CHAPTER 4

THE IMAGE PROCESSING TOOLBOX AT A GLANCE

WHAT WILL WE LEARN?

- How do I read an image from a file using MATLAB?
- What are the main data classes used for image representation and how can I convert from one to another?
- Why is it important to understand the data class and range of pixel values of images stored in memory?
- How do I display an image using MATLAB?
- How can I explore the pixel contents of an image?
- How do I save an image to a file using MATLAB?

```
imfinfo('pout.tif');
```

The resulting structure (stored in variable ans) will contain the following information:[1]

```
        Filename: '/.../pout.tif'
    FileModDate: '04-Dec-2000 13:57:50'
        FileSize: 69004
          Format: 'tif'
```

4.2.3 Data Classes and Data Conversions

The IPT ability to read images of any type, store their contents into arrays, and make them available for further processing and display does not preclude the need to understand how the image contents are represented in memory. This section follows up on our discussion in Section 3.2.2 and explains the issue of data classes, data conversions, and when they may be needed.

The most common data classes for images are as follows:

- `uint8`: 1 byte per pixel, in the [0, 255] range
- `double`: 8 bytes per pixel, usually in the [0.0, 1.0] range
- `logical`: 1 byte per pixel, representing its value as *true* (1 or white) or *false* (0 or black)

4.2.4 Displaying the Contents of an Image

MATLAB has several functions for displaying images:

- image: displays an image using the current color map.5
- imagesc: scales image data to the full range of the current color map and displays the image.
- imshow: displays an image and contains a number of optimizations and optional parameters for property settings associated with image display.
- imtool: displays an image and contains a number of associated tools that can be used to explore the image contents.

```
I = imread('pout.tif');
imshow(I)
figure, imshow(I,[])
figure, imshow(I,[100 160])
```

(a)

(b)

(c)

imtool function
which provides all the image display capabilities of imshow as well as access to other tools for navigating and exploring images, such as the ***Pixel Region*** tool (Figure 4.2), ***Image Information*** **tool** (Figure 4.3), and ***Adjust Contrast*** **tool** (Figure 4.4). These tools can also be directly accessed using their library functions **impixelinfo, imageinfo, and imcontrast, respectively.**


Image Tool 1 - I
File Tools Window Help
100%
Pixel Info: (X, Y) Intensity
Display range: [0 255]
Overview (Image...
File Edit Window Help
Pixel Region (Image Tool 1)
File Edit Window Help
Pixel Info: (X, Y) Intensity

- impixel: returns the red, green, and blue color values of image pixels specified with the mouse.
- imdistline: creates a *Distance* tool—a draggable, resizable line that measures the distance between its endpoints. You can use the mouse to move and resize the line to measure the distance between any two points within an image (Figure 4.5).

```
I = imread('peppers.png');
imwrite(I, 'pep75.jpg');
imwrite(I, 'pep05.jpg', 'quality', 5);
imwrite(I, 'pep95.jpg', 'quality', 95);
```

(a)

(b)

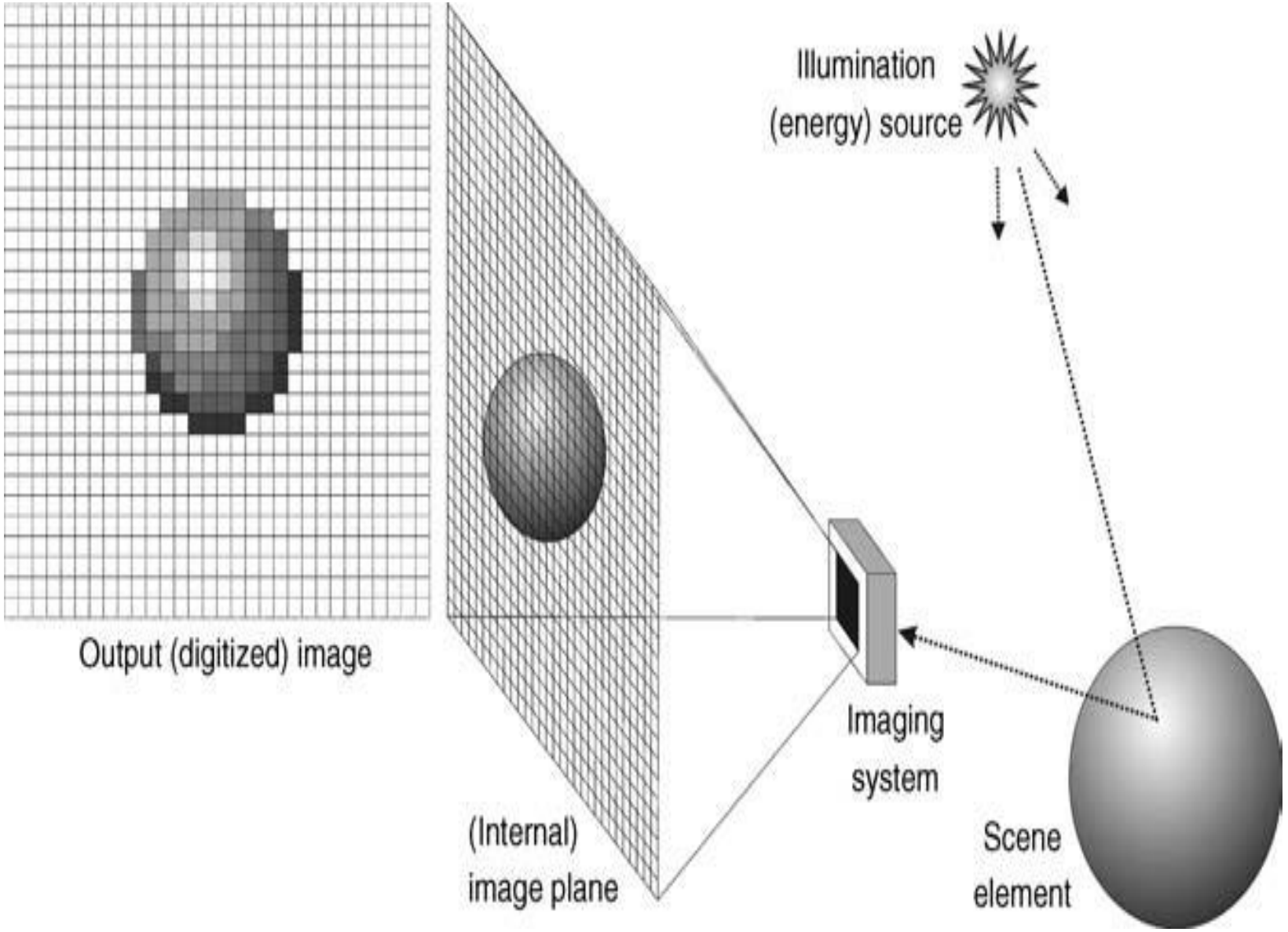
(c)

(d)

CHAPTER 5
IMAGE SENSING AND ACQUISITION

WHAT WILL WE LEARN?

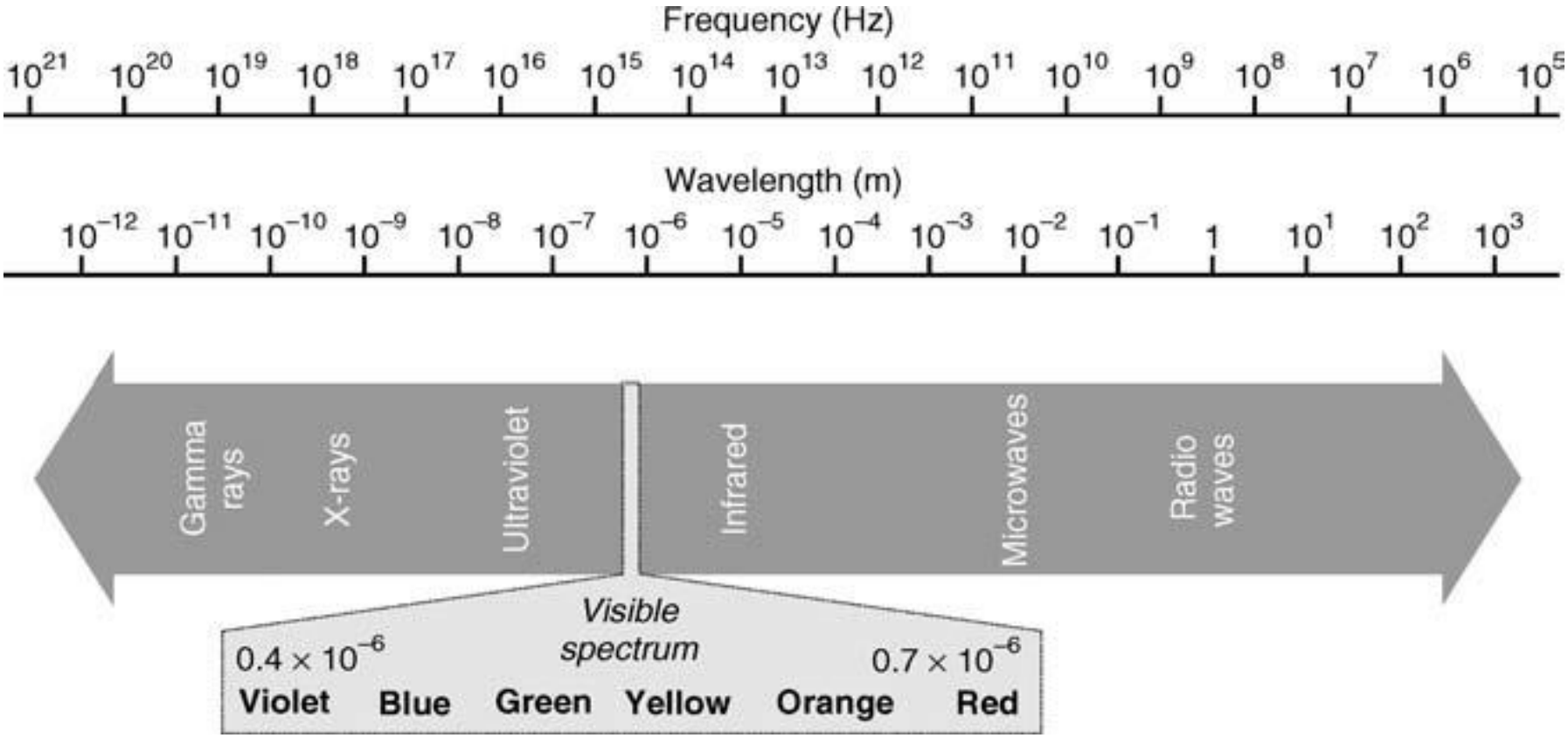
- What are the main parameters involved in the design of an image acquisition solution?
- How do contemporary image sensors work?
- What is image digitization and what are the main parameters that impact the digitization of an image or video clip?
- What is sampling?
- What is quantization?
- How can I use MATLAB to resample or requantize an image?


Illumination
(energy) source
Output (digitized) image
Imaging
system
(Internal)
image plane
Scene
element

5.2.1 Light and Electromagnetic Spectrum

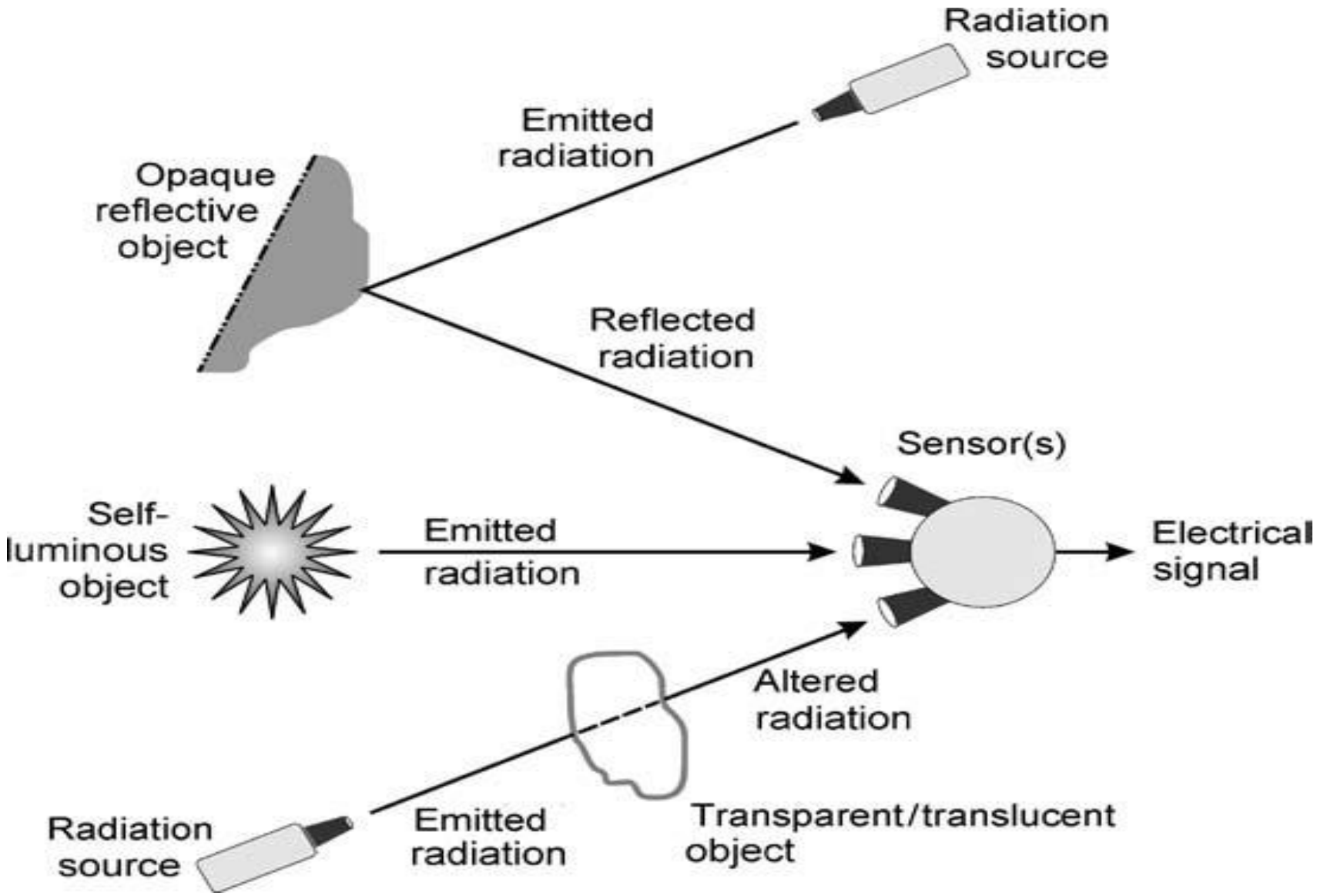
Light can be described in terms of electromagnetic waves or particles, called *photons*.

A photon is a tiny packet of vibrating electromagnetic energy that can be characterized by its wavelength or frequency.



The human visual system (HVS) is sensitive to photons of wavelengths between 400 and 700 nm, where 1 nm = 10^{-9} m. As shown in Figure 5.2, this is a fairly narrow slice within the EM spectrum, which ranges from *radio waves* (wavelengths of 1m or longer) at one end to *gamma rays* (wavelengths of 0.01 nm or shorter) at the other end.

5.2.2 Types of Images



• ***Reflection Images*:** These are the result of radiation that has been reflected from the surfaces of objects. The radiation may be *ambient* or *artificial*. Most of the images we perceive in our daily experiences are reflection images. The type of information that can be extracted from reflection images is primarily about surfaces of objects, for example, their shapes, colors, and textures.

• ***Emission Images*:** These are the result of objects that are self-luminous, such as stars and light bulbs (both within the visible light range), and—beyond visible light range—thermal and infrared images.

• ***Absorption Images*:** These are the result of radiation that passes through an object and results in an image that provides information about the object's internal structure. The most common example is X-ray image.

5.2.3 Light and Color Perception

Light is a particular type of EM radiation that can be sensed by the human eye.

Colors perceived by humans are determined by the nature of the light reflected by the object, which is a function of the spectral properties of the light source as well as the absorption and reflectance properties of the object.

The radiance (physical power) of a light source is expressed in terms of its **spectral power distribution** (SPD). Figure 5.5 shows examples of SPDs of physical light sources commonly found in imaging systems: **sunlight, tungsten lamp, light-emitting diode (LED), mercury arc lamp, and helium–neon laser.** The human perception of each of these light sources will vary—from the yellowish nature of light produced by tungsten light bulbs to the extremely bright and pure red laser beam.

• ***Brightness*****:** The *subjective* perception of (achromatic) luminous intensity, or “the attribute of a visual sensation according to which an area appears to emit more or less light” [Poy03].

• ***Hue*****:** “The attribute of a visual sensation according to which an area appears to be similar to one of the perceived colors, red, yellow, green and blue, or a combination of two of them” [Poy03]. From a spectral viewpoint, hue can be associated with the dominant wavelength of an SPD.

• ***Saturation*****:** “The colorfulness of an area judged in proportion to its brightness” [Poy03], which usually translates into a description of the whiteness of the light source. From a spectral viewpoint, the more an SPD is concentrated at one wavelength, the more saturated will be the associated color. The addition of white light, that is, light that contains power at all wavelengths, causes color desaturation.

5.3 IMAGE ACQUISITION

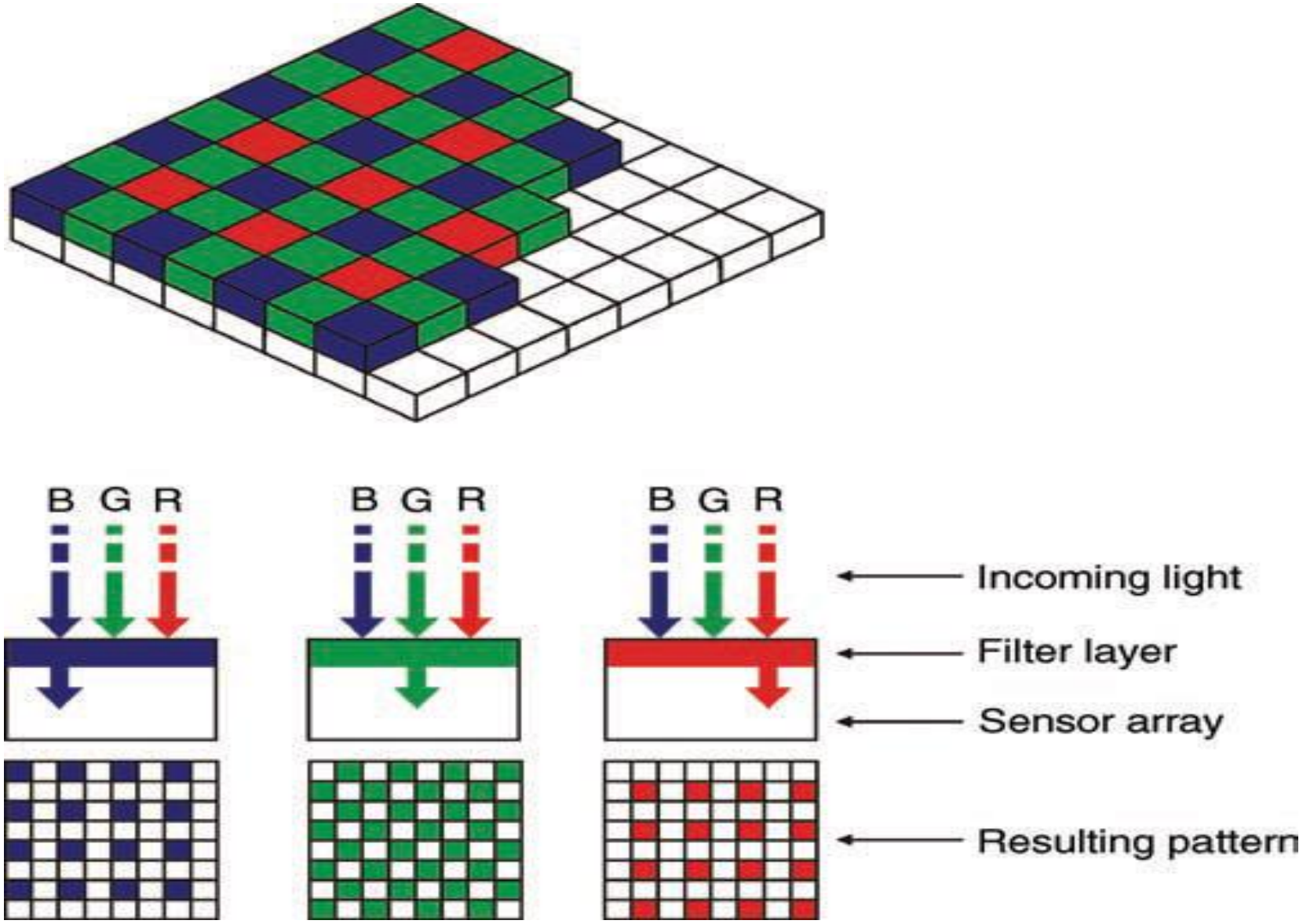
The main goal of an image sensor is to convert EM energy into electrical signals that can be processed, displayed, and interpreted as images.

Two of the most popular and relatively inexpensive devices used for image acquisition are the **digital camera and the flatbed scanner**. Cameras typically use 2D (area) CCD sensors, whereas scanners employ 1D (line) CCDs that move across the image as each row is scanned.

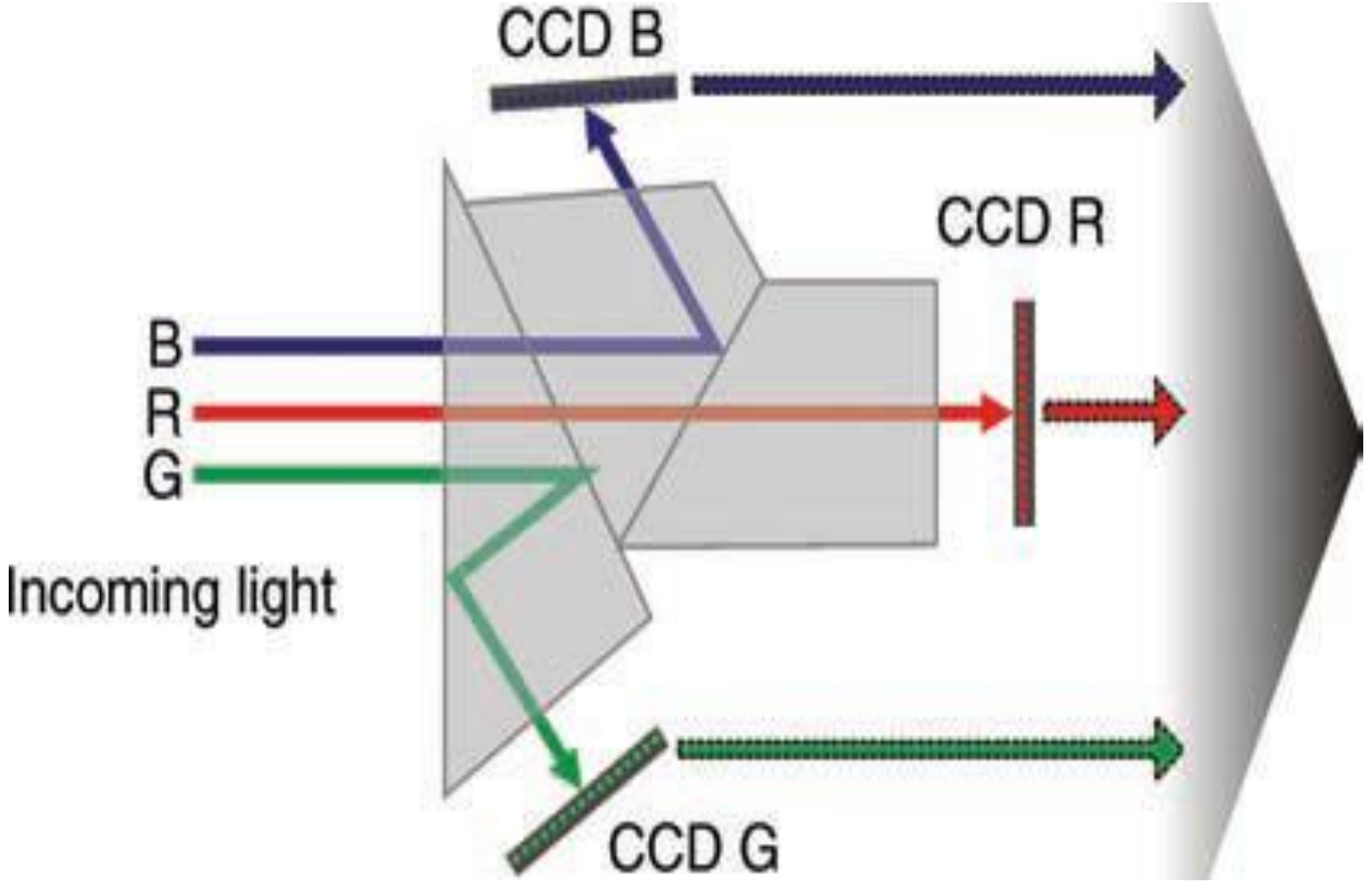
A CCD sensor is made up of an array of light-sensitive cells called ***photosites*****,** manufactured in silicon, each of which produces a voltage proportional to the intensity of light falling on them. **A photosite** has a finite capacity of about 106 energy carriers, which imposes an upper bound on the brightness of the objects to be imaged. A saturated photosite can overflow, corrupting its neighbors and causing a defect known as ***blooming***.

The *nominal resolution* of aCCD sensor is the size of the *scene element* that images to *a single pixel* on the image plane. For example, if a 20 cm × 20 cm square sheet of paper is imaged to form a 500 × 500 digital image, then the nominal resolution of the sensor is 0.04 cm.

A CCD camera sometimes plugs into a computer board, called ***frame buffer***, which contains fast access memory (**typically 0.1 ms per image**) for the images captured by the camera. After being captured and temporarily stored in the frame buffer, images can be processed or copied to a long-term storage device

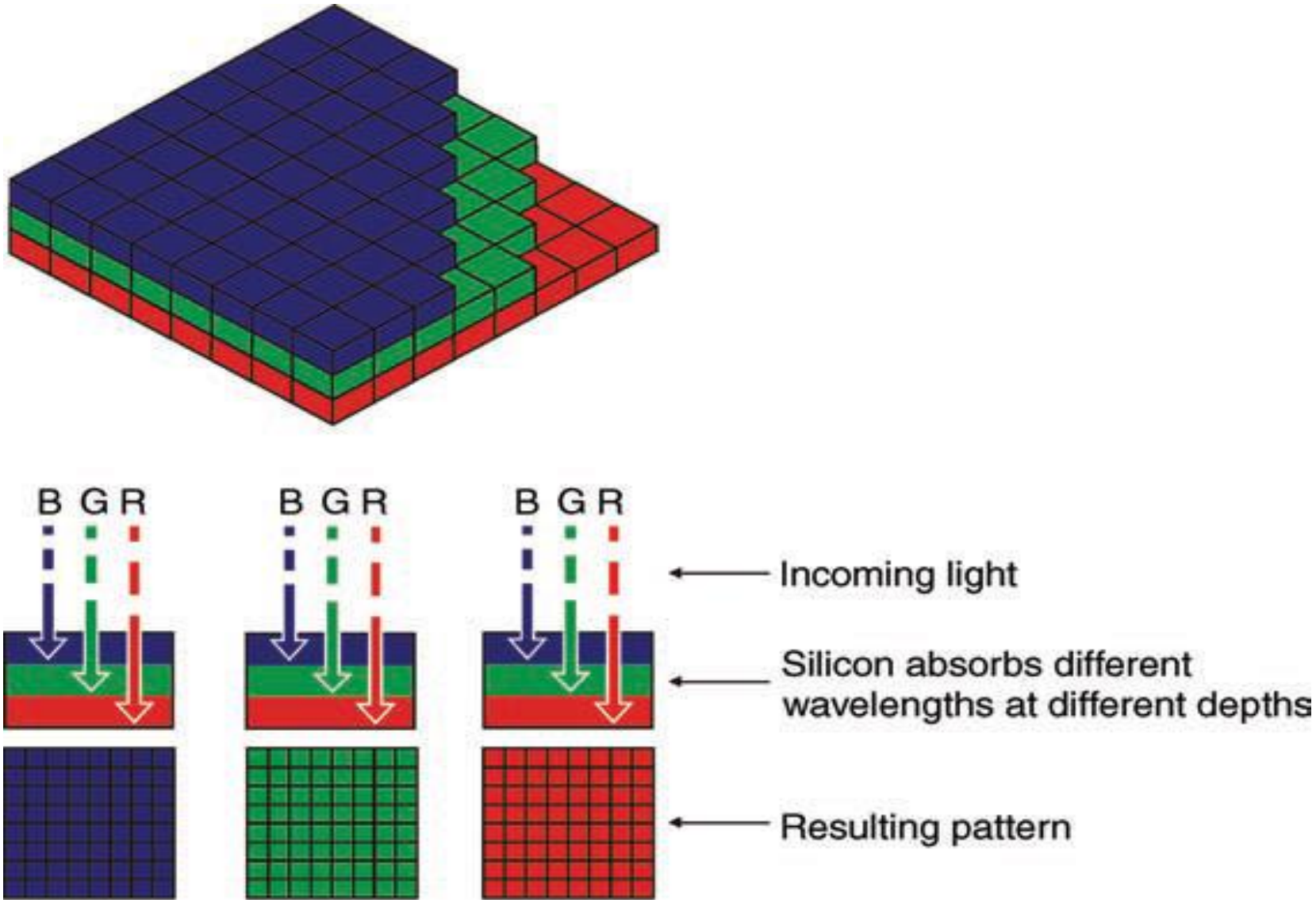


The Bayer pattern for single-CCD cameras

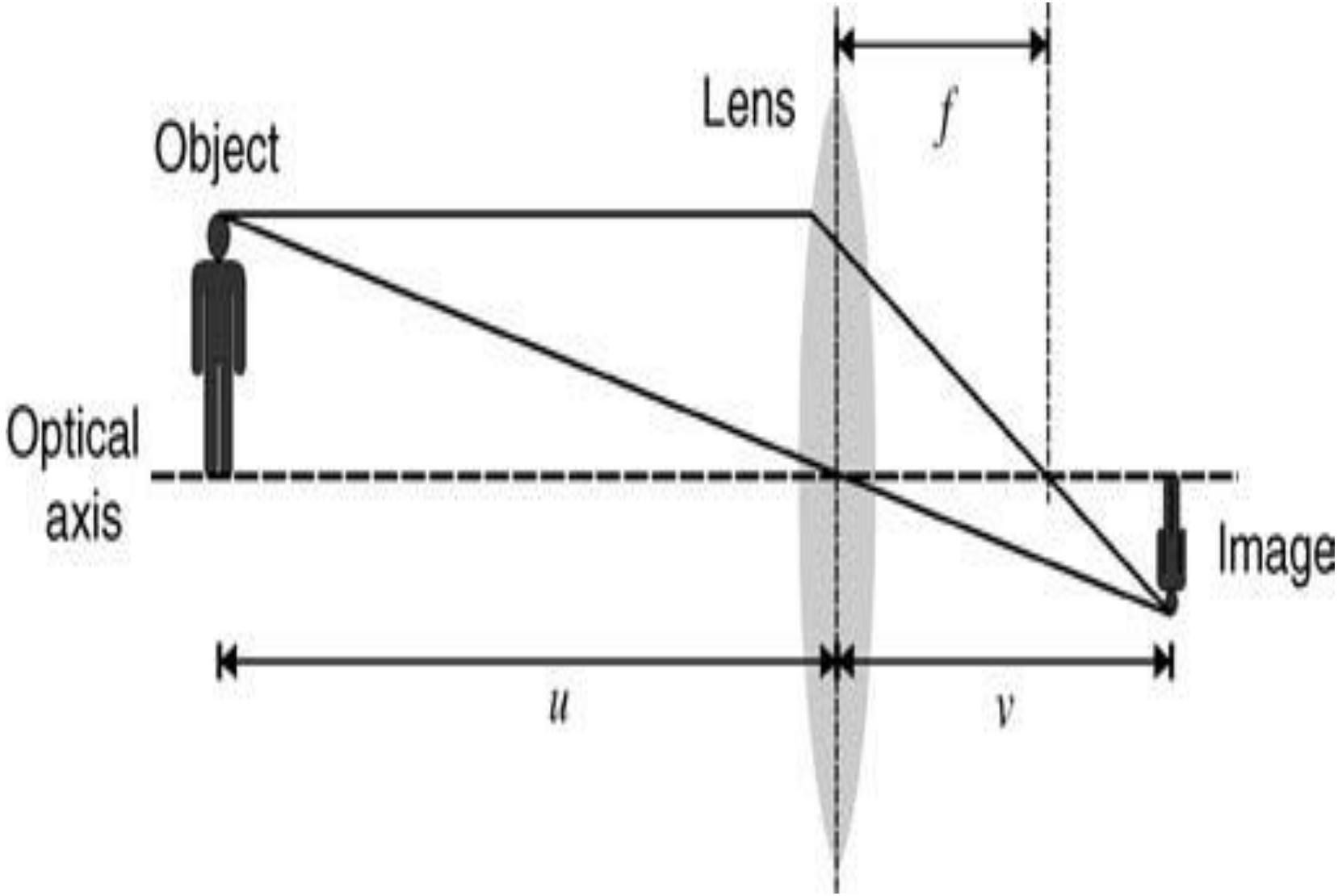


The beam splitter for three-CCD color cameras.

An alternative technology to CCDs is CMOS. CMOS chips have the advantages of being cheaper to produce and requiring less power to operate than comparable CCD chips. Their main disadvantage is the increased susceptibility to noise, which limits their performance at low illumination levels. CMOS sensors were initially used in low-end cameras, such as webcams, but have recently been extended to much more sophisticated cameras, including the Panavision HDMAX 35 mm video camera.



X3 color sensor.



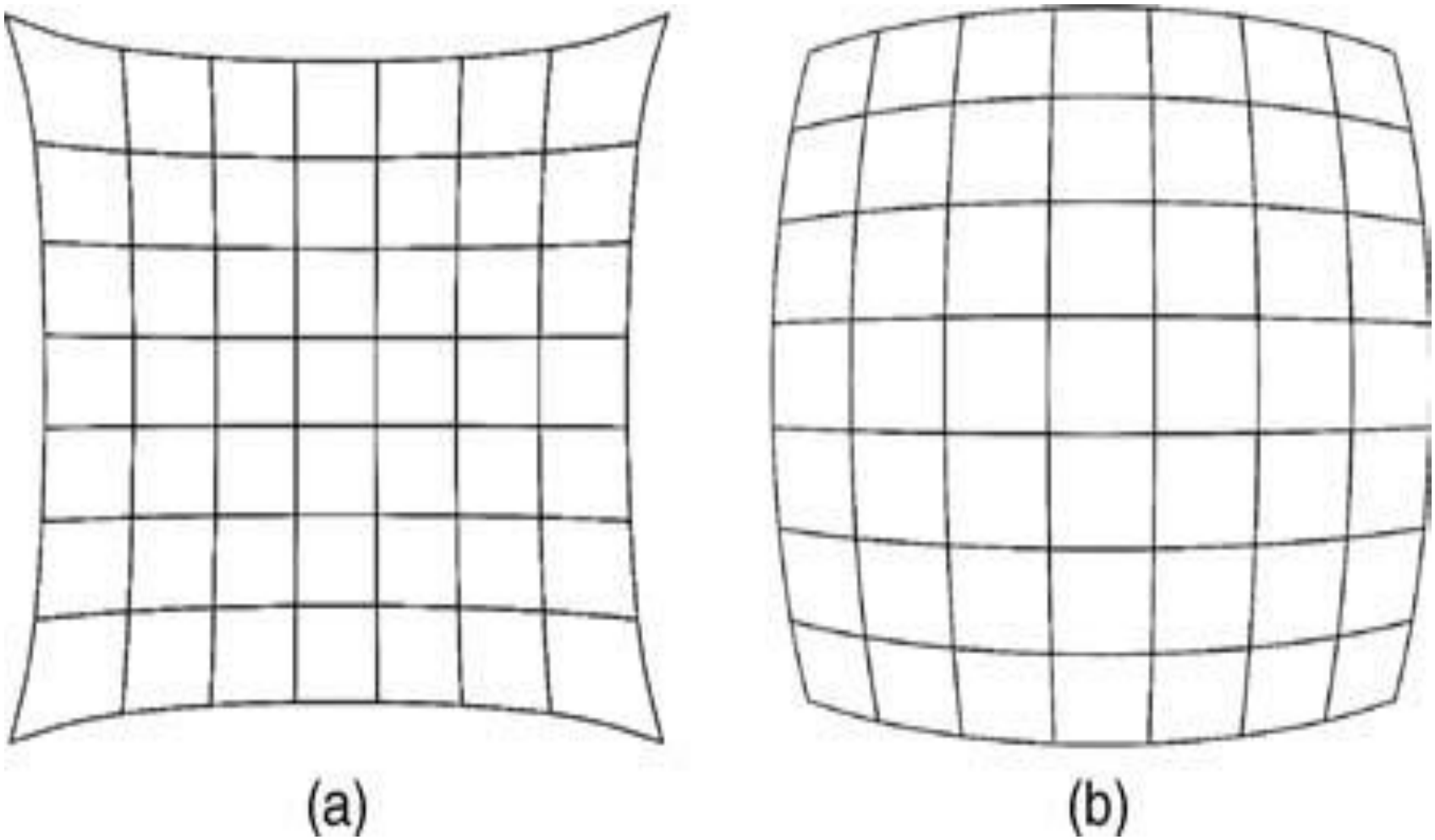
5.3.2 Camera Optics

magnification factor (*m*), which is the ratio between image size and object size:

$$m = \frac{v}{u}$$

$$f = \frac{um}{m+1}$$

The light gathering capacity of a camera lens is determined by its *aperture*, which is often expressed as an "*f* number"—a dimensionless value that represents the ratio between focal length and aperture diameter. Most lenses have a sequence of fixed apertures (e.g., *f* 2.8, *f* 4, *f* 5.6, *f* 8, *f* 11) that progressively reduce the total amount of light reaching the sensor by half.



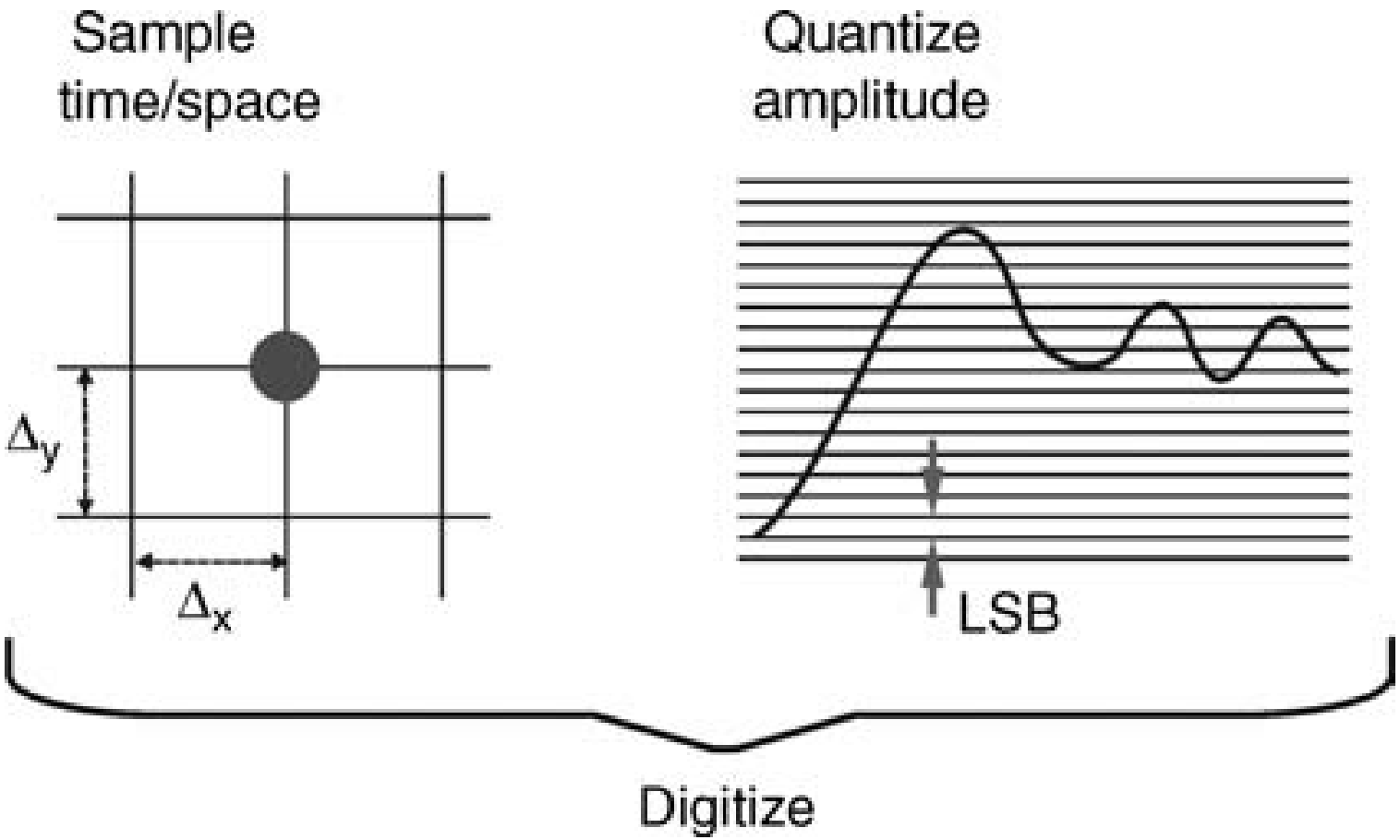
Examples of lens aberrations: (a) pincushion distortion; (b) barrel distortion

5.4 IMAGE DIGITIZATION

Digitization involves two processes: *sampling* (in time or space) and *quantization* (in amplitude). These operations may occur in any sequence, but usually sampling precedes quantization.

Sampling involves selecting a finite number of points within an interval.

quantization implies assigning an amplitude value (within a finite range of possible values) to each of those points.



Digitization = sampling + quantization.
Redrawn from [Poy03].

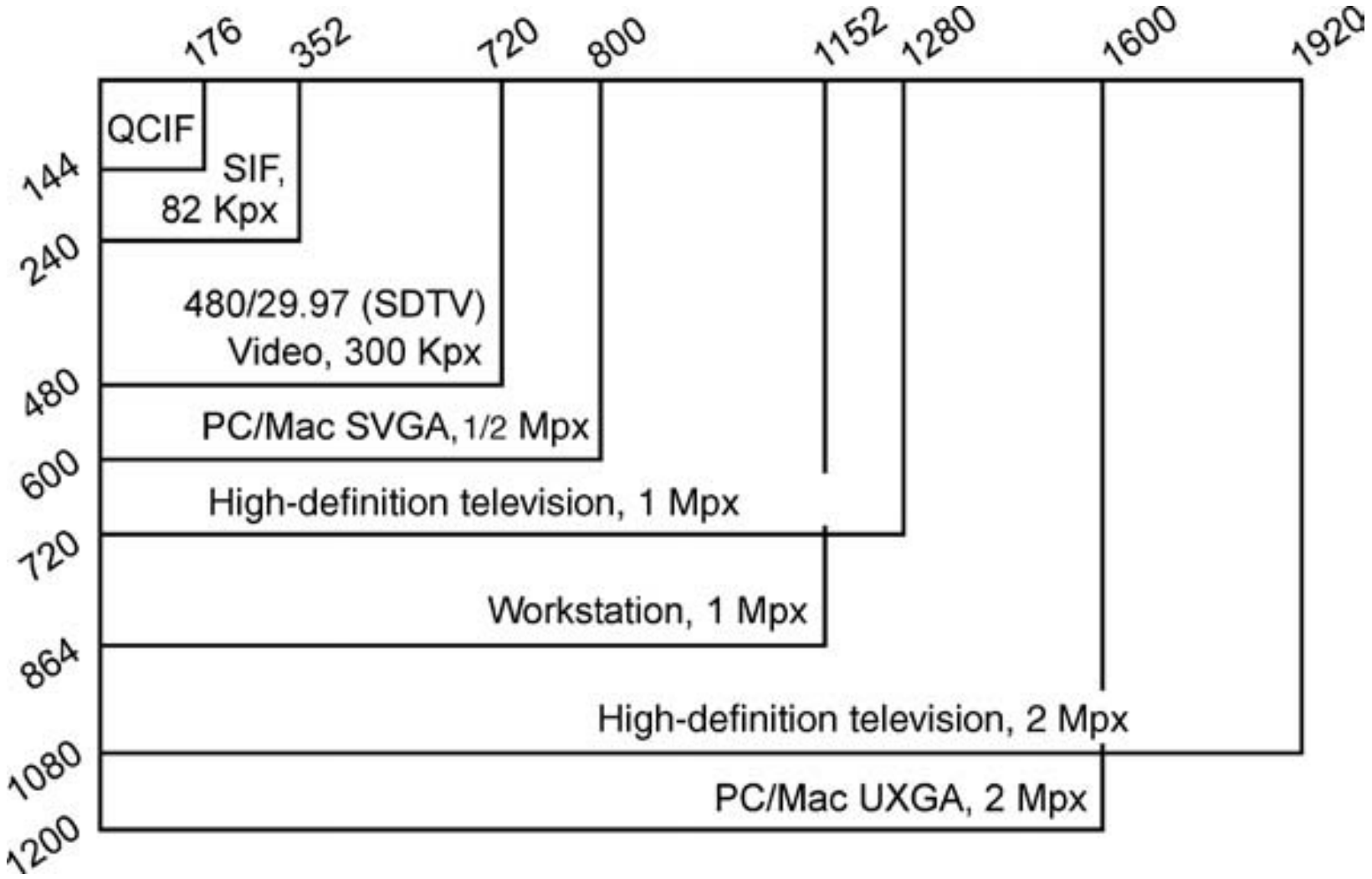
The result of the digitization process is **a *pixel array***, which is a rectangular matrix of picture elements whose values correspond to their intensities (for **monochrome** images) or color components (for **color images**).

5.4.1 Sampling

Sampling is the process of measuring the value of a 2D function at discrete intervals along the *x* and *y* dimensions.

A system that has equal horizontal and vertical sampling densities is said to have ***square sampling***.

Several imaging and video systems use sampling lattices where the horizontal and the vertical sample pitch are unequal, that is, ***nonsquare sampling***.



Pixel arrays of several imaging standards.
Redrawn from [Poy03].

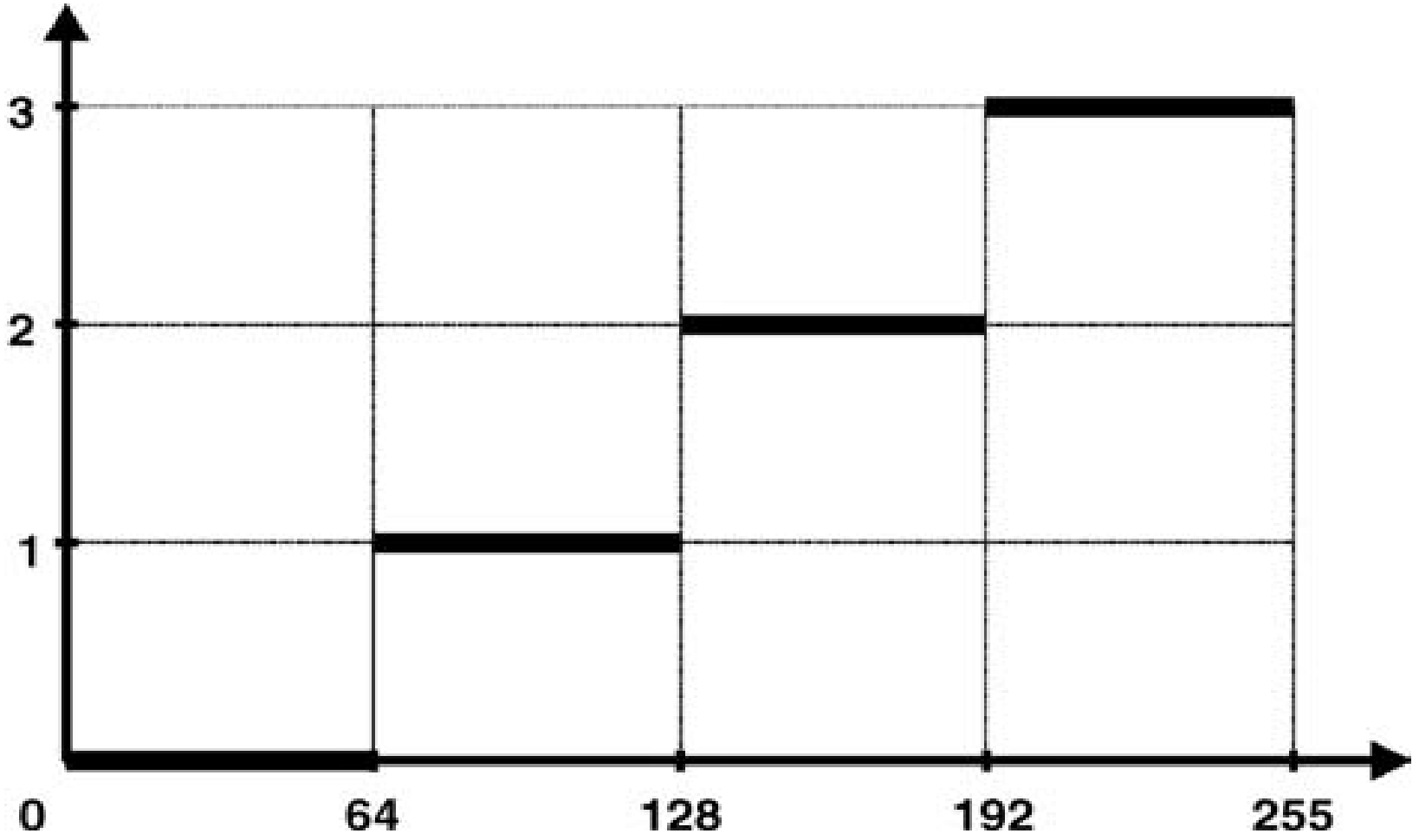
Two parameters must be taken into account when sampling images:

1. **The *sampling rate***, that is, the number of samples across the height and width of the image. The choice of an appropriate sampling rate will impact image qualityknown as *aliasing*, which will be discussed later.

2. **The *sampling pattern*,** that is, the physical arrangement of the samples. A rectangular pattern, in which pixels are aligned horizontally and vertically into rows and columns, is by far the most common form, but other arrangements are possible, for example, the *hexagonal* and *log-polar* sampling patterns.

5.4.2 Quantization

Quantization is the process of replacing a continuously varying function with a discrete set of quantization levels. In the case of images, the function is $f(x, y)$ and the quantization levels are also known as *gray levels*. It is common to adopt N quantization levels for image digitization, where N is usually an integral power of 2 that is, $N = 2n$, where n is the number of bits needed to encode each pixel value. The case where $n = 2^8 = 256$ produces images where each pixel is represented by an unsigned byte, with values ranging from 0 (black) to 255 (white).



A mapping function for uniform quantization ($N = 4$).

CHAPTER 6

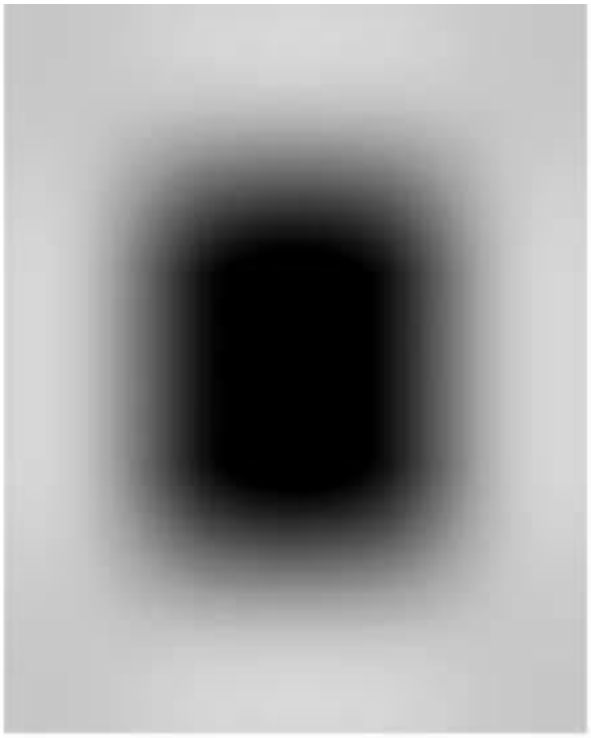
ARITHMETIC AND LOGIC OPERATIONS

6.1 ARITHMETIC OPERATIONS: FUNDAMENTALS AND APPLICATIONS

$$X \quad opn \quad Y = \quad Z$$

where *opn* is a binary arithmetic $(+, -, \times, /)$ operator.

(a)

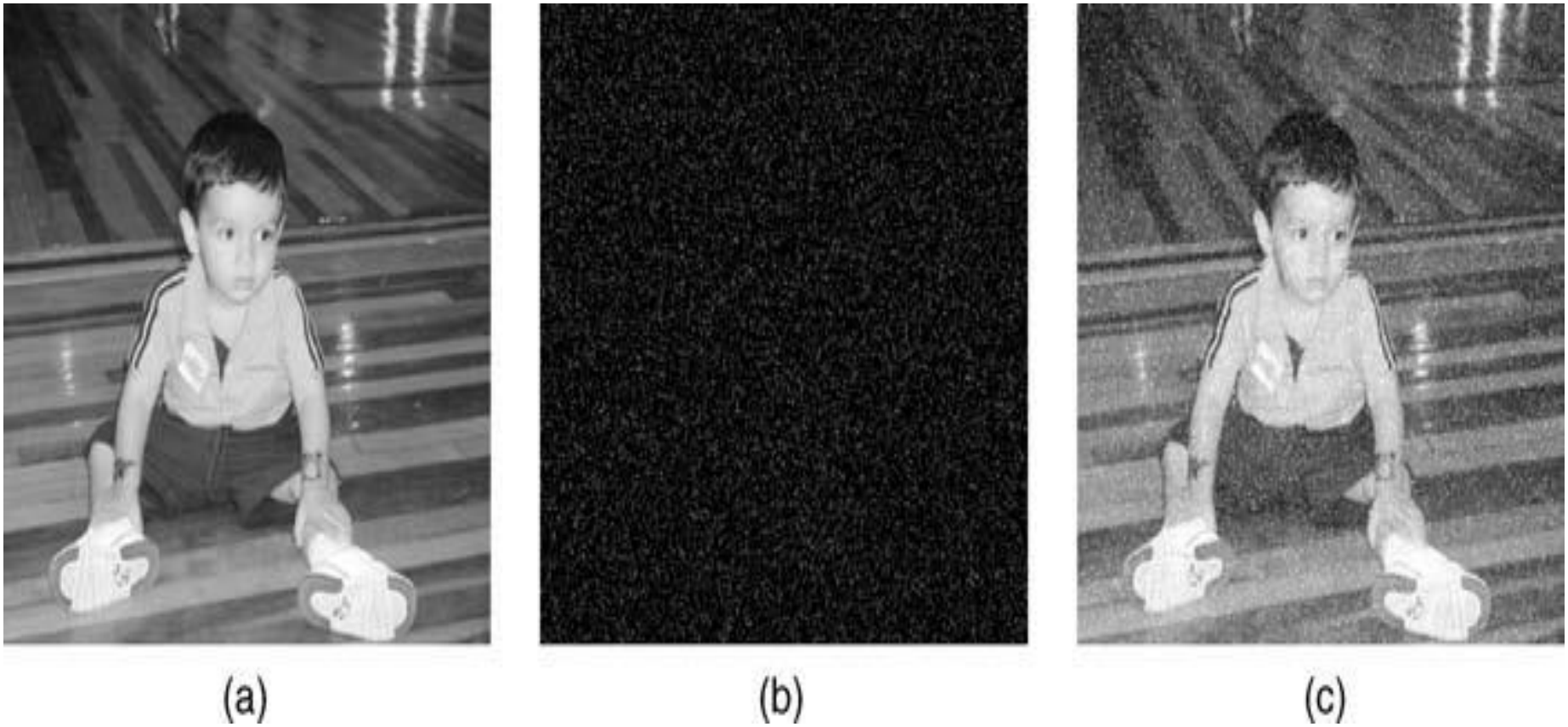


(b)

(c)

Adding two images: (a) first image (X); (b) second image (Y); (c) result ($Z = X + Y$).

Additive image offset: (a) original image (X); (b) brighter version ($Z = X + ۷۵$



Adding noise to an image: (a) original image (X); (b) zero-mean Gaussian white noise (variance = 0.01) (N); (c) result ($Z = X + N$).

When adding two images, you must be careful with values that exceed the maximum pixel value for the data type being used. There are two ways of dealing with this *overflow* issue: ***normalization*** **and** ***truncation***. Normalization consists in storing the intermediate result in a temporary variable (*W*) and calculating each resulting pixel value in *Z* using equation (6.2).

$$g = \frac{L_{\max}}{f_{\max} - f_{\min}}(f - f_{\min})$$

where *f* is the current pixel in *W*, L_{max} is the maximum possible intensity value (e.g., 255 for uint8 or 1.0 for double), *g* is the corresponding pixel in *Z*, f_{max} is the maximum pixel value in *W*, and f_{min} is the minimum pixel value in *W*.

Truncation consists in simply limiting the results to the maximum positive number that can be represented with the adopted data type.

6.1.2 Subtraction

(a) (b)

Subtractive image offset: (a) original image (X); (b) darker version ($Z = X - 75$).

When subtracting one image from another or a constant (scalar) from an image, you must be careful with the possibility of obtaining negative pixel values as a result.

There are two ways of dealing with this ***underflow*** issue: treating subtraction as absolute difference (which will always result in positive values proportional to the difference between the two original images without indicating, however, which pixel was brighter or darker) and **truncating** the result, so that negative intermediate values become zero.

Image subtraction can also be used to obtain the *negative* of an image (Figure 6.5):

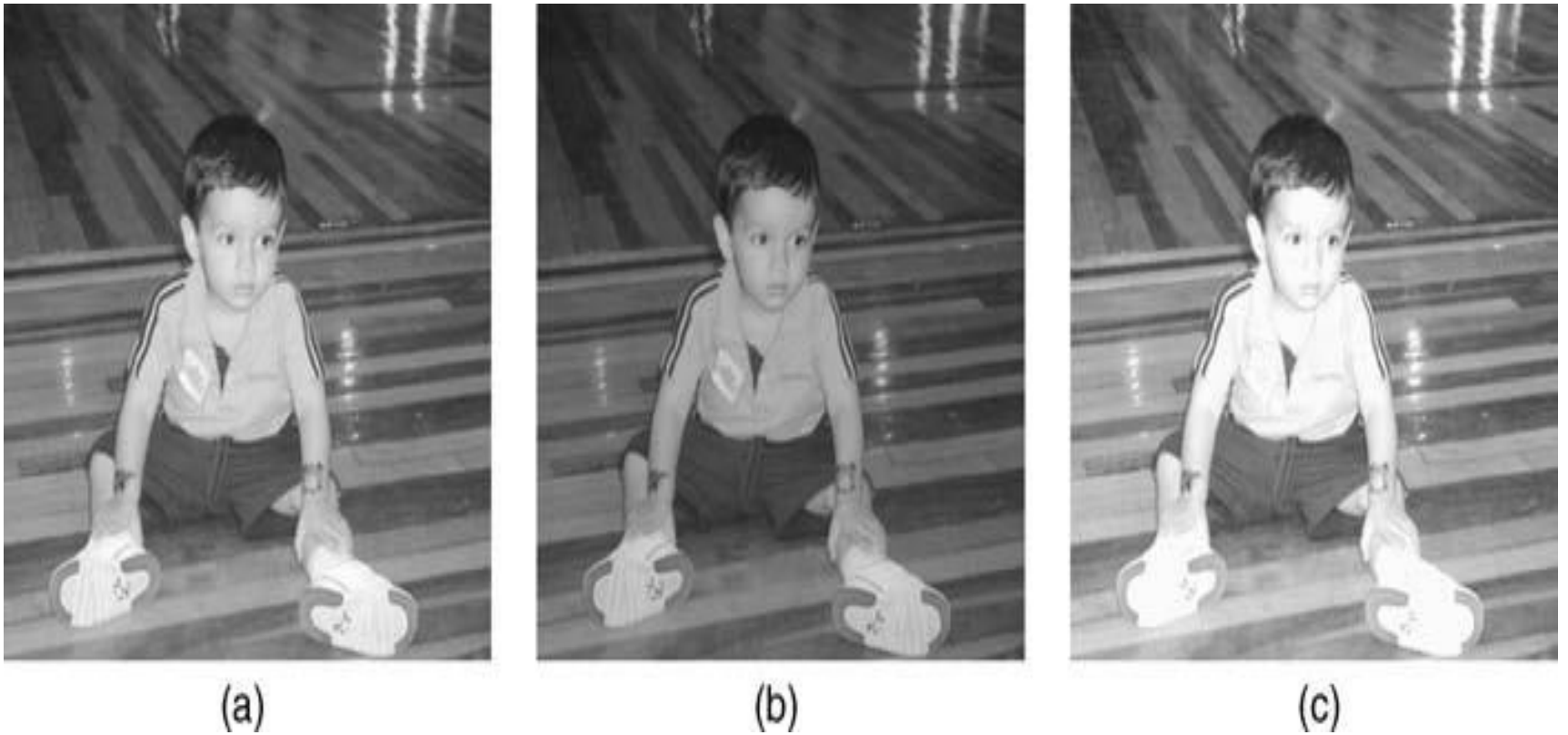
$$g = -f + L_{max} \qquad (6.3)$$

where L_{max} is the maximum possible intensity value (e.g., 255 for uint8 or 1.0 for double), f is the pixel value in X, g is the corresponding pixel in Z.

Example of an image negative: (a) original image; (b) negative image.

6.1.3 Multiplication and Division

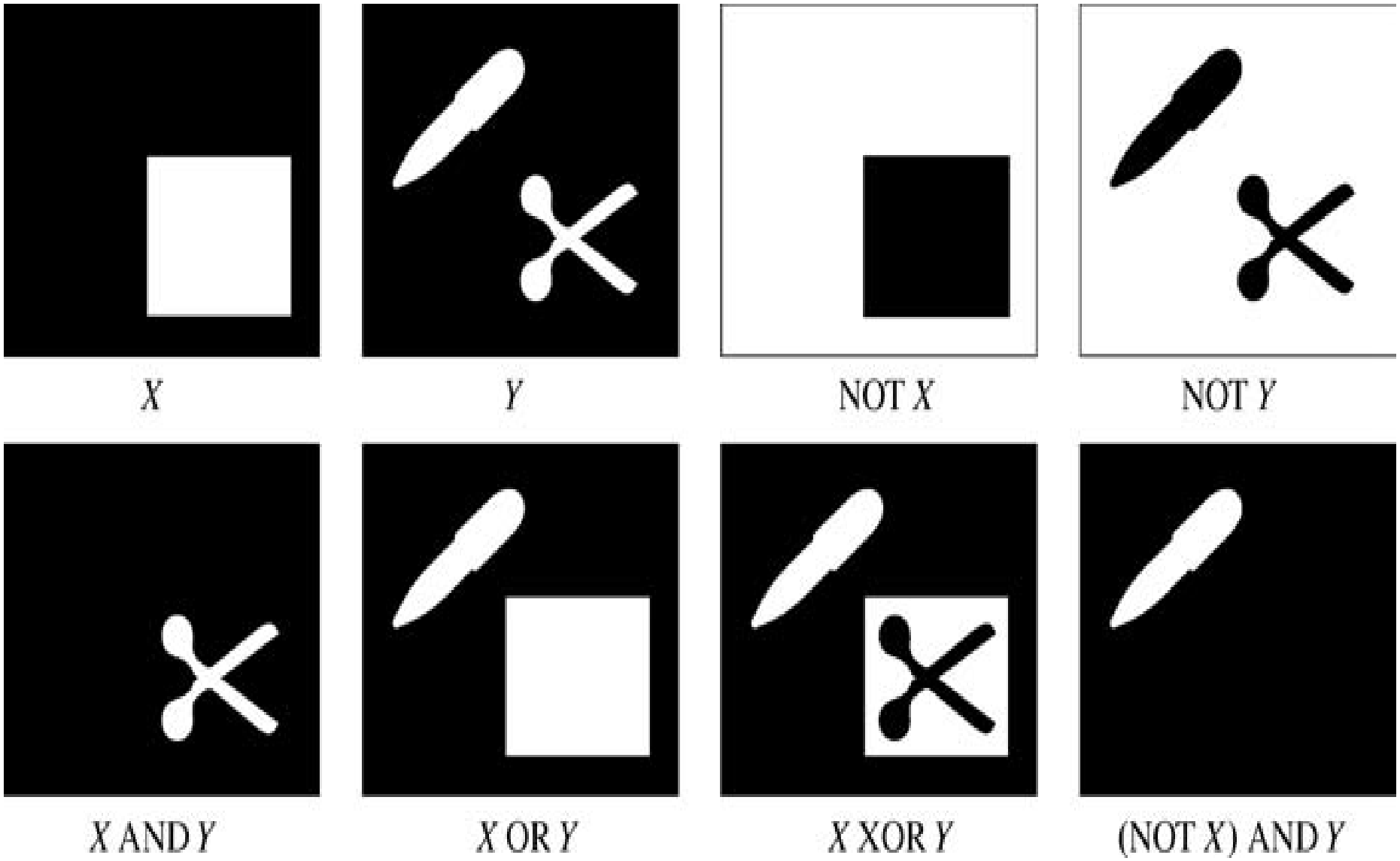
Multiplication and division by a scalar are often used to perform brightness adjustments on an image. This process—sometimes referred to as *multiplicative image scaling*—makes each pixel value brighter (or darker) by multiplying its original value by a scalar factor: if the value of the scalar multiplication factor is greater than one, the result is a brighter image; if it is greater than zero and less than one, it results in a darker image (Figure 6.6). Multiplicative image scaling usually produces better subjective results than the additive image offset process described previously.



Multiplication and division by a constant: (a) original image (*X*); (b) multiplication result (*X* × 0.7); (c) division result (*X*/0.7).

6.2 LOGIC OPERATIONS: FUNDAMENTALS AND APPLICATIONS

The AND, XOR, and OR operators require two or more arguments,
whereas the NOT operator requires only one argument.



Logic operations on binary images.

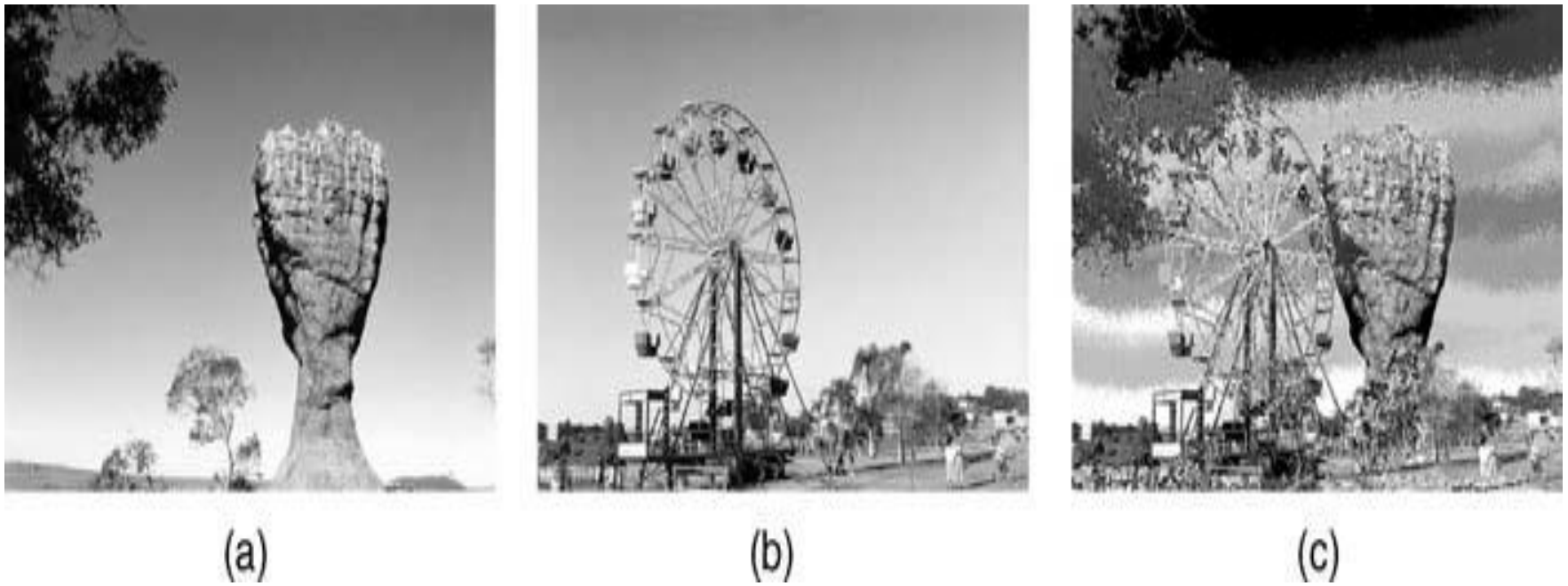
AND, OR, XOR, and NOT operations on monochrome images.

- The AND and OR operations can be used to combine images for special effects purposes. They are also used in masking operations, whose goal is to extract a region of interest (ROI) from an image (see Tutorial 6.2).

The AND operation applied to monochrome images: (a) *X*; (b) *Y*; (c) *X* AND *Y*.

The OR operation applied to monochrome images: (a) *X*; (b) *Y*; (c) *X* OR *Y*.

- The XOR operation is often used to highlight differences between two monochrome images. It is, therefore, equivalent to calculating the absolute difference between two images.

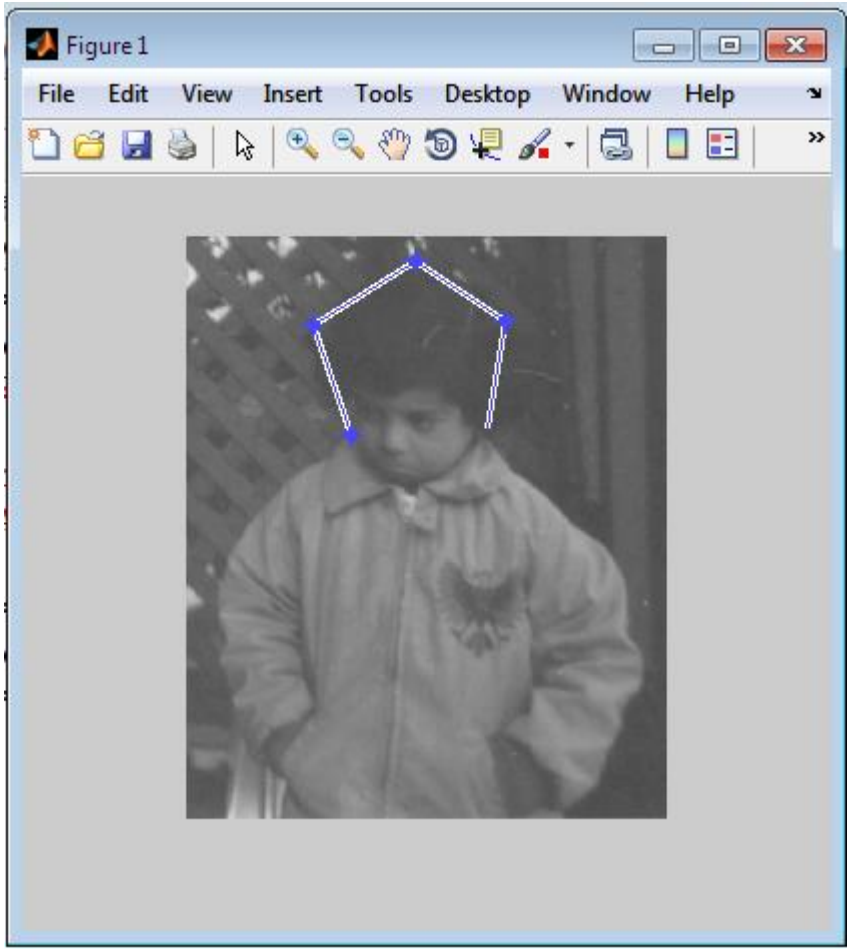


The XOR operation applied to monochrome images: (a) *X*; (b) *Y*; (c) *X* XOR *Y*.

- The NOT operation extracts the binary complement of each pixel value, which is equivalent to applying the “negative” effect on an image.

The NOT operation applied to a monochrome image: (a) *X*; (b) NOT *X*.

```
I = imread('pout.tif');
bw = roipoly(I);
```



Question 1 How do we add points to the polygon?
Question 2 How do we delete points from the polygon?
Question 3 How do we end the process of creating a polygon?