 2D-3D Point Correspondences

The camera model of the previous section is used, and we assume that the camera has been calibrated to obtain the intrinsic and extrinsic parameters. The problem of determining object pose from n point correspondences between 3D model object points and 2D image points is inherently a non-linear one. Non-linear methods for estimating the pose parameters are necessary; however, under some conditions, an approximate, linear solution can be found.

Let $[x, y, z]$ be the coordinates of object point ${}^{\mathbf{M}}\mathbf{P}$ in its model coordinate system. Let the object coordinate system and the camera coordinate system be related by a transformation ${}^{\mathbf{C}}_{\mathbf{M}}\mathbf{Tr} = \{\mathbf{R}, \mathbf{T}\}$, described in the form of a rotation matrix $\mathbf{R}$ and a translation vector $\mathbf{T} = [t_x, t_y, t_z]$. Then, the perspective projection of ${}^{\mathbf{M}}\mathbf{P}$ onto the image plane yields image plane coordinates $[u, v]$, where

$$u = f\frac{r_{11}x + r_{12}y + r_{13}z + t_x}{r_{31}x + r_{32}y + r_{33}z + t_z} \tag{13.75}$$

and

$$v = f\frac{r_{21}x + r_{22}y + r_{23}z + t_y}{r_{31}x + r_{32}y + r_{33}z + t_z} \tag{13.76}$$

and f is the focal length of the camera, which is now known.

The transformation between object model frame and camera frame corresponds to the pose of the object with respect to the camera frame. Using our perspective imaging model as before, we have nine rotation parameters and three translation parameters involved in 12 equations of the following form.

$$\mathbf{B}\ \mathbf{w} = \mathbf{0} \tag{13.77}$$

where

$$\mathbf{B} = \begin{pmatrix} fx_1 & fy_1 & fz_1 & 0 & 0 & 0 & -u_1x_1 & -u_1y_1 & -u_1z_1 & f & 0 & -u_1 \\ 0 & 0 & 0 & fx_1 & fy_1 & fz_1 & -v_1x_1 & -v_1y_1 & -v_1z_1 & 0 & f & -v_1 \\ fx_2 & fy_2 & fz_2 & 0 & 0 & 0 & -u_2x_2 & -u_2y_2 & -u_2z_2 & f & 0 & -u_2 \\ 0 & 0 & 0 & fx_2 & fy_2 & fz_2 & -v_2x_2 & -v_2y_2 & -v_2z_2 & 0 & f & -v_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ fx_6 & fy_6 & fz_6 & 0 & 0 & 0 & -u_6x_6 & -u_6y_6 & -u_6z_6 & f & 0 & -u_6 \\ 0 & 0 & 0 & fx_6 & fy_6 & fz_6 & -v_6x_6 & -v_6y_6 & -v_6z_6 & 0 & f & -v_6 \end{pmatrix} \tag{13.78}$$

and

$$\mathbf{w} = \begin{pmatrix} r_{11} & r_{12} & r_{13} & r_{21} & r_{22} & r_{23} & r_{31} & r_{32} & r_{33} & t_x & t_y & t_z \end{pmatrix}^T. \tag{13.79}$$

However, if one is interested in finding independent pose parameters, and not simply an affine transformation that aligns the projected model points with the image points, conditions need to be imposed on the elements of R such that it satisfies all the criteria a true 3D rotation matrix must satisfy. In particular, a rotation matrix needs to be orthonormal: its row vectors must have magnitude equal to one, and they must be orthogonal to each other. This can be written as:

$$\begin{aligned} ||R_1|| &= r_{11}^2 + r_{12}^2 + r_{13}^2 = 1 \\ ||R_2|| &= r_{21}^2 + r_{22}^2 + r_{23}^2 = 1 \\ ||R_3|| &= r_{31}^2 + r_{32}^2 + r_{33}^2 = 1 \end{aligned} \tag{13.80}$$

and

$$\begin{aligned} R_1 \circ R_2 = 0 \\ R_1 \circ R_3 = 0 \\ R_2 \circ R_3 = 0 \end{aligned} \tag{13.81}$$

The conditions imposed on $\mathbf{R}$ turn the problem into a non-linear one. If the conditions on the magnitudes of the row vectors of $\mathbf{R}$ are imposed one at a time, and computed independently, a linear constrained optimization technique can be used to compute the constrained row vector of $\mathbf{R}$. (See the Faugeras 1993 reference for a similar procedure.)

13.8.2 Constrained Linear Optimization

Given the system of equations 13.77, the problem at hand is to find the solution vector w that minimizes $||\mathbf{Bw}||$ subject to the constraint $||\mathbf{w}'||^2 = 1$, where $\mathbf{w}'$ is a subset of the elements of $\mathbf{w}$. If the constraint is to be imposed on the first row vector of $\mathbf{R}$, then

$$\mathbf{w}' = \begin{pmatrix} r_{11} \\ r_{12} \\ r_{13} \end{pmatrix} .$$

To solve the above problem, it is necessary to rewrite the original system of equations $\mathbf{Bw} = \mathbf{0}$ in the following form

$$\mathbf{Cw}' + \mathbf{Dw}'' = \mathbf{0},$$

where $\mathbf{w}''$ is a vector with the remaining elements of $\mathbf{w}$. Using the example above, i.e., if the constraint is imposed on the first row of $\mathbf{R}$,

$$\mathbf{w}'' = \begin{pmatrix} r_{21} & r_{22} & r_{23} & r_{31} & r_{32} & r_{33} & t_x & t_y & t_z \end{pmatrix}^T .$$

The original problem can be stated as: minimize the objective function $\mathbf{O} = \mathbf{Cw}' + \mathbf{Dw}''$, that is

$$\min_{w',w''} ||\mathbf{Cw}' + \mathbf{Dw}''||^2 \tag{13.82}$$

subject to the constraint $||\mathbf{w}'||^2 = 1$. Using a Lagrange multiplier technique, the above is equivalent to

$$\min_{w',w''} \left[||\mathbf{Cw}' + \mathbf{Dw}''||^2 + \lambda(1 - ||\mathbf{w}'||^2) \right] . \tag{13.83}$$

The minimization problem above can be solved by taking partial derivatives of the objective function with respect to $\mathbf{w}'$ and $\mathbf{w}''$ and equating them to zero:

$$\frac{\partial \mathbf{O}}{\partial \mathbf{w}'} = 2\mathbf{C}^{\mathbf{T}}(\mathbf{C}\mathbf{w}' + \mathbf{D}\mathbf{w}'') - 2\lambda\mathbf{w}' = 0 \tag{13.84}$$

$$\frac{\partial \mathbf{O}}{\partial \mathbf{w}''} = 2\mathbf{D}^{\mathbf{T}}(\mathbf{C}\mathbf{w}' + \mathbf{D}\mathbf{w}'') = 0 \tag{13.85}$$

Equation 13.85 is equivalent to

$$\mathbf{w}'' = -(\mathbf{D}^{\mathbf{T}}\mathbf{D})^{-\mathbf{1}}\mathbf{D}^{\mathbf{T}}\mathbf{C}\mathbf{w}'. \tag{13.86}$$

Substituting equation 13.86 into equation 13.84 yields

$$\lambda\mathbf{w}' = [\mathbf{C}^{\mathbf{T}}\mathbf{C} - \mathbf{C}^{\mathbf{T}}\mathbf{D}(\mathbf{D}^{\mathbf{T}}\mathbf{D})^{-\mathbf{1}}\mathbf{D}^{\mathbf{T}}\mathbf{C}]\mathbf{w}'. \tag{13.87}$$

It can be seen that λ is an eigenvector of the matrix

$$\mathbf{M} = \mathbf{C}^{\mathbf{T}}\mathbf{C} - \mathbf{C}^{\mathbf{T}}\mathbf{D}(\mathbf{D}^{\mathbf{T}}\mathbf{D})^{-\mathbf{1}}\mathbf{D}^{\mathbf{T}}\mathbf{C}. \tag{13.88}$$

Therefore, the solution sought for $\mathbf{w}'$ corresponds to the smallest eigenvector associated with matrix $\mathbf{M}$. The corresponding $\mathbf{w}''$ can be directly computed from equation 13.86. It is important to notice that since the magnitude constraint was imposed only on one of the rows of $\mathbf{R}$, the results obtained for $\mathbf{w}''$ are not reliable and therefore should not be used. However, solution vector $\mathbf{w}''$ provides an important piece of information regarding the sign of the row vector on which the constraint was imposed. The constraint imposed was $||\mathbf{w}'||^2 = 1$, but the sign of $\mathbf{w}'$ is not restricted by this constraint. Therefore, it is necessary to check whether or not the resulting $\mathbf{w}'$ yields a solution that is physically possible. In particular, the translation t_z must be positive in order for the object to be located in front of the camera as opposed to behind it. If the element of vector $\mathbf{w}''$ that corresponds to t_z is negative, it means that the magnitude of the computed $\mathbf{w}'$ is correct, but its sign is not and must be changed. Thus, the final expression for the computed $\mathbf{w}'$ is

$$\mathbf{w}' = \mathrm{sign}(w_9'')\mathbf{w}'. \tag{13.89}$$

13.8.3 Computing the Transformation $\mathbf{Tr} = \{\mathbf{R}, \mathbf{T}\}$

Row vector $\mathbf{R_1}$ is computed first by computing $\mathbf{w}'$ as described above, since in this case $\mathbf{R_1} = \mathbf{w}'$. Matrices $\mathbf{C}$ and $\mathbf{D}$ are

$$\mathbf{C} = \begin{pmatrix} x_1 & y_1 & z_1 \\ 0 & 0 & 0 \\ x_2 & y_2 & z_2 \\ 0 & 0 & 0 \\ \vdots & \vdots & \vdots \\ x_6 & y_6 & z_6 \\ 0 & 0 & 0 \end{pmatrix} \tag{13.90}$$

and

$$\mathbf{D} = \begin{pmatrix} 0 & 0 & 0 & -u_1x_1 & -u_1y_1 & -u_1z_1 & f & 0 & -u_1 \\ fx_1 & fy_1 & fz_1 & -v_1x_1 & -v_1y_1 & -v_1z_1 & 0 & f & -v_1 \\ 0 & 0 & 0 & -u_2x_2 & -u_2y_2 & -u_2z_2 & 0 & f & -u_2 \\ fx_2 & fy_2 & fz_2 & -v_2x_2 & -v_2y_2 & -v_2z_2 & 0 & f & -v_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & -u_6x_6 & -u_6y_6 & -u_6z_6 & f & 0 & -u_6 \\ fx_6 & fy_6 & fz_6 & -v_6x_6 & -v_6y_6 & -v_6z_6 & 0 & f & -v_6 \end{pmatrix}. \tag{13.91}$$

Then row vector $\mathbf{R_2}$ is computed using the same technique, except that now the constraint is imposed on its magnitude, thus $\mathbf{R_2} = \mathbf{w}'$. In this case, matrices $\mathbf{C}$ and $\mathbf{D}$ are

$$\mathbf{C} = \begin{pmatrix} 0 & 0 & 0 \\ fx_1 & fy_1 & fz_1 \\ 0 & 0 & 0 \\ fx_2 & fy_2 & fz_2 \\ \vdots & \vdots & \vdots \\ 0 & 0 & 0 \\ fx_6 & fy_6 & fz_6 \end{pmatrix} \tag{13.92}$$

and

$$D = \begin{pmatrix} fx_1 & fy_1 & fz_1 & -u_1x_1 & -u_1y_1 & -u_1z_1 & f & 0 & -u_1 \\ 0 & 0 & 0 & -v_1x_1 & -v_1y_1 & -v_1z_1 & 0 & f & -v_1 \\ fx_2 & fy_2 & fz_2 & -u_2x_2 & -u_2y_2 & -u_2z_2 & f & 0 & -u_2 \\ 0 & 0 & 0 & -v_2x_2 & -v_2y_2 & -v_2z_2 & 0 & f & -v_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ fx_6 & fy_6 & fz_6 & -u_6x_6 & -u_6y_6 & -u_6z_6 & f & 0 & -u_6 \\ 0 & 0 & 0 & -v_6x_6 & -v_6y_6 & -v_6z_6 & 0 & f & -v_6 \end{pmatrix}. \tag{13.93}$$

$\mathbf{R_3}$ could also be computed the same way as $\mathbf{R_1}$ and $\mathbf{R_2}$ above, but that would not guarantee it to be normal to $\mathbf{R_1}$ and $\mathbf{R_2}$. Instead, $\mathbf{R_3}$ is computed as follows:

$$\mathbf{R_3} = \frac{\mathbf{R_1} \times \mathbf{R_2}}{||\mathbf{R_1} \times \mathbf{R_2}||}. \tag{13.94}$$

All the constraints on the row vectors of $\mathbf{R}$ have been satisfied, except one: there is no guarantee that $\mathbf{R_1}$ is orthogonal to $\mathbf{R_2}$. In order to solve this undesired situation, $\mathbf{R_1}, \mathbf{R_2}$, and $\mathbf{R_3}$ need to go through an orthogonalization process, such that the rotation matrix $\mathbf{R}$ is assured to be orthonormal. This can be accomplished by fixing $\mathbf{R_1}$ and $\mathbf{R_3}$ as computed above and recomputing $\mathbf{R_2}$ as:

$$\mathbf{R_2} = \mathbf{R_3} \times \mathbf{R_1}. \tag{13.95}$$

This way, all the rotation parameters have been calculated and they all satisfy the necessary constraints. The translation vector $\mathbf{T}$ is computed using a least squares technique on a new, non-homogeneous, over-constrained system of twelve equations:

$$\mathbf{A}\ \mathbf{t} = \mathbf{b}, \tag{13.96}$$

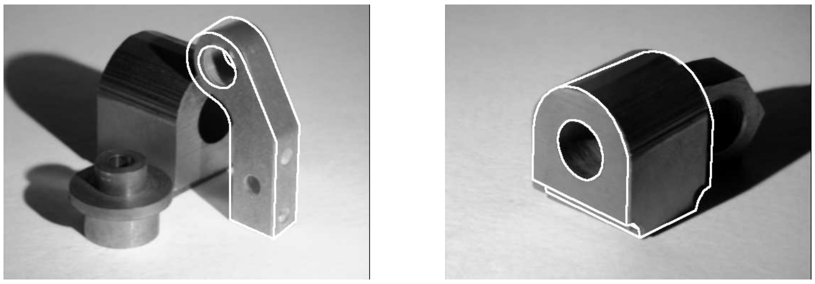


Figure 13.27: Examples of pose computed from six point correspondences using constrained linear optimization.

where

$$\mathbf{A} = \begin{pmatrix} f & 0 & -u_1 \\ 0 & f & -v_1 \\ f & 0 & -u_2 \\ 0 & f & -v_2 \\ \vdots & \vdots & \vdots \\ f & 0 & -u_6 \\ 0 & f & -v_6 \end{pmatrix}, \tag{13.97}$$

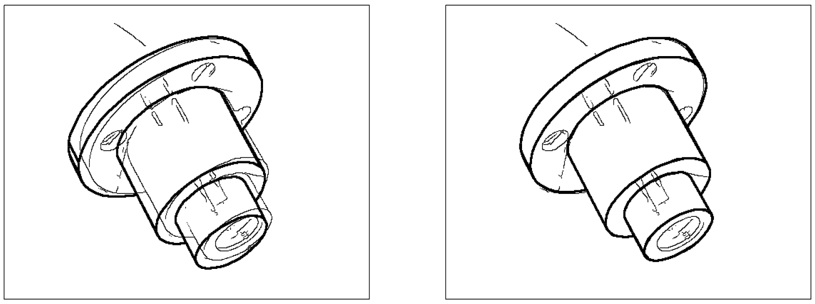
and

$$\mathbf{b} = \begin{pmatrix} -f(r_{11}x_1 + r_{12}y_1 + r_{13}z_1) + u_1(r_{31}x_1 + r_{32}y_1 + r_{33}z_1) \\ -f(r_{21}x_1 + r_{22}y_1 + r_{23}z_1) + v_1(r_{31}x_1 + r_{32}y_1 + r_{33}z_1) \\ -f(r_{11}x_2 + r_{12}y_2 + r_{13}z_2) + u_1(r_{31}x_2 + r_{32}y_2 + r_{33}z_2) \\ -f(r_{21}x_2 + r_{22}y_2 + r_{23}z_2) + v_1(r_{31}x_2 + r_{32}y_2 + r_{33}z_2) \\ \vdots \\ -f(r_{11}x_6 + r_{12}y_6 + r_{13}z_6) + u_1(r_{31}x_6 + r_{32}y_6 + r_{33}z_6) \\ -f(r_{21}x_6 + r_{22}y_6 + r_{23}z_6) + v_1(r_{31}x_6 + r_{32}y_6 + r_{33}z_6) \end{pmatrix}. \tag{13.98}$$

13.8.4 Verification and Optimization of Pose

A quantitative measure is useful for evaluating the quality of the pose parameters. One was already used in our development of the affine calibration procedure: it was the sum of the squared distances between projected posed model points and their corresponding image points. Some correspondences must be dropped as outliers, however, due to occlusions of some object points by the same or by other objects. Other distance measures can be used; for example, the Hausdorf or modified Hausdorf distance. (See the references by Hutenlocher *et al* Dubuisson and Jain). Verification can also be done using other features, such as edges, corners, or holes.

A measure of pose quality can be used to improve the estimated pose parameters. Conceptually, we can evaluate small variations of the parameters and keep only the best ones. Brute force search of 10 variations of each of six rotation and translation parameters would mean that one million sets of pose parameters would have to be evaluated — computational effort that is not often done. A nonlinear optimization method, such as Newton's method or



(a) Initial pose (b) Final pose

Figure 13.28: Example pose hypothesis and final pose after nonlinear optimization.

Powell's method (see *Numerical Recipes*) will be much faster. Figure 13.28 shows an initial pose estimate for a single-object image as well as the final result after nonlinear optimization has been applied to that initial solution. The improved pose is clearly better for visual inspection tasks and possibly better for grasping the object, but perhaps not necessary for recognition.

13.9 3D Object Reconstruction

3D sensing is important for 3D model building. We can acquire range images of an existant object in order to *create* a computer model of it. This process of *3D object reconstruction* has applications in medicine and industrial vision, and is also used for producing the object models needed for virtual reality environments. By necessity, this section includes some discussion of object modeling, which is the major topic of the next chapter. The object reconstruction process has four major steps:

1. 3D data acquisition
2. registration
3. surface construction
4. optimization

In the data acquisition phase, range data must be acquired from a set of views that cover the surface of the object. Often, 8-10 views are enough, but for complex objects or strict accuracy requirements, a larger number may be necessary. Of course, more views also mean more computation, so more is not necessarily better.

Each view obtained consists of a single range image of a portion of the object and often a registered gray tone or color image. The range data from all of the views will be combined to produce a surface model of the object. The intensity data can be used in the registration procedure, but is really meant for use in texture mapping the objects for realistic viewing in graphics applications. The process of combining the range data by transforming them all to a single 3D coordinate system is the *registration* process.

Once the data have been registered, it is possible to view a *cloud of 3D points*, but it takes more work to create an object model. Two possible 3D representations for the object are 1) a connected *mesh* of 3D points and edges connecting them representing the object surface and 2) a set of 3D cubes or *voxels* representing the entire volume of the object. (See Chapter 14 for full explanations of these representations.) It is possible to convert from one representation to the other.

Exercise 31 objects with hidden surfaces

Some objects have hidden surfaces that cannot be imaged regardless of the number of views taken. Sketch one. For simplicity, you may work with a 2D model and world.

13.9.1 Data Acquisition

Range images registered to color images can be obtained from most modern commercial scanners. Here, we describe a laboratory system made with off-the-shelf components, and emphasize its fundamental operations. Figure 13.29 illustrates a specially built active stereo vision system that is used for acquiring range/color data. The system employs four color video cameras mounted on an aluminum bar. The cameras are connected to a digitizing board, which can switch between the four inputs under computer control, and produces images at 640×480 resolution. Below the cameras, a slide projector sits on a computer-controlled turntable. The slide projector emits a vertical stripe of white light, which is manually focused for the working volume. The light stripes are projected in a darkened room; the two side lights are turned on to take a color image after the range data has been obtained.

The cameras are calibrated simultaneously using Tsai's algorithm, described in Section 13.7. They can be used for either standard two-camera stereo or a more robust four-camera stereo algorithm. In either case, the projector is used to project a single vertical light stripe onto the object being scanned. It is controlled by computer to start at the left of the object and move, via the turntable, from left to right at fixed intervals chosen by the user to allow for either coarse or fine resolution. At each position, the cameras take a picture of the light stripe on the object in the darkened room. On each image, the intersection of the light stripe with an epipolar line provides a point for the stereo matching. Figure 13.30 illustrates the triangulation process using two cameras and a single light stripe. The two matched pixels are used to produce a point in 3D space. For a single light stripe, 3D points are computed at each pixel along that light stripe. Then the projector turns, a new light stripe is projected, new images are taken, and the process is repeated. The result is a dense range map, with 3D data associated with each pixel in the left image if that point was also visible to both the light projector and the right camera.

We can increase the reliability of the image acquisition system by using more than two cameras. One camera is our base camera in whose coordinate frame the range image will be computed. A surface point on the object must be visible from the base camera, the light projector, and at least one of the other three cameras. If it is visible in only one of the three, then the process reduces to two-camera stereo. If it is visible in two or three of the

Figure 13.29: A 4-camera stereo acquisition system.

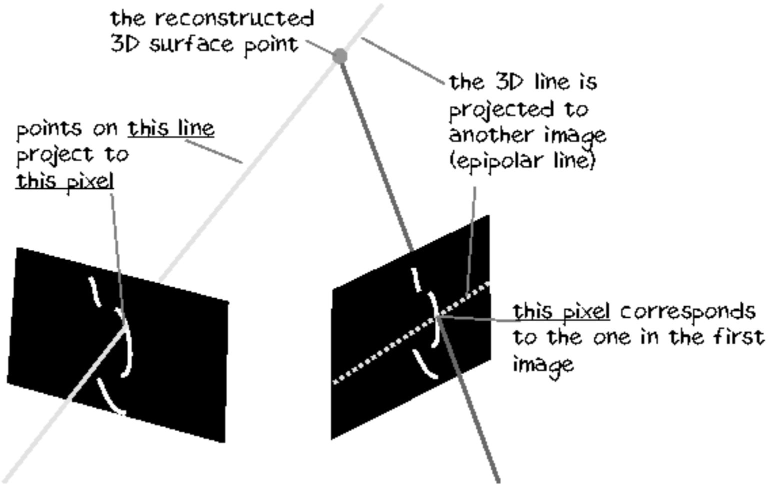


Figure 13.30: The intersections of light stripes with epipolar lines in two images gives a pair of corresponding points.

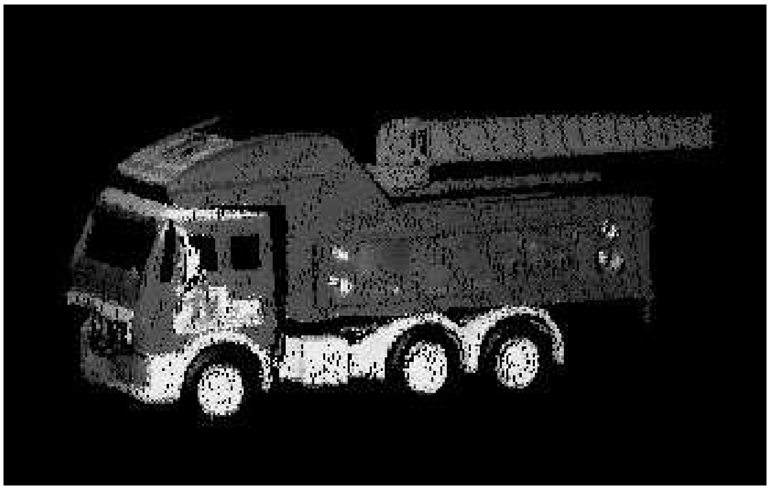


Figure 13.31: A range data set for a toy truck obtained via the 4-camera active stereo system. The range points were colored with intensity data for display purposes.

three, then the extra images can be used to make the process more robust. In the case of the base camera plus two more images, we have three points, so there are three different correspondences that can be triangulated to compute the 3D coordinates. It is likely that the three separate results will all be different. If they differ only by a small amount (say, they are all within a seven cubic millimeter volume), then they are all considered valid, and the average of the three 3D points can be used as the final result for that pixel. Or the pair of cameras with the widest baseline can be used as a more reliable estimate than the others. If the point is visible in all four cameras, then there are six possible combinations. Again we can check whether they all lie in a small volume, throw out outliers, and use an average value or the nonoutlier value that comes from the camera pair with the widest baseline. This procedure gives better accuracy than just using a fixed pair of cameras. (In measuring human body position in cars, as shown in Figure 13.1, the expected error was about 2mm in x,y, and z using this procedure.) A range image of a toy truck computed by this method is shown in Figure 13.31. The 3D truck dataset clearly shows the shape of the truck.

13.9.2 Registration of Views

In order to thoroughly cover the surface of the object, range data must be captured from multiple views. A transformation ${}^2_1\mathbf{T}$ from view 1 to view 2 is obtained either from precise mechanical movement or from image correspondence. When a highly accurate device, such as a calibrated robot or a coordinate measurement machine, is available that can move the camera system or the object in a controlled way, then the approximate transformations between the views can be obtained automatically from the system. If the movement of the cameras or object is not machine controlled, then there must be a method for detecting correspondences between views that will allow computation of the rigid transformation that maps the data from one view into that for another view. This can be done automatically using 3D features such as corner points and line segments that lead to a number of 3D-3D point correspondences from which the transformation can be computed. It can also be done interactively, allowing the user to select point correspondences in a pair of images of the object. In either case, the initial transformation is not likely to be perfect. Robots and machines will have some associated error that becomes larger as more moves take place. The automatic correspondence finder will suffer from the problems of any matching algorithm and may find false correspondences, or the features may be a little off. Even the human point clicker will make some errors, or quantization can lead to the right pixel, but the wrong transformation.

To solve these problems, most registration procedures employ an iterative scheme that begins with an initial estimate of the transformation ${}^2_1\mathbf{T}$, no matter how obtained, and refines it via a minimization procedure. For example, the iterative closest point (ICP) algorithm minimizes the sum of the distances between 3D points ${}^2_1\mathbf{T}\ {}^1\mathbf{P}$ and ${}^2\mathbf{P}$, where ${}^1\mathbf{P}$ is from one view and ${}^2\mathbf{P}$ is the closest point to it in the other view. A variation of this approach looks for a point in the second view along a normal extended from the point ${}^2_1\mathbf{T}\ {}^1\mathbf{P}$ to the *surface* interpolating a neighborhood in the second view. (See the references by Medioni *et al* (1992) and Dorai *et al* (1994) for example.) When color data is available, it is possible to project the color data from one view onto the color image from another view via the estimated transformation and define a distance measure based on how well they line up. This distance can also be iteratively minimized, leading to the best transformation from one set of 3D points to the other. Figure 13.32 illustrates the registration process for two views of a sofa using an ICP algorithm.

13.9.3 Surface Reconstruction

Once the data have been registered into a single coordinate system, reconstruction can begin. We would like the reconstructed object to come as close as possible to the shape of the actual object and to preserve its topology. Figure 13.33 illustrates problems that can occur in the reconstruction process. The registered range data is dense, but quite noisy. There are extra points outside the actual chair volume and, in particular, between the spokes of the back. The reconstruction shown in the middle is naive in that it considered the range data only as a cloud of 3D points and did not take into account object geometry or neighbor relationships between range data points. It fails to preserve the topology of the object. The reconstruction shown on the right is better in that it has removed most of the noise and the holes between the spokes of the back have been preserved. This reconstruction was produced by a *space-carving* algorithm described below.

13.9.4 Space-Carving

The space-carving approach was developed by Curless and Levoy, and the method described here was implemented by Pulli *et al.*. Figure 13.34 illustrates the basic concept. At the left is an object to be reconstructed from a set of views. In the center, there is one camera viewing the object. The space can be partitioned into areas according to where the points lie with respect to the object and the camera. The left and bottom sides of the object are visible to the camera. The volume between the scanned surface and the camera (light gray)

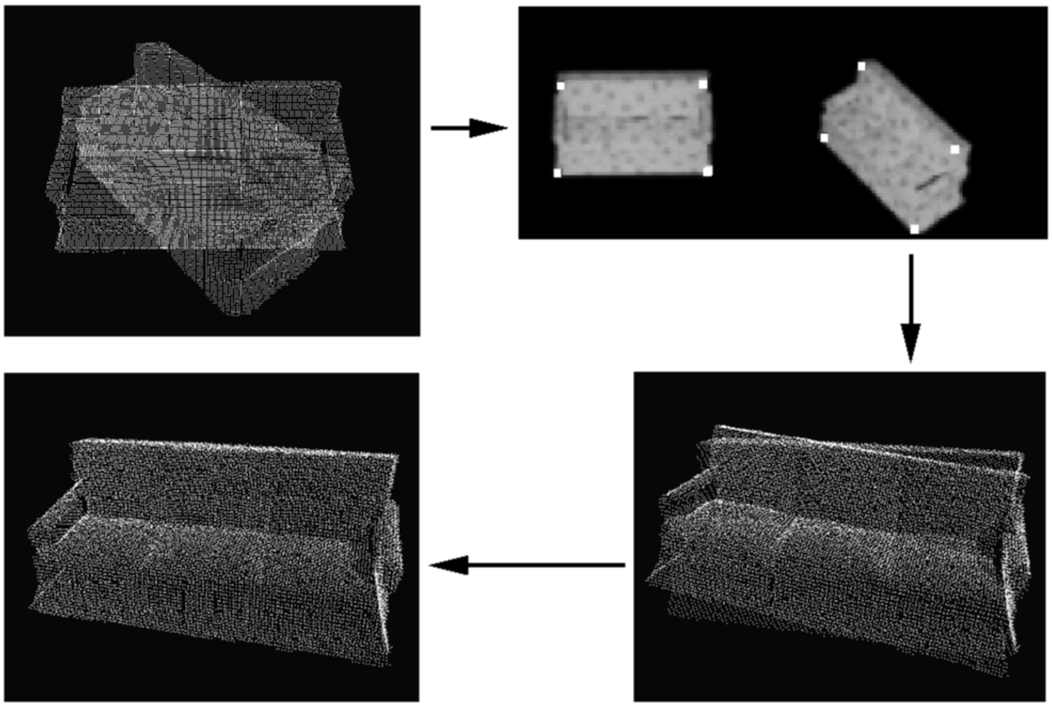


Figure 13.32: Registration of two range data sets shown at top left. The user has selected four points on intensity images corresponding to the two range views (upper right). The initial transformation is slightly off (lower right). After several iterations, the two range datasets line up well (lower left).

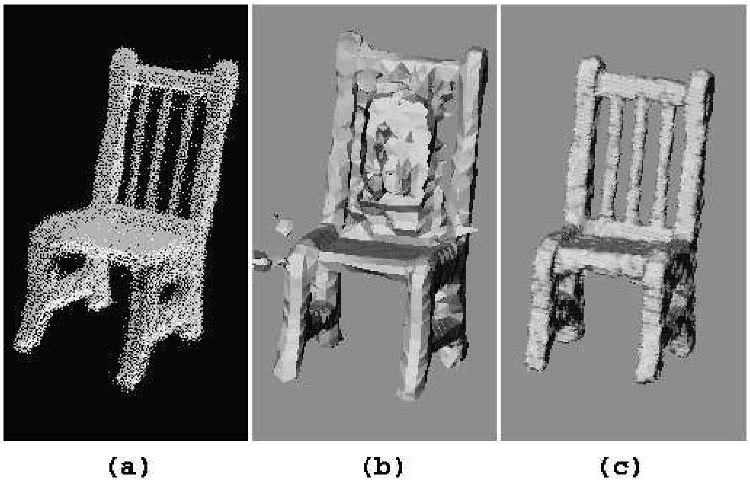


Figure 13.33: The registered range data for a chair object (left), problems that can occur in reconstruction (middle), and a topologically correct rough mesh model (right).

is in front of the object and can be removed. If there is data from the background, as well as the object, even more volume (darker gray) can be removed. On the other hand, points behind the object cannot be removed, because the single camera cannot tell if they are part of the object or behind it. However, another camera viewing the object (at right) can carve away more volume. A sufficient number of views will carve away most of the unwanted free space, leaving a voxel model of the object.

The space-carving algorithm discretizes space into a collection of cubes or voxels that can be processed one at a time. Figure 13.35 illustrates how the status of a single cube with respect to a single camera view is determined:

- In case (a) the cube lies between the range data and the sensor. Therefore the cube must lie outside of the object and can be discarded.
- In case (b) the whole cube is behind the range data. As far as the sensor is concerned, the cube is assumed to lie inside of the object.
- In case (c) the cube is neither fully in front of the data or behind the data and is assumed to intersect the object surface.

The cube labeling step can be implemented as follows. The eight corners of the cube are projected to the sensor's image plane, where their convex hull generally forms a hexagon. The rays from the sensor to the hexagon form a cone, which is truncated so that it just encloses the cube. If all the data points projecting onto the hexagon are behind the truncated cone (are farther than the farthest corner of the cube from the sensor), the cube is outside of the object. If all those points are closer than the closest cube corner, the cube is inside the object. Otherwise, it is a boundary cube.

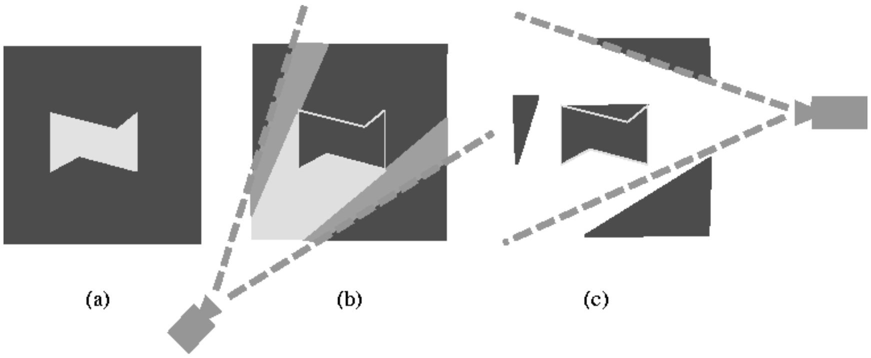


Figure 13.34: The concept of space carving: (a) object cross section; (b) camera view 1 can remove all material of light shading; (c) camera view 2 can remove other material. (Extra material still remains, however.)

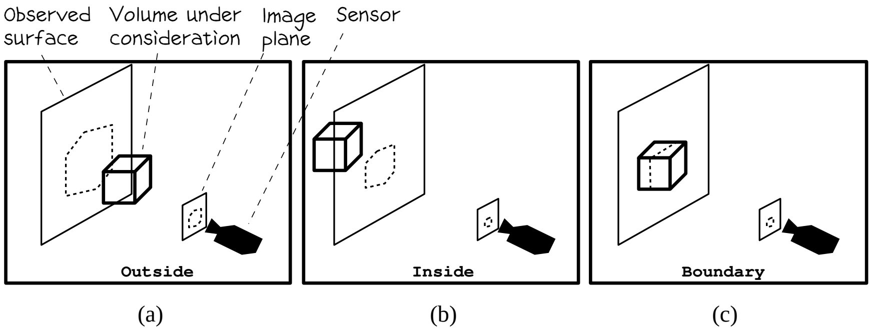


Figure 13.35: The three possible positions of a cube in space in relation to the object being reconstructed.

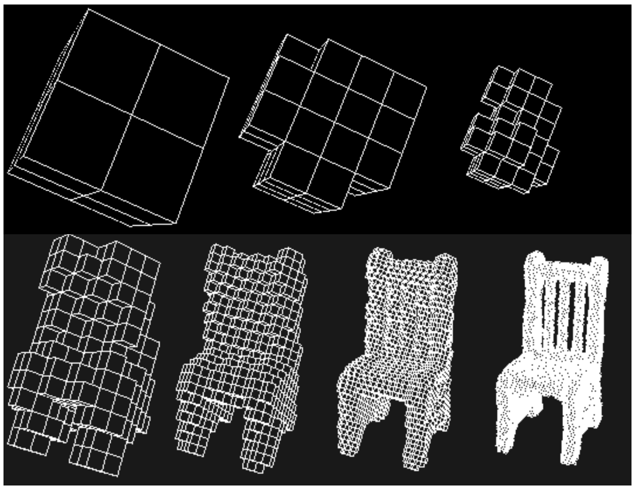


Figure 13.36: Hierarchical space carving: seven iterations to produce the chair mesh.

So far, we've looked at one cube and one sensor. It takes a number of sensors (or views) to carve out the free space. The cube labeling step for a given cube is performed for all of the sensors. If even one sensor says that the cube is outside of the object, then it is outside. If all of the sensors say that the cube is inside of the object, then it is inside as far as we can tell. Some other view may determine that it is really outside, but we don't have that view. In the third case, if the cube is neither inside nor outside, it is a boundary cube.

Instead of using a set of fixed-size cubes, it is more efficient to perform the cube labeling in a hierarchic fashion, using an octree structure. The *octree* is described in detail in the next chapter, but we can use it intuitively here without confusion. Initially a large cube surrounds the data. Since by definition this large cube intersects the data, it is broken into eight smaller cubes. Those cubes that are outside the object can be discarded, while those that are fully inside can be marked as part of the object. The boundary cubes are further subdivided and the process continues up to the desired resolution. The resultant octree represents the 3D object. Figure 13.36 illustrates the hierarchical space carving procedure for the chair object.

The octree representation can be converted into a 3D mesh for viewing purposes as shown in Figure 13.36. After the initial mesh is created, it can be optimized by a method that tries to simplify the mesh and better fit the data. Figure 13.37 shows the registered range data for the dog object (a), the initial mesh (b), and several steps in the optimization procedure defined by Hoppe *et al.* (c)-(f). The final mesh (f) is much more concise and smoother than the initial mesh. It can now be used in a graphics system for rendering realistic views of the object as in Figure 13.38 or in model-based object recognition as in Chapter 14.

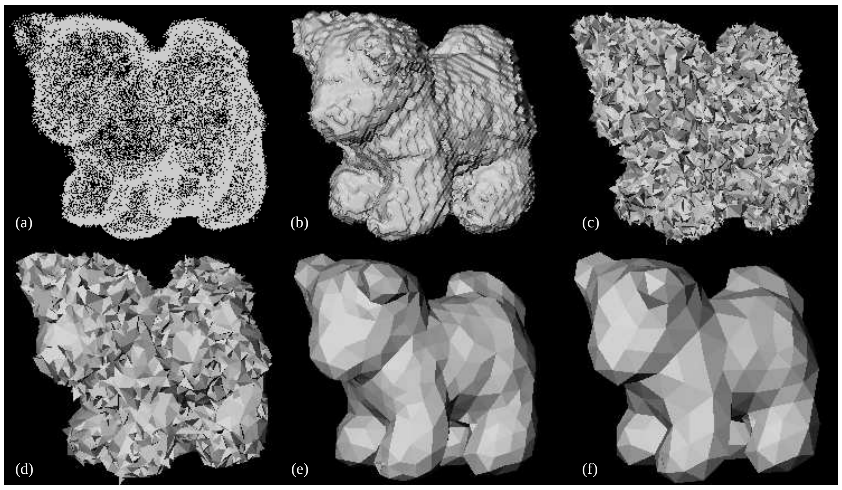


Figure 13.37: The registered range data and five steps in the creation of a dog mesh.

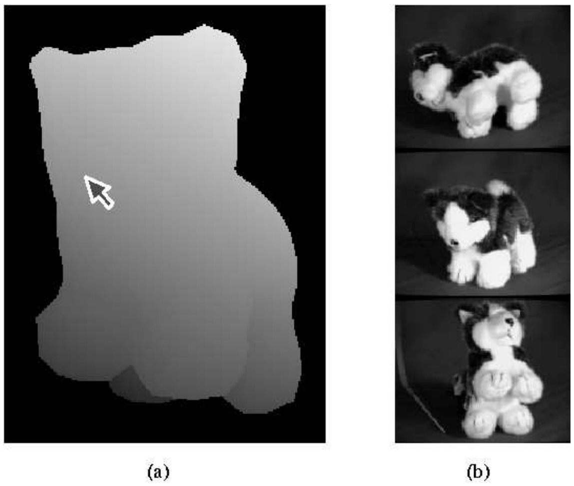


Figure 13.38: (a) A false color rendition of the dog model that a user can manipulate to select a desired view. (b) The 3D point on the model's nose marked by the arrow is projected onto 3 color images of the dog to select pixels that can be used to produce realistic color renderings of the dog.

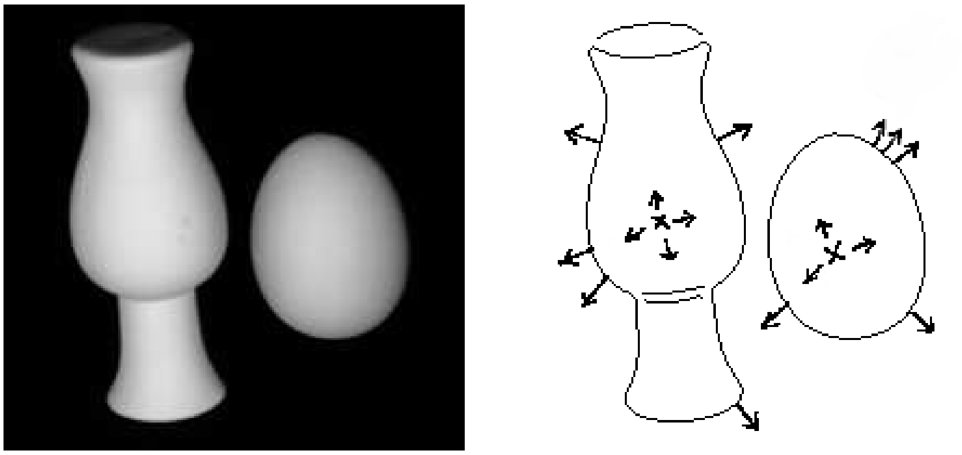


Figure 13.39: (Left) Shaded image of Lambertian objects taken with light source near the camera. (Right) Surface normals sketched on boundaries detected by Canny operator.

13.10 Computing Shape from Shading

Chapters 6 and 12 both discussed how light reflecting off smooth curved objects creates a shaded image. Below, we briefly show how, under certain assumptions, the shape of the object can be computed from the shading in the image.

Humans tend to see a smoothly darkening surface as one that is turning away from the view direction. Using makeup on our face, we can change how others perceive us. Makeup that is darker than our skin color applied to our outer cheeks makes the face look narrower, because the darker shading induces the perception that the surface is turning away from the viewer faster than it actually does. Similarly, makeup lighter than our skin color will have the opposite effect, inducing the perception of a fuller face. Using the formula for Lambertian reflectance, it is possible to map an intensity value (shading) into a surface normal for the surface element being imaged. Early work by Horn studied methods of determining the topography of the moon illuminated by the distant sun and viewed from the distant earth. The family of methods that have evolved to map back from shading in an image to a surface normal in the scene has been called *shape from shading(SFS).*

1 Definition *A* **shape-from-shading** *method computes surface shape* $\mathbf{n} = f(x, y)$ *from a shaded image, where* $\mathbf{n}$ *is the normal to the surface at point* $[x, y]$ *of the image and* ${}^F I[x, y]$ *is the intensity.*

Figure 13.39 motivates shape from shading methods. At the left is an image of objects whose surfaces are approximated nicely by the Lambertian reflectance model: image intensity is proportional to the angle between the surface normal and the direction of illumination. At the right of Figure 13.39 is a sketch of the surface normals at several points on the surfaces of the objects. Clearly, the brightest image points indicate where the surface normal points directly toward the light source: the surface normals pointing back at us appear as X's in the figure. At limb points the surface normal is perpendicular to both the view direction and the surface boundary: this completely constrains the normal in 3D space. Using these constraints, surface normals can be propagated to all the image points. This creates a partial intrinsic image. To obtain the depth z at each image point, we can assign an arbitrary value of z_0 to one of the brightest points and then propagate depth across the image using the changes in the normals.

Exercise 32

Assume a cube with a Lambertian surface posed such that all face normals make an angle of at least $\pi/6$ with the direction of illumination. Clearly, the brightest image points do not correspond to normals pointing at the light source. Why is it true for the egg and vase and not for the cube?

Exercise 33

Using shading, how might it be known whether an object boundary is a limb or blade?

The Lambertian reflectance model is $i = c\ cos\theta$, where constant c results from a combination of source energy, surface albedo, and the distances between the surface element and source and sensor: all these factors are assumed to be constant. The assumption on the distances will hold when the object is many diameters away from both source and sensor. Often, it is also assumed that the illuminant direction is known, but as observed above, illuminant direction can sometimes be computed from weaker assumptions.

The orthographic projection is most convenient for the current development. Also, we use the camera frame as our only frame for 3D space $[x, y, z]$. The observed surface is $z = f(x, y)$: the problem is to compute function f at each image point from the observed intensity values ${}^F I[x, y]$. By rewriting and taking derivatives, we arrive at $\frac{\partial f}{\partial x}\Delta x + \frac{\partial f}{\partial y}\Delta y - \Delta z = 0$, or in vector terms $[p, q, -1] \circ [\Delta x, \Delta y, \Delta z] = 0$, where p and q denote the partial derivatives of f with respect to x and y, respectively. The last equation defines the tangent plane at the surface point $[x, y, f(x, y)]$ that has (nonunit) normal direction $[p, q, -1]$. If we know that $[x_0, y_0, z_0]$ is on the surface, and if we know p and q, then the above planar approximation allows us to compute surface point $[x_0 + \Delta x, y_0 + \Delta y, z_0 + \Delta z]$ by just moving within the approximating tangent plane. We can do this if we can estimate p and q from the intensity image and our assumptions.

We can relate surface normals to intensity by observing a known object. Whenever we know the surface orientation $[p, q, -1]$ for a point $[x, y, f(x, y)]$ observed at $I[x, y]$, we contribute tuple $< p, q, I[x, y] >$ to a mapping so the surface orientation is associated with the shading it produces. Figure 13.40 shows how this is a many to one mapping: all surface orientations making a cone of angle θ with the illuminant direction will yield the same observed intensity. The best object to use for such calibration is a sphere, because (a) it exhibits all surface orientations and (b) the surface normal is readily known from the location of the image point relative to the center and radius of the sphere. Figure 13.41 shows the results of viewing a calibration sphere with two different light sources. For each light source, a reflectance map can be created that stores with each observed intensity the set of all surface orientations that produce that intensity.

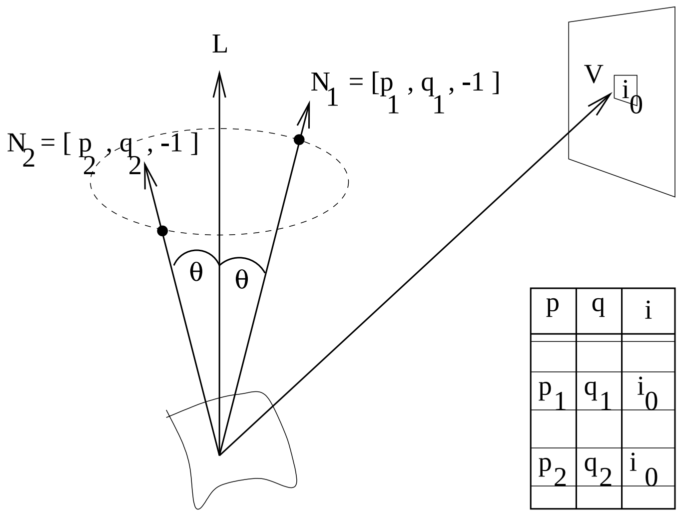


Figure 13.40: (Left) An entire cone of possible surface normals will produce the same observed intensity. (Right) A reflectance map relates surface normals to intensity values: it is a many-to-one mapping.

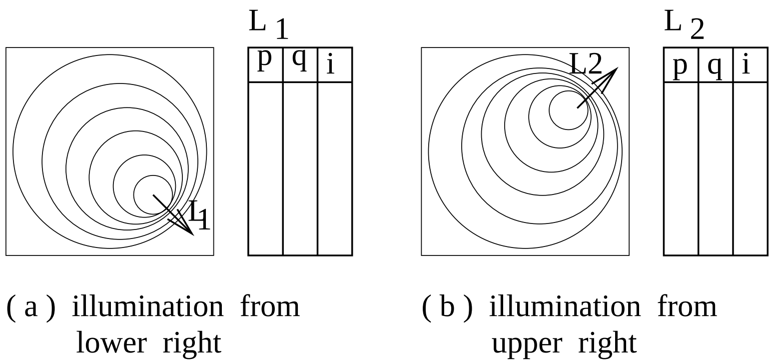


Figure 13.41: Reflectance maps can be created by using a Lambertian calibration sphere of the same material as objects to be sensed. In the observed image of the sphere, p and q are known by analytical geometry for each point $I[x, y]$: we can insert a tuple $< p, q, I[x, y] >$ in the mapping for each image point. A different mapping is obtained for each separately used light source.

Exercise 34

Given a calibration sphere of radius r located at $[0, 0, 100]$ in camera frame $\mathbf{C} : [x, y, z]$, derive the formulas for p and q in terms of location $[x, y]$ in the image. Recall that orthographic projection effectively drops the z-coordinate.

As Figures 13.39 and 13.41 show, image intensity gives a strong constraint on surface orientation, but not a unique surface normal. Additional constraints are needed. There are two general approaches. The first approach is to use information from the spatial neighborhood; for example, a pixel and its 4-neighborhood yield five instances of the shading equation that can be integrated to solve for a smooth surface passing through those five points. The second approach uses more than one intensity image so that multiple equations can be applied to single pixels without regard to neighbors: this has been called *photometric stereo* for obvious reasons.

13.10.1 Photometric Stereo

The photometric stereo method takes multiple images of an object illuminated in sequence by different light sources. A set of intensities is obtained for each image pixel, these can be looked up in a table to obtain the corresponding surface normal. The table is constructed by an offline photometric calibration procedure as shown in Figure 13.41. Algorithm 4 sketches this procedure. Photometric stereo is a fast method that has been shown to work well in a controlled environment. Rajarshi Ray reported excellent results even with specular objects using three balanced light sources. However, if the environment can be tightly controlled to support shape from shading, we can do better using structured light as demonstrated by recent trends in industry.

13.10.2 Integrating Spatial Constraints

Several different methods have been proposed for determining a smooth surface function $z = f(x, y)$ by applying the shading constraint across spatial neighborhoods. One such method is to propagate the surface from the brightest image points as mentioned above. Minimization approaches find the *best* function fitting the available constraints. Figure 13.42 shows results from one such algorithm: a mesh describing the computed surface is shown for two synthetic objects and one real object. These results may or may not be good, depending on the task the data are to support. The method is not considered reliable enough to use in practice.

Shape from shading work has proven that shading information gives strong constraint on surface shape. It is a wonderful example of a "pure" computer vision problem — the input, output, and assumptions are very cleary defined. Many of the mathematical algorithms work well in some cases; however, none work well across a variety of scenes. The interested reader should consult the references to obtain more depth in this subject, especially for the mathematical algorithms, which were only sketched here.

13.11 Structure from Motion

Humans perceive a great deal of information about the 3D structure of the environment by moving through it. When we move or when objects move, or both, we obtain information from images sensed over time. From flow vectors or from corresponding points, the 3D scene surfaces and corners can be reconstructed, as well as the trajectory of the sensor through

Compute surface normals $[p, q]$ for points of a scene viewed with multiple images ^1I, ^2I, ^3I using different light sources ^1L, ^2L, ^3L.

Offline Calibration:

1. Place calibration sphere in the center of the scene.
2. For each of the three light sources ^jL.
   (a) Switch on light source ^jL.
   (b) Record image of calibration sphere.
   (c) Create reflectance map $^j\mathbf{R} = \{< p, q,\ ^jI[x, y] >\}$ where (p, q) is associated with some intensity $^jI[x, y]$ of image ^jI.

Online Surface Sensing:

1. The object to be sensed appears in center of scene.
2. Take three separate images ^jI in rapid succession using each light source ^jL individually.
3. For each image point $[x, y]$
   (a) Use intensity $i_j =\ ^jI[x, y]$ to index reflectance map $^j\mathbf{R}$ and access the set of tuples $R_j = \{(p, q)\}$ associated with intensity i_j.
   (b) "Intersect" the three sets: $S = R_1 \cap R_2 \cap R_3$.
   (c) If S is empty, then set $N[x, y] = NULL$
   else set $N[x, y]$ to the average direction vector in S
4. Return $N[x, y]$ as that part of the intrinsic image storing surface normals.

Algorithm 4: Photometric Stereo with Three Light Sources

Exercise 35 improve Algorithm 4

Improve the efficiency of Algorithm 4 by moving all the set intersection operations into the offline procedure. Justify why this can be done. What data structure is appropriate for storing the results for use in the online procedure?

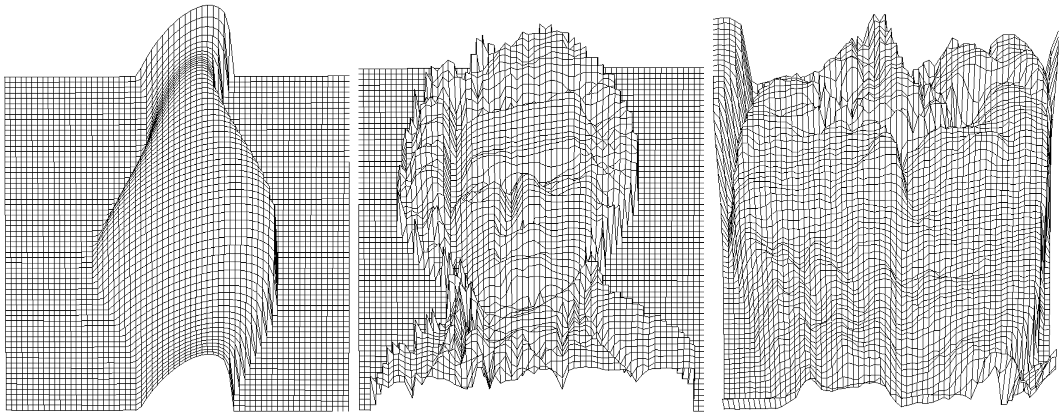


Figure 13.42: Results of the Tsai-Shah algorithm on synthetic and real images. (Left) Surface obtained by algorithm from image generated applying a diffuse lighting model to a CAD model of a vase; (center) surface obtained by algorithm from a synthetic image of a bust of Mozart; (right) surface obtained by algorithm from a real image of a green bell pepper. (Images courtesy of Mubarak Shah.)

the scene. This intuitive process can be refined into an entire class of more specifically defined mathematical problems. Construction of useful computer vision algorithms to compute scene structure and object and observer motion has been difficult: progress has been steady but slow.

Figure 13.43 illustrates a general situation where both the observer and scene objects may be moving. The relative motion of objects and observer produces flow vectors in the image; these might be computable via point matching or optical flow. Figure 13.44 shows the case of two significantly different views of five 3D points. The many cases reported in the literature vary in both the problem definition and the algorithm achieved.

The 3D objects used in the problem definition may be

- points
- lines
- planar surface patches
- curved surface patches

Given the assumptions, an algorithm should yield not only the 3D object structure, but also their motion relative to the camera coordinate system. Much algorithm development has been done assuming that the 3D object points have been reliably sensed and matched. Sensing and matching is difficult and error prone and few convincing demonstrations have been made including them. Algorithms based on image flow use small time steps between images and attempt to compute dense 3D structure, whereas algorithms based on feature correspondences can tolerate larger time steps but can only compute sparse 3D structure.

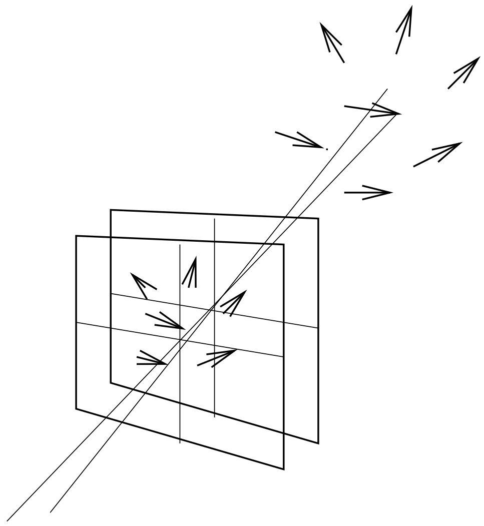


Figure 13.43: An observer moves through a scene of moving objects. The motion of a 3D point projects to a 2D flow vector spanning two images close in time.

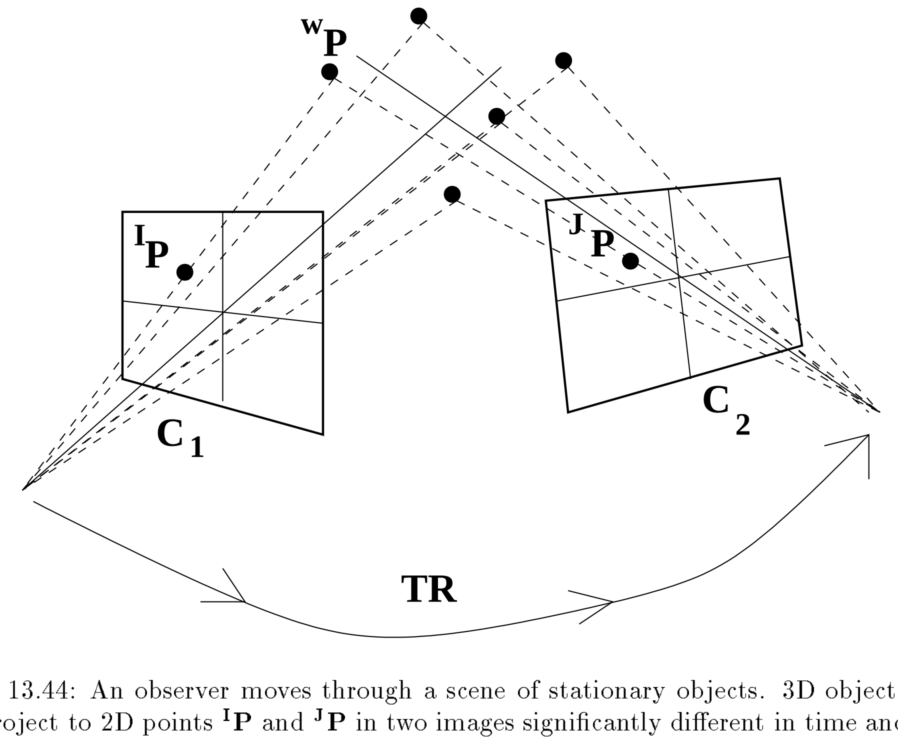


Figure 13.44: An observer moves through a scene of stationary objects. 3D object points $^{\mathbf{W}}\mathbf{P}$ project to 2D points $^{\mathbf{I}}\mathbf{P}$ and $^{\mathbf{J}}\mathbf{P}$ in two images significantly different in time and space so that point correspondence may be difficult. Given image point correspondences, the problem is to compute both the relative motion $\mathbf{TR}$ and the 3D coordinates of the points $^{\mathbf{W}}\mathbf{P}$.

Ullman (1979) reported early results showing that the structure and motion of a rigid configuration of four points could be computed, in theory, from three orthographic projections of these four points using a stationary camera. Ten years later, Huang and Lee (1989) showed that the problem could not be solved using only two orthographic projections. While *minimalist* mathematical models for *shape-from-motion* are interesting and may be difficult to solve, they appear to be impractical because of the errors due to noise or mismatched points. Haralick and Shapiro treat several mathematical approaches and show methods for making the computations robust. Brodsky *et al* (1999) have recently shown good practical results for computing dense shape from a rigidly moving video camera viewing a static scene. In the first chapter of this book, we asked whether or not we might make a 3D model of Notre Dame Cathedral from a video made of it. This is one version of the structure-from-motion problem, and there is now a commercially available solution using computer vision. For a summary of the methods used, refer to the paper by Faugeras *et al* (1998). Having introduced the general problem of computing structure from motion and several of its aspects, we urge the reader who wants to delve deeper to consult the published literature.

13.12 References

The method for affine camera calibration was derived from the earlier work of Ballard and Brown (1982) and Hall *et al* (1982). The latter article also describes a structured light system using a calibrated camera and projector. Several different viable camera calibration methods are available. For object recognition, the affine method for perspective and even weak perspective are often accurate enough. However, for inspection or accurate pose computation, methods that model radial distortion are needed: the widely-used Tsai procedure was reported in Tsai (1987). Calibration is appropriate for many machine vision applications. However, much can be done with an *uncalibrated camera*: this is, in fact, what we have when we scan the world with our video camera. We don't know what the focal length is at every point in time and we don't know any pose parameters relative to any global coordinate system. However, humans do perceive the 3D structure of the world from such imagery. 3D structure can be computed within an unknown scale factor, assuming only that perspective projection applies. The work of Faugeras *et al* (1998) shows how to construct texture-mapped 3D models of buildings from image sequences. Brodsky *et al* (1999) show results for computing the structure of more general surfaces.

Our P3P solution followed closely the work of Ohmura *et al* (1988). A similar work from Linainmaa *et al* (1988) appeared about the same time. However, note that Fischler and Bolles (1981) had studied this same problem and had published a closed form solution to it. Iterative solutions appear to have advantages when the object is being tracked in a sequence of frames, because a starting point is available, which also helps to discard a false solution. A good alternative using a weak perspective projection model is given by Huttenlocher and Ullman (1988). Their method is a fast approximation and the derivation is constructive. M. Fischler and R. Bolles (1981) were the first to formally define and study the *perspective N point* problem and gave a closed form solution for P3P. They also showed how to use it by hypothesizing N correspondences, computing object pose, and then verifying that other model points projected to correponding points in the image. They called their algorithm RANSAC since they suggested randomly choosing correspondences — something that should be avoided if properties of feature points are available.

In recent years there has been much activity in making 3D object models from multiple views in laboratory controlled environments. Many systems and procedures have been developed. The system for object reconstruction that we reported was constructed at The University of Washington by K. Pulli, H. Abi-Rached, P. Neal, and L. Shapiro.

1. D. Ballard and C. Brown (1982), **Computer vision**, Prentice-Hall.
2. P. Ballard and G. Stockman (1995), *Controlling a Computer via Facial Ascpect*, IEEE-Trans-SMC April 95.
3. T. Brodsky, C. Fermuller and Y. Aloimonos (1999), *Shape from Video*, **Proceedings of IEEE CVPR 1999**, Ft Collins, Co. (23-25 June 1999) 146-151.
4. T. S. Huang and C. H. Lee (1989), *Motion and Structure from Orthographic Views*, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol.11, (1989)536-540.
5. Y. Chen and G. Medioni (1992), *Object modeling by registration of multiple range images*, **International Journal of Image and Vision Computing**, VOl. 10, No. 3 (April 1992)145-155.
6. J. Craig (1986) **Introduction to Robotics Mechanics and Control**, Addison-Wesley, 1986.
7. B. Curless and M. Levoy (1996), *A volumetric method for building complex models from range images*, **ACM Siggraph 96**, 301-312.
8. C. Dorai, J. Weng and A. Jain (1994), *Optimal registration of multiple range views*, **Proceedings of the 12th International Conference on Pattern Recognition, Vol.1**, Jerusalem, Israel (Oct 1994)569-571.
9. M.-P. Dubuisson and A. K. Jain (1984), "A modified hausdorff distance for object matching," *Proceedings of the 12th International Conference on Pattern Recognition*, Jerusalem, Israel, 1994.
10. O. Faugeras (1993), *Three-Dimensional Computer Vision, a geometric viewpoint*, The MIT Press, Cambridge, Massachussets, 1993.
11. O. Faugeras, L. Robert, S. Laveau, G. Csurka, C. Zeller, C. Gauclin and I. Zoghlami (1998), *3-D Reconstruction of Urban Scenes from Image Sequences*, **Computer Vision and Image Understanding, Vol. 69**, No. 3 (March 98)292-309.
12. M. Fischler and R. Bolles (1981), *Random concensus: A paradigm for model fitting with applications in image analysis and automated cartography*, **Communications of the ACM, Vol. 24**, (1981)381-395.
13. D. Forsyth *et al* (1991), "Invariant descriptors for 3-d object recognition and pose", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(10):971–991, 1991.

14. E. Hall, J. Tio, C. McPherson and F. Sadjadi (1982), *Measuring curved surfaces for robot vision*, **Computer**, Vol. 15, No. 12 (Dec 1982)385-394.

15. M. Heath (1997), **Scientific Computing: an Introductory Survey**, McGraw-Hill,Inc.

16. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle (1992), *Surface reconstruction from unorganized points*, **Proceedings of SIGGRAPH 92**, 71-78.

17. G.Hu and G.Stockman (1989), *3-D Surface Solution Using Structured Light and Constraint Propagation*, IEEE-TPAMI-11-4 (Apr1989)390-402.

18. D. Huttenlocher and S. Ullman (1988), *Recognizing Solid Objects by Alignment*, **Proc. DARPA Spring Meeting**1114-1122.

19. D. P. Huttenlocher, G. A. Klanderman, and W. J. Rucklidge (1993), *Comparing images using the Hausdorf distance*, **IEEE Transactions on Pattern Analysis and Machine Intelligence**, Vol. 15, No. 9, 850-863.

20. Q. Ji, M. S. Costa, R. M. Haralick, and L. G. Shapiro (1998), "An Integrated Technique for Pose Estimation from Different Geometric Features," *Proceedings of Vision Interface '98*, Vancouver (June 18-20, 1998)77-84.

21. S. Linnainmaa, D. Harwood, and L. Davis (1988), *Pose Determination of a Three-Dimensional Object Using Triangle Pairs*, **IEEE Trans. on Pattern Analysis and Machine Intelligence**, Vol. 10, No. 5 (Sep 1988)634-647.

22. W. H. Press et. al. (1992), *Numerical Recipes in C.* Cambridge University Press, 1992.

23. K. Pulli, H. Abi-Rached, T. Duchamp, L. G. Shapiro, and W. Stuetzle (1998), *Acquisition and visualization of colored 3D objects*, **Proceedings of ICPR 98**, 11-15.

24. R. Ray, J. Birk and R. Kelley (1983), *Error analysis of surface normals determined by radiometry*,**IEEE-TPAMI**, Vol 5, No. 6 (Nov 1983)631-644.

25. N. Shrikhande and G. Stockman (1989), *Surface Orientation from a Projected Grid*,IEEE-TPAMI-11-4(June 1989)650-655.

26. P.-S. Tsai and M. Shah (1992), *A fast linear shape from shading*, **Proceedings IEEE Conf. on Computer Vision and Pattern Recognition**, (June 1992)734-736.

27. R. Tsai (1987), *A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf cameras and lenses*, **IEEE Transactions on Robotics and Automation**, Vol. 3, No. 4.

28. S. Ullman (1979) **The Interpretations of Visual Motion** MIT Press, Cambridge, Mass. (1979).

29. R. Zhang, P.-S. Tsai, J. Cryer and M. Shah, *Shape from Shading: A Survey*, **IEEE-TPAMI** Vol. 21, No. 8, (Aug 1999)690-706.