1 Introduction

1.1 Images and Pictures

As we mentioned in the Preface, human beings are predominantly visual creatures: we rely heavily on our vision to make sense of the world around us. We not only look at things to identify and classify them, but we can scan for differences, and obtain an overall rough "feeling" for a scene with a quick glance.

Humans have evolved very precise visual skills: we can identify a face in an instant; we can differentiate colors; we can process a large amount of visual information very quickly.

However, the world is in constant motion: stare at something for long enough and it will change in some way. Even a large solid structure, like a building or a mountain, will change its appearance depending on the time of day (day or night); amount of sunlight (clear or cloudy), or various shadows falling upon it.

We are concerned with single images: snapshots, if you like, of a visual scene. Although image processing can deal with changing scenes, we shall not discuss it in any detail in this text.

For our purposes, an *image* is a single picture that represents something. It may be a picture of a person, people, or animals, an outdoor scene, a microphotograph of an electronic component, or the result of medical imaging. Even if the picture is not immediately recognizable, it will not be just a random blur.

1.2 What Is Image Processing?

Image processing involves changing the nature of an image in order to either

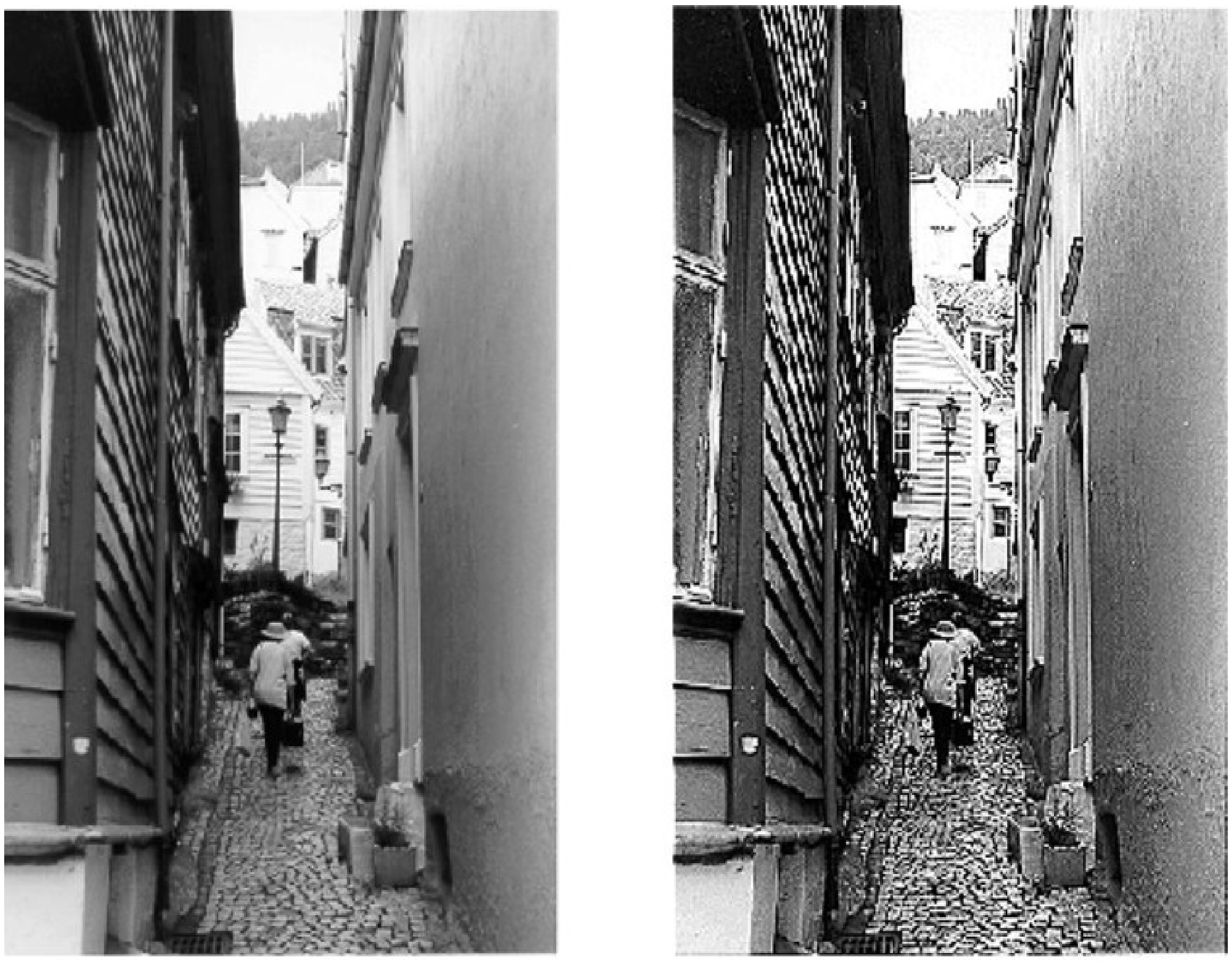
1. Improve its pictorial information for human interpretation
2. Render it more suitable for autonomous machine perception

We shall be concerned with *digital image processing*, which involves using a computer to change the nature of a *digital image* (see Section 1.4). It is necessary to realize that these two aspects represent two separate but equally important aspects of image processing. A procedure that satisfies condition (1)—a procedure that makes an image "look better"—may be the very worst procedure for satisfying condition (2). Humans like their images to be sharp, clear, and detailed; machines prefer their images to be simple and uncluttered.

Examples of (1) may include:

- Enhancing the edges of an image to make it appear sharper; an example is shown in Figure 1.1. Note how the second image appears "cleaner"; it is a more pleasant image. Sharpening edges is a vital component of printing: in order for an image to appear "at its best" on the printed page; some sharpening is usually performed.

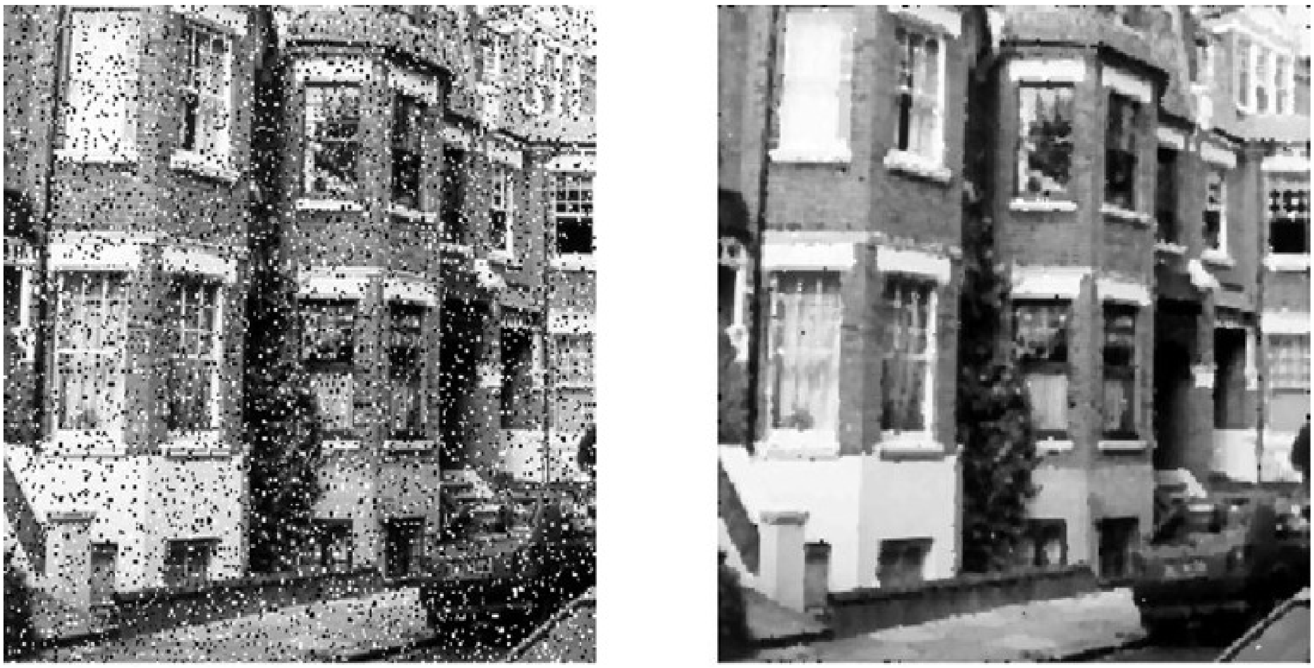
Figure 1.1: Image sharpening



(a) The original image (b) Result after "sharpening"

- Removing "noise" from an image; noise being random errors in the image. An example is given in Figure 1.2. Noise is a very common problem in data transmission: all sorts of electronic components may affect data passing through them, and the results may be undesirable. As we shall see in Chapter 8, noise may take many different forms, and each type of noise requires a different method of removal.

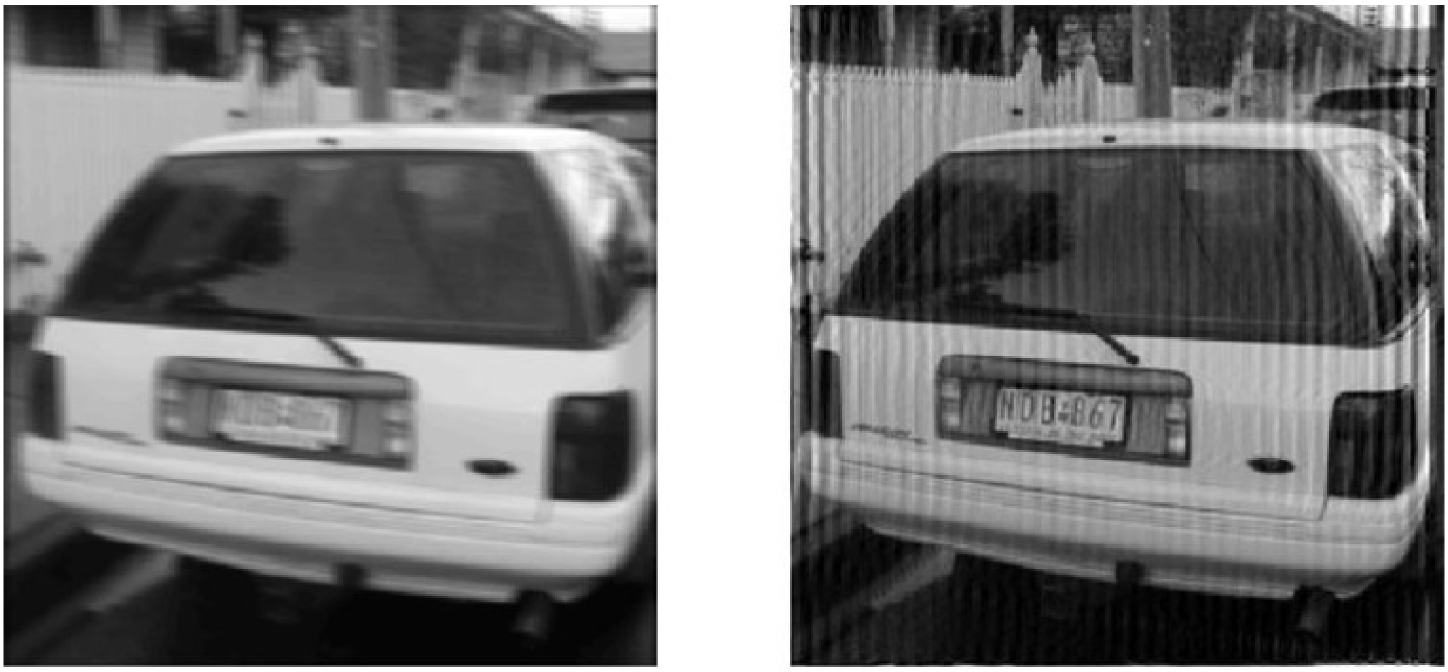
Figure 1.2: Removing noise from an image



(a) The original image (b) After removing noise

- • Removing motion blur from an image. An example is given in Figure 1.3. Note that in the deblurred image (b) it is easier to read the numberplate, and to see the spikes on the fence behind the car, as well as other details not at all clear in the original image (a). Motion blur may occur when the shutter speed of the camera is too long for the speed of the object. In photographs of fast moving objects—athletes, vehicles for example—the problem of blur may be considerable.

Figure 1.3: Image deblurring

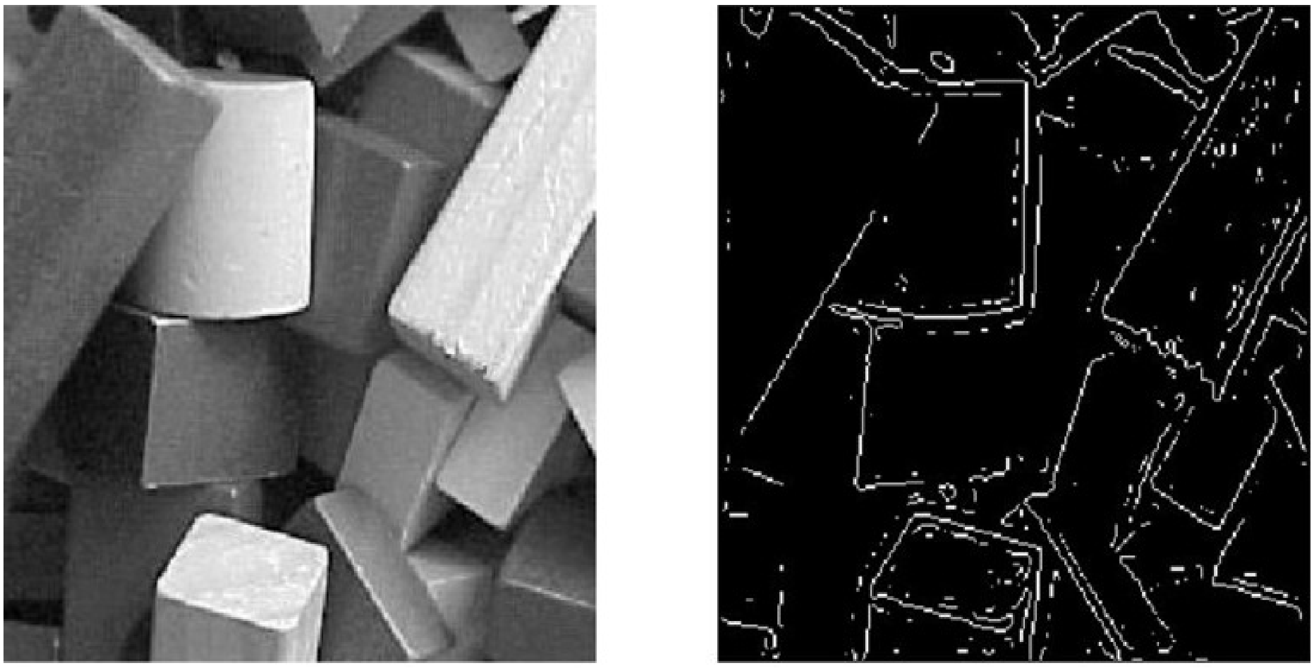


(a) The original image (b) After removing the blur

Examples of (2) may include:

• Obtaining the edges of an image. This may be necessary for the measurement of objects in an image; an example is shown in Figures 1.4. Once we have the edges we can measure their spread, and the area contained within them. We can also use edge detection algorithms as a first step in edge enhancement, as we saw previously. From the edge result, we see that it may be necessary to enhance the original image slightly, to make the edges clearer.

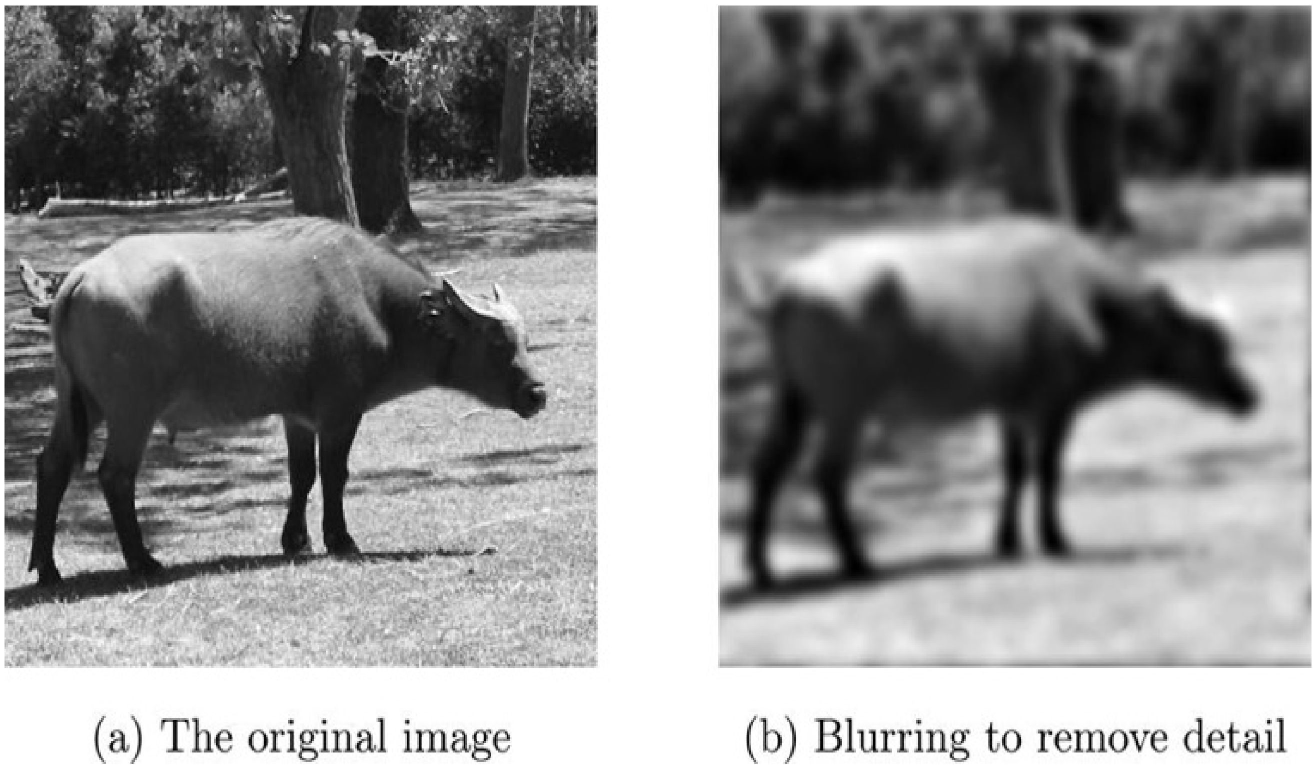
Figure 1.4: Finding edges in an image



(a) The original image (b) Its edge image

• Removing detail from an image. For measurement or counting purposes, we may not be interested in all the detail in an image. For example, a machine inspected items on an assembly line, the only matters of interest may be shape, size, or color. For such cases, we might want to simplify the image. Figure 1.5 shows an example: in image (a) is a picture of an African buffalo, and image (b) shows a blurred version in which extraneous detail (like the logs of wood in the background) have been removed. Notice that in image (b) all the fine detail is gone; what remains is the coarse structure of the image. We could for example, measure the size and shape of the animal without being “distracted” by unnecessary detail.

Figure 1.5: Blurring an image



1.3 Image Acquisition and Sampling

Sampling refers to the process of digitizing a continuous function. For example, suppose we take the function

$$y = \sin(x) + \frac{1}{3}\sin(3x)$$

and sample it at ten evenly spaced values of *x* only. The resulting sample points are shown in Figure 1.6. This shows an example of *undersampling*, where the number of points is not sufficient to reconstruct the function. Suppose we sample the function at 100 points, as shown in Figure 1.7. We can clearly now reconstruct the function; all its properties can be determined from this sampling. In order to ensure that we have enough sample points, we require that the sampling period is not greater than one-half the finest detail in our function. This is known as the *Nyquist criterion*, and can be formulated more precisely in terms of “frequencies,” which are discussed in Chapter 7. The Nyquist criterion can be stated as the *sampling theorem*, which says, in effect, that a continuous function can be reconstructed from its samples provided that the sampling frequency is at least twice the maximum frequency in the function. A formal account of this theorem is provided by Castleman [7].

Sampling an image again requires that we consider the Nyquist criterion, when we consider an image as a continuous function of two variables, and we wish to sample it to produce a digital image.

An example is shown in Figure 1.8 where an image is shown, and then with an undersampled version. The jagged edges in the undersampled image are examples of *aliasing*.

Figure 1.6: Sampling a function—undersampling

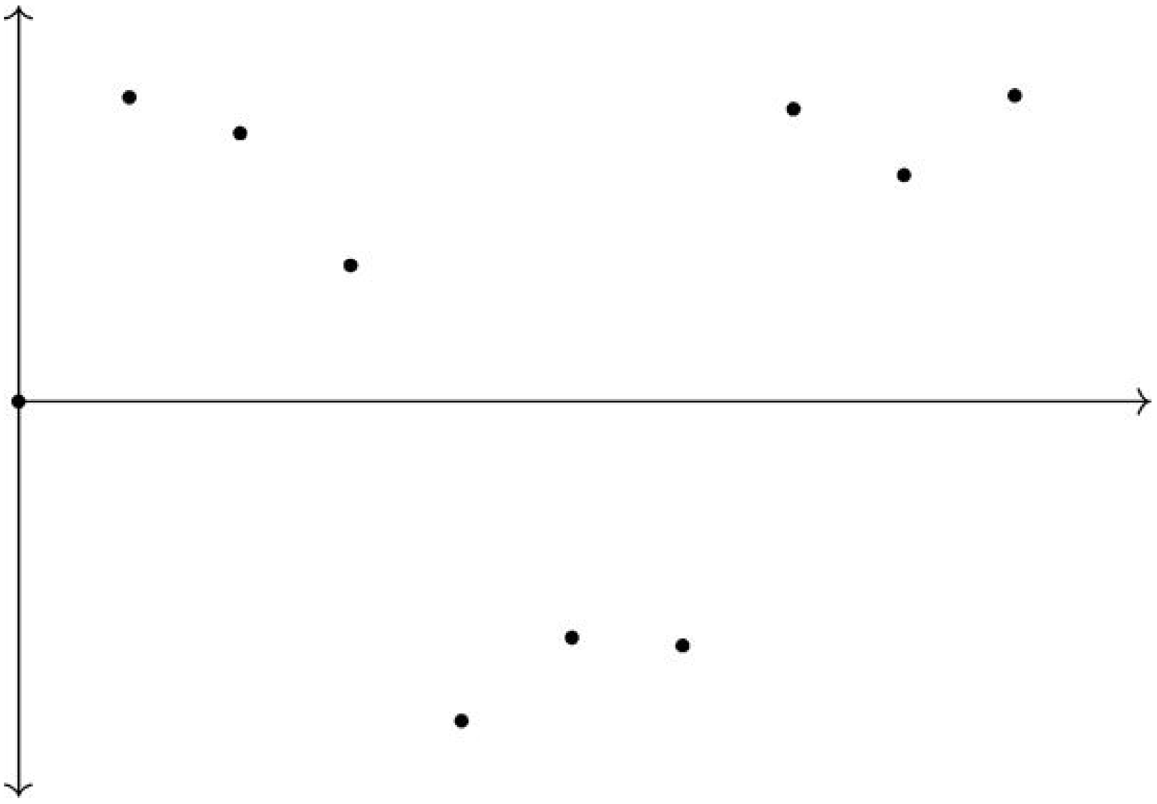


Figure 1.7: Sampling a function with more points

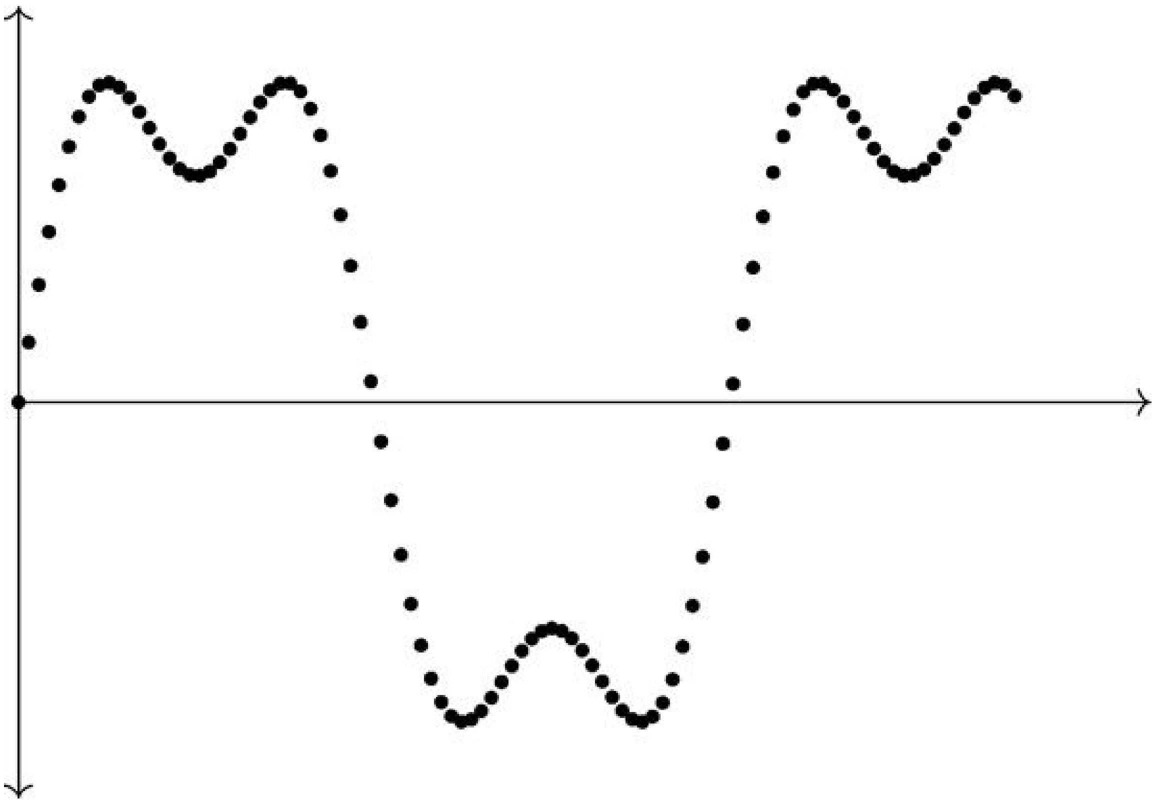
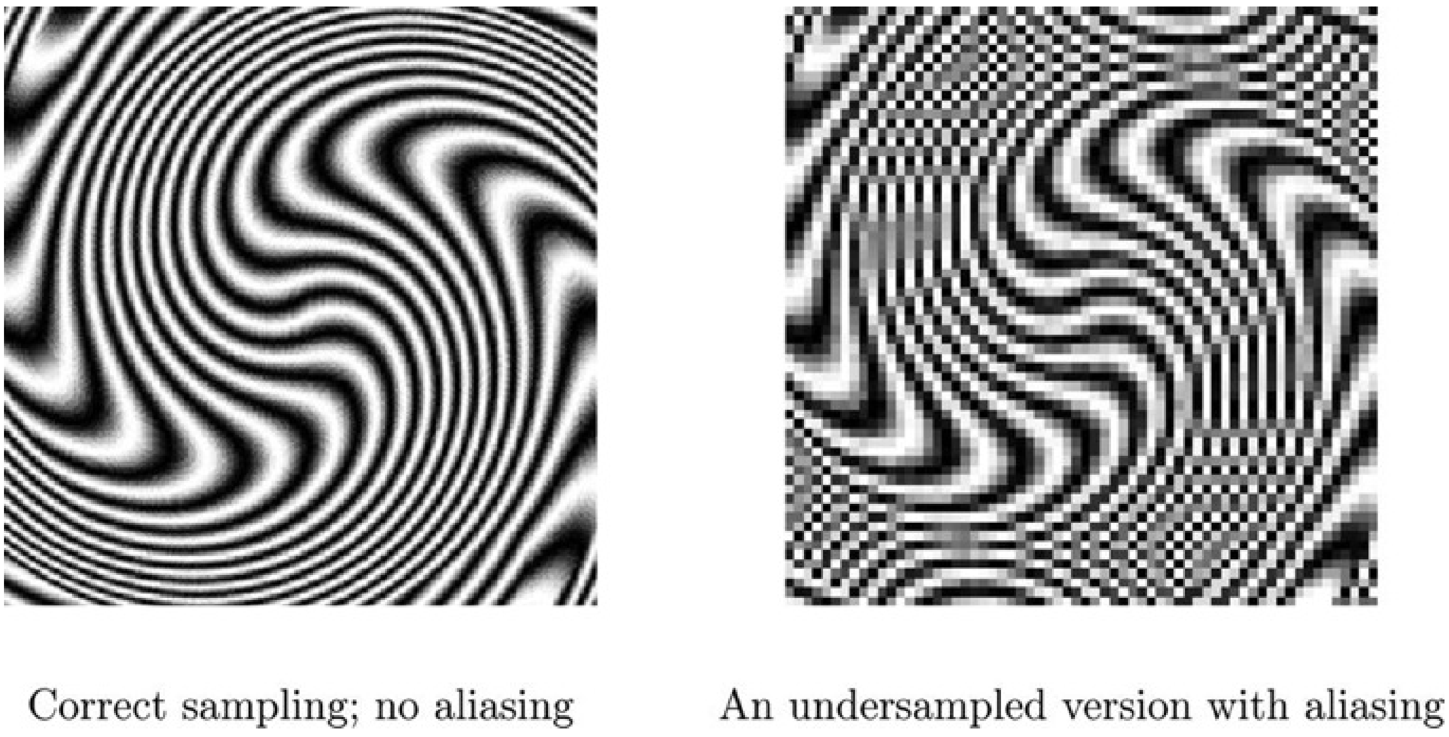


Figure 1.8: Effects of sampling


Correct sampling; no aliasing
An undersampled version with aliasing

The sampling rate will of course affect the final resolution of the image; we discuss this in Chapter 3. In order to obtain a sampled (digital) image, we may start with a continuous representation of a scene. To view the scene, we record the energy reflected from it; we may use visible light, or some other energy source.

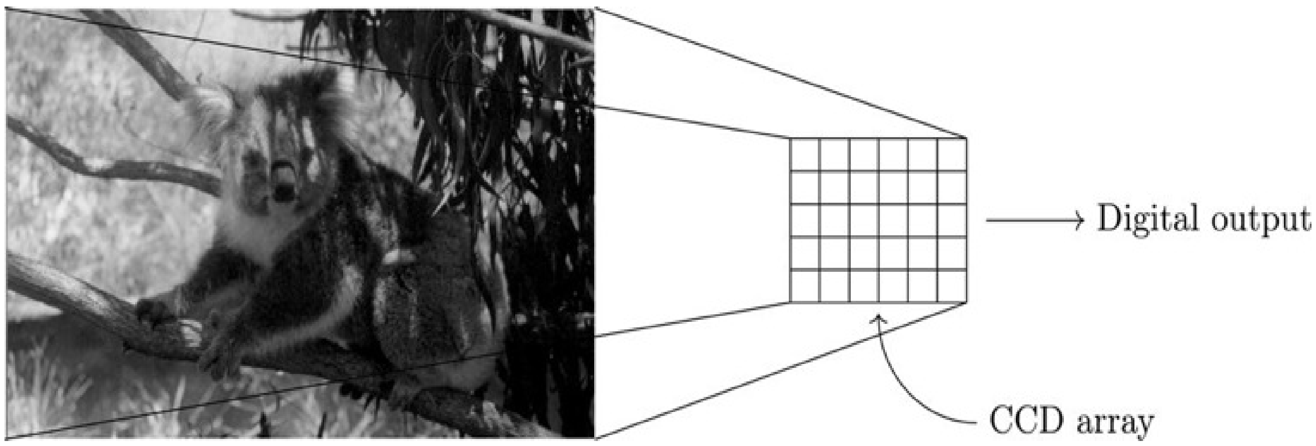
Using Light

Light is the predominant energy source for images; simply because it is the energy source that human beings can observe directly. We are all familiar with photographs, which are a pictorial record of a visual scene.

Many digital images are captured using visible light as the energy source; this has the advantage of being safe, cheap, easily detected, and readily processed with suitable hardware. Two very popular methods of producing a digital image are with a digital camera or a flat-bed scanner.

CCD camera. Such a camera has, in place of the usual film, an array of *photosites*; these are silicon electronic devices whose voltage output is proportional to the intensity of light falling on them. For a camera attached to a computer, information from the photosites is then output to a suitable storage medium. Generally this is done on hardware, as being much faster and more efficient than software, using a *frame-grabbing card*. This allows a large number of images to be captured in a very short time—in the order of one ten-thousandth of a second each. The images can then be copied onto a permanent storage device at some later time. This is shown schematically in Figure 1.9.

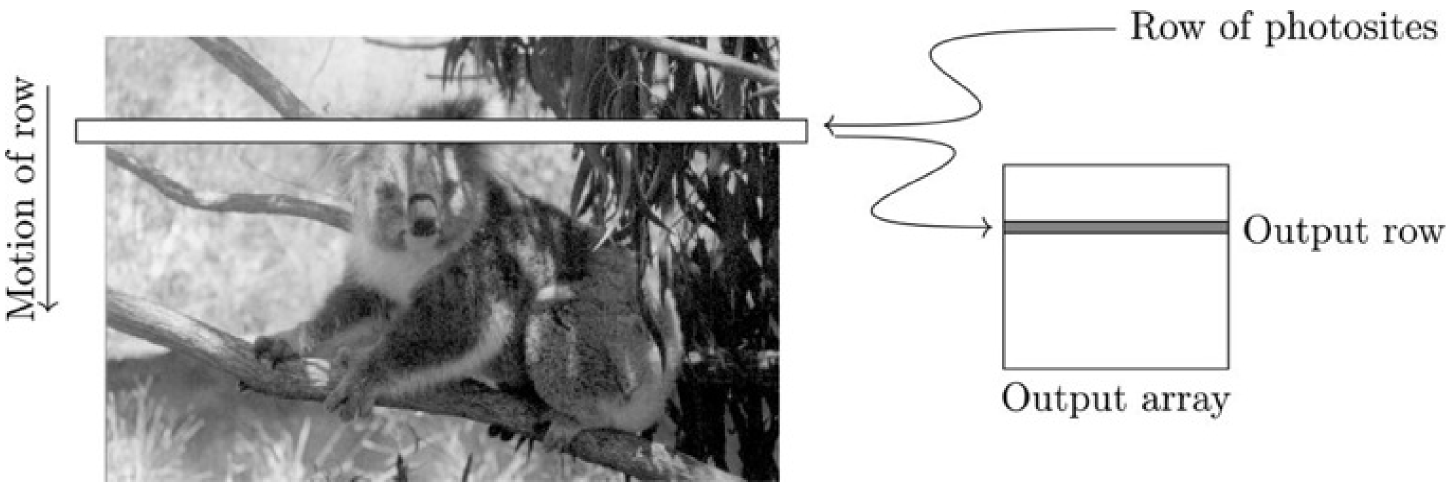
Figure 1.9: Capturing an image with a CCD array



The output will be an array of values; each representing a sampled point from the original scene. The elements of this array are called *picture elements*, or more simply *pixels*. Digital still cameras use a range of devices, from floppy discs and CDs, to various specialized cards and "memory sticks." The information can then be downloaded from these devices to a computer hard disk.

Flat bed scanner. This works on a principle similar to the CCD camera. Instead of the entire image being captured at once on a large array, a single row of photosites is moved across the image, capturing it row-by-row as it moves. This is shown schematically in Figure 1.10. Since this is a much slower process than taking a picture with a camera, it is quite reasonable to allow all capture and storage to be processed by suitable software.

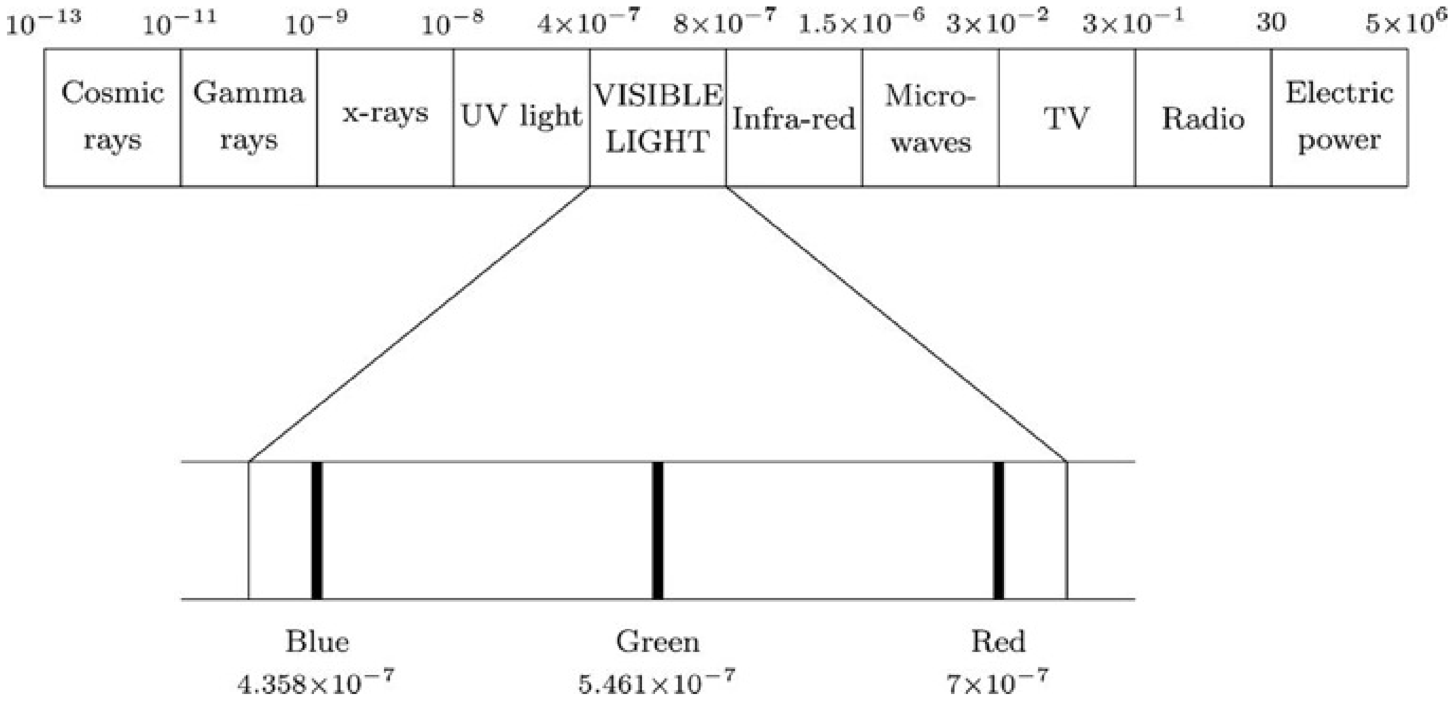
Figure 1.10: Capturing an image with a CCD scanner



Other Energy Sources

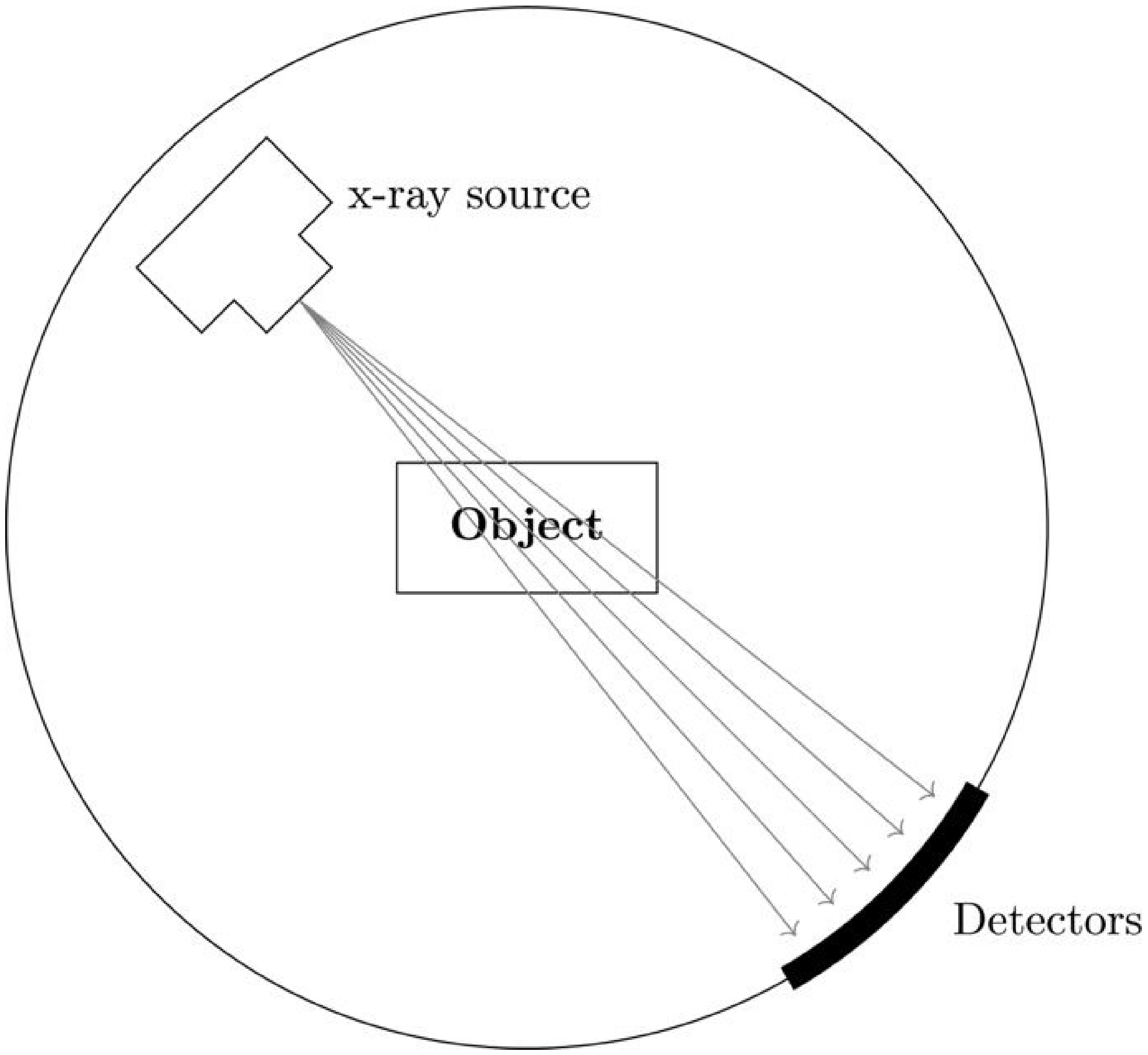
Although light is popular and easy to use, other energy sources may be used to create a digital image. Visible light is part of the *electromagnetic spectrum*: radiation in which the energy takes the form of waves of varying wavelength. These range from cosmic rays of very short wavelength, to electric power, which has very long wavelength. Figure 1.11 illustrates this. For microscopy, we may use x-rays or electron beams. As we can see from Figure 1.11, x-rays have a shorter wavelength than visible light, and so can be used to resolve smaller objects than are possible with visible light. See Clark [8] for a good introduction to this. X-rays are of course also useful in determining the structure of objects usually hidden from view, such as bones.

Figure 1.11: The electromagnetic spectrum



A further method of obtaining images is by the use of *x-ray tomography*, where an object is encircled by an x-ray beam. As the beam is fired through the object, it is detected on the other side of the object, as shown in Figure 1.12. As the beam moves around the object, an image of the object can be constructed; such an image is called a *tomogram*. In a CAT (Computed Axial Tomography) scan, the patient lies within a tube around which x-ray beams are fired. This enables a large number of tomographic “slices” to be formed, which can then be joined to produce a three-dimensional image. A good account of such systems (and others) is given by Siedband [48].

Figure 1.12: X-ray tomography



1.4 Images and Digital Images

Suppose we take an image, a photo, say. For the moment, let's make things easy and suppose the photo is monochromatic (that is, shades of gray only), so no color. We may consider this image as being a two-dimensional function, where the function values give the brightness of the image at any given point, as shown in Figure 1.13. We may assume that in such an image brightness values can be any real numbers in the range 0.0 (black) to 1.0 (white). The ranges of *x* and *y* will clearly depend on the image, but they can take all real values between their minima and maxima.

Such a function can of course be plotted, as shown in Figure 1.14. However, such a plot is of limited use to us in terms of image analysis. The concept of an image as a function, however, will be vital for the development and implementation of image processing techniques.

A *digital image* differs from a photo in that the *x, y*, and *f*(*x, y*) values are all *discrete*. Usually they take on only integer values, so the image shown in Figure 1.13 will have *x* and *y* ranging from 1 to 256 each, and the brightness values also ranging from 0 (black) to 255 (white). A digital image, as we have seen above, can be considered a large array of sampled points from the continuous image, each of which has a particular quantized brightness; these points are the *pixels*, which constitute the digital image. The pixels surrounding a given pixel constitute its *neighborhood*. A neighborhood can be characterized by its shape in the same way as a matrix: we can speak, for example, of a 3 × 3 neighborhood or of a 5 × 7

neighborhood. Except in very special circumstances, neighborhoods have odd numbers of rows and columns; this ensures that the current pixel is in the center of the neighborhood. An example of a neighborhood is given in Figure 1.15. If a neighborhood has an even number of rows or columns (or both), it may be necessary to specify which pixel in the neighborhood is the "current pixel."

Figure 1.13: An image as a function

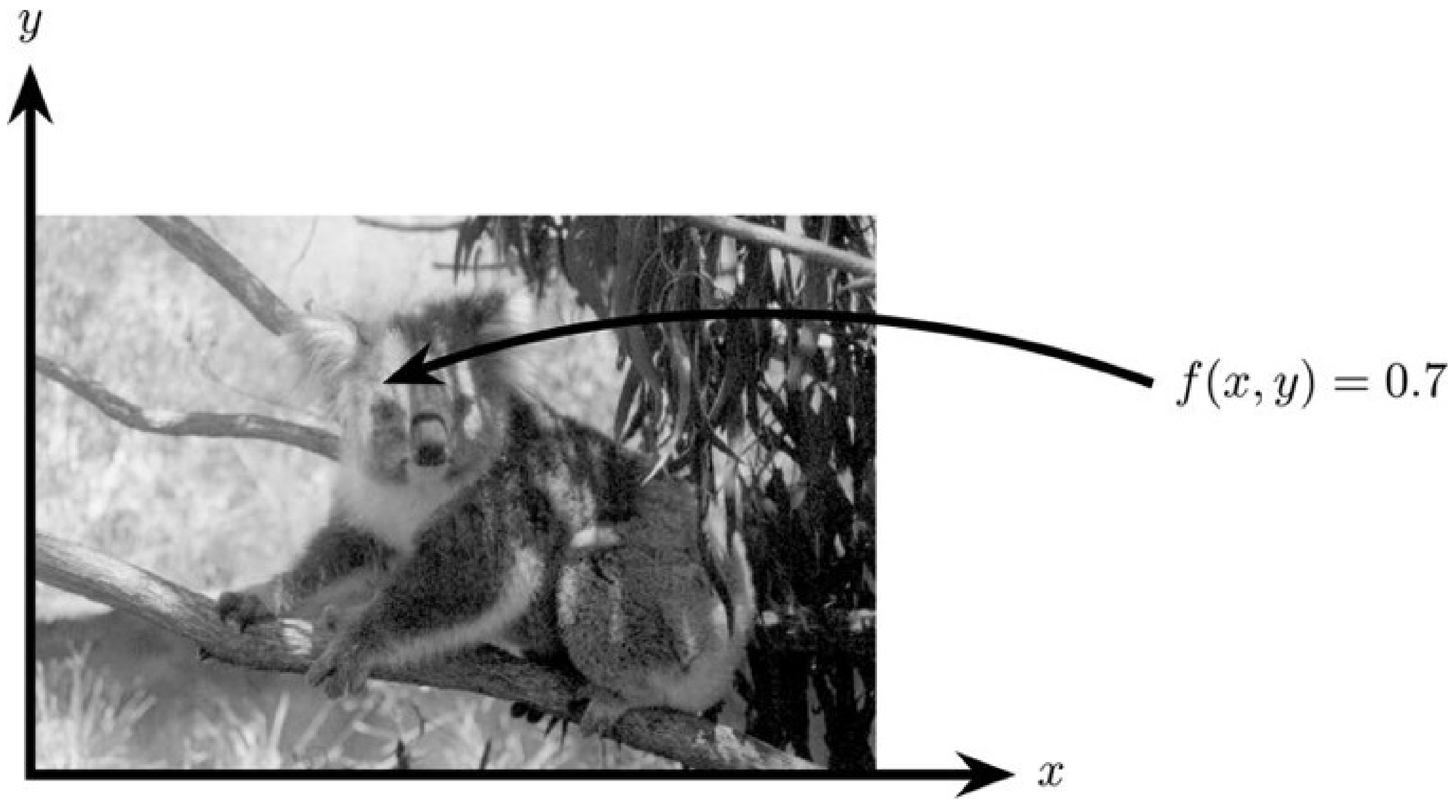


Figure 1.14: The image of Figure 1.13 plotted as a function of two variables

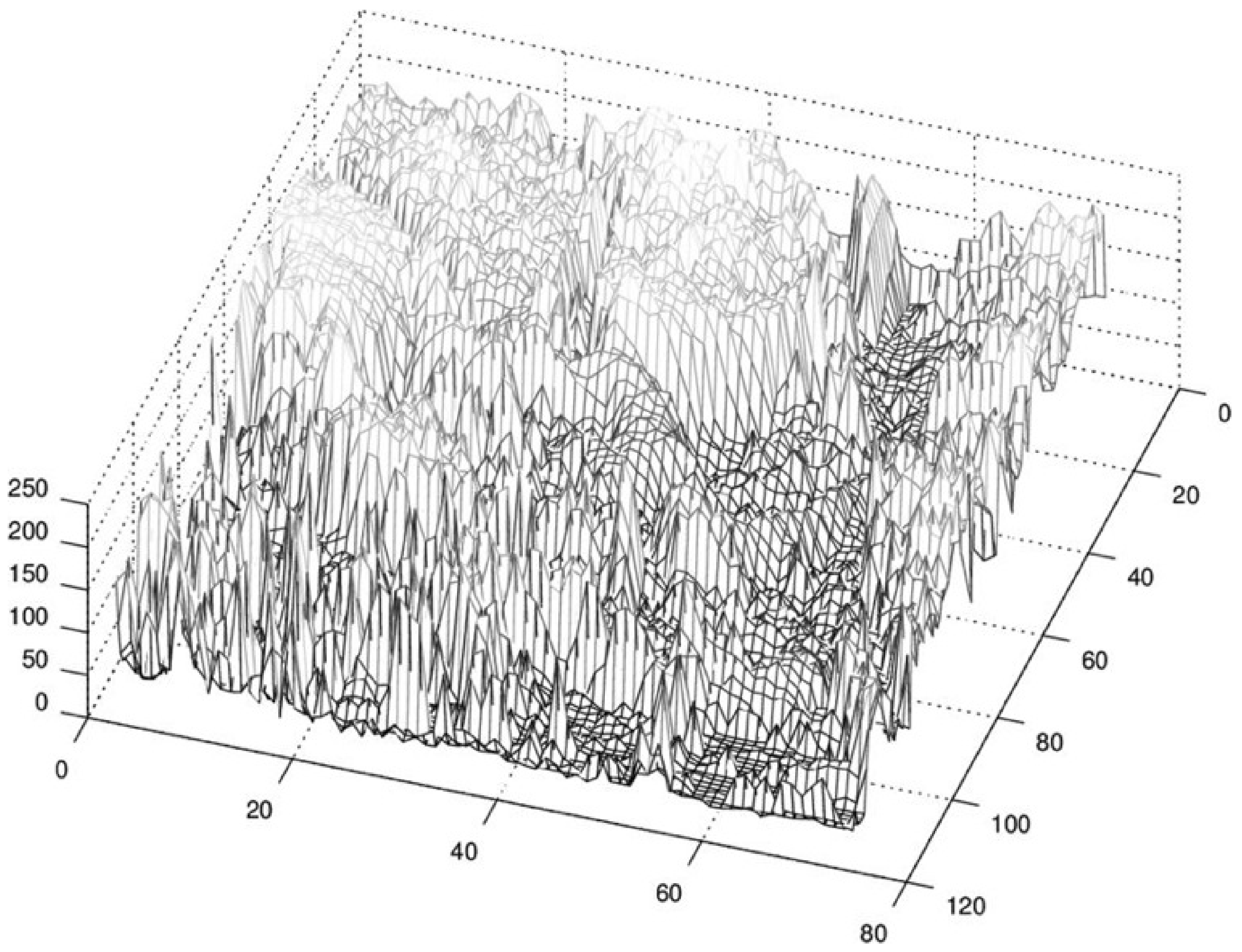
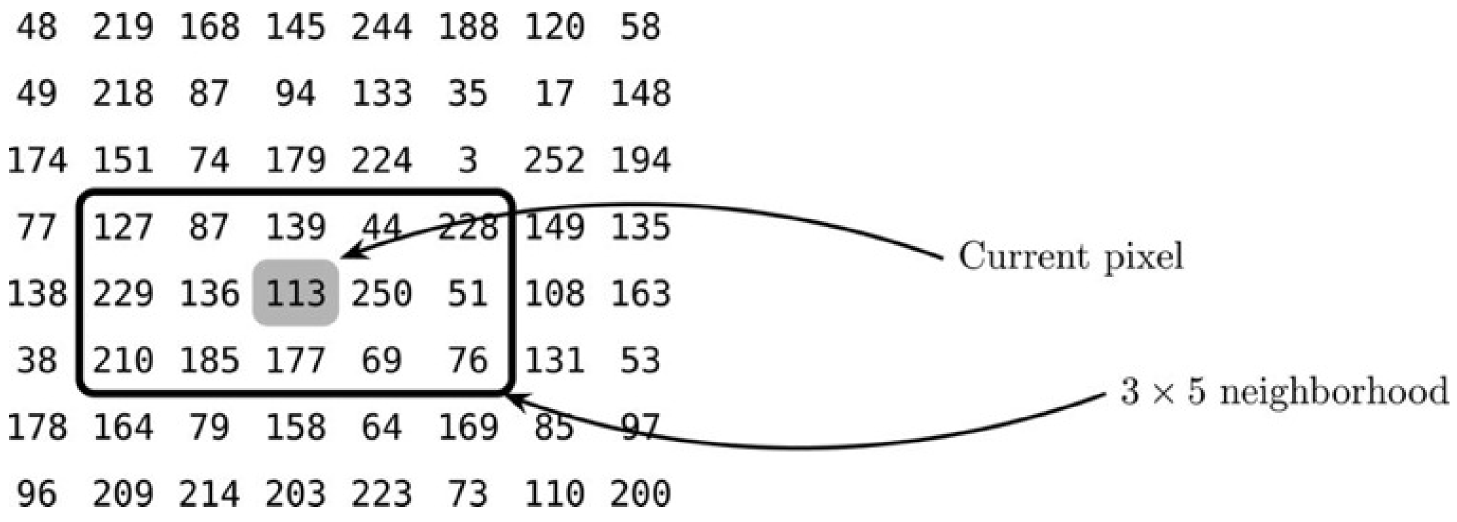


Figure 1.15: Pixels, with a neighborhood



48	219	168	145	244	188	120	58
49	218	87	94	133	35	17	148
174	151	74	179	224	3	252	194
77	127	87	139	44	228	149	135
138	229	136	113	250	51	108	163
38	210	185	177	69	76	131	53
178	164	79	158	64	169	85	97
96	209	214	203	223	73	110	200

1.5 Some Applications

Image processing has an enormous range of applications; almost every area of science and technology can make use of image processing methods. Here is a short list just to give some indication of the range of image processing applications.

1. Medicine
    - Inspection and interpretation of images obtained from x-rays, MRI, or CAT scans
    - Analysis of cell images, of chromosome karyotypes
2. Agriculture
    - Satellite/aerial views of land, for example to determine how much land is being used for different purposes, or to investigate the suitability of different regions for different crops
    - Inspection of fruit and vegetables—distinguishing good and fresh produce from old
3. Industry
    - Automatic inspection of items on a production line
    - Inspection of paper samples
4. Law enforcement
    - Fingerprint analysis
    - Sharpening or deblurring of speed-camera images

1.6 Image Processing Operations

It is convenient to subdivide different image processing algorithms into broad subclasses. There are different algorithms for different tasks and problems, and often we would like to distinguish the nature of the task at hand.

Image enhancement. This refers to processing an image so that the result is more suitable for a particular application. Examples include:

- Sharpening or deblurring an out of focus image
- Highlighting edges
- Improving image contrast, or brightening an image
- Removing noise

Image restoration. This may be considered as reversing the damage done to an image by a known cause, for example:

- Removing of blur caused by linear motion
- Removal of optical distortions
- Removing periodic interference

Image segmentation. This involves subdividing an image into constituent parts, or isolating certain aspects of an image:

- Finding lines, circles, or particular shapes in an image

- In an aerial photograph, identifying cars, trees, buildings, or roads

Image registration. This involves “matching” distinct images so that they can be compared, or processed together. The initial images must all be joined to share the same coordinate system. In this text, registation as such is not covered, but some tasks that are vital to registration are discussed, for example corner detection.

These classes are not disjoint; a given algorithm may be used for both image enhancement or for image restoration. However, we should be able to decide what it is that we are trying to do with our image: simply make it look better (enhancement) or removing damage (restoration).

1.7 An Image Processing Task

We will look in some detail at a particular real-world task, and see how the above classes may be used to describe the various stages in performing this task. The job is to obtain, by an automatic process, the postcodes from envelopes. Here is how this may be accomplished:

Acquiring the image. First we need to produce a digital image from a paper envelope. This can be done using either a CCD camera or a scanner.

Preprocessing. This is the step taken before the “major” image processing task. The problem here is to perform some basic tasks in order to render the resulting image more suitable for the job to follow. In this case, it may involve enhancing the contrast, removing noise, or identifying regions likely to contain the postcode.

Segmentation. Here is where we actually “get” the postcode; in other words, we extract from the image that part of it that contains just the postcode.

Representation and description. These terms refer to extracting the particular features which allow us to differentiate between objects. Here we will be looking for curves, holes, and corners, which allow us to distinguish the different digits that constitute a postcode.

Recognition and interpretation. This means assigning labels to objects based on their descriptors (from the previous step), and assigning meanings to those labels. So we identify particular digits, and we interpret a string of four digits at the end of the address as the postcode.

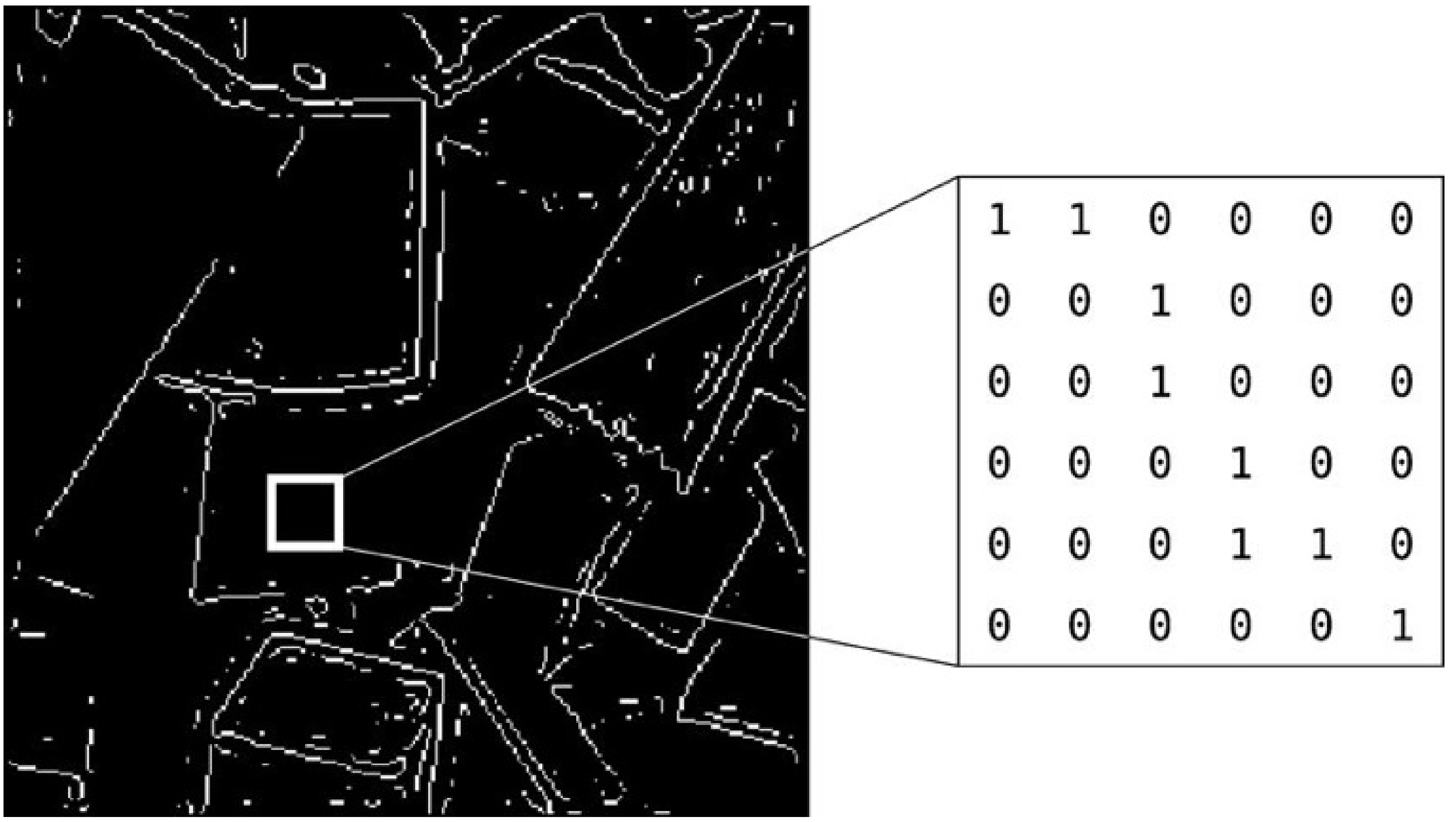
1.8 Types of Digital Images

We shall consider four basic types of images:

Binary. Each pixel is just black or white. Since there are only two possible values for each pixel, we only need one bit per pixel. Such images can therefore be very efficient in terms of storage. Images for which a binary representation may be suitable include text (printed or handwriting), fingerprints, or architectural plans.

An example was the image shown in Figure 1.4(b). In this image, we have only the two colors: white for the edges, and black for the background. See Figure 1.16.

Figure 1.16: A binary image

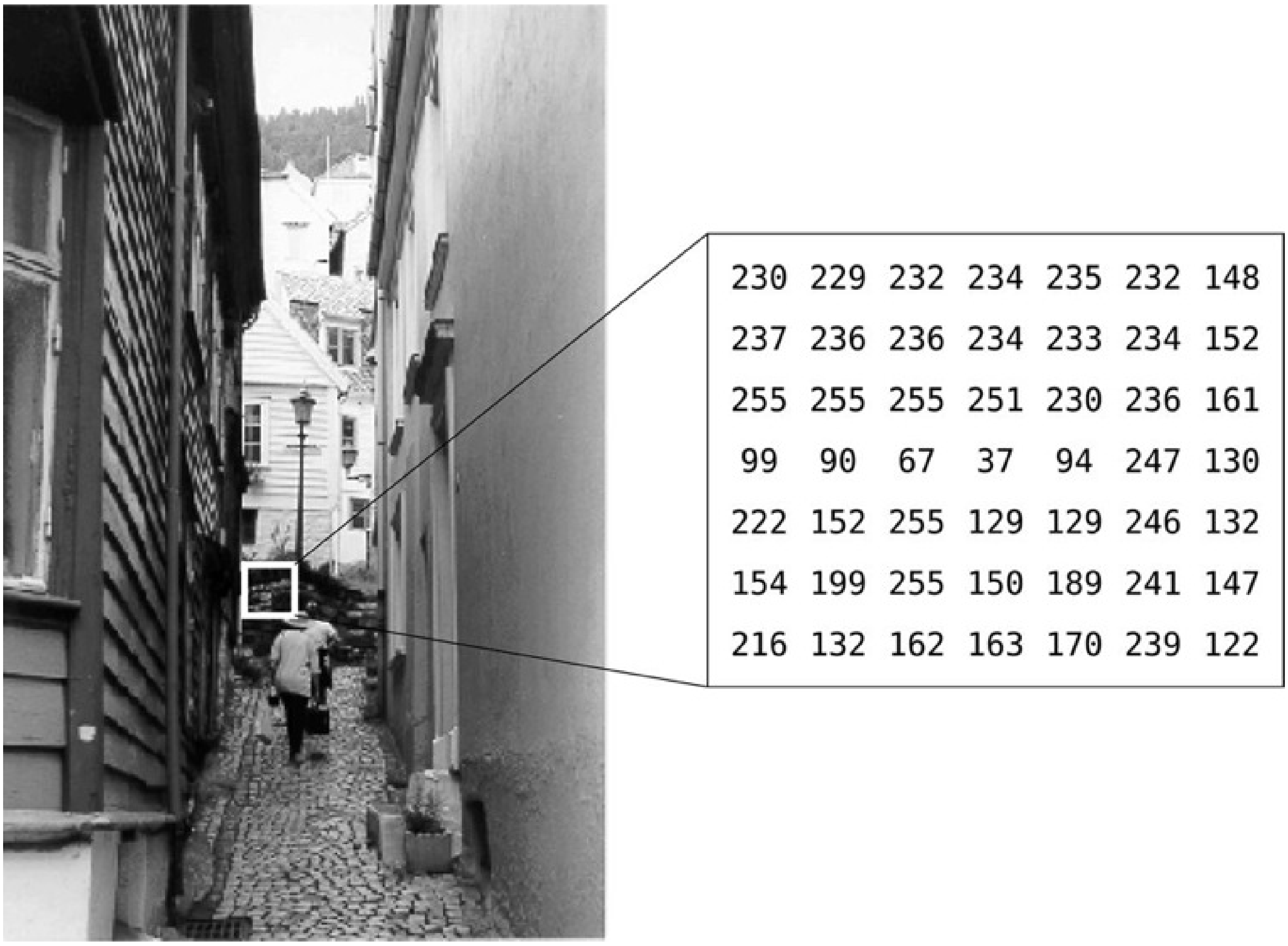


Grayscale. Each pixel is a shade of gray, normally from 0 (black) to 255 (white). This range means that each pixel can be represented by eight bits, or exactly one byte. This is a very natural range for image file handling. Other grayscale ranges are used,

but generally they are a power of 2. Such images arise in medicine (X-rays), images of printed works, and indeed 256 different gray levels is sufficient for the recognition of most natural objects.

An example is the street scene shown in Figure 1.1, and in Figure 1.17.

Figure 1.17: A grayscale image



230	229	232	234	235	232	148
237	236	236	234	233	234	152
255	255	255	251	230	236	161
99	90	67	37	94	247	130
222	152	255	129	129	246	132
154	199	255	150	189	241	147
216	132	162	163	170	239	122

True color, or RGB. Here each pixel has a particular color; that color being described by the amount of red, green, and blue in it. If each of these components has a range 0–255, this gives a total of 255^3 = 16,777,216 different possible colors in the image. This is enough colors for any image. Since the total number of bits required for each pixel is 24, such images are also called 24-*bit color images*.

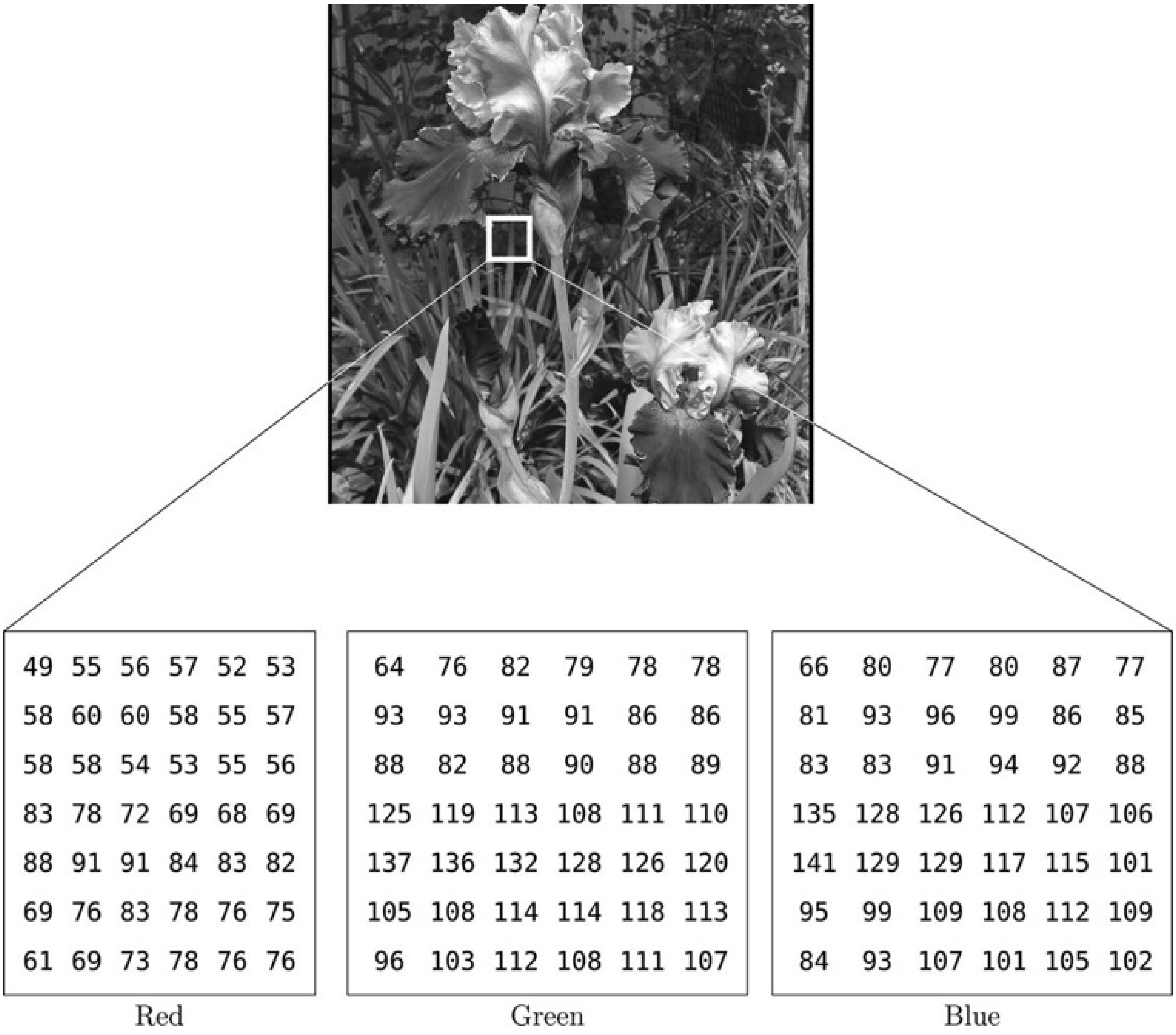
Such an image may be considered as consisting of a “stack” of three matrices; representing the red, green, and blue values for each pixel. This means that for every pixel there are three corresponding values.

An example is shown in Figure 1.18.

Indexed. Most color images only have a small subset of the more than sixteen million possible colors. For convenience of storage and file handling, the image has an associated *color map, or color palette*, which is simply a list of all the colors used in that image. Each pixel has a value which does not give its color (as for an RGB image), but an index to the color in the map.

It is convenient if an image has 256 colors or less, for then the index values will only require one byte each to store. Some image file formats (for example, Compuserve GIF) allow only 256 colors or fewer in each image, for precisely this reason.

Figure 1.18: SEE COLOR INSERT A true color image



Red

49	55	56	57	52	53
58	60	60	58	55	57
58	58	54	53	55	56
83	78	72	69	68	69
88	91	91	84	83	82
69	76	83	78	76	75
61	69	73	78	76	76

Green

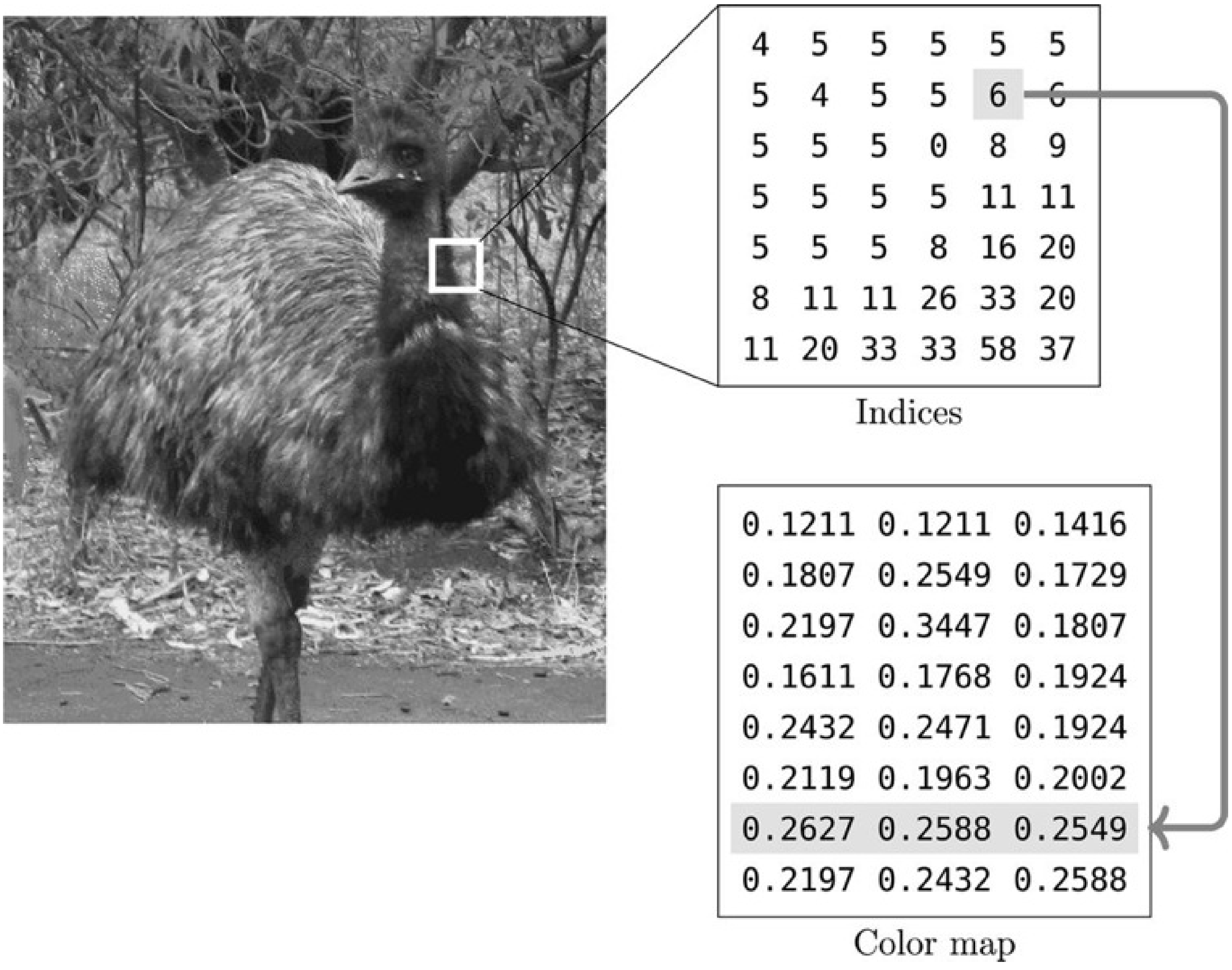
64	76	82	79	78	78
93	93	91	91	86	86
88	82	88	90	88	89
125	119	113	108	111	110
137	136	132	128	126	120
105	108	114	114	118	113
96	103	112	108	111	107

Blue

66	80	77	80	87	77
81	93	96	99	86	85
83	83	91	94	92	88
135	128	126	112	107	106
141	129	129	117	115	101
95	99	109	108	112	109
84	93	107	101	105	102

Figure 1.19 shows an example. In this image the indices, rather then being the gray values of the pixels, are simply indices into the color map. Without the color map, the image would be very dark and colorless. In the figure, for example, pixels labelled `5` correspond to `0.2627 0.2588 0.2549`, which is a dark grayish color.

Figure 1.19: SEE COLOR INSERT An indexed color image



4	5	5	5	5	5
5	4	5	5	6	6
5	5	5	0	8	9
5	5	5	5	11	11
5	5	5	8	16	20
8	11	11	26	33	20
11	20	33	33	58	37

0.1211	0.1211	0.1416
0.1807	0.2549	0.1729
0.2197	0.3447	0.1807
0.1611	0.1768	0.1924
0.2432	0.2471	0.1924
0.2119	0.1963	0.2002
0.2627	0.2588	0.2549
0.2197	0.2432	0.2588

1.9 Image File Sizes

Image files tend to be large. We shall investigate the amount of information used in different image types of varying sizes. For example, suppose we consider a 512 × 512 binary image. The number of bits used in this image (assuming no compression, and neglecting, for the sake of discussion, any header information) is

$$\begin{aligned} 512 \times 512 \times 1 &= 262{,}144 \\ &= 32{,}768 \text{ bytes} \\ &= 32.768 \text{ Kb} \\ &\approx 0.033 \text{ Mb.} \end{aligned}$$

(Here we use the convention that a kilobyte is 1000 bytes, and a megabyte is one million bytes.)

A grayscale image of the same size requires:

$$\begin{aligned} 512 \times 512 \times 1 &= 262{,}144 \text{ bytes} \\ &= 262.14 \text{ Kb} \\ &\approx 0.262 \text{ Mb.} \end{aligned}$$

If we now turn our attention to color images, each pixel is associated with 3 bytes of color information. A 512 × 512 image thus requires

$$\begin{aligned} 521 \times 512 \times 3 &= 786{,}432 \text{ bytes} \\ &= 786.43 \text{ Kb} \\ &\approx 0.786 \text{ Mb.} \end{aligned}$$

Many images are of course larger than this; satellite images may be of the order of several thousand pixels in each direction.

1.10 Image Perception

Much of image processing is concerned with making an image appear "better" to human beings. We should therefore be aware of the limitations of the human visual system. Image perception consists of two basic steps:

1. Capturing the image with the eye
2. Recognizing and interpreting the image with the *visual cortex* in the brain

The combination and immense variability of these steps influences the ways in which we perceive the world around us.

There are a number of things to bear in mind:

1. *Observed intensities* vary as to the background. A single block of gray will appear darker if placed on a white background than if it were placed on a black background. That is, we don't perceive gray scales "as they are," but rather as they differ from their surroundings. In Figure 1.20, a gray square is shown on two different backgrounds. Notice how much darker the square appears when it is surrounded by a light gray. However, the two central squares have exactly the same intensity.
2. We may observe non-existent intensities as bars in continuously varying gray levels. See, for example, Figure 1.21. This image varies continuously from light to dark as we travel from left to right. However, it is impossible for our eyes not to see a few horizontal edges in this image.
3. Our visual system tends to undershoot or overshoot around the boundary of regions of different intensities. For example, suppose we had a light gray blob on a dark gray background. As our eye travels from the dark background to the light region, the boundary of the region appears lighter than the rest of it. Conversely, going in the other direction, the boundary of the background appears *darker* than the rest of it.

Figure 1.20: A gray square on different backgrounds

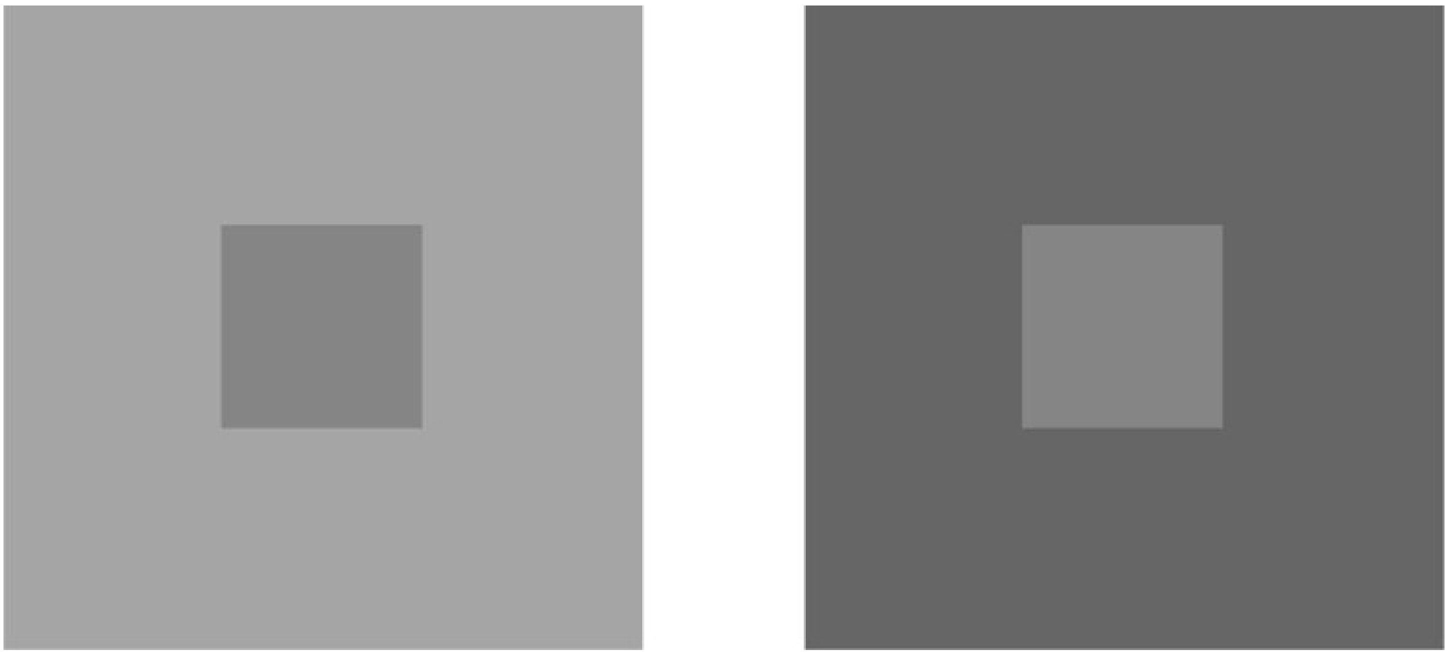
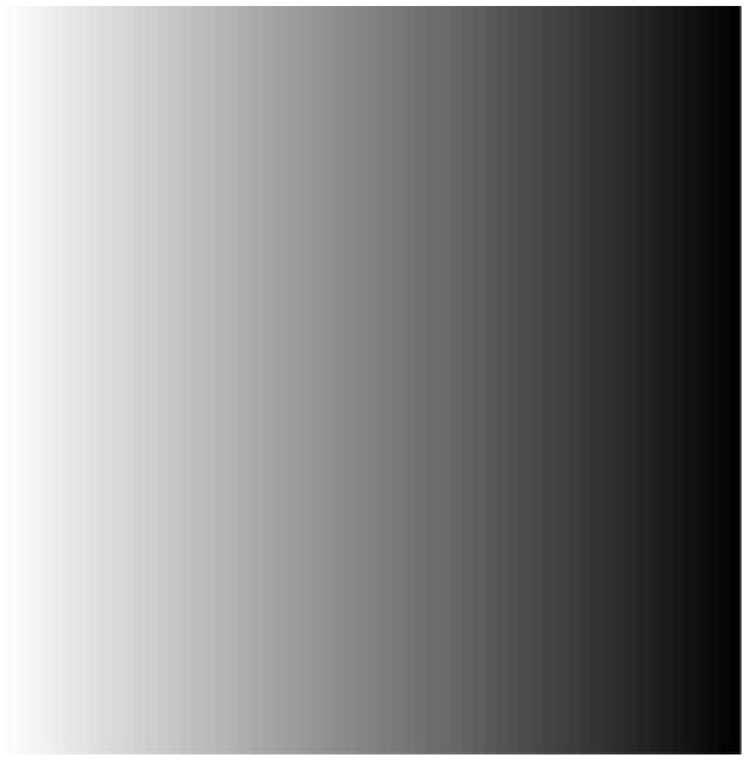


Figure 1.21: Continuously varying intensities



Exercises

1. Watch the TV news, and see if you can observe any examples of image processing.
2. If your TV set allows it, turn down the color as far as you can to produce a monochromatic display. How does this affect your viewing? Is there anything that is hard to recognize without color?
3. Look through a collection of old photographs. How can they be enhanced or restored?
4. For each of the following, list five ways in which image processing could be used:

    - Medicine

- Astronomy
- Sport
- Music
- Agriculture
- Travel

5. Image processing techniques have become a vital part of the modern movie production process. Next time you watch a film, take note of all the image processing involved.
6. If you have access to a scanner, scan in a photograph, and experiment with all the possible scanner settings.

   (a) What is the smallest sized file you can create which shows all the detail of your photograph?
   (b) What is the smallest sized file you can create in which the major parts of your image are still recognizable?
   (c) How do the color settings affect the output?

7. If you have access to a digital camera, again photograph a fixed scene, using all possible camera settings.

   (a) What is the smallest file you can create?
   (b) How do the light settings affect the output?

8. Suppose you were to scan in a monochromatic photograph, and then print out the result. Then suppose you scanned in the printout, and printed out the result of that, and repeated this a few times. Would you expect any degradation of the image during this process? What aspects of the scanner and printer would minimize degradation?
9. Look up *ultrasonography*. How does it differ from the image acquisition methods discussed in this chapter? What is it used for? If you can, compare an ultrasound image with an x-ray image. How to they differ? In what ways are they similar?
10. If you have access to an image viewing program (other than MATLAB, Octave or Python) on your computer, make a list of the image processing capabilities it offers. Can you find imaging tasks it is unable to do?

---

2 Images Files and File Types

We shall see that matrices can be handled very efficiently in MATLAB, Octave and Python. Images may be considered as matrices whose elements are the pixel values of the image. In this chapter we shall investigate how the matrix capabilities of each system allow us to investigate images and their properties.

2.1 Opening and Viewing Grayscale Images

Suppose you are sitting at your computer and have started your system. You will have a prompt of some sort, and in it you can type:

```
>> w = imread('wombats.png');
```

MATLAB/Octave