Chapter 6

COMPUTER VISION

If a robot is to interact with its environment, then the robot must be able to sense its environment. Computer vision is one of the most powerful sensing modalities that currently exist. Therefore, in this chapter we present a number of basic concepts from the field of computer vision. It is not our intention here to cover the now vast field of computer vision. Rather, we aim to present a number of basic techniques that are applicable to the highly constrained problems that often present themselves in industrial applications. The material in this chapter, when combined with the material of previous chapters, should enable the reader to implement a rudimentary vision-based robotic manipulation system. For example, using techniques presented in this chapter, one could design a system that locates objects on a conveyor belt, and determines the positions and orientations of those objects. This information could then be used in conjunction with the inverse kinematic solution for the robot, along with various coordinate transformations, to command the robot to grasp these objects.

We begin by examining the geometry of the image formation process. This will provide us with the fundamental geometric relationships between objects in the world and their projections in an image. We then describe a calibration process that can be used to determine the values for the various camera parameters that appear in these relationships. We then consider image segmentation, the problem of dividing the image into distinct regions that correspond to the background and to objects in the scene. When there are multiple objects in the scene, it is often useful to deal with them individually; therefore, we next present an approach to component labelling. Finally, we describe how to compute the positions and orientations of objects in the image.

6.1 The Geometry of Image Formation

A digital image is a two-dimensional array of pixels that is formed by focusing light onto a two-dimensional array of sensing elements. A lens with focal length λ is used to focus the light onto the sensing array, which is often composed of CCD (charge-coupled device) sensors. The lens and sensing array are packaged together in a camera, which is connected to a digitizer or frame grabber. In the case of analog cameras, the digitizer converts the analog video signal that is output by the camera into discrete values that are then transferred to the pixel array by the frame grabber. In the case of digital cameras, a frame grabber merely transfers the digital data from the camera to the pixel array. Associated with each pixel in the digital image is a gray level value, typically between 0 and 255, which encodes the intensity of the light incident on the corresponding sensing element.

In robotics applications, it is often sufficient to consider only the geometric aspects of image formation. Therefore in this section we will describe only the geometry of the image formation

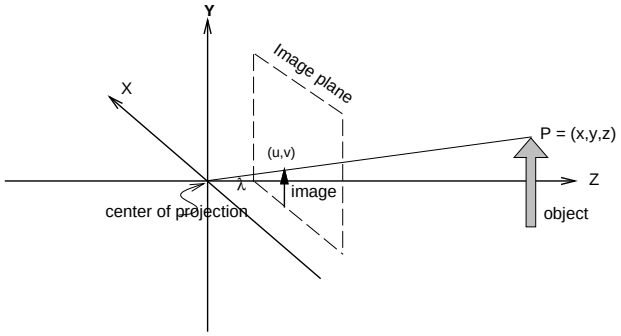


Figure 6.1: Camera coordinate frame.

process. We will not deal with the photometric aspects of image formation (e.g., we will not address issues related to depth of field, lens models, or radiometry).

We will begin the section by assigning a coordinate frame to the imaging system. We then discuss the popular pinhole model of image formation, and derive the corresponding equations that relate the coordinates of a point in the world to its image coordinates. Finally, we describe camera calibration, the process by which all of the relevant parameters associated with the imaging process can be determined.

6.1.1 The Camera Coordinate Frame

In order to simplify many of the equations of this chapter, it often will be useful to express the coordinates of objects relative to a camera centered coordinate frame. For this purpose, we define the camera coordinate frame as follows. Define the image plane, π, as the plane that contains the sensing array. The axes x_c and y_c form a basis for the image plane, and are typically taken to be parallel to the horizontal and vertical axes (respectively) of the image. The axis z_c is perpendicular to the image plane and aligned with the optic axis of the lens (i.e., it passes through the focal center of the lens). The origin of the camera frame is located at a distance λ behind the image plane. This point is also referred to as the *center of projection*. The point at which the optical axis intersects the image plane is known as the *principal point*. This coordinate frame is illustrated in Figure 6.1.

With this assignment of the camera frame, any point that is contained in the image plane will have coordinates (u, v, λ). Thus, we can use (u, v) to parameterize the image plane, and we will refer to (u, v) as image plane coordinates.

6.1.2 Perspective Projection

The image formation process is often modeled by the pinhole lens approximation. With this approximation, the lens is considered to be an ideal pinhole, and the pinhole is located at the focal center of the lens[1]. Light rays pass through this pinhole, and intersect the image plane.

Let P be a point in the world with coordinates x, y, z (relative to the camera frame). Let p denote the projection of P onto the image plane with coordinates (u, v, λ). Under the pinhole assumption, P, p and the origin of the camera frame will be collinear. This can is illustrated in

[1]Note that in our mathematical model, illustrated in Figure 6.1, we have placed the pinhole behind the image plane in order to simplify the model.

Figure 6.1. Thus, for some unknown positive k we have

$$k \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} u \\ v \\ \lambda \end{pmatrix} \tag{6.1}$$

which can be rewritten as the system of equations:

$$kx = u, \tag{6.2}$$
$$ky = v, \tag{6.3}$$
$$kz = \lambda. \tag{6.4}$$

This gives $k = \lambda/z$, which can be substituted into (6.2) and (6.3) to obtain

$$u = \lambda \frac{x}{z}, \qquad v = \lambda \frac{y}{z}. \tag{6.5}$$

These are the well-known equations for perspective projection.

6.1.3 The Image Plane and the Sensor Array

As described above, the image is a discrete array of gray level values. We will denote the row and column indices for a pixel by the coordinates (r, c). In order to relate digital images to the 3D world, we must determine the relationship between the image plane coordinates, (u, v), and indices into the pixel array of pixels, (r, c).

We typically define the origin of this pixel array to be located at a corner of the image (rather than the center of the image). Let the pixel array coordinates of the pixel that contains the principal point be given by (O_r, O_c). In general, the sensing elements in the camera will not be of unit size, nor will they necessarily be square. Denote the s_x and s_y the horizontal and vertical dimensions (respectively) of a pixel. Finally, it is often the case that the horizontal and vertical axes of the pixel array coordinate system point in opposite directions from the horizontal and vertical axes of the camera coordinate frame[2]. Combining these, we obtain the following relationship between image plane coordinates and pixel array coordinates,

$$-\frac{u}{s_x} = (r - O_r), \qquad -\frac{v}{s_y} = (c - O_c). \tag{6.6}$$

which gives,

$$u = -s_x(r - O_r), \qquad v = -s_y(c - O_c). \tag{6.7}$$

Note that the coordinates (r, c) will be integers, since they are the discrete indices into an array that is stored in computer memory. Therefore, it is not possible to obtain the exact image plane coordinates for a point from the (r, c) coordinates.

6.2 Camera Calibration

The objective of camera calibration is to determine all of the parameters that are necessary to predict the image pixel coordinates (r, c) of the projection of a point in the camera's field of view,

[2]This is an artifact of our choice to place the center of projection behind the image plane. The directions of the pixel array axes may vary, depending on the frame grabber.

given that the coordinates of that point with respect to the world coordinate frame are know. In other words, given the coordinates of P relative to the world coordinate frame, after we have calibrated the camera we will be able to predict (r, c), the image pixel coordinates for the projection of this point.

6.2.1 Extrinsic Camera Parameters

To this point, in our derivations of the equations for perspective projection, we have dealt only with coordinates expressed relative to the camera frame. In typical robotics applications, tasks will be expressed in terms of the world coordinate frame, and it will therefore be necessary to perform coordinate transformations. If we know the position and orientation of the camera frame relative to the world coordinate frame we have

$$x^w = \mathbf{R}_c^w x^c + O_c^w \tag{6.8}$$

or, if we know x^w and wish to solve for x^c,

$$x^c = \mathbf{R}_w^c (x^w - O_c^w) \tag{6.9}$$

In the remainder of this section, to simplify notation we will define

$$\mathbf{R} = \mathbf{R}_w^c, \quad \mathbf{T} = -\mathbf{R}_w^c O_c^w. \tag{6.10}$$

Thus,

$$x^c = \mathbf{R}x^w + \mathbf{T} \tag{6.11}$$

Cameras are typically mounted on tripods, or on mechanical positioning units. In the latter case, a popular configuration is the pan/tilt head. A pan/tilt head has two degrees of freedom: a rotation about the world z axis and a rotation about the pan/tilt head's x axis. These two degrees of freedom are analogous to the those of a human head, which can easily look up or down, and can turn from side to side. In this case, the rotation matrix $\mathbf{R}$ is given by

$$\mathbf{R} = \mathbf{R}_{z,\theta}\mathbf{R}_{x,\alpha}, \tag{6.12}$$

where θ is the pan angle and α is the tilt angle. More precisely, θ is the angle between the world x-axis and the camera x-axis, about the world z-axis, while α is the angle between the world z-axis and the camera z-axis, about the camera x-axis.

6.2.2 Intrinsic Camera Parameters

Using the pinhole model, we obtained the following equations that map the coordinates of a point expressed with respect to the camera frame to the corresponding pixel coordinates:

$$r = -\frac{u}{s_x} + O_r, \qquad c = -\frac{v}{s_y} + O_c, \qquad u = \lambda\frac{x}{z} \qquad v = \lambda\frac{y}{z}. \tag{6.13}$$

These equations can be combined to give

$$r = -\frac{\lambda}{s_x}\frac{x}{z} + O_r, \qquad c = -\frac{\lambda}{s_y}\frac{y}{z} + O_c, \tag{6.14}$$

Thus, once we know the values of the parameters $\lambda, s_x, O_r, s_y, O_c$ we can determine (r, c) from (x, y, z), where (x, y, z) are coordinates relative to the camera frame. In fact, we don't need to know all of λ, s_x, s_y; it is sufficient to know the ratios

$$f_x = -\frac{\lambda}{s_x} \qquad f_y = -\frac{\lambda}{s_y}. \tag{6.15}$$

These parameters, f_x, O_r, f_y, O_c are known as the intrinsic parameters of the camera. They are constant for a given camera, and do not change when the camera moves.

6.2.3 Determining the Camera Parameters

The task of camera calibration is to determine the intrinsic and extrinsic parameters of the camera. We will proceed by first determining the parameters associated with the image center, and then solving for the remaining parameters.

Of all the camera parameters, O_r, O_c (the image pixel coordinates of the principal point) are the easiest to determine. This can be done by using the idea of vanishing points. Although a full treatment of vanishing points is beyond the scope of this text, the idea is simple: a set of parallel lines in the world will project onto image lines that intersect at a single point, and this intersection point is known as a *vanishing point*. The vanishing points for three mutually orthogonal sets of lines in the world will define a triangle in the image. The orthocenter of this triangle (i.e., the point at which the three altitudes intersect) is the image principal point (a proof of this is beyond the scope of this text). Thus, a simple way to compute the principal point is to position a cube in the workspace, find the edges of the cube in the image (this will produce the three sets of mutually orthogonal parallel lines), compute the intersections of the image lines that correspond to each set of parallel lines in the world, and determine the orthocenter for the resulting triangle.

Once we know the principal point, we proceed to determine the remaining camera parameters. This is done by constructing a linear system of equations in terms of the known coordinates of points in the world and the pixel coordinates of their projections in the image. The unknowns in this system are the camera parameters. Thus, the first step in this stage of calibration is to acquire a data set of the form $r_1, c_1, x_1, y_1, z_1, r_2, c_2, x_2, y_2, z_2, \cdots r_N, c_N, x_N, y_N, z_N$, in which r_i, c_i are the image pixel coordinates of the projection of a point in the world with coordinates x_i, y_i, z_i relative to the world coordinate frame. This acquisition is often done manually, e.g., by having a robot move a small bright light to known x, y, z coordinates in the world, and then hand selecting the corresponding image point.

Once we have acquired the data set, we proceed to set up the linear system of equations. The extrinsic parameters of the camera are given by

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \quad \mathbf{T} = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}. \tag{6.16}$$

With respect to the camera frame, the coordinates of a point in the world are thus given by

$$x^c = r_{11}x + r_{12}y + r_{13}z + T_x \tag{6.17}$$

$$y^c = r_{21}x + r_{22}y + r_{23}z + T_y \tag{6.18}$$

$$z^c = r_{31}x + r_{32}y + r_{33}z + T_z. \tag{6.19}$$

Combining this with (6.14) we obtain

$$r - O_r = -f_x \frac{x^c}{z^c} = -f_x \frac{r_{11}x + r_{12}y + r_{13}z + T_x}{r_{31}x + r_{32}y + r_{33}z + T_z} \tag{6.20}$$

$$c - O_c = -f_y \frac{y^c}{z^c} = -f_y \frac{r_{21}x + r_{22}y + r_{23}z + T_y}{r_{31}x + r_{32}y + r_{33}z + T_z}. \tag{6.21}$$

Since we know the coordinates of the principal point, we an simplify these equations by using the simple coordinate transformation

$$r \leftarrow r - O_r, \qquad c \leftarrow c - O_c. \tag{6.22}$$

We now write the two transformed projection equations as functions of the unknown variables: $r_{ij}, T_x, T_y, T_z, f_x, f_y$. This is done by solving each of these equations for z^c, and setting the resulting equations to be equal to one another. In particular, for the data points r_i, c_i, x_i, y_i, z_i we have

$$r_i f_y (r_{21}x_i + r_{22}y_i + r_{23}z_i + T_y) = c_i f_x (r_{11}x_i + r_{12}y_i + r_{13}z_i + T_x). \tag{6.23}$$

We define $\alpha = f_x/f_y$ and rewrite this as:

$$r_i r_{21}x_i + r_i r_{22}y_i + r_i r_{23}z_i + r_i T_y - \alpha c_i r_{11}x_i - \alpha c_i r_{12}y_i - \alpha c_i r_{13}z_i - \alpha c_i T_x = 0. \tag{6.24}$$

We can combine the N such equations into the matrix equation

$$\begin{bmatrix} r_1x_1 & r_1y_1 & r_1z_1 & r_1 & -c_1x_1 & -c_1y_1 & -c_1z_1 & -c_1 \\ r_2x_2 & r_2y_2 & r_2z_2 & r_2 & -c_2x_2 & -c_2y_2 & -c_2z_2 & -c_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ r_Nx_N & r_Ny_N & r_Nz_N & r_N & -c_Nx_N & -c_Ny_N & -c_Nz_N & -c_N \end{bmatrix} \begin{bmatrix} r_{21} \\ r_{22} \\ r_{23} \\ T_y \\ \alpha r_{11} \\ \alpha r_{12} \\ \alpha r_{13} \\ \alpha T_x \end{bmatrix} = 0 \tag{6.25}$$

This is an equation of the form $A\mathbf{x} = 0$. As such, if $\bar{\mathbf{x}} = [\bar{x}_1, \cdots, \bar{x}_8]^T$ is a solution, for (6.25) we only know that this solution is some scalar multiple of the desired solution, $\mathbf{x}$, i.e.,

$$\bar{\mathbf{x}} = k[r_{21}, r_{22}, r_{23}, T_y, \alpha r_{11}, \alpha r_{12}, \alpha r_{13}, \alpha T_x]^T, \tag{6.26}$$

in which k is an unknown scale factor.

In order to solve for the true values of the camera parameters, we can exploit constraints that arise from the fact that $\mathbf{R}$ is a rotation matrix. In particular,

$$(\bar{x}_1^2 + \bar{x}_2^2 + \bar{x}_3^2)^{\frac{1}{2}} = (k^2(r_{21}^2 + r_{22}^2 + r_{23}^2))^{\frac{1}{2}} = |k|, \tag{6.27}$$

and likewise

$$(\bar{x}_5^2 + \bar{x}_6^2 + \bar{x}_7^2)^{\frac{1}{2}} = (\alpha^2 k^2(r_{21}^2 + r_{22}^2 + r_{23}^2))^{\frac{1}{2}} = \alpha|k| \tag{6.28}$$

(note that by definition, $\alpha > 0$).

Our next task is to determine the sign of k. Using equations (6.14) we see that $r \times x^c < 0$ (recall that we have used the coordinate transformation $r \leftarrow r - O_r$). Therefore, to determine the

sign of k, we first assume that $k > 0$. If $r(r_{11}x + r_{12}y + r_{13}z + T_x) < 0$, then we know that we have made the right choice and $k > 0$; otherwise, we know that $k < 0$.

At this point, we know the values for $k, \alpha, r_{21}, r_{22}, r_{23}, r_{11}, r_{12}, r_{13}, T_x, T_Y$, and all that remains is to determine T_z, f_x, f_y. Since $\alpha = f_x/f_y$, we need only determine T_z and f_x. Returning again to the projection equations, we can write

$$r = -f_x \frac{x^c}{z^c} = -f_x \frac{r_{11}x + r_{12}y + r_{13}z + T_x}{r_{31}x + r_{32}y + r_{33}z + T_z} \tag{6.29}$$

Using an approach similar to that used above to solve for the first eight parameters, we can write this as the linear system

$$r(r_{31}x + r_{32}y + r_{33}z + T_z) = -f_x(r_{11}x + r_{12}y + r_{13}z + T_x) \tag{6.30}$$

which can easily be solved for T_Z and f_x.

6.3 Segmentation by Thresholding

Segmentation is the process by which an image is divided into meaningful components. Segmentation has been the topic of computer vision research since its earliest days, and the approaches to segmentation are far too numerous to survey here. These approaches are sometimes concerned with finding *features* in an image (e.g., edges), and sometimes concerned with partitioning the image into homogeneous regions (region-based segmentation). In many practical applications, the goal of segmentation is merely to divide the image into two regions: one region that corresponds to an object in the scene, and one region that corresponds to the background. In many industrial applications, this segmentation can be accomplished by a straight-forward thresholding approach. Pixels whose gray level is greater than the threshold are considered to belong to the object, and pixels whose gray level is less than or equal to the threshold are considered to belong to the background.

In this section we will describe an algorithm that automatically selects a threshold. This basic idea behind the algorithm is that the pixels should be divided into two groups (background and object), and that the intensities of the pixels in a particular group should all be fairly similar. To quantify this idea, we will use some standard techniques from statistics. Thus, we begin the section with a quick review of the necessary concepts from statistics and then proceed to describe the threshold selection algorithm.

6.3.1 A Brief Statistics Review

Many approaches to segmentation exploit statistical information contained in the image. In this section, we briefly review some of the more useful statistical concepts that are used by segmentation algorithms.

The basic premise for most of these statistical concepts is that the gray level value associated with a pixel in an image is a random variable that takes on values in the set $\{0, 1, \cdots N-1\}$. Let $P(z)$ denote the probability that a pixel has gray level value z. In general, we will not know this probability, but we can estimate it with the use of a *histogram*. A histogram is an array, H, that encodes the number of occurrences of each gray level value. In particular, the entry $H[z]$ is the number of times gray level value z occurs in the image. Thus, $0 \leq H[z] \leq N_{rows} \times N_{cols}$ for all z. An algorithm to compute the histogram for an image is shown in figure 6.2.

```
For i = 0 to N − 1
    H[i] ← 0
For r = 0 to N_rows − 1
    For c = 0 to N_cols − 1
        Index ← Image(r, c)
        H[Index] ← H[Index] + 1
```

Figure 6.2: Pseudo-code to compute an image histogram.

Given the histogram for the image, we estimate the probability that a pixel will have gray level z by

$$P(z) = \frac{H[z]}{N_{rows} \times N_{cols}}. \tag{6.31}$$

Thus, the image histogram is a scaled version of our approximation of P.

Given P, we can compute the *average*, or *mean* value of the gray level values in the image. We denote the mean by μ, and compute it by

$$\mu = \sum_{z=0}^{N-1} zP(z). \tag{6.32}$$

In many applications, the image will consist of one or more objects against some background. In such applications, it is often useful to compute the mean for each object in the image, and also for the background. This computation can be effected by constructing individual histogram arrays for each object, and for the background, in the image. If we denote by H_i the histogram for the i^{th} object in the image (where $i = 0$ denotes the background), the mean for the i^{th} object is given by

$$\mu_i = \sum_{z=0}^{N-1} z \, \frac{H_i[z]}{\sum_{z=0}^{N-1} H_i[z]}, \tag{6.33}$$

which is a straightforward generalization of (6.32). The term

$$\frac{H_i[z]}{\sum_{z=0}^{N-1} H_i[z]}$$

is in fact an estimate of the probability that a pixel will have gray level value z given that the pixel is a part of object i in the image. For this reason, μ_i is sometimes called a *conditional mean*.

The mean conveys useful, but very limited information about the distribution of grey level values in an image. For example, if half or the pixels have gray value 127 and the remaining half have gray value 128, the mean will be $\mu = 127.5$. Likewise, if half or the pixels have gray value 255 and the remaining half have gray value 0, the mean will be $\mu = 127.5$. Clearly these two images are very different, but this difference is not reflected by the mean. One way to capture this difference is to compute the the average deviation of gray values from the mean. This average would be small for the first example, and large for the second. We could, for example, use the average value of $|z - \mu|$; however, it will be more convenient mathematically to use the square of this value instead. The resulting quantity is known as the *variance*, which is defined by

$$\sigma^2 = \sum_{z=0}^{N-1} (z-\mu)^2 P(z). \tag{6.34}$$

As with the mean, we can also compute the conditional variance, σ_i^2 for each object in the image

$$\sigma_i^2 = \sum_{z=0}^{N-1} (z-\mu_i)^2 \frac{H_i[z]}{\sum_{z=0}^{N-1} H_i[z]}. \tag{6.35}$$

6.3.2 Automatic Threshold Selection

We are now prepared to develop an automatic threshold selection algorithm. We will assume that the image consists of an object and a background, and that the background pixels have gray level values less than or equal to some threshold while the object pixels are above the threshold. Thus, for a given threshold value, z_t, we divide the image pixels into two groups: those pixels with gray level value $z \leq z_t$, and those pixels with gray level value $z > z_t$. We can compute the means and variance for each of these groups using the equations of Section 6.3.1. Clearly, the conditional means and variances depend on the choice of z_t, since it is the choice of z_t that determines which pixels will belong to each of the two groups. The approach that we take in this section is to determine the value for z_t that minimizes a function of the variances of these two groups of pixels.

In this section, it will be convenient to rewrite the conditional means and variances in terms of the pixels in the two groups. To do this, we define $q_i(z_t)$ as the probability that a pixel in the image will belong to group i for a particular choice of threshold, z_t. Since all pixels in the background have gray value less than or equal to z_t and all pixels in the object have gray value greater than z_t, we can define $q_i(z_t)$ for $i = 0, 1$ by

$$q_0(z_t) = \frac{\sum_{z=0}^{z_t} H[z]}{(N_{rows} \times N_{cols})}, \quad q_1(z_t) = \frac{\sum_{z=z_t+1}^{N-1} H[z]}{(N_{rows} \times N_{cols})}. \tag{6.36}$$

We now rewrite (6.33) as

$$\mu_i = \sum_{z=0}^{N-1} z \frac{H_i[z]}{\sum_{z=0}^{N-1} H_i[z]} = \sum_{z=0}^{N-1} z \frac{H_i[z]/(N_{rows} \times N_{cols})}{\sum_{z=0}^{N-1} H_i[z]/(N_{rows} \times N_{cols})}$$

Using again the fact that the two pixel groups are defined by the threshold z_t, we have

$$\frac{H_0[z]}{(N_{rows} \times N_{cols})} = \frac{P(z)}{q_0(z_t)}, \quad z \leq z_t \qquad \text{and} \qquad \frac{H_1[z]}{(N_{rows} \times N_{cols})} = \frac{P(z)}{q_1(z_t)}, \quad z > z_t. \tag{6.37}$$

Thus, we can write the conditional means for the two groups as

$$\mu_0(z_t) = \sum_{z=0}^{z_t} z \frac{P(z)}{q_0(z_t)}, \quad \mu_1(z_t) = \sum_{z=z_t+1}^{N-1} z \frac{P(z)}{q_1(z_t)}. \tag{6.38}$$

Similarly, we can write the equations for the conditional variances by

$$\sigma_0^2(z_t) = \sum_{z=0}^{z_t} (z-\mu_0(z_t))^2 \frac{P(z)}{q_0(z_t)}, \quad \sigma_1^2(z_t) = \sum_{z=z_t+1}^{N} (z-\mu_1(z_t))^2 \frac{P(z)}{q_1(z_t)}. \tag{6.39}$$

We now turn to the selection of z_t. If nothing is known about the true values of μ_i or σ_i^2, how can we determine the optimal value of z_t? To answer this question, recall that the variance is a measure of the average deviation of pixel intensities from the mean. Thus, if we make a good choice for z_t, we would expect that the variances $\sigma_i^2(z_t)$ would be small. This reflects the assumption that pixels belonging to the object will be clustered closely about μ_1, pixels belonging to the background will be clustered closely about μ_0. We could, therefore, select the value of z_t that minimizes the sum of these two variances. However, it is unlikely that the object and background will occupy the same number of pixels in the image; merely adding the variances gives both regions equal importance. A more reasonable approach is to weight the two variances by the probability that a pixel will belong to the corresponding region,

$$\sigma_w^2(z_t) = q_0(z_t)\sigma_0^2(z_t) + q_1(z_t)\sigma_1^2(z_t). \tag{6.40}$$

The value σ_w^2 is known as the *within-group variance*. The approach that we will take is to minimize this within-group variance.

At this point we could implement a threshold selection algorithm. The naive approach would be to simply iterate over all possible values of z_t and select the one for which $\sigma_w^2(z_t)$ is smallest. Such an algorithm performs an enormous amount of calculation, much of which is identical for different candidate values of the threshold. For example, most of the calculations required to compute $\sigma_w^2(z_t)$ are also required to compute $\sigma_w^2(z_t + 1)$; the required summations change only slightly from one iteration to the next. Therefore, we now turn our attention to an efficient algorithm.

To develop an efficient algorithm, we take two steps. First, we will derive the between-group variance, σ_b^2, which depends on the within-group variance and the variance over the entire image. The between-group variance is a bit simpler to deal with than the within-group variance, and we will show that maximizing the between-group variance is equivalent to minimizing the within-group variance. Then, we will derive a recursive formulation for the between-group variance that lends itself to an efficient implementation.

To derive the between-group variance, we begin by expanding the equation for the total variance of the image, and then simplifying and grouping terms. The variance of the gray level values in the image is given by (6.34), which can be rewritten as

$$\begin{aligned}
\sigma^2 &= \sum_{z=0}^{N-1}(z-\mu)^2 P(z) \\
&= \sum_{z=0}^{z_t}(z-\mu)^2 P(z) + \sum_{z=z_t+1}^{N-1}(z-\mu)^2 P(z) \\
&= \sum_{z=0}^{z_t}(z-\mu_0+\mu_0-\mu)^2 P(z) + \sum_{z=z_t+1}^{N-1}(z-\mu_1+\mu_1-\mu)^2 P(z) \\
&= \sum_{z=0}^{z_t}[(z-\mu_0)^2 + 2(z-\mu_0)(\mu_0-\mu) + (\mu_0-\mu)^2]P(z) \\
&\quad + \sum_{z=z_t+1}^{N-1}[(z-\mu_1)^2 + 2(z-\mu_1)(\mu_1-\mu) + (\mu_1-\mu)^2]P(z).
\end{aligned} \tag{6.41}$$

Note that the we have not explicitly noted the dependence on z_t here. In the remainder of this section, to simplify notation, we will refer to the group probabilities and conditional means and variances as q_i, μ_i, and σ_i^2, without explicitly noting the dependence on z_t. This last expression

(6.41) can be further simplified by examining the cross-terms

$$\begin{aligned}\sum(z-\mu_i)(\mu_i-\mu)P(z) &= \sum z\mu_i P(z) - \sum z\mu P(z) - \sum \mu_i^2 P(z) + \sum \mu_i \mu P(z) \\ &= \mu_i \sum zP(z) - \mu \sum zP(z) - \mu_i^2 \sum P(z) + \mu_i \mu \sum P(z) \\ &= \mu_i(\mu_i q_i) - \mu(\mu_i q_i) - \mu_i^2 q_i + \mu_i \mu q_i \\ &= 0,\end{aligned}$$

in which the summations are taken for z from 0 to z_t for the background pixels (i.e., $i = 0$) and z from $z_t + 1$ to $N - 1$ for the object pixels (i.e., $i = 1$). Therefore, we can simplify (6.41) to obtain

$$\begin{aligned}\sigma^2 &= \sum_{z=0}^{z_t}[(z-\mu_0)^2 + (\mu_0-\mu)^2]P(z) + \sum_{z=z_t+1}^{N-1}[(z-\mu_1)^2 + (\mu_1-\mu)^2]P(z) \\ &= q_0\sigma_0^2 + q_0(\mu_0-\mu)^2 + q_1\sigma_1^2 + q_1(\mu_1-\mu)^2 \\ &= \{q_0\sigma_0^2 + q_1\sigma_1^2\} + \{q_0(\mu_0-\mu)^2 + q_1(\mu_1-\mu)^2\} \\ &= \sigma_w^2 + \sigma_b^2 \qquad (6.42)\end{aligned}$$

in which

$$\sigma_b^2 = q_0(\mu_0-\mu)^2 + q_1(\mu_1-\mu)^2. \qquad (6.43)$$

Since σ^2 does not depend on the threshold value (i.e., it is constant for a specific image), minimizing σ_w^2 is equivalent to maximizing σ_b^2. This is preferable because σ_b^2 is a function only of the q_i and μ_i, and is thus simpler to compute than σ_w^2, which depends also on the σ_i^2. In fact, by expanding the squares in (6.43), using the facts that $q_1 = 1 - q_0$ and $\mu = q_1\mu_0 + q_1\mu_1$, we obtain

$$\sigma_b^2 = q_0(1-q_0)(\mu_0-\mu_1)^2. \qquad (6.44)$$

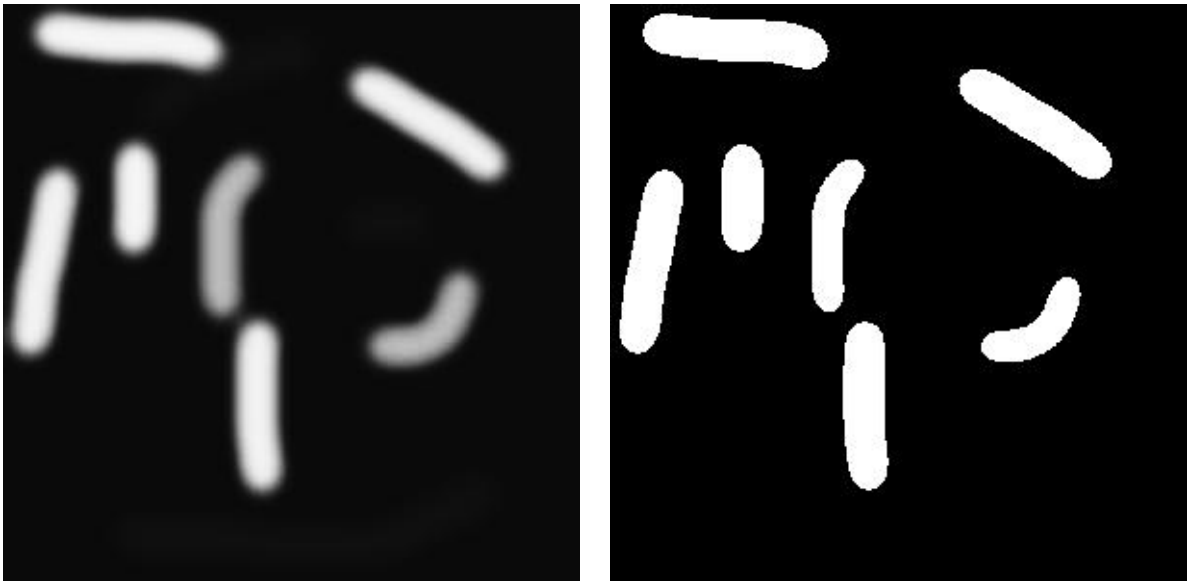
The simplest algorithm to maximize σ_b^2 is to iterate over all possible threshold values, and select the one that maximizes σ_b^2. However, as discussed above, such an algorithm performs many redundant calculations, since most of the calculations required to compute $\sigma_b^2(z_t)$ are also required to compute $\sigma_b^2(z_t+1)$. Therefore, we now turn our attention to an efficient algorithm that maximizes $\sigma_b^2(z_t)$. The basic idea for the efficient algorithm is to re-use the computations needed for $\sigma_b^2(z_t)$ when computing $\sigma_b^2(z_t + 1)$. In particular, we will derive expressions for the necessary terms at iteration $z_t + 1$ in terms of expressions that were computed at iteration z_t. We begin with the group probabilities, and determine the recursive expression for q_0 as

$$q_0(z_t+1) = \sum_{z=0}^{z_t+1} P(z) = P(z_t+1) + \sum_{z=0}^{z_t} P(z) = P(z_t+1) + q_0(z_t). \qquad (6.45)$$

In this expression, $P(z_t + 1)$ can be obtained directly from the histogram array, and $q_0(z_t)$ is directly available because it was computed on the previous iteration of the algorithm. Thus, given the results from iteration z_t, very little computation is required to compute the value for q_0 at iteration $z_t + 1$.

For the conditional mean $\mu_0(z_t)$ we have

$$\mu_0(z_t+1) = \sum_{z=0}^{z_t+1} z\frac{P(z)}{q_0(z_t+1)} \qquad (6.46)$$



(a) (b)

Figure 6.3: (a) A 300×300 pixel image with 256 gray levels. (b) Thresholded version of the image in (a).

$$= \frac{(z_t+1)P(z_t+1)}{q_0(z_t+1)} + \sum_{z=0}^{z_t} z\frac{P(z)}{q_0(z_t+1)} \tag{6.47}$$

$$= \frac{(z_t+1)P(z_t+1)}{q_0(z_t+1)} + \frac{q_0(z_t)}{q_0(z_t+1)}\sum_{z=0}^{z_t} z\frac{P(z)}{q_0(z_t)} \tag{6.48}$$

$$= \frac{(z_t+1)P(z_t+1)}{q_0(z_t+1)} + \frac{q_0(z_t)}{q_0(z_t+1)}\mu_0(z_t) \tag{6.49}$$

Again, all of the quantities in this expression are available either from the histogram, or as the results of calculations performed at iteration z_t of the algorithm.

To compute $\mu_1(z_t+1)$, we use the relationship $\mu = q_0\mu_0 + q_1\mu_1$, which can be easily obtained using (6.32) and (6.38). Thus, we have

$$\mu_1(z_t+1) = \frac{\mu - q_0(z_t+1)\mu_0(z_t+1)}{q_1(z_t+1)} = \frac{\mu - q_0(z_t+1)\mu_0(z_t+1)}{1 - q_0(z_t+1)}. \tag{6.50}$$

We are now equipped to construct a highly efficient algorithm that automatically selects a threshold value that minimizes the within-group variance. This algorithm simply iterates from 0 to $N-1$ (where N is the total number of gray level values), computing q_0, μ_0, μ_1 and σ_b^2 at each iteration using the recursive formulations given in (6.45), (6.49), (6.50) and (6.44). The algorithm returns the value of z_t for which σ_b^2 is largest. Figure **??** shows a grey level image and the binary, thresholded image that results from the application of this algorithm.

6.4 Connected Components

It is often the case that multiple objects will be present in a single image. When this occurs, after thresholding there will be multiple connected components with gray level values that are above

the threshold. In this section, we will first make precise the notion of a connected component, and then describe an algorithm that assigns a unique label to each connected component, i.e., all pixels within a single connected component have the same label, but pixels in different connected components have different labels.

In order to define what is meant by a connected component, it is first necessary to define what is meant by connectivity. For our purposes, it is sufficient to say that a pixel, A, with image pixel coordinates (r, c) is adjacent to four pixels, those with image pixel coordinates $(r-1, c)$, $(r+1, c)$, $(r, c+1)$, and $(r, c-1)$. In other words, each image pixel A (except those at the edges of the image) has four neighbors: the pixel directly above, directly below, directly to the right and directly to the left of pixel A. This relationship is sometimes referred to as *4-connectivity*, and we say that two pixels are 4-connected if they are adjacent by this definition. If we expand the definition of adjacency to include those pixels that are diagonally adjacent (i.e., the pixels with coordinates $(r-1, c-1)$, $(r-1, c+1)$, $(r+1, c-1)$, and $(r+1, c+1)$), then we say that adjacent pixels are 8-connected. In this text, we will consider only the case of 4-connectivity.

A connected component is a collection of pixels, S, such that for any two pixels, say P and P', in S, there is a 4-connected path between them and this path is contained in S. Intuitively, this definition means that it is possible to move from P to P' by "taking steps" only to adjacent pixels without ever leaving the region S. The purpose of a component labeling algorithm is to assign a unique label to each such S.

There are many component labeling algorithms that have been developed over the years. Here, we describe a simple algorithm that requires two passes over the image. This algorithm performs two raster scans of the image (note: a raster scan visits each pixel in the image by traversing from left to right, top to bottom, in the same way that one reads a page of text). On the first raster scan, when an object pixel P, (i.e., a pixel whose gray level is above the threshold value), is encountered, its previously visited neighbors (i.e., the pixel immediately above and the pixel immediately to the left of P) are examined, and if they have gray value that is below the threshold (i.e., they are background pixels), a new label is given to P. This is done by using a global counter that is initialized to zero, and is incremented each time a new label is needed. If either of these two neighbors have already received labels, then P is given the smaller of these, and in the case when both of the neighbors have received labels, an equivalence is noted between those two labels. For example, in Figure 6.4, after the first raster scan labels (2,3,4) are noted as equivalent. In the second raster scan, each pixel's label is replaced by the smallest label to which it is equivalent. Thus, in the example of Figure 6.4, at the end of the second raster scan labels 3 and 4 have been replaced by the label 2.

After this algorithm has assigned labels to the components in the image, it is not necessarily the case that the labels will be the consecutive integers $(1, 2, \cdots)$. Therefore, a second stage of processing is sometimes used to relabel the components to achieve this. In other cases, it is desirable to give each component a label that is very different from the labels of the other components. For example, if the component labelled image is to be displayed, it is useful to increase the contrast, so that distinct components will actually appear distinct in the image (a component with the label 2 will appear almost indistinguishable from a component with label 3 if the component labels are used as pixel gray values in the displayed component labelled image). The results of applying this process to the image in Figure 6.3 are shown in Figure 6.5.

When there are multiple connected object components, it is often useful to process each component individually. For example, we might like to compute the sizes of the various components. For this purpose, it is useful to introduce the *indicator function* for a component. The indicator function for component i, denoted by $\mathcal{I}_i$, is a function that takes on the value 1 for pixels that are

0	0	0	0	0	0	0	0	0	0
0	X	X	X	0	0	0	0	0	0
0	X	X	X	0	0	0	0	0	0
0	X	X	X	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	X	0	0	X	X	0
0	0	0	0	X	0	0	X	X	0
0	0	0	0	X	X	X	X	X	0
0	X	X	X	X	X	X	X	X	0
0	X	X	X	X	X	X	X	X	0
0	0	0	0	0	0	0	0	0	0

(a)

0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	2	0	0	3	3	0
0	0	0	0	2	0	0	3	3	0
0	0	0	0	2	2	2	2	2	0
0	4	4	4	2	2	2	2	2	0
0	4	4	4	2	2	2	2	2	0
0	0	0	0	0	0	0	0	0	0

(b)

0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	2	0	0	3	3	0
0	0	0	0	2	0	0	3	3	0
0	0	0	0	2	2	2	2	2	0
0	4	4	4	X	2	2	2	2	0
0	4	4	4	2	2	2	2	2	0
0	0	0	0	0	0	0	0	0	0

(c)

0	0	0	0	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	1	1	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	2	0	0	2	2	0
0	0	0	0	2	0	0	2	2	0
0	0	0	0	2	2	2	2	2	0
0	2	2	2	2	2	2	2	2	0
0	2	2	2	2	2	2	2	2	0
0	0	0	0	0	0	0	0	0	0

(d)

Figure 6.4: The image in (a) is a simple binary image. Background pixels are denoted by *0* and object pixels are denoted by *X*. Image (b) shows the assigned labels after the first raster scan. In image (c), an *X* denotes those pixels at which an equivalence is noted during the first raster scan. Image (d) shows the final component labelled image.

Figure 6.5: The image of Figure 6.3 after connected components have been labelled.

contained in component i, and the value 0 for all other pixels:

$$\mathcal{I}_i(r,c) = \begin{cases} 1 & : \quad \text{pixel } r,c \text{ is contained in component } i \\ 0 & : \quad \text{otherwise} \end{cases} . \tag{6.51}$$

We will make use of the indicator function below, when we discuss computing statistics associated with the various objects in the image.

6.5 Position and Orientation

The ultimate goal of a robotic system is to manipulate objects in the world. In order to achieve this, it is necessary to know the positions and orientations of the objects that are to be manipulated. In this section, we address the problem of determining the position and orientation of objects *in the image*. If the camera has been calibrated, it is then possible to use these image position and orientations to infer the 3D positions and orientations of the objects. In general, this problem of inferring the 3D position and orientation from image measurements can be a difficult problem; however, for many cases that are faced by industrial robots we an obtain adequate solutions. For example, when grasping parts from a conveyor belt, the depth, z, is fixed, and the perspective projection equations can be inverted if z is known.

We begin the section with a general discussion of moments, since moments will be used in the computation of both position and orientation of objects in the image.

6.5.1 Moments

Moments are functions defined on the image that can be used to summarize various aspects of the shape and size of objects in the image. The i,j moment for the k^{th} object, denoted by $m_{ij}(k)$, is defined by

$$m_{ij}(k) = \sum_{r,c} r^i c^j \mathcal{I}_k(r,c). \tag{6.52}$$

From this definition, it is evident that m_{00} is merely number of pixels in the object. The order of a moment is defined to be the sum $i+j$. The first order moments are of particular interest when computing the centroid of an object, and they are given by

$$m_{10}(k) = \sum_{r,c} r\mathcal{I}_k(r,c), \qquad m_{01}(k) = \sum_{r,c} c\mathcal{I}_k(r,c). \tag{6.53}$$

It is often useful to compute moments with respect to the object center of mass. By doing so, we obtain characteristics that are invariant with respect to translation of the object. These moments are called *central moments*. The i,j central moment for the k^{th} object is defined by

$$C_{ij}(k) = \sum_{r,c} (r-\bar{r})^i (c-\bar{c})^j \mathcal{I}_k(r,c), \tag{6.54}$$

in which $(\bar{r},\bar{c})$ are the coordinates for the center of mass, or centroid, of the object.

6.5.2 The Centroid of an Object

It is convenient to define the position of an object to be the object's center of mass or centroid. By definition, the center of mass of an object is that point $(\bar{r},\bar{c})$ such that, if all of the object's mass were concentrated at $(\bar{r},\bar{c})$ the first moments would not change. Thus, we have

$$\sum_{r,c} \bar{r}_i \mathcal{I}_i(r,c) = \sum_{r,c} r\mathcal{I}_i(r,c) \qquad \Rightarrow \qquad \bar{r}_i = \frac{\sum_{r,c} r\mathcal{I}_i(r,c)}{\sum_{r,c} \mathcal{I}_i(r,c)} = \frac{m_{10}(i)}{m_{00}(i)} \tag{6.55}$$

$$\sum_{r,c} \bar{c}_i \mathcal{I}_i(r,c) = \sum_{r,c} c\mathcal{I}_i(r,c) \qquad \Rightarrow \qquad \bar{c}_i = \frac{\sum_{r,c} c\mathcal{I}_i(r,c)}{\sum_{r,c} \mathcal{I}_i(r,c)} = \frac{m_{01}(i)}{m_{00}(i)}. \tag{6.56}$$

Figure 6.6(a) shows the centroids for the connected components of the image of Figure 6.3.

6.5.3 The Orientation of an Object

We will define the orientation of an object in the image to be the orientation of an axis that passes through the object such that the second moment of the object about that axis is minimal. This axis is merely the two-dimensional equivalent of the axis of least inertia.

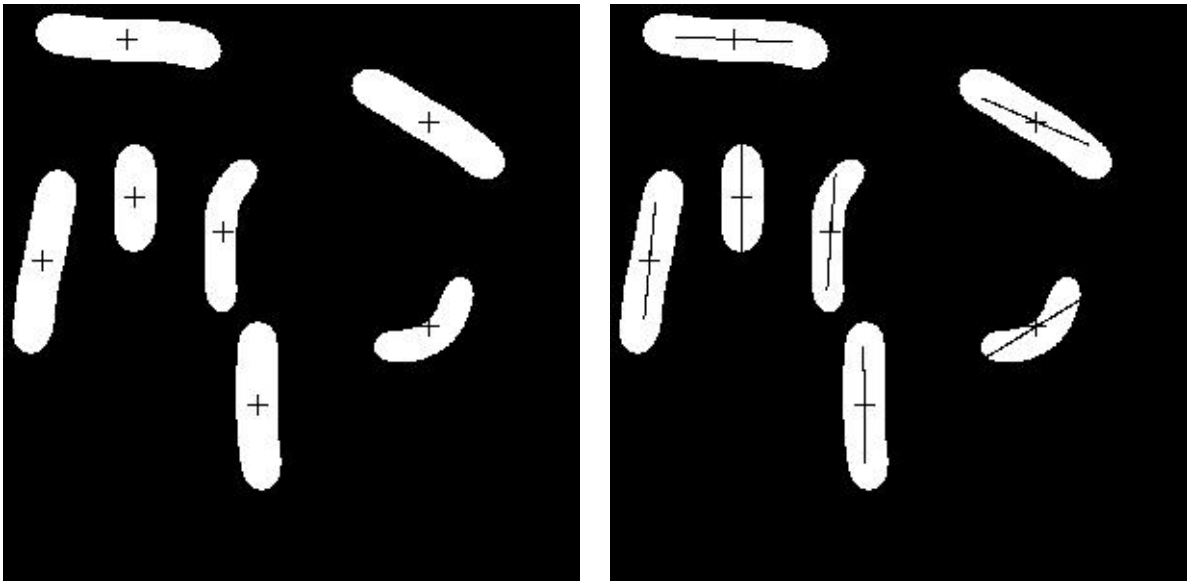
For a given line in the image, the second moment of the object about that line is given by

$$\mathcal{L} = \sum_{r,c} d^2(r,c)\mathcal{I}(r,c) \tag{6.57}$$

in which $d(r,c)$ is the minimum distance from the pixel with coordinates (r,c) to the line. Our task is to minimize $\mathcal{L}$ with respect to all possible lines in the image plane. To do this, we will use the ρ,θ parameterization of lines, and compute the partial derivatives of $\mathcal{L}$ with respect to ρ and θ. We find the minimum by setting these partial derivatives to zero.

With the ρ,θ parameterization, a line consists of all those points x,y that satisfy

$$x\cos\theta + y\sin\theta - \rho = 0. \tag{6.58}$$



(a) (b)

Figure 6.6: (a) The image of figure 6.3 showing the centroids of each component. (b) The image of figure 6.3 showing the orientation of each component.

Thus, $(\cos\theta, \sin\theta)$ gives the unit normal to the line, and ρ gives the perpendicular distance to the line from the origin. Under this parameterization, the distance from the line to the point with coordinates (r, c) is given by

$$d(r, c) = r\cos\theta + c\sin\theta - \rho. \tag{6.59}$$

Thus, our task is to find

$$\mathcal{L}^\star = \min_{\rho,\theta} \sum_{r,c} (r\cos\theta + c\sin\theta - \rho)^2 \mathcal{I}(r, c) \tag{6.60}$$

We compute the partial derivative with respect to ρ as

$$\begin{aligned}
\frac{d}{d\rho}\mathcal{L} &= \frac{d}{d\rho}\sum_{r,c}(r\cos\theta + c\sin\theta - \rho)^2\mathcal{I}(r, c) && (6.61)\\
&= -2\sum_{r,c}(r\cos\theta + c\sin\theta - \rho)\mathcal{I}(r, c) && (6.62)\\
&= -2\cos\theta\sum_{r,c} r\mathcal{I}(r, c) - 2\sin\theta\sum_{r,c} c\mathcal{I}(r, c) + 2\rho\sum_{r,c}\mathcal{I}(r, c) && (6.63)\\
&= -2(\cos\theta m_{10} + \sin\theta m_{01} - \rho m_{00}) && (6.64)\\
&= -2(m_{00}\bar{r}\cos\theta + m_{00}\bar{c}\sin\theta - \rho m_{00}) && (6.65)\\
&= -2m_{00}(\bar{r}\cos\theta + \bar{c}\sin\theta - \rho). && (6.66)
\end{aligned}$$

Now, setting this to zero we obtain

$$\bar{r}\cos\theta + \bar{c}\sin\theta - \rho = 0. \tag{6.67}$$

But this is just the equation of a line that passes through the point $(\bar{r}, \bar{c})$, and therefore we conclude that the inertia is minimized by a line that passes through the center of mass. We can use this

knowledge to simplify the remaining computations. In particular, define the new coordinates (r', c') as

$$r' = r - \bar{r}, \qquad c' = c - \bar{c}. \tag{6.68}$$

The line that minimizes $\mathcal{L}$ passes through the point $r' = 0, c' = 0$, and therefore its equation can be written as

$$r' \cos\theta + c' \sin\theta = 0. \tag{6.69}$$

Before computing the partial derivative of $\mathcal{L}$ (expressed in the new coordinate system) with respect to θ, it is useful to perform some simplifications.

$$\begin{aligned}
\mathcal{L} &= \sum_{r,c} (r' \cos\theta + c' \sin\theta)^2 \mathcal{I}(r,c) && (6.70)\\
&= \cos^2\theta \sum_{r,c} (r')^2 \mathcal{I}(r,c) + 2\cos\theta\sin\theta \sum_{r,c} (r'c')\mathcal{I}(r,c) + \sin^2\theta \sum_{r,c} (c')^2 \mathcal{I}(r,c) && (6.71)\\
&= C_{20}\cos^2\theta + 2C_{11}\cos\theta\sin\theta + C_{02}\sin^2\theta && (6.72)
\end{aligned}$$

in which the C_{ij} are the central moments given in (6.54). Note that the central moments depend on neither ρ nor θ.

The final set of simplifications that we will make all rely on the double angle identities:

$$\begin{aligned}
cos^2\theta &= \frac{1}{2} + \frac{1}{2}\cos 2\theta && (6.73)\\
sin^2\theta &= \frac{1}{2} - \frac{1}{2}\cos 2\theta && (6.74)\\
cos\theta \sin\theta &= \frac{1}{2}\sin 2\theta. && (6.75)
\end{aligned}$$

Substituting these into our expression for $\mathcal{L}$ we obtain

$$\begin{aligned}
\mathcal{L} &= C_{20}(\frac{1}{2} + \frac{1}{2}\cos 2\theta) + 2C_{11}(\frac{1}{2}\sin 2\theta) + C_{02}(\frac{1}{2} - \frac{1}{2}\cos 2\theta) && (6.76)\\
&= \frac{1}{2}(C_{20} + C_{02}) + \frac{1}{2}(C_{20} - C_{02})\cos 2\theta + \frac{1}{2}C_{11}\sin 2\theta && (6.77)
\end{aligned}$$

It is now easy to compute the partial derivative with respect to θ:

$$\begin{aligned}
\frac{d}{d\theta}\mathcal{L} &= \frac{d}{d\theta}\frac{1}{2}(C_{20} + C_{02}) + \frac{1}{2}(C_{20} - C_{02})\cos 2\theta + \frac{1}{2}C_{11}\sin 2\theta && (6.78)\\
&= -(C_{20} - C_{02})\sin 2\theta + C_{11}\cos 2\theta, && (6.79)
\end{aligned}$$

and we setting this to zero we obtain

$$\tan 2\theta = \frac{C_{11}}{C_{20} - C_{02}}. \tag{6.80}$$

Figure 6.6(b) shows the orientations for the connected components of the image of Figure 6.3.

6.6 Problems

1. For a camera with $\lambda = 10$mm, find the image plane coordinates for the 3D points whose coordinates with respect to the camera frame are given below. Indicate if any of these points will not be visible to a physical camera.

(a) $(25, 25, 50)$

(b) $(-25, -25, 50)$

(c) $(20, 5, -50)$

(d) $(15, 10, 25)$

(e) $(0, 0, 50)$

(f) $(0, 0, 100)$

2. Repeat problem 1 for the case when the coordinates of the points are given with respect to the world frame. Suppose that the optic axis of the camera is aligned with the world x-axis, that the camera x-axis is parallel to the world y-axis, and that the center of projection has coordinates (0,0,100).

3. Consider two cameras with coordinate frames $O_1x_1y_1z_1$ and $O_2x_2y_2z_2$ such that

$$\mathbf{H}_2^1 = \begin{bmatrix} 1 & 0 & 0 & B \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Here, B is called the baseline distance between the two cameras. Suppose that a 3D point P projects onto these two images with image plane coordinates (u_1, v_1) in the first camera and (u_2, v_2) in the second camera. Determine the depth of the point P.

4. Show that the projection of a 3D line is a line in the image.

5. Show that

$$\sum_{i=1}^{N}(X - \mu)^2 P(X) = \left[\sum_{i=1}^{N} X^2 P(X) \right] - \mu^2,$$

i.e., show that the variance of X is equal to the difference between the expected value of X^2 and the square of the mean.

6. Suppose that an image consists of a light object on a dark background. Further, suppose that the image is hand segmented, giving histograms for both the object and background. Thus, it is a simple matter to compute $P_0(z)$ (the probability that a pixel with intensity value z belongs to the background) and $P_1(z)$ (the probability that a pixel with intensity value z belongs to the object). Give an expression for the probability that a pixel will be misclassified if the threshold value of t is selected.

7. Suppose again that an image consists of a light object on a dark background, and that the image has been hand segmented, giving $P_0(z)$ and $P_1(z)$. Give an algorithm that determines t^*, the optimal threshold value (i.e., the threshold value that minimizes the probability of misclassification of an image pixel). Your algorithm should employ a recursive formulation whenever possible.