Chapter 2

Machine Vision Background

2.1 Introduction

The goal of a machine vision system is to create a model of the real world from images. *A machine vision system recovers useful information about a scene from its two-dimensional projections* [2]. Since images are two-dimensional projections of the three-dimensional world, the information is not directly available and must be recovered. This recovery requires the inversion of a many-to-one mapping. To recover the information, knowledge about the objects in the scene and projection geometry is required.

2.2 Relationships to Other Fields

Many fields are related to machine vision [42]. As we will see, techniques developed from many areas are used for recovering information from images. In this section, we briefly describe some very closely related fields. No effort is made to relate machine vision (also called computer vision) to other fields exhaustively.

Image processing. Image processing techniques usually transform images into other images; the task of information recovery is left to a human user. This field includes topics such as image enhancement, image compression, and correcting blurred or out-of-focus images. On the other hand, machine vision algorithms take images as inputs but produce other types of outputs, such as representations for the object contours in an image. Thus, emphasis in machine vision is on recovering information automatically, with minimal interaction with a human. Image processing algorithms are useful in early stages of a machine vision system. They are usually used to enhance particular information and suppress noise.

Computer graphics generates images from geometric primitives such as lines, circles, and free-form surfaces. Computer graphics techniques play a significant role in visualization and virtual reality. Machine vision is the inverse problem: estimating the geometric primitives and other features from the image. Thus, computer graphics is the synthesis of images, and machine vision is the analysis of images. In the early days of these two fields, there was not much relationship between them, but in the last few years these two fields have been growing closer. Machine vision is using curve and surface representations and several other techniques from computer graphics, and computer graphics is using many techniques from machine vision to enter models into the computer for creating realistic images. Visualization and virtual reality are bringing these two fields closer.

Pattern recognition classifies numerical and symbolic data. Many statistical and syntactical techniques have been developed for classification of patterns. Techniques from pattern recognition play an important role in machine vision for recognizing objects. In fact, many industrial applications rely heavily on pattern recognition. Object recognition in machine vision usually requires many other techniques. We will discuss some aspects of statistical pattern recognition briefly in the follows chapters.

Artificial intelligence is concerned with designing systems that are intelligent and with studying computational aspects of intelligence. Artificial intelligence is used to analyze scenes by computing a symbolic representation of the scene contents after the images have been processed to obtain features. Artificial intelligence may be viewed as having three stages: perception, cognition, and action. Perception translates signals from the world into symbols, cognition manipulates symbols, and action translates symbols into signals that effect changes in the world. Many techniques from artificial intelligence play important roles in all aspects of computer vision. In fact, computer vision is often considered a subfield of artificial intelligence.

Design and analysis of *neural networks* has become a very active field in the last decade [38, 39]. Neural networks are being increasingly applied to solve some machine vision problems. Since this field is in its infancy, there are no established techniques for machine vision yet.

Psychophysics, along with cognitive science, has studied human vision for a long time. Many techniques in machine vision are related to what is known about human vision. Many researchers in computer vision are more interested in preparing computational models of human vision than in designing machine vision systems. Our emphasis in this thesis will be on designing machine vision systems; we will not make any effort to relate the techniques discussed here to those in psychophysics.

Machine vision produces measurements or abstractions from geometrical properties. It may be useful to remember the equation

$$Vision = Geometry + Measurement + Interpretation \tag{2.1}$$

2.3 Role of Knowledge

Decision making always requires knowledge of the application or goal. As we will see, at every stage in machine vision decisions must be made by the system. Emphasis in machine vision systems is on maximizing automatic operation at each stage, and these systems should use knowledge to accomplish this. The knowledge used by the system includes models of features, image formation, models of objects, and relationships among objects. Without explicit use of knowledge, machine vision systems can be designed to work only in a very constrained environment for very limited applications [49]. To provide more flexibility and robustness, knowledge is represented explicitly and used by the system. One objective in this thesis is to point out the types of knowledge used by machine vision systems at different stages, that should be considered to make the system more adaptive and robust. We will see that knowledge is used by designers of systems in many implicit as well as explicit forms. The efficacy and efficiency of a system is usually governed by the quality of the knowledge used by the system. Difficult problems are often solvable only by identifying the proper source of knowledge and appropriate mechanisms to use it in the system.

2.4 Image Geometry

There are two parts to the image formation process [19]:

1. The geometry of image formation, which determines where in the image plane the projection of a point in the scene will be located.

2. The physics of light, which determines the brightness of a point in the image plane as a function of scene illumination and surface properties.

This section introduces the first of these two parts, the geometry of image formation. Because an understanding of the physics of light is not necessary for understanding the fundamentals of most vision algorithms, optics will be not covered.

The basic model for the projection of points in the scene onto the image plane is diagrammed in Figure 2.1. In this model, the imaging system's center of projection coincides with the origin of the three-dimensional coordinate system. The coordinate system for points in the scene is the three-dimensional space spanned by the unit vectors **x**, **y**, and **z** that form the axes of the coordinate system. A point in the scene has coordinates (**x**, **y**, **z**). The x coordinate is the horizontal position of the point in space as seen from the camera, the y coordinate is the vertical position of the point in space as seen from the camera, and the z coordinate is the distance from the camera to the point in space along a line parallel to the z axis. *The line of sight* of a point in the scene is the line that passes through the point of interest and the center of projection. The line drawn in Figure 2.1 is a line of sight.

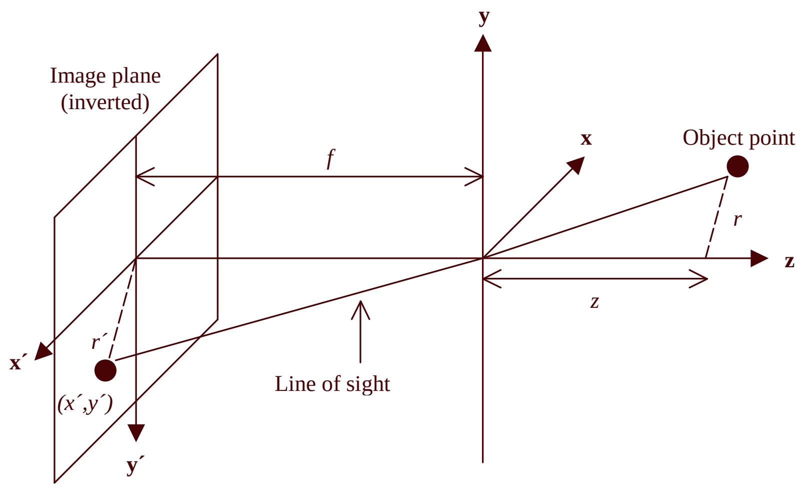


Figure 2.1: The point on the image plane that corresponds to a particular point in the scene is found by following the line that passes through the scene point and the center of projection.

The image plane is parallel to the **x** and **y** axes of the coordinate system at a distance f from the center of projection, as shown in Figure 2.1. Note that the image plane in an actual camera is at a distance of f behind the center of projection (as shown in Figure 2.1) and the projected image is inverted. It is customary to avoid this inversion by assuming that the image plane is in front of the center of projection as shown in Figure 2.2. The image plane is spanned by the vectors **x′** and **y′** to form a two-dimensional coordinate system for specifying the position of points in the image plane. The position of a point in the image plane is specified by the two coordinates x' and y'. The point (0,0) in the image plane is the origin of the image plane. The position in the image plane of a point in the scene is found by intersecting the line of sight with the image plane according to the projection scheme as described in the following sections.

2.4.1 Perspective Projection

The position (x', y') in the image plane of a point at position (x, y, z) in the scene is found by computing the coordinates (x', y') of the intersection of the line of sight passing through the scene point (x, y, z) with the image plane as shown in Figure 2.2.

The distance of the point (x, y, z) from the z axis is $r = \sqrt{x^2 + y^2}$, and the distance of the projected point (x', y') from the origin of the image plane is $r' = \sqrt{x'^2 + y'^2}$. The z axis, the line of sight to point (x,y,z), and the line segment of length r from point (x, y, z) to the z axis (perpendicular to the z axis)

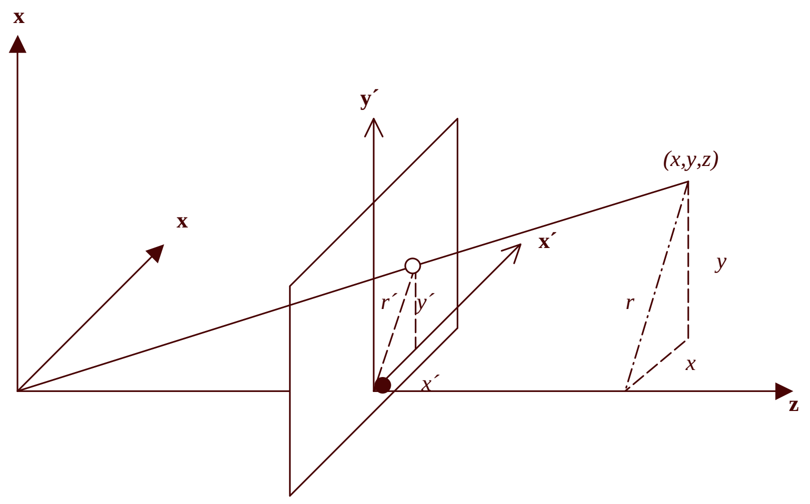


Figure 2.2: An illustration showing the line of sight that is used to calculate the projected point (x',y') from the object point (x,y,z).

form a triangle. The z axis, the line of sight to point (x', y') in the image plane, and the line segment of length r′ from point (x', y') to the z axis (perpendicular to the z axis) form another triangle. The two triangles are similar, so the ratios of the corresponding sides of the triangles must be the same:

$$\frac{f}{z} = \frac{r'}{r} \tag{2.2}$$

The triangle formed from the x and y coordinates and the perpendicular distance r and the triangle formed from the image plane coordinates x', y' and the perpendicular distance r' are also similar triangles:

$$\frac{x'}{x} = \frac{y'}{y} = \frac{r'}{r} \tag{2.3}$$

Combining Equations 2.2 and 2.3 yields the equations for perspective projection:

$$\frac{x'}{x} = \frac{f}{z}$$

$$\frac{y'}{y} = \frac{f}{z} \tag{2.4}$$

The position of a point (x, y, z) in the ′mage plane is given by the equations

$$x' = \frac{f}{z}x \tag{2.5}$$

$$y' = \frac{f}{z} y \tag{2.6}$$

2.4.2 Coordinate Systems

In this presentation, we have assumed that the center of projection coincides with the origin of the three-dimensional space and that the camera axes are aligned with the coordinate system used to specify the location of a point in the scene. In general, the camera is displaced and rotated with respect to the three-dimensional coordinate system used for specifying the coordinates of points in the scene. The coordinates (x_a, y_a, z_a) in the absolute coordinate system must be transformed into the coordinates (x_c, y_c, z_c) of the point in the camera coordinate system before projecting the points onto the image plane. The general case is presented in Chapter 10 on calibration, which covers the mathematics of transforming coordinates between coordinate systems. Absolute coordinates are also called world coordinates.

Individual objects may have their own coordinate system, called model coordinates. The scene consists of object models that have been placed (rotated and translated) into the scene, yielding object coordinates in the coordinate system of the scene (absolute coordinates). The scene coordinates in the absolute coordinate system are transformed to camera coordinates before projection onto the image plane.

Throughout the thesis, subscripts are used to denote the coordinate system, and primes are used to denote the image plane coordinates of a point after projection onto the image plane. Subscripts are omitted when the coordinate system is clear, and the primes are omitted when it is clear that coordinates are in the image plane. These conventions allow us to be precise about the coordinate system when necessary and to distinguish between a point in camera coordinates and its projection onto the image plane, but they allow the notation to be simplified to the common usage for point coordinates in analytic geometry when we are presenting the equations for curves and surfaces.

In keeping with the convention, we will omit primes from image plane coordinates for the rest of this chapter.

2.5 Sampling and Quantization

A continuous function cannot be represented exactly in a digital computer. The interface between the optical system that projects a scene onto the image plane and the computer must sample the image at a finite number of points and represent each sample within the finite word size of the computer. This is sampling and quantization. Each image sample is called a pixel.

Each pixel is represented in the computer as a small integer. Frequently, the pixel is represented as an unsigned 8-bit integer in the range [0,255], with

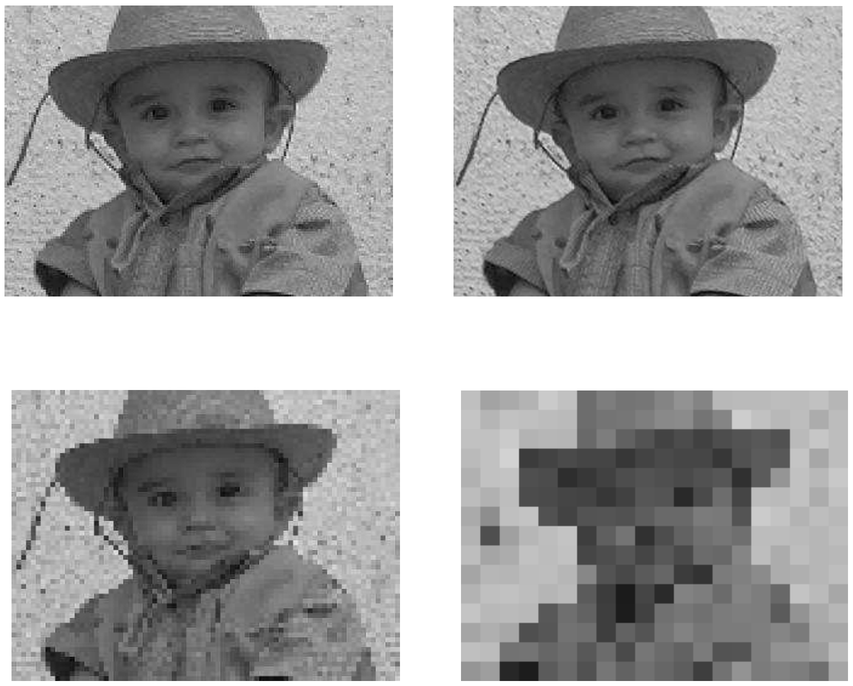


Figure 2.3: An image shown at many different spatial resolution. *Top left:* Original image sampled at 320 x 240 and 256 gray levels. *Top right:* 160 x 120. *Bottom left:* 80 x 60. *Bottom right:* 20 x 15.

0 corresponding to black, 255 corresponding to white, and shades of gray distributed over the middle values.

Many cameras acquire an analog image, which is then sampled and quantized to convert it to a digital image. The sampling rate determines how many pixels the digital image will have (the image resolution), and quantization determines how many intensity levels will be used to represent the intensity value at each sample point. As shown in Figures 2.3 and 2.4, an image looks very different at different sampling rates and quantization levels. In most machine vision applications, the sampling and quantizing rates are predetermined due to the limited choice of available cameras and image acquisition hardware; but in many applications it may be important to know the effects of sampling and quantizing. The image processing book [30] discusses the factors that should be considered in selecting appropriate sampling and quantization rates to retain the important information in images.

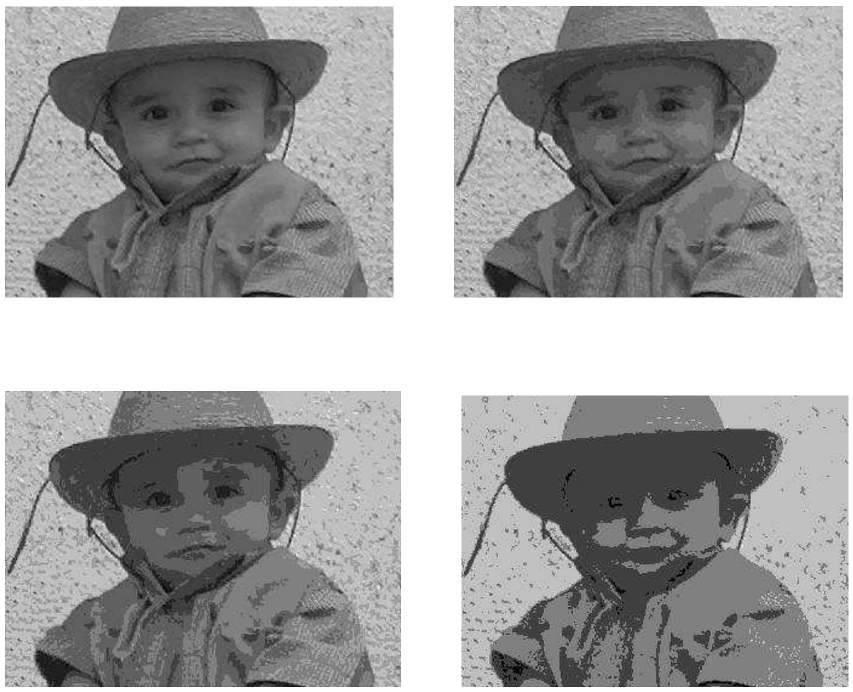


Figure 2.4: An image shown at many different gray level resolutions. *Top left:* Original image sampled at 256 gray levels. *Top right:* 64 gray levels. *Bottom left:* 32 gray levels. *Bottom right:* 8 gray levels.

2.6 Image Definitions

It is important to understand the relationship between the geometry of image formation, described in previous sections, and the representation for images in the computer. There must be a bridge from the mathematical notation used to develop machine vision algorithms to the algorithmic notation used in programs [56].

A pixel is a sample of the image intensity quantized to an integer value. An image is a two-dimensional array of pixels. The row and column indices [*i*, *j*] of a pixel are integer values that specify the row and column in the array of pixel values. Pixel [0,0] is located at the top left corner of the image. The index *i* points down, and j points to the right. This index notation corresponds closely to the array syntax used in computer programs. The positions of points in the image plane have x and y coordinates. The y coordinate corresponds to the vertical direction, and the *x* coordinate corresponds to the horizontal direction. The *y* axis points up, and the *x* axis points to the right. Note that the directions corresponding to the two indices *i* and *j* in the pixel index [*i*,*j*] are the reverse of the directions corresponding to the respective coordinates in the position (*x*,*y*).

The x and y coordinates are real numbers, stored as floating-point numbers in the computer. Image plane coordinates (*x*, y) can be computed from pixel coordinates [*i*,*j*] of an *n* by *m* pixel array using the formulas

$$x = j - \frac{m-1}{2} \tag{2.7}$$

$$y = -\left(i - \frac{n-1}{2}\right) \tag{2.8}$$

which assume that the origin of the image plane coordinate system corresponds to the center of the image array.

In an imaging system, each pixel occupies some finite area on the image plane. Machine vision algorithms that depend on the exact shape of the pixel footprint will not be covered in this thesis, so we may assume for concreteness that the pixels partition the image plane into equal-sized squares. Positions in the image plane can be represented to fractions of a pixel. The coordinates (x_{ij}, y_{ij}) of the pixel with indices [*i*, *j*] are the location of the center of the pixel in the coordinate system of the image plane. Since we are concerned only with the location of the center of the pixel in the image plane (the location at which the image sample was taken), the pixel may be further abstracted to a point in the image plane. The array of pixels in the computer program corresponds to the grid of image plane locations at which the samples were obtained, as illustrated in Figure 2.5.

In diagramming images, we may show the image as a square tessellation of a rectangular region, with each square shaded to indicate the image intensity of that pixel.

This is purely a technique for visualization and does not imply that the shape of the pixel matters to the algorithm being discussed. For the purposes of nearly all of the algorithms discussed in this thesis, the image can be modeled as a square grid of samples of the image intensity, represented in the computer as an array of pixel values. The camera and digitizing electronics are designed to ensure that this assumption is satisfied. Some variations, such as different spacing between the rows and columns in the grid, distortions due to lens imperfections, and errors in the construction of the camera, can be removed through calibration without changing the algorithms that process the image.

To summarize, a pixel is both a gray value, which is a quantized sample of the continuous image intensity, and an image location, specified as the row and column indices in the image array. The image array is obtained by sampling the image intensity at points on a rectangular grid. Points in the image plane, specified with coordinates x and y, may lie between the grid locations at which pixels were sampled.

2.7 Levels of Computation

An image usually contains several objects. A vision application usually involves computing certain properties of an object, not the image as a whole. To compute properties of an object, individual objects must first be identified as separate objects; then object properties can be computed by applying calculations to the separate objects [59].

Definitions and algorithms for connectivity and segmentation that will allow different objects to be represented as distinct subimages will be presented in the next chapters. For now, consider computer vision algorithms from the viewpoint of locality of computation. Consider each algorithm in terms of its input-output characteristics. Here our aim is to characterize operations so that we can discuss the nature of input and output and how best to implement these operations. Note that the input to a computer vision system is an image, and the output, unlike that of image processing systems, is some Symbolic quantity denoting identity or location of an object, for instance. The amount of data processed by a vision system is very large, and that makes the computational requirements of a computer vision system very demanding. The last few years have witnessed many special architectures designed for computer vision. Since we want to discuss characteristics of operations to predict their computational requirements, we classify the levels of operations and study their general characteristics.

2.7.1 Point Level

Some operations produce an output based on only a point in an image. Thresholding is an example. A thresholding algorithm produces output values that depend only on the input value, for a preset threshold. Thus,

$$f_B[i,j] = O_{\text{point}}\{f_A[i,j]\} \tag{2.9}$$

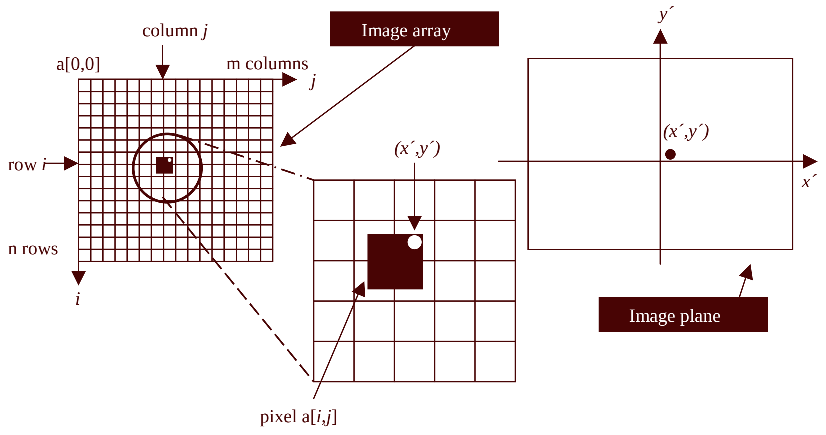


Figure 2.5: Relationship between image plane coordinates and image array indices. Note that the location of the origin of the *x*-*y* plane is arbitrary with respect to the image array.

where f_A and f_B are the input and output images, respectively. This operation can be efficiently implemented using a lookup table (see Figure 2.6).

2.7.2 Local Level

A local operation produces an output image in which the intensity at a point depends on the neighborhood of the corresponding point in the input image. Thus,

$$f_B[i,j] = O_{\text{local}}\{f_A[i_k,j_l]; [i_k,j_l] \in N[i,j]\} \tag{2.10}$$

An example of such an operation is shown in Figure 2.7. Smoothing and edge detection are local operations. Since these operations require values from a neighborhood in the input image, array processors or Single Instruction, Multiple Data (SIMD) machines may be suitable for implementing these operations. In general, these operations can be easily implemented on parallel machines and can often be performed in real time.

2.7.3 Global Level

The output of certain operators depends on the whole picture. Such operations are called global operations:

$$P = O_{global}\{f[i,j]\} \tag{2.11}$$

This operation is shown in Figure 2.8.

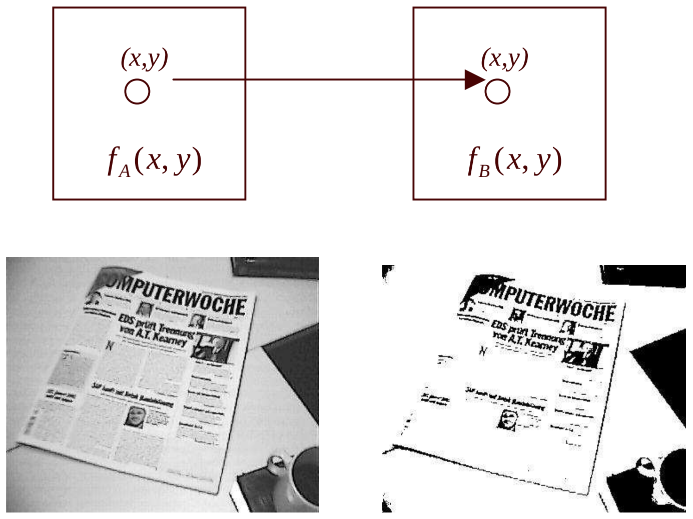


Figure 2.6: *Top:* Point operations are applied to individual image pixels and produce an output image as the result. *Bottom left:* Original image. *Bottom right:* Thresholded image where pixels from the original image with a gray level value greater than 128 are set to white, while the remaining pixels are set to black.

The output of these operators may be an image or it may be symbolic output. A histogram of intensity values and the Fourier transform are global operations. Global operations are responsible for the slowness of vision systems. We will see that most operations at higher levels are global in nature and pose the biggest challenge to designers of algorithms and architectures.

2.7.4 Object Level

Most applications of computer vision require properties to be computed at the object level. Size, average intensity, shape, and other characteristics of an object must be computed for the system to recognize it. Many other characteristics of an object must be determined for defect detection. Operators restricted to those pixels belonging to an object are occasionally applied to determine these properties. This leads to very difficult questions: What is an object? How do we find objects?.

We will see that an object is defined in a particular context. In fact, many operations in machine vision are performed to find where a particular object is located in an image. We must use all points that belong to an object to compute some of its characteristics, but we must use those characteristics to identify those points. We will see that significant efforts are spent to solve the figure-ground problem (separation of foreground pixels from background pixels) to

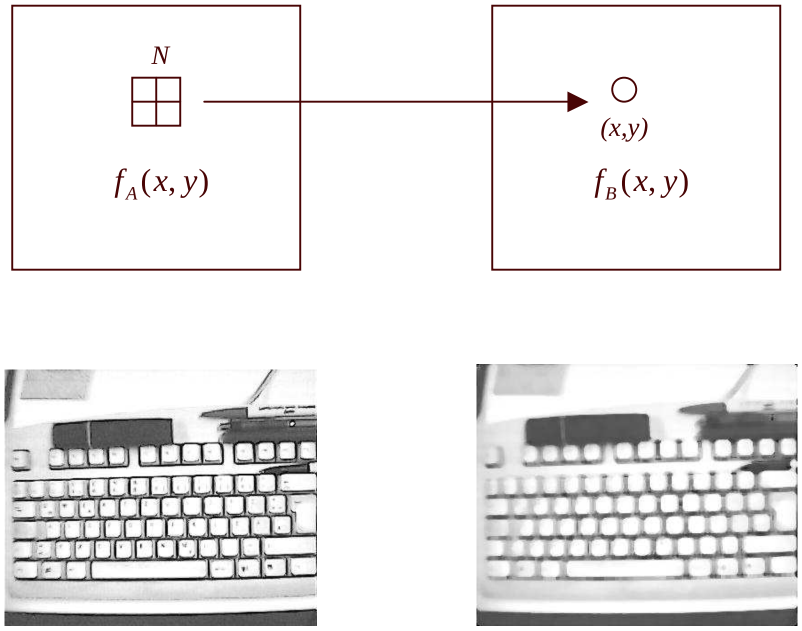
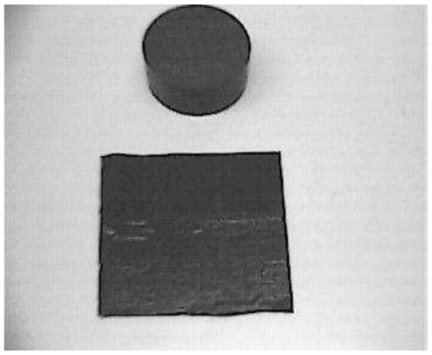


Figure 2.7: *Top:* Local operations are applied to pixel neighborhoods and produce an output image as the result. *Bottom left:* Original image. *Bottom right:* Smoothed image where each pixel value is the average gray value calculated from its 5 x 5 local neighborhood in the original image.



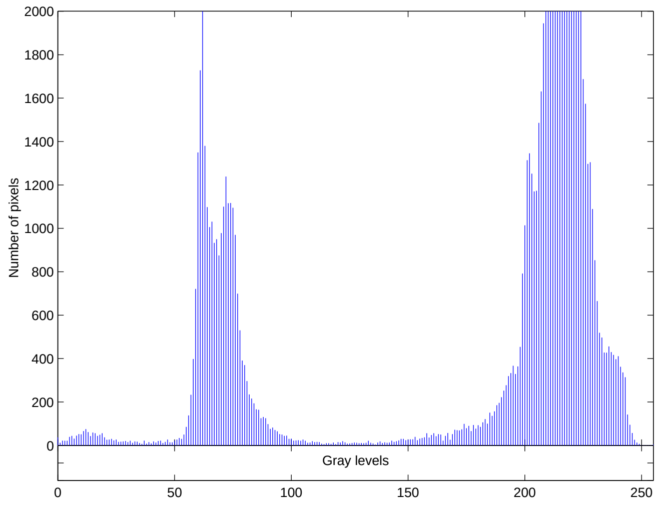


Figure 2.8: An example of a global operation: an image (Top) and its histogram (Bottom). A histogram is a plot of the number of pixels at each gray value contained in the image.

group points into objects.

At this point, we should just remember that to understand the contents of an image, a machine vision system must perform several operations at the object level.

2.8 Binary Image Processing

An image contains a continuum of intensity values before it is quantized to obtain a digital image. The information in an image is in these gray values. To interpret an image, the variations in the intensity values must be analyzed. The most commonly used number of quantization levels for representing image intensities is 256 different gray levels. It is not uncommon, however, to see digital images quantized to 32, 64, 128, or 512 intensity levels for certain applications, and even up to 4096 (12 bits) are used in medicine. Clearly, more intensity levels allow better representation of the scene at the cost of more storage.

In the early days of machine vision, the memory and computing power available was very limited and expensive. These limitations encouraged designers of vision applications to focus their efforts on binary vision systems. A binary image contains only two gray levels. The difference this makes in the representation of a scene is shown in Figure 2.9.

In addition designers noted that people have no difficulty in understanding line drawings, silhouettes, and other images formed using only two gray levels. Encouraged by this human capability, they used binary images in many applications.

Even though computers have become much more powerful, binary vision systems are still useful. First of all, the algorithms for computing properties of binary images are well understood. They also tend to be less expensive and faster than vision systems that operate on gray level or color images. This is due to the significantly smaller memory and processing requirements of binary vision. The memory requirements of a gray level system working with 256 gray levels will be eight times that of a system working with a binary image of the same size. The storage size may be further reduced by using techniques such as run-length encoding. The processing time requirements are lower because many operations on binary images may be performed as logical operations instead of integer arithmetic operations.

Smaller memory requirements and faster execution times are not the only reasons for studying binary vision systems. Many techniques developed for these systems are also applicable to vision systems which use gray scale images. A convenient way to represent an object in a gray level or color image is to use its mask. The mask of an object is a binary picture in which the object points are 1 and other points are 0. After an object has been separated from the background, its geometric and topological properties may be required in decision making. These properties can be computed from its binary image. All the techniques discussed in this chapter can be applied to a region in a gray image. Thus, though we will discuss these techniques in the context of binary

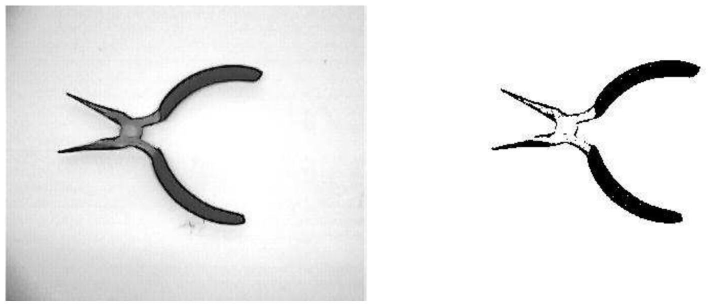


Figure 2.9: A gray level image and its corresponding binary image.

images, their application is not limited to binary images.

In general, binary vision systems are useful in cases where a silhouette contains enough information to allow recognition of an object and where the environment can be adequately controlled. To obtain a good silhouette, the objects must be easily separated from the background. This can be achieved by using special illumination techniques and by having only a few objects in the scene. There are many industrial situations that fulfill these requirements. For example, binary vision systems have found application in optical character recognition, chromosome analysis, and recognition of industrial parts. In these cases, the binary vision system usually uses a threshold to separate objects from the background. The proper value of this threshold depends on illumination and on reflectance characteristics of objects. The resulting binary picture allows computation of geometric and topological properties (*features*) of objects for the given task. In many applications, these characteristics are enough for recognition of objects.

It should be mentioned here, however, that with the increase in the complexity of applications, more and more vision systems are using gray scale images. This is due to the fact that in many material handling and assembly tasks, the illumination cannot be controlled to obtain good contrast between objects and background. Care has to be exercised to make the system insensitive to small changes in illumination and reflectance characteristics of other objects in a scene. In many applications, this becomes a formidable task. Similarly, in inspection tasks, it may not be possible to recover subtle information using only two intensity levels. Internal details of an object may be lost in thresholding and may make the task of detecting surface defects very difficult.

Certain generally used conventions concerning binary images will be followed in this chapter. Object pixels will have the value 1 and background pixels will have 0. In displaying pictures, 0 is white and 1 is black; thus, in binary images, the background is white and objects are black. We will also assume that pictures are of size n x m pixels and are represented in a computer as a two-dimensional array. This representation allows us to visualize images with the spatial relationships between points maintained in the form familiar to people.

The techniques discussed in this chapter, though simple, have played a very important role in robotic vision. We will study the following aspects of binary vision systems in this chapter:

- Formation of binary images
- Geometric properties
- Topological properties
- Object recognition in binary images.

Many concepts discussed here are used in all aspects of machine vision. Many definitions are related to digital geometry and are useful in discussions related to sampled images. In general, after an image has been segmented into several objects, each object is represented as a region. Discussions related to these object regions use the terminology and concepts discussed in this chapter.

2.9 Thresholding

One of the most important problems in a vision system is to identify the subimages that represent objects. This operation, which is so natural and so easy for people, is surprisingly difficult for computers. The partitioning of an image into regions is called segmentation. Ideally, a partition represents an object or part of an object [22]. Formally, segmentation can be defined as a method to partition an image, F[i,j], into subimages, called regions, P_i,..., P_k, such that each subimage is an object candidate.

Definition 1.1 A region is a subset of an image.

Definition 1.2 Segmentation is grouping pixels into regions, such that

- $\cup_{i=1}^{k} P_i$= Entire image ($\{P_i\}$ is an exhaustive partitioning.)
- $P_i \cap P_j = \emptyset, i \neq j$ ($\{P_i\}$ is an exclusive partitioning.)
- Each region Pi satisfies a predicate; that is, all points of the partition have some common property.
- Pixels belonging to adjacent regions, when taken jointly, do not satisfy the predicate. As shown above, a partition satisfies a predicate.

This predicate may be as simple as has uniform intensity but is more complex in most applications. Segmentation is a very important step in understanding images.

A binary image is obtained using an appropriate segmentation of a gray scale image. If the intensity values of an object are in an interval and the intensity values of the background pixels are outside this interval, a binary image can be obtained using a thresholding operation that sets the points in that interval to 1 and points outside that range to 0. Thus, for binary vision, segmentation and thresholding are synonymous. Many cameras have been designed to perform this thresholding operation in hardware. The output of such a camera is a binary image. In most applications, however, cameras give a gray scale image and the binary image is obtained using thresholding.

Thresholding is a method to convert a gray scale image into a binary image so that objects of interest are separated from the background. For thresholding to be effective in *object-background separation*, it is necessary that the objects and background have sufficient contrast and that we know the intensity levels of either the objects or the background. In a fixed thresholding scheme, these intensity characteristics determine the value of the threshold.

Let us assume that a binary image *B*[*i, j*] is the same as a thresholded gray image F_T[*i, j*] which is obtained using a threshold *T* for the original gray image *F*[*i,j*]. Thus,

$$B[i,j] = F_T[i,j] \tag{2.12}$$

where for a darker object on a lighter background

$$F_T[i,j] = \begin{cases} 1 \text{ if } F[i,j] \leq T \\ 0 \text{ otherwise} \end{cases} \tag{2.13}$$

If it is known that the object intensity values are in a range [T_1, T_2], then we may use

$$F_T[i,j] = \begin{cases} 1 \text{ if } T_1 \leq F[i,j] \leq T_2 \\ 0 \text{ otherwise} \end{cases} \tag{2.14}$$

A general thresholding scheme in which the intensity levels for an object may come from several disjoint intervals may be represented as

$$F_T[i,j] = \begin{cases} 1 \text{ if } F[i,j] \in Z \\ 0 \text{ otherwise} \end{cases} \tag{2.15}$$

Note how knowledge about the application domain is incorporated into the thresholding algorithm. It has, in fact, been tailored for the domain; therefore, the same threshold values may not work in a new domain. The threshold is usually selected on the basis of experience with the application domain. In some cases, the first few runs of the system may be used for interactively analyzing a scene and determining an appropriate value for the threshold.

Automatic thresholding of images is often the first step in the analysis of images in machine vision systems. Many techniques have been developed for utilizing the intensity distribution in an image and the knowledge about the objects of interest for selecting a proper threshold value automatically. This was briefly introduced in Figure 2.8, where an image and its histogram are given.

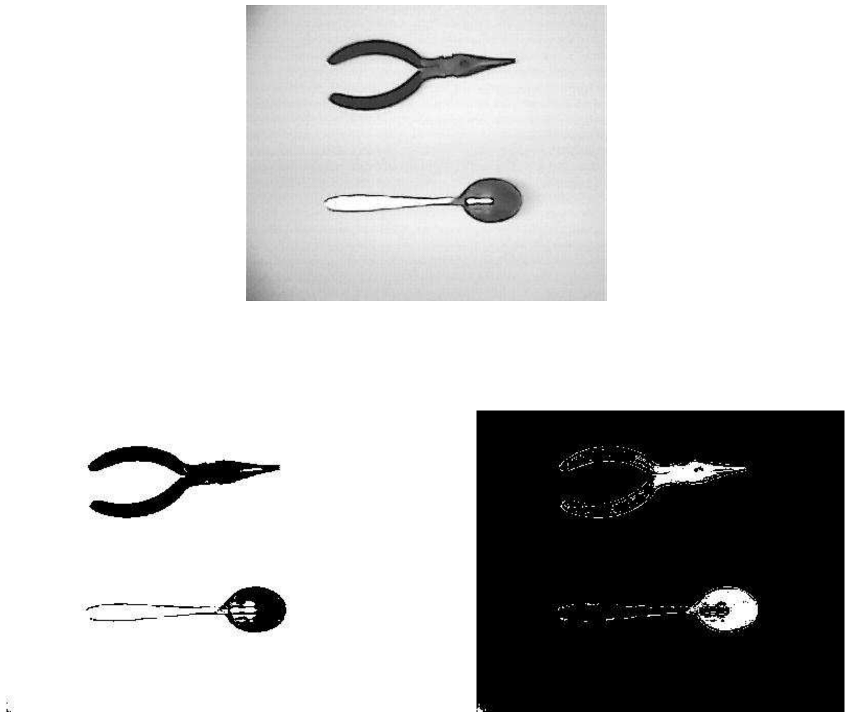


Figure 2.10: A gray level image and its resulting binary images using different thresholds. *Top:* Original gray-level image. *Left:* Image threshold with T=125. *Right*: T_1=70 and T_2=160.

2.10 Geometric Properties

Suppose that a thresholding scheme has given us objects in an image. The next step is to recognize and locate objects. In most robotic applications, the camera location and the environment are known [23]. Using simple geometry, one may find the three-dimensional locations of objects from their two-dimensional positions in images. Moreover, in most applications the number of different objects is not large. If the objects are different in size and shape, the size and shape features of objects may be determined from their images to help the system recognize them. Many applications in robotic have utilized some simple features of regions for determining the locations of objects and for recognizing them (e.g., size, position, orientation).

If there are several objects, one can compute these features for each object. An object is usually represented by a connected component or a region. The concept of connectedness and the algorithms for finding connected components in an image will be discussed later in this chapter. For the present discussion, let us assume that an image has only one object.

2.10.1 Size

In general, for a binary image it is well known that the area A is given by

$$A = \sum_{i=1}^{n} \sum_{j=1}^{m} B[i,j] \tag{2.16}$$

This is the zeroth-order moment.

2.10.2 Position

The position of an object in an image plays an important role in many applications. There are different ways to specify the position of an object, such as using its enclosing rectangle or centroid. In robotic applications, objects usually appear on a known surface, such as a table, and the position of the camera is known with respect to the table. In such cases, an object's position in the image determines its spatial location. The position of an object in an image may be defined using the center of area of the object image. Though other methods such as the enclosing rectangle of the object image may be used, the center of area is a point and is relatively insensitive to noise in the image.

The center of area in binary images is the same as the center of mass if we consider the intensity at a point as the mass at that point. To calculate the position of the object, we use

$$\bar{x} \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} B[i,j] = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} jB[i,j] \tag{2.17}$$

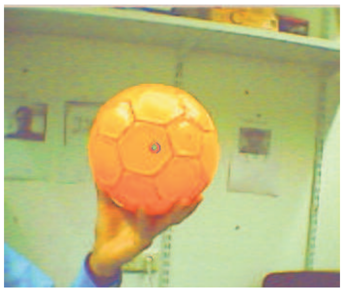
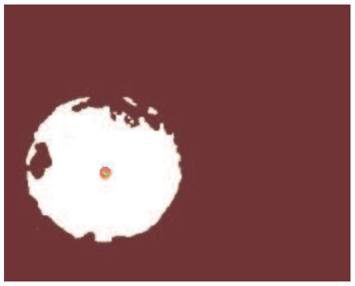



Figure 2.11: A soccer ball image and its resulting binary image with their respective centroid.*Left:* Original image. *Right:* calculated centroid.

$$\bar{y}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} B[i,j] = -\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} iB[i,j] \tag{2.18}$$

where $\bar{x}$ and $\bar{y}$ are the coordinates of the center of the region measured with respect to the top left pixel. Thus, the position of an object is

$$\bar{x} = \frac{\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} jB[i,j]}{A} \tag{2.19}$$

$$\bar{y} = \frac{-\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} iB[i,j]}{A} \tag{2.20}$$

These are the first-order moments. The position calculated using first moments is not necessarily an integer and usually lies between the integer values of the image array indices. We emphasize that this does *not* imply that the calculated position is better than the resolution of pixel coordinates.

An example of these concepts can be applied to find the centroid a soccer ball in an image. First the image is transformed of the RGB model to the HSV model, which is more appropriate for the color segmentation, later the image is divided in its characteristic planes H,S and V. From those planes we take only the S plane and then we apply a threshold value of 220. Once the object is segmented, then we can use the equations 2.19 and 2.20 to identify the centroid. The figure 2.11 shows the result of this process.

These are the first-order moments. The position calculated using first moments is not necessarily an integer and usually lies between the integer values of the image array indices. We emphasize that this does not imply that the calculated position is better than the resolution of pixel coordinates.

2.11 Orientation

Calculating the orientation of an object is a little more complex than calculating its position [26]. For some shapes (such as circles), orientation is not unique. To define unique orientation, an object must be elongated. If so, the orientation of the axis of elongation can be used to define the orientation of the object. Usually, the axis of least second moment, which in 2-D is equivalent to the axis of least inertia, is used as the axis of elongation.

The axis of second moment for an object image is that line for which the sum of the squared distances between object points and the line is minimum. Given a binary image, *B*[*i*,*j*], compute the least-squares fit of a line to the object points in the binary image. Minimize the sum of the squared perpendicular distances of all object points from the line,

$$\chi^2 = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} r_{ij}^2 B[i,j] \tag{2.21}$$

where $r_{11}$is the perpendicular distance from an object point [*i*,*j*] to the line. To avoid numerical problems when the line is nearly vertical, represent the line in polar coordinates:

$$\rho = x \cos\theta + y \sin\theta \tag{2.22}$$

As shown in Figure 2.12, $\theta$is the orientation of the normal to the line with the *x* axis, and *p* is the distance of the line from the origin. The distance r of a point (*x*, *y*) is obtained by plugging the coordinates of the point into the equation for the line:

$$r^2 = (x \cos\theta + y \sin\theta - \rho)^2 \tag{2.23}$$

Plugging the representation of the line into the minimization criterion yields the regression problem for fitting a straight line to the object points. Determine the model parameters ρ and θ by minimizing

$$\chi^2 = \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} (x_{ij} \cos\theta + y_{ij} \sin\theta - \rho)^2 B[i,j] \tag{2.24}$$

Setting the derivative of χ^2 with respect to ρ to zero and solving for ρ yields

$$\rho = \bar{x} \cos\theta + \bar{y} \sin\theta \tag{2.25}$$

which shows that the regression line passes through the center of object points ($\bar{x}$,$\bar{y}$). After substituting this value of ρin the above equation for $\chi^2$and replacing

$$x' = x - \bar{x}$$

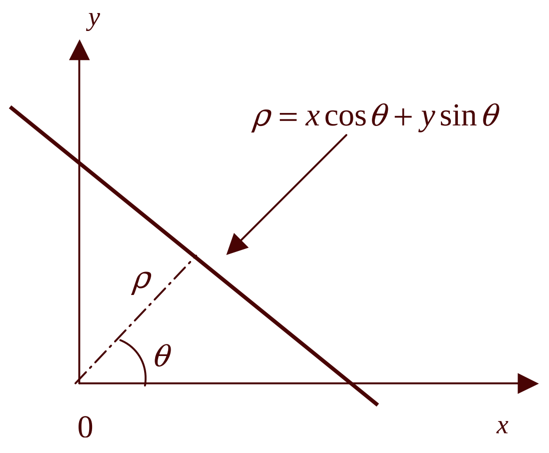


Figure 2.12: Polar representation of a straight line.

$$y' = y - \bar{y} \tag{2.26}$$

the minimizations problem becomes

$$\chi^2 = a\cos^2\theta + b\sin\theta\cos\theta + c\sin^2\theta \tag{2.27}$$

The parameters

$$a = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1}(x'_{ij})^2 B[i,j] \tag{2.28}$$

$$b = 2\sum_{i=0}^{n-1}\sum_{j=0}^{m-1} x'_{ij}y'_{ij}B[i,j] \tag{2.29}$$

$$c = \sum_{i=0}^{n-1}\sum_{j=0}^{m-1}(y'_{ij})^2 B[i,j] \tag{2.30}$$

are the second-order moments. The expression for χ^2 can be rewritten as

$$\chi^2 = \frac{1}{2}(a+c) + \frac{1}{2}(a-c)\cos\theta + \frac{1}{2}b\sin\theta \tag{2.31}$$

Differentiating χ^2, setting the result to zero, and solving for θ yield

$$\tan 2\theta = \frac{b}{a-c} \tag{2.32}$$

The orientation of the axis is given by

$$\sin 2\theta = \pm \frac{b}{\sqrt{b^2 + (a-c)^2}} \tag{2.33}$$

$$\cos 2\theta = \pm \frac{a-c}{\sqrt{b^2 + (a-c)^2}} \tag{2.34}$$

The axis of orientation is obtained for the minimal value of χ^2. Note that if *b*=0 and *a*=*c*, the object does not have a unique axis of orientation. The elongation *E* of the object is the ratio of largest to smallest values for χ.

$$E = \frac{\chi_{\max}}{\chi_{\min}} \tag{2.35}$$

When the expressions for $\sin 2\theta$ and $\cos 2\theta$ are substituted into equation 2.31, their signs determine whether χ^2 is a maximum or minimum. Note that the elongation is 1 for a circle and that this is lower bound on *E*.