Universidad Veracruzana

Digital Image Fundamentals

Dr. Héctor Gabriel Acosta Mesa

Instituto de Investigaciones en Inteligencia Artificial
Maestría en Inteligencia Artificial
heacosta@uv.mx
www.uv.mx/heacosta

Image Analysis

Preview

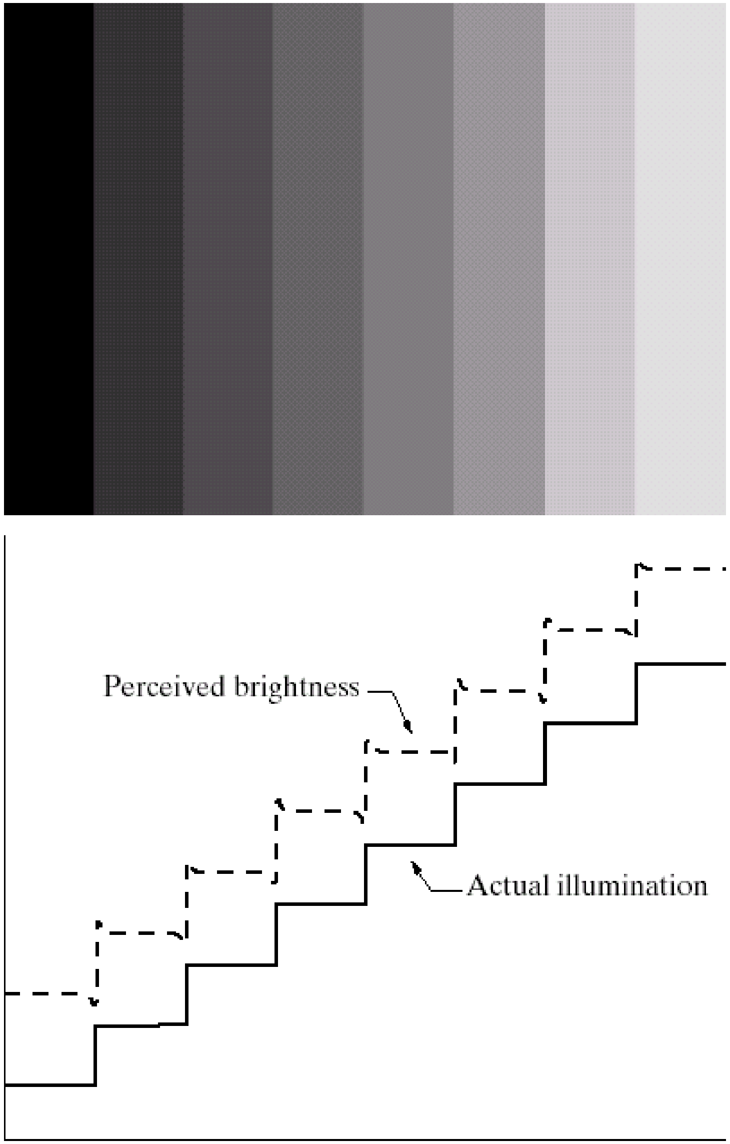
- Developing a basic understanding of human visual perception as a first step in our journey through this field.

- In particular, our interest lies in the mechanics and parameters related to how images are formed in the eye.

- We are interested in learning the physical limitations of human vision in terms of factors that also are used in our work with digital images.

- Consult file I-B .

Brightness adaptation and discrimination



Perceived brightness is not a simple function of intensity. The figure 2.7 shows an example of this phenomenon: *mach bands.*

Why?

a

b

Figure 2.7 (a) An example showing that perceived brightness is not a simple function of intensity. The relative vertical positions between the two profiles in (b) have no special significance: they were chosen for clarity.

Brightness adaptation and discrimination

Because digital images are displayed as a discrete set of intensities, the eye´s ability to discriminate between different intensity levels is an important consideration in presenting image-processing results.

Subjective brightness is a logarithmic function of the light intensity incident on the eye. Figure 2.4 illustrates that the visual system cannot operate over such a range *simultaneously*. Changes in its overall sensitivity, known as *brightness adaptation level.*

Brightness adaptation and discrimination

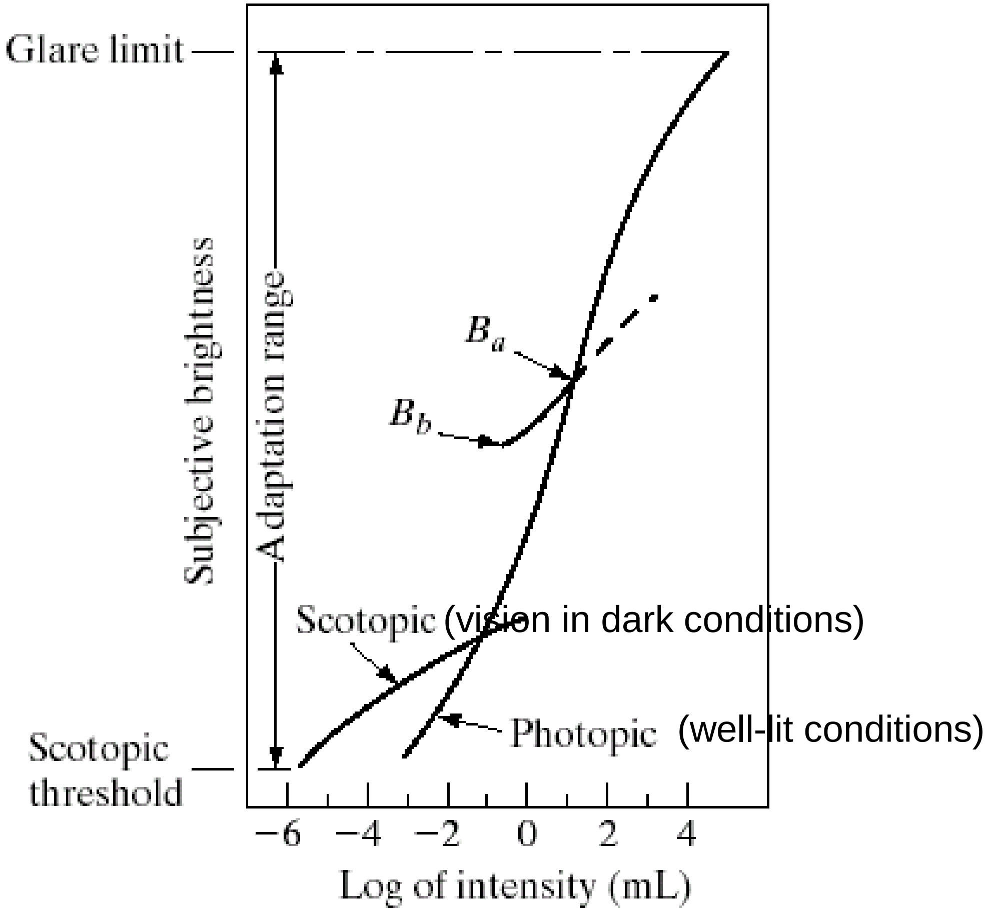


Figure 2.4 Range of subjective brightness sensations showing a particular adaptation level

Brightness adaptation and discrimination

Figure 2.5 shows a classic experiment. The quantity $\Delta I_c/I$ is called the *Weber ratio*. A small value of $\Delta I_c/I$, means that a small percentage change in intensity is discriminable. This represents "good" brightness discrimination.

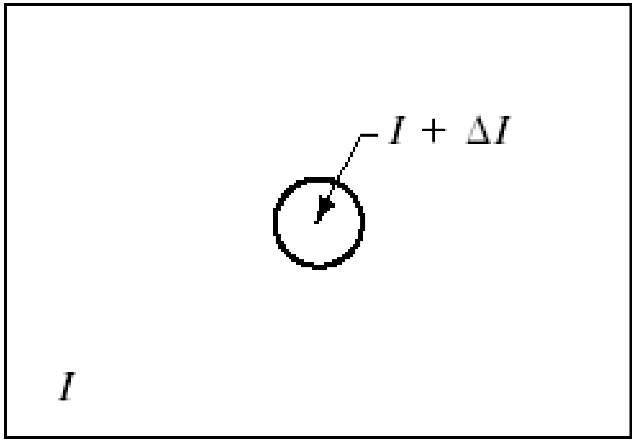
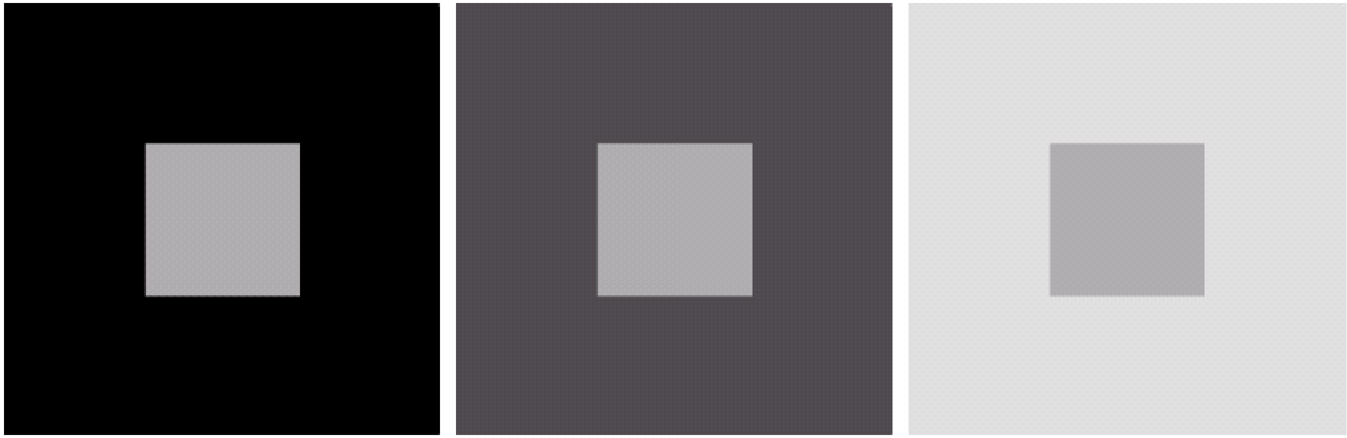


Figure 2.5 Basic experimental setup used to characterize brightness discrimination

Brightness adaptation and discrimination

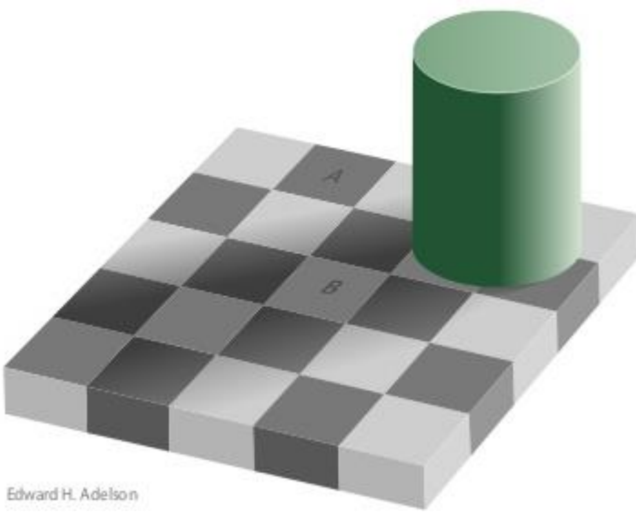
Figure 2.8 shows the phenomenon called *simultaneous contrast*

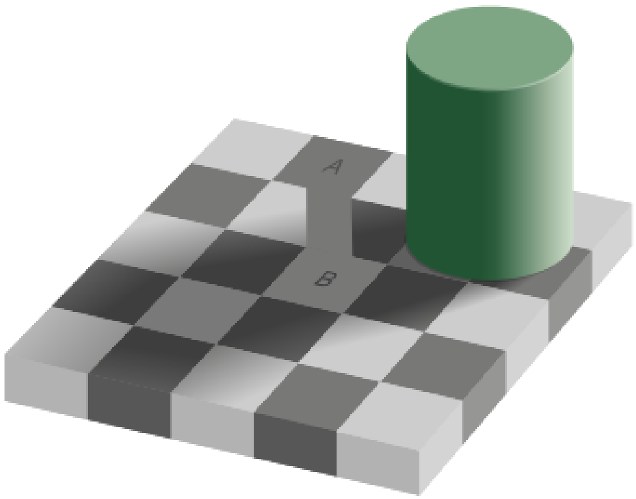


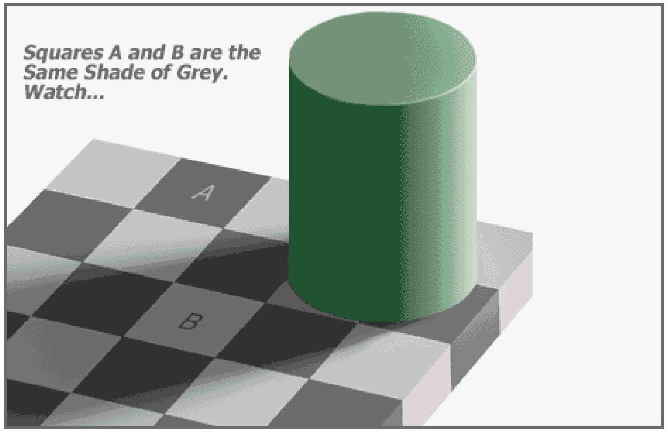
a b c

Figure 2.8 Examples of simultaneous contrast. All the inner squares have the same intensity, but they appear progressively darker as the background becomes lighter.

Checkershadow







https://www.blogodisea.com/ilusiones-opticas-checkershadow.html

Light and the electromagnetic spectrum

As shown in figure 2.10, the range of colors we perceive in visible light represents a very small portion of the electromagnetic spectrum.

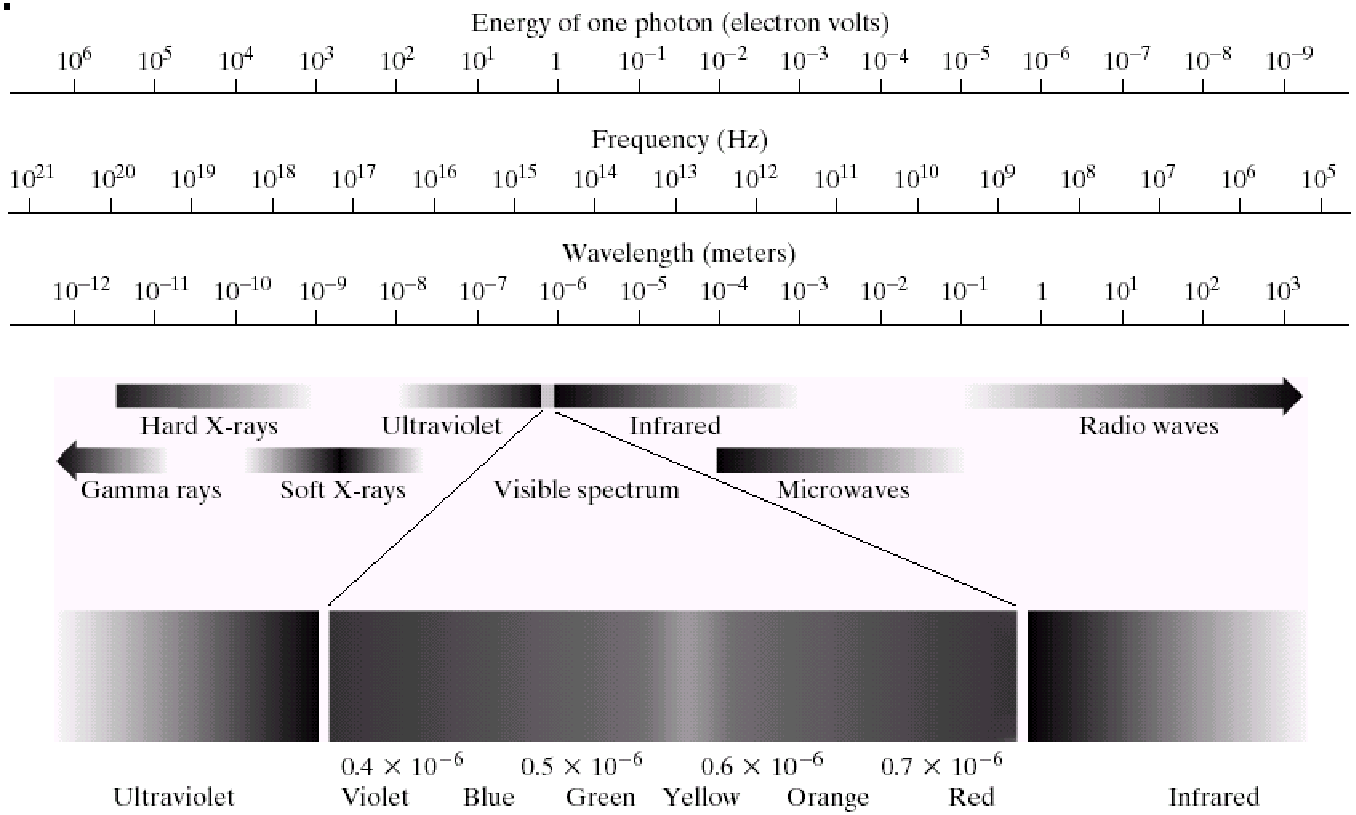


Figure 2.10 The electromagnetic spectrum. The visible spectrum is shown zoomed to facilitate explanation, but note that the visible spectrum is a rather narrow portion of the EM spectrum

Light and the electromagnetic spectrum

- Three basic quantities are used to describe the quality of a chromatic light source: radiance; luminance; and brightness
- *Radiance*, energy that flows from the light source (W)
- *Luminance*, energy an observer perceives from a light source (lm)
- *Brightness*, is a subjective descriptor of light perception

Light and the electromagnetic spectrum

In principle, if a sensor can be developed that is capable of detecting energy radiated by a bande of the electromagnetic spectrum, we can image events of interest in that band.

Image sensing and acquisition

The types of images in which we are interested are generated by the combination of an “illumination” source and the reflection or absorption of energy from that source by the elements of the “scene” being imaged.

Image sensing and acquisition

Figure 2.12 shows the three principal sensor arrangements used to transform illumination energy into digital images.

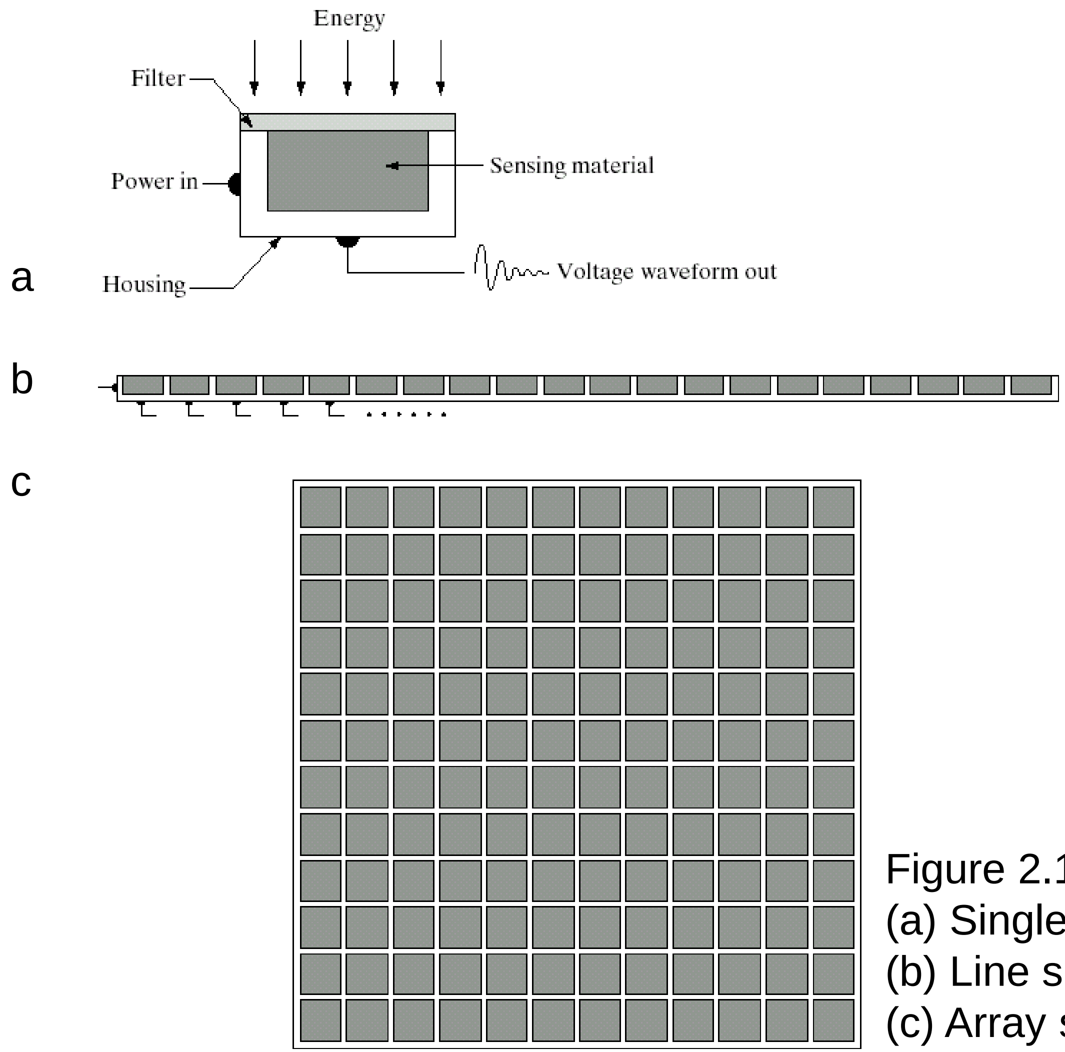


Figure 2.12
(a) Single imaging sensor.
(b) Line sensor.
(c) Array sensor.

Image acquisition using a single sensor

In order to generate a 2-D image using a single sensor, there has to be relative displacements in both the *x*- and *y*-directions between the sensor and the area to be imaged.

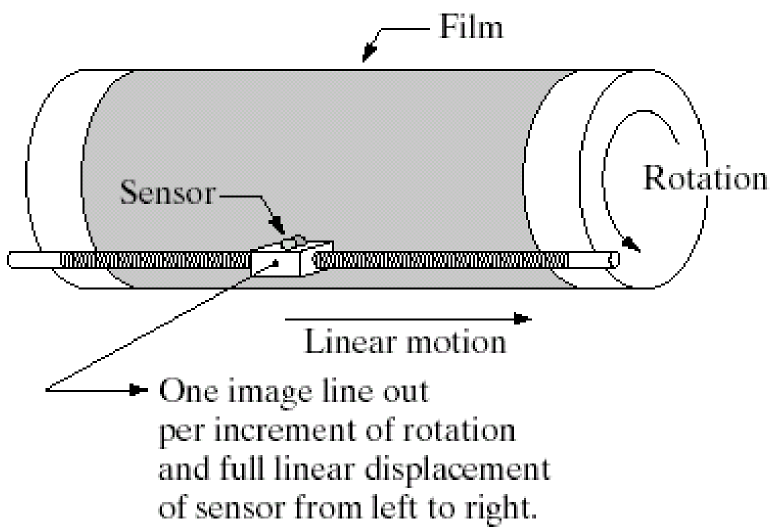


Figure 2.13 Combining a single sensor with motion to generate a 2-D image

Image acquisition using a single sensor

NXT Image Scanner (http://www.norgesgade14.dk/nxt_scanner.php)

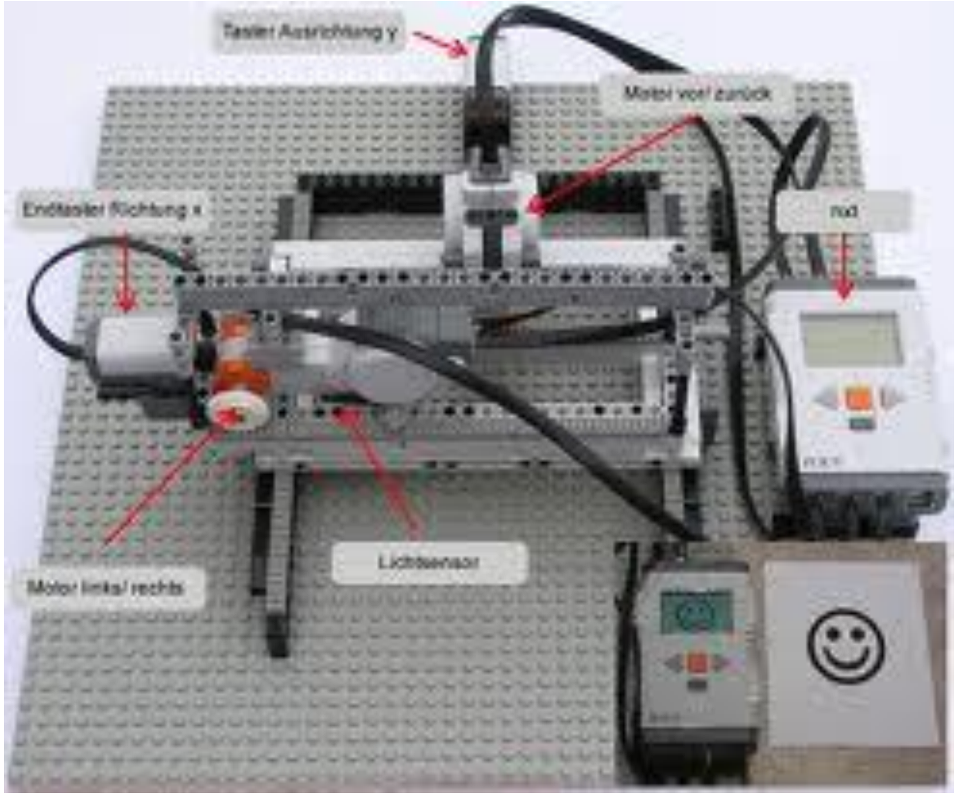


Image acquisition using sensor strips

Sensor strips mounted in a ring configuration are used in medical and industrial imaging to obtain cross-sectional ("slice") images of 3-D objects, as figure 2.14(b).

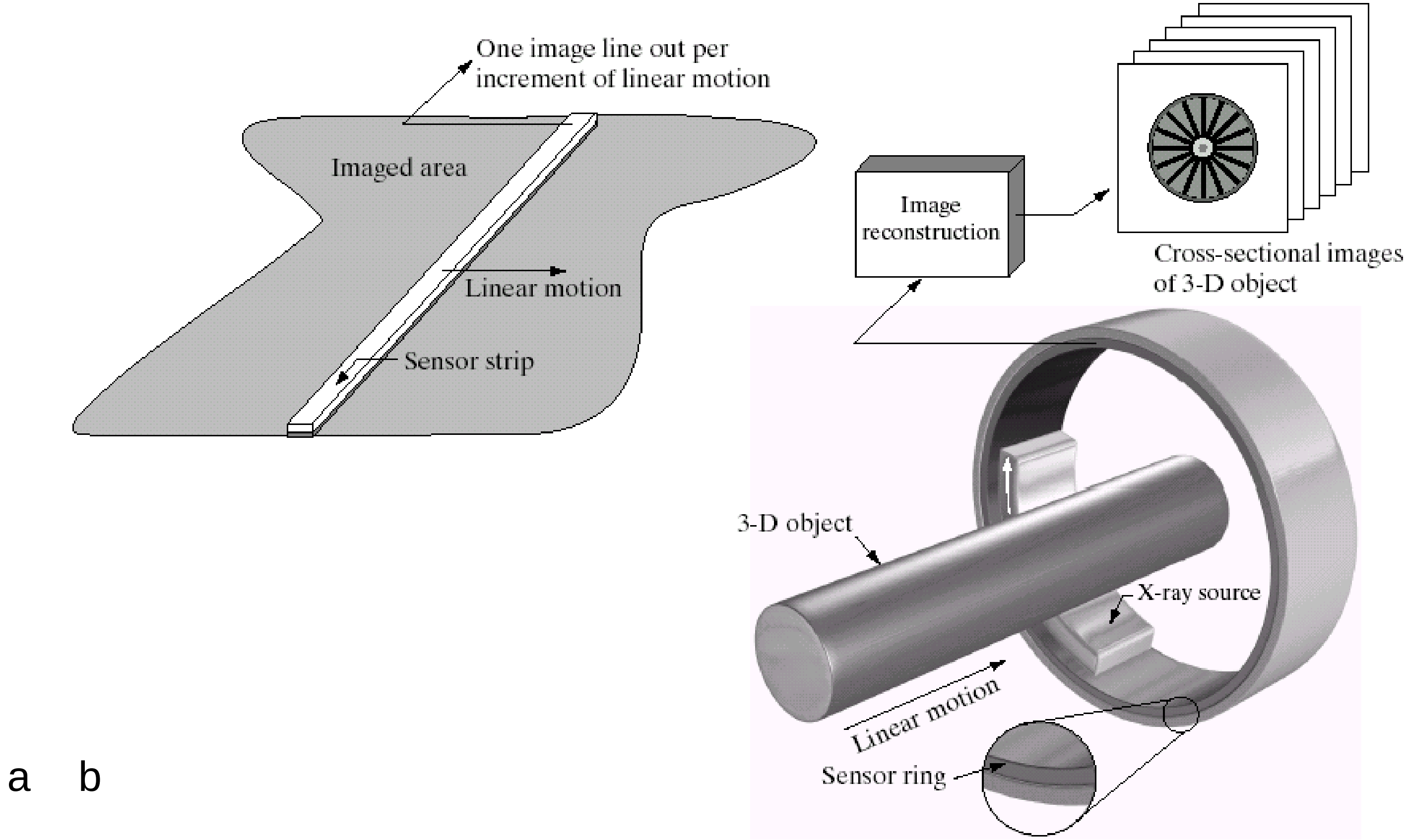


Figure 2.14 (a) Image acquisition using a linear sensor strip. (b) Image acquisition using a circular sensor strip

Image acquisition using sensor strips

This is the basis for medical and industrial computerized axial tomography (CAT)

Other modalities of imaging based on the CAT principle include magnetic resonance imaging (MRI) and positron emission tomography (PET)

Image acquisition using sensor arrays

This is also the predominant arrangement found in digital cameras. A typical sensor for these cameras is a CCD (charge-coupled device) array, can be packaged in rugged arrays of 4000 × 4000 elements.

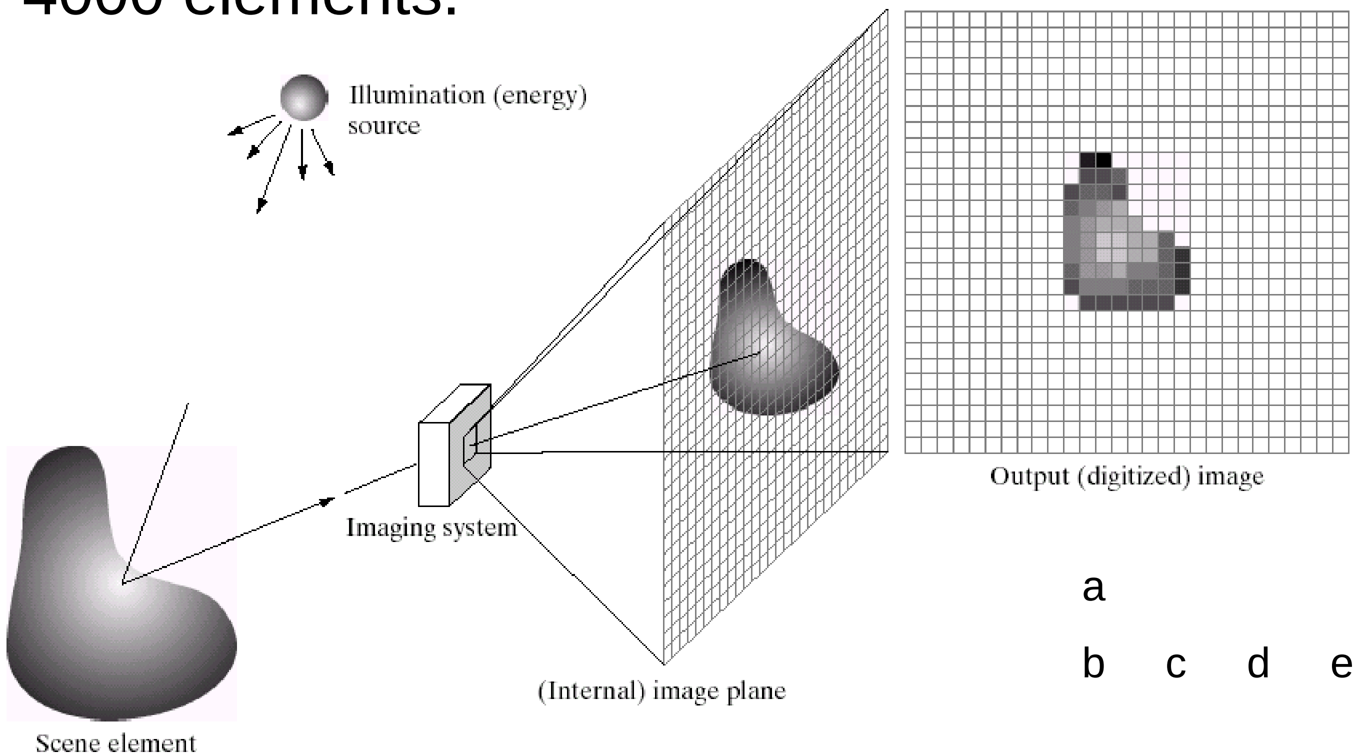


Figure 2.15 An example of the digital image acquisition process. (a) Energy ("illumination") source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

Representing digital images

The complete $M \times N$ digital image in the following compact matrix form:

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

Each element of this matrix array is called *image element*, *picture element*, *pixel*, or *pel.*

DIP in MATLAB

```
>> I=imread('Fig2.21(a).jpg');
>> whos I
>> image(I)
>> imagesc(I)
>> imshow(I)
>> imtool(I)
```

Image formation model

We denote images by two-dimensional functions of the form $f(x,y)$. The value or amplitude of f at spatial coordinates (x,y) is a positive scalar. In most of the images in which we are interested, its values are proportional to energy radiated by a physical source.

$$0 < f(x,y) < \infty$$

The function $f(x,y)$ may be characterized by two components: *illumination* is the amount of source illumination incident on the scene being viewed, and *reflectance* is the amount of illumination reflected by the objects in the scene.

Image formation model

The two functions combine as a product to form $f(x,y)$:

$$f(x,y) = i(x,y)r(x,y)$$

where

$$0 < i(x,y) < \infty$$

and

$$0 < r(x,y) < 1$$

Note: reflectance is bounded by 0 (total absorption) and 1 (total reflectance)

Image formation model

We call the intensity of a monochrome image at any coordinates (x_o,y_o) the *gray level* (ℓ) of the image at that point. That is,

$$\ell = f(x_o,y_o)$$

ℓ lies in the range

$$L_{min} \leq \ell \leq L_{max}$$

In practice, $L_{min} = i_{min}\, r_{min}$ and $L_{max} = i_{max}\, r_{max}$

Image formation model

The interval $[L_{min}, L_{max}]$ is called the *gray scale*. Common practice is to shift this interval numerically to the interval $[0, L - 1]$, where $\ell = 0$ is considered black and $\ell = L - 1$ is considered white on the gray scale.

Representing digital images

Figure 2.18 shows the coordinate convention used throughout this course.

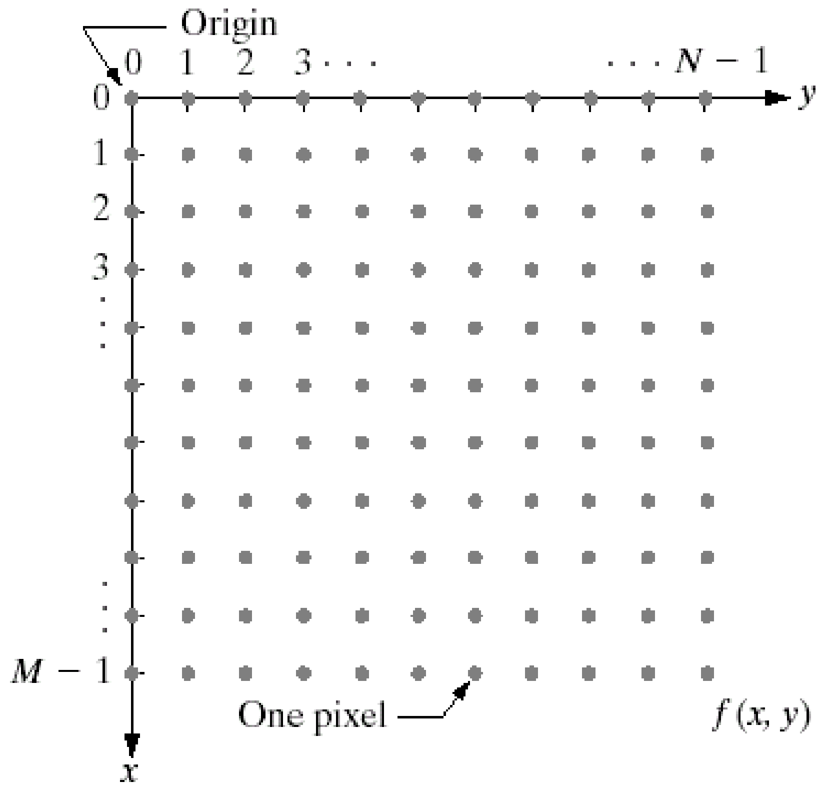


Figure 2.18 Coordinate convention used in this course to represent digital images

Image sampling and quantization

To create a digital image, we need to convert the continuos sensed data into digital form. This involves two processes: *sampling* and *quantization*.

Digitizing the coordinate values is called sampling. Digitizing the amplitude values is called quantization.

Image quantization

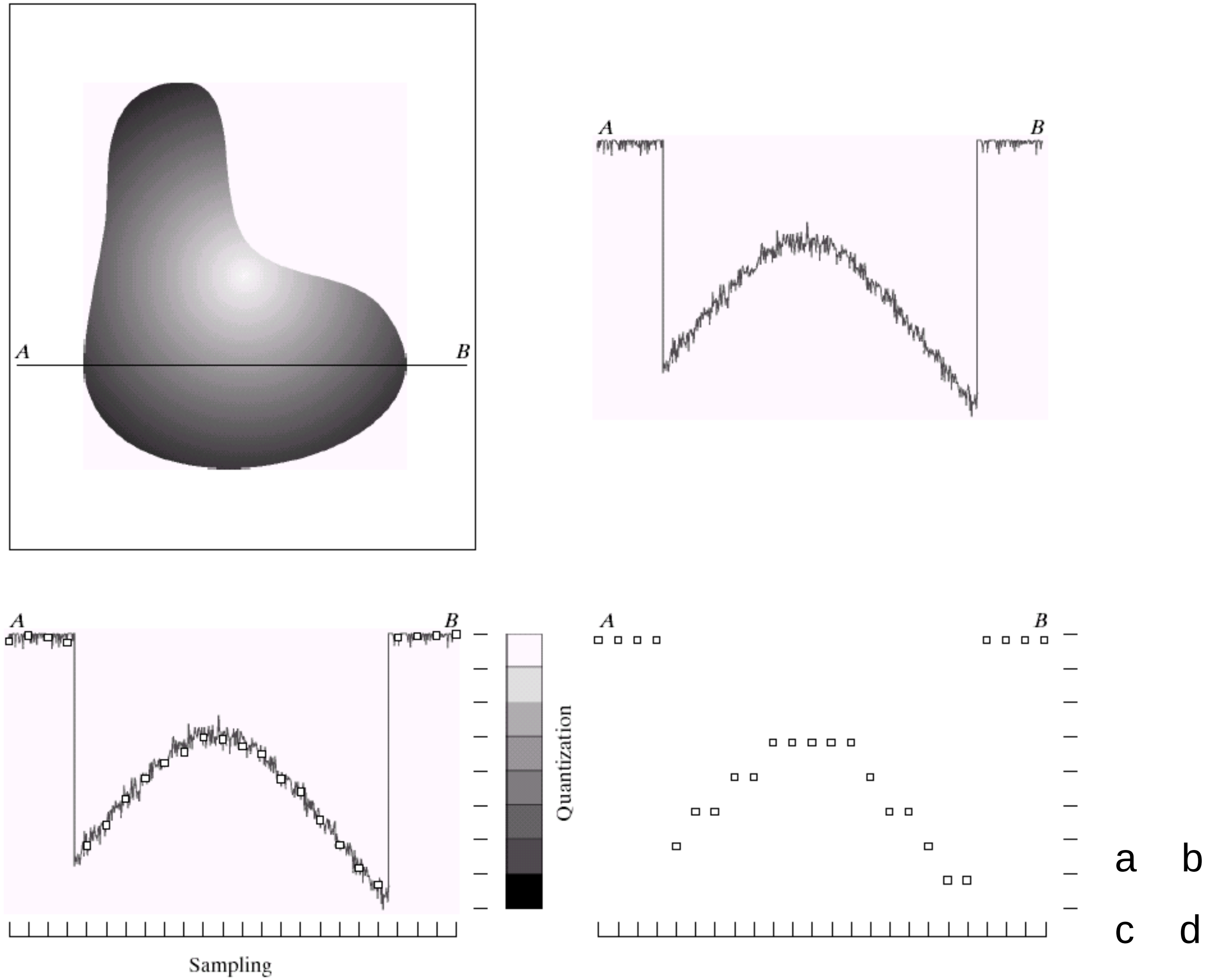


Figure 2.16 Generating a digital image. (a) Continuos image. (b) A scan line from A to B in the continuos image, used to illustrate the concepts of sampling and quantization. (c) Sampling and quantization. (d) Digital scan line

Lectura de Imágenes MATLAB

```
>> I(25,50) % Compare with imtool

>> line=I(226,:);

>> plot(line);

>> surface(I)

>> mesh(I)

>> imagesc(I')
```

Program: Find the positions with minimun value.

Image quantization

The number of gray levels typically is an integer power of 2:

$$L = 2^k$$

The range of values spanned by the gray scale is called the *dynamic range* of an image. The number, b, of bits required to store a digitized image is:

$$b = M \times N \times k$$

The number of gray levels is usually an integer power of 2, the most common number is 8 bits.

Image sampling and quantization

The table 2.1 shows the number of bits required to store square images with various values of N and k

Table 2.1 Number of storage bits for various values of N and k

N/k	**1 (L = 2)**	**2 (L = 4)**	**3 (L = 8)**	**4 (L = 16)**	**5 (L = 32)**	**6 (L = 64)**	**7 (L = 128)**	**8 (L = 256)**
32	1,024	2,048	3,072	4,096	5,120	6,144	7,168	8,192
64	4,096	8,192	12,288	16,384	20,480	24,576	28,672	32,768
128	16,384	32,768	49,152	65,536	81,920	98,304	114,688	131,072
256	65,536	131,072	196,608	262,144	327,680	393,216	458,752	524,288
512	262,144	524,288	786,432	1,048,576	1,310,720	1,572,864	1,835,008	2,097,152
1024	1,048,576	2,097,152	3,145,728	4,194,304	5,242,880	6,291,456	7,340,032	8,388,608
2048	4,194,304	8,388,608	12,582,912	16,777,216	20,971,520	25,165,824	29,369,128	33,554,432
4096	16,777,216	33,554,432	50,331,648	67,108,864	83,886,080	100,663,296	117,440,512	134,217,728
8192	67,108,864	134,217,728	201,326,592	268,435,456	335,544,320	402,653,184	469,762,048	536,870,912

Image sampling and quantization

Effects produced on image quality by varying N and k independently

a b c

Figure 2.22 (a) Image with a low level of detail. (b) Image with a medium level of detail. (c) Image with a relatively large amount of detail

Image sampling and quantization

Isopreference curves in the *N*-*k* plane

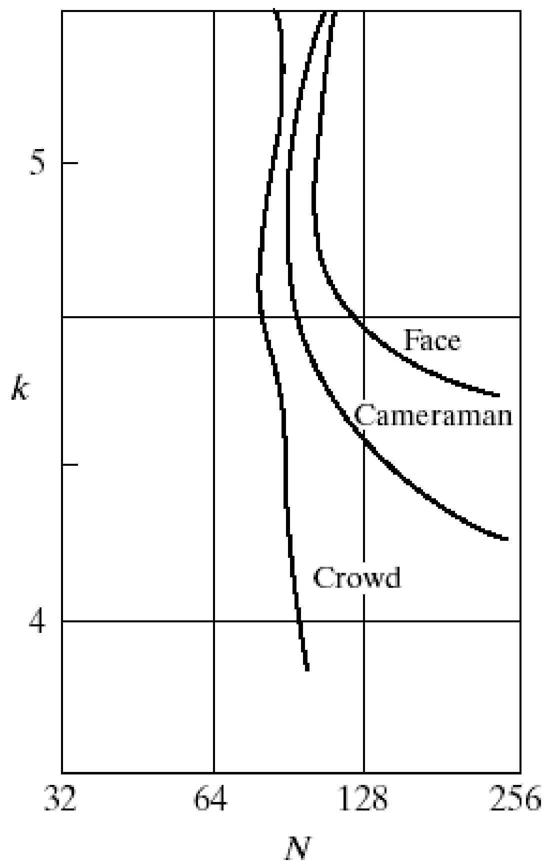


Figure 2.23 Representative isopreference curves for the three types of images in figure 20.

Image quantization

In this example, we keep the number of samples constant and reduce the number of gray levels from 256 to 2, in integer powers of 2.

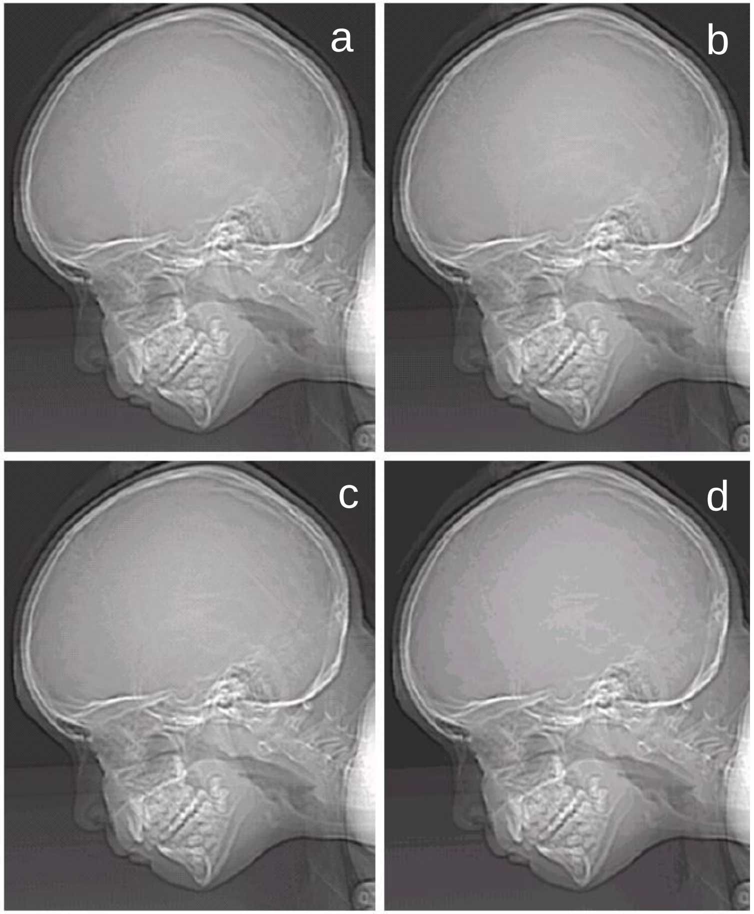


Figure 2.21 (a) 452×374, 256-level image. (b)-(d) Image displayed in 128, 64, and 32 gray levels, while keeping the spatial resolution constant.

Image quantization

The effect caused by the use of an insuficient number of gray levels in smooth areas of a digital image, is called *false contouring*

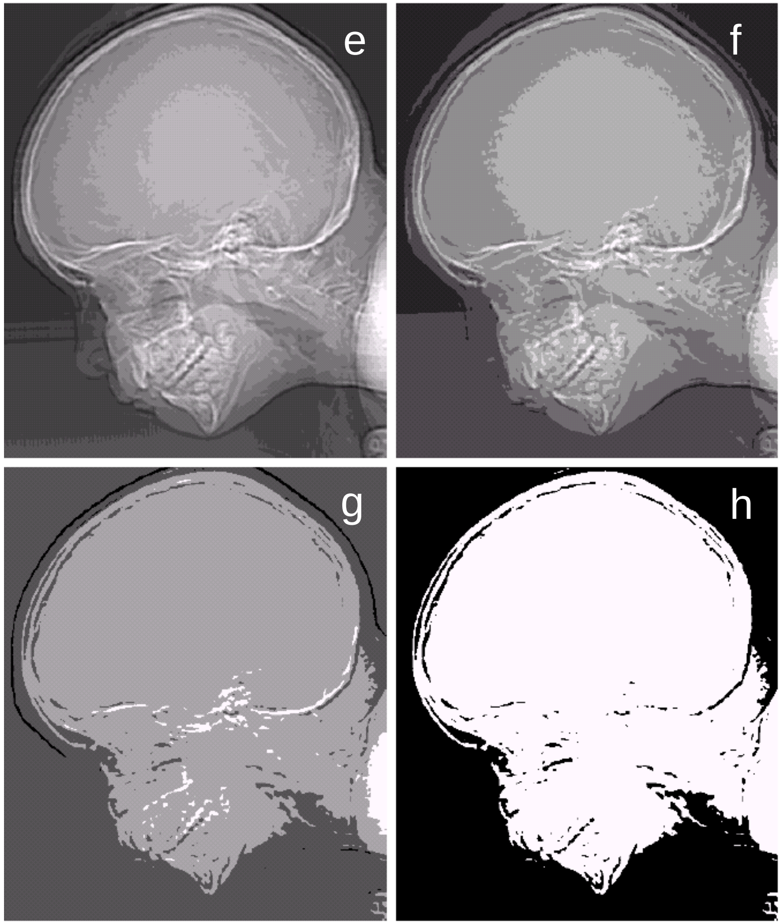
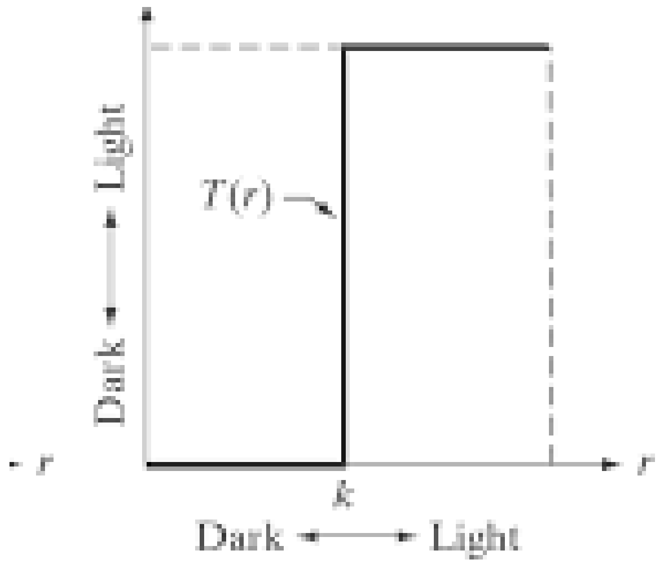


Figure 2.21 (*Continued)* (e)-(h) Image displayed in 16, 8, 4, and 2 gray levels

Image quantization



Project 02-02

Reducing the number of gray levels in an image

(a) Write a computer program capable of reducing the number of gray levels in an image from 256 to 2, in integer powers of 2. The desired number of gray levels needs to be a variable input to your program

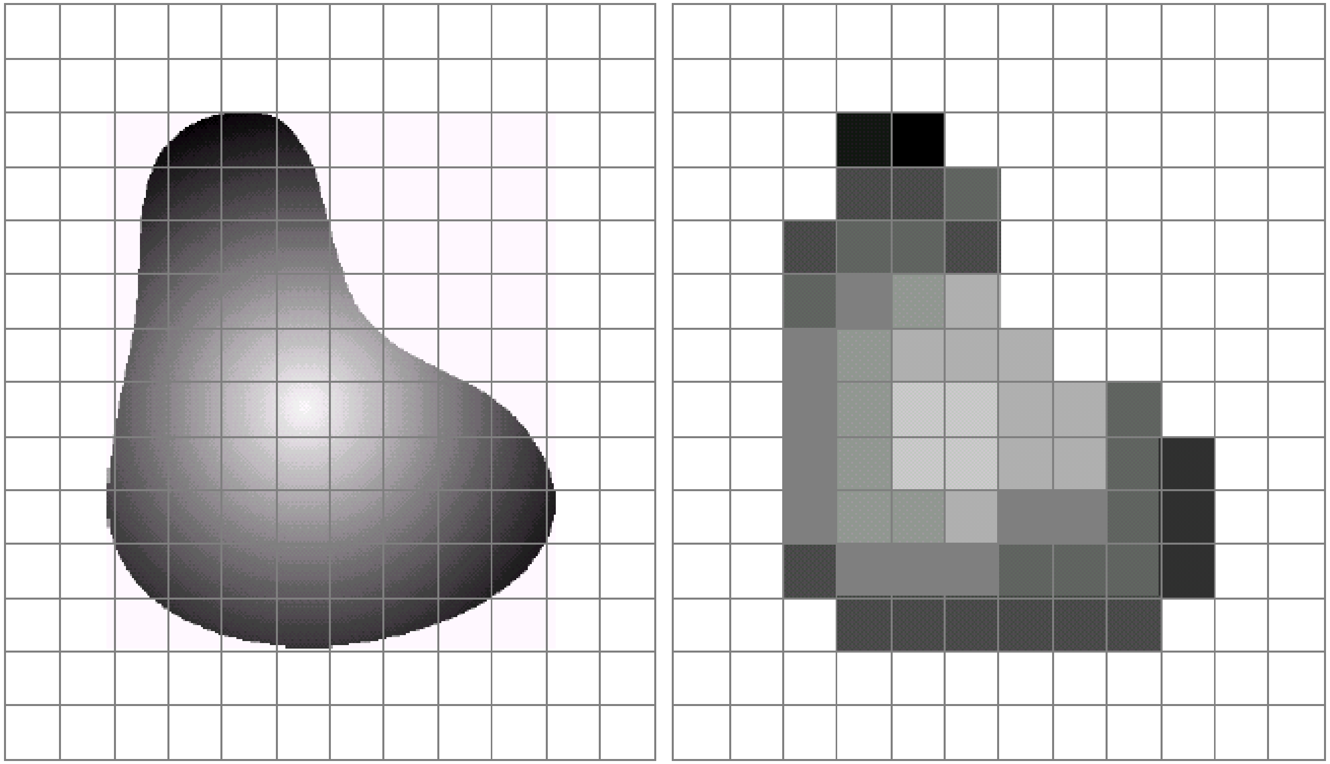
(b) Download fig 2.21 (a) 1 and duplicate the results shown in figure 2.21 of the book.

Image sampling

In practice, the method of sampling is determined by the sensor arrangement used to generate the image.

When a sensing array is used for image acquisition, there is no motion and the number of sensors in the array establishes the limits of sampling in both directions.

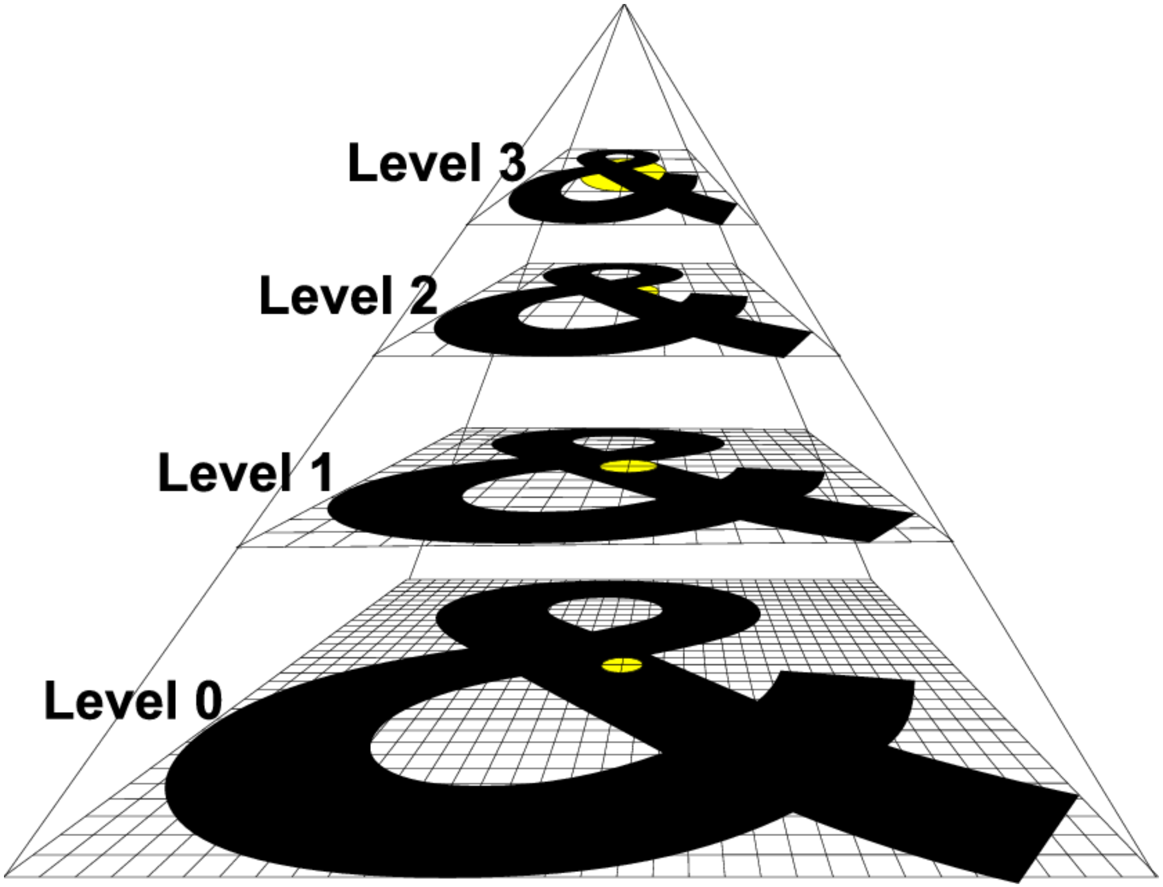
Image sampling



a b

Figure 2.17 (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization

Image downsampling



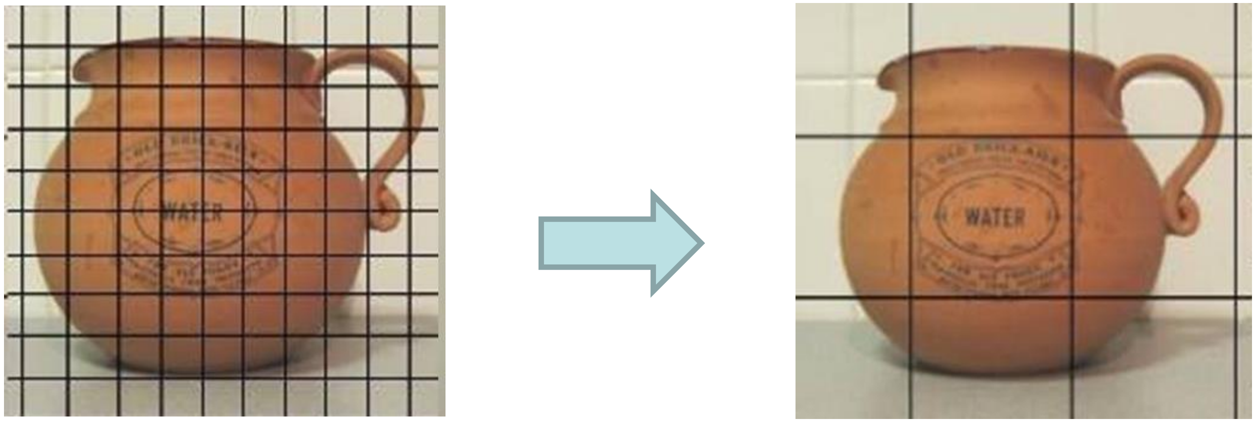
Shrinking digital images

Spatial resolution is the smallest discernible detail in an image.

Figure 2.19 A 1024 × 1024, 8-bit image subsampled down to size 32 × 32 pixels. The number of allowable gray levels was kept at 256

Shrinking digital images

- Shrinkingis laying an imaginary big grid over the original image.



Modified from: https://slideplayer.com/slide/7801257/

Shrinking digital images

Effects resulting from a reduction

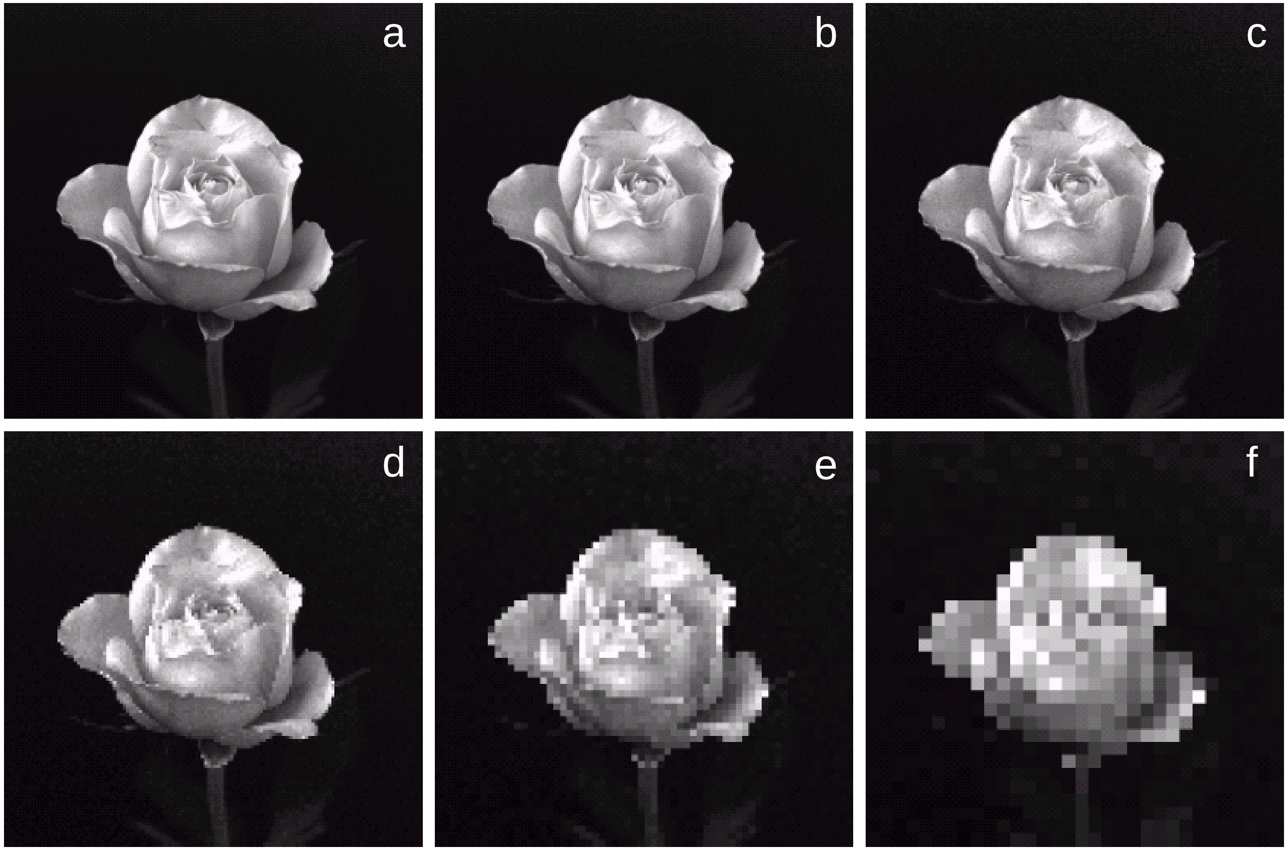


Figure 2.20 (a) 1024×1024, 8-bit image. (b) 512×512 image resampled into 1024×1024 pixels by row and column duplication. (c) through (f) 256×256, 128×128, 64×64, and 32×32 images resampled into 1024×1024 pixels

MATLAB: imresize

B = imresize(A, SCALE) returns an image that is SCALE times the size of A, which is a grayscale, RGB, or binary image.

B = imresize(A, [NUMROWS NUMCOLS]) resizes the image so that it has the specified number of rows and columns.

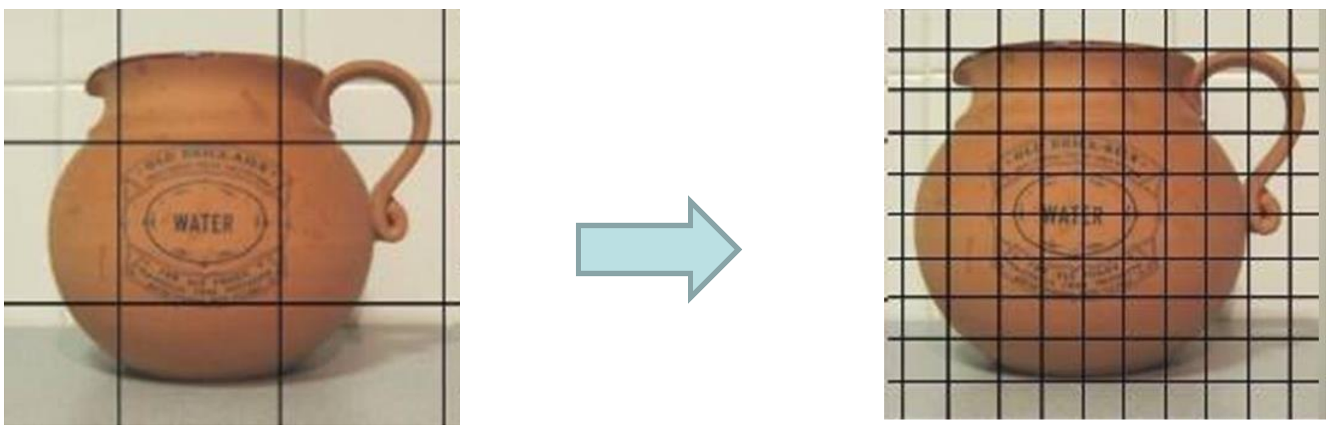
Either NUMROWS or NUMCOLS may be NaN, in which case imresize computes the number of rows or columns automatically in order to preserve the image aspect ratio.

Image upsampling

- Img=imread('Fig2.19(a)');
- Img2=imresize(Img, [128 128]);
- Image(img2)

Image upsampling

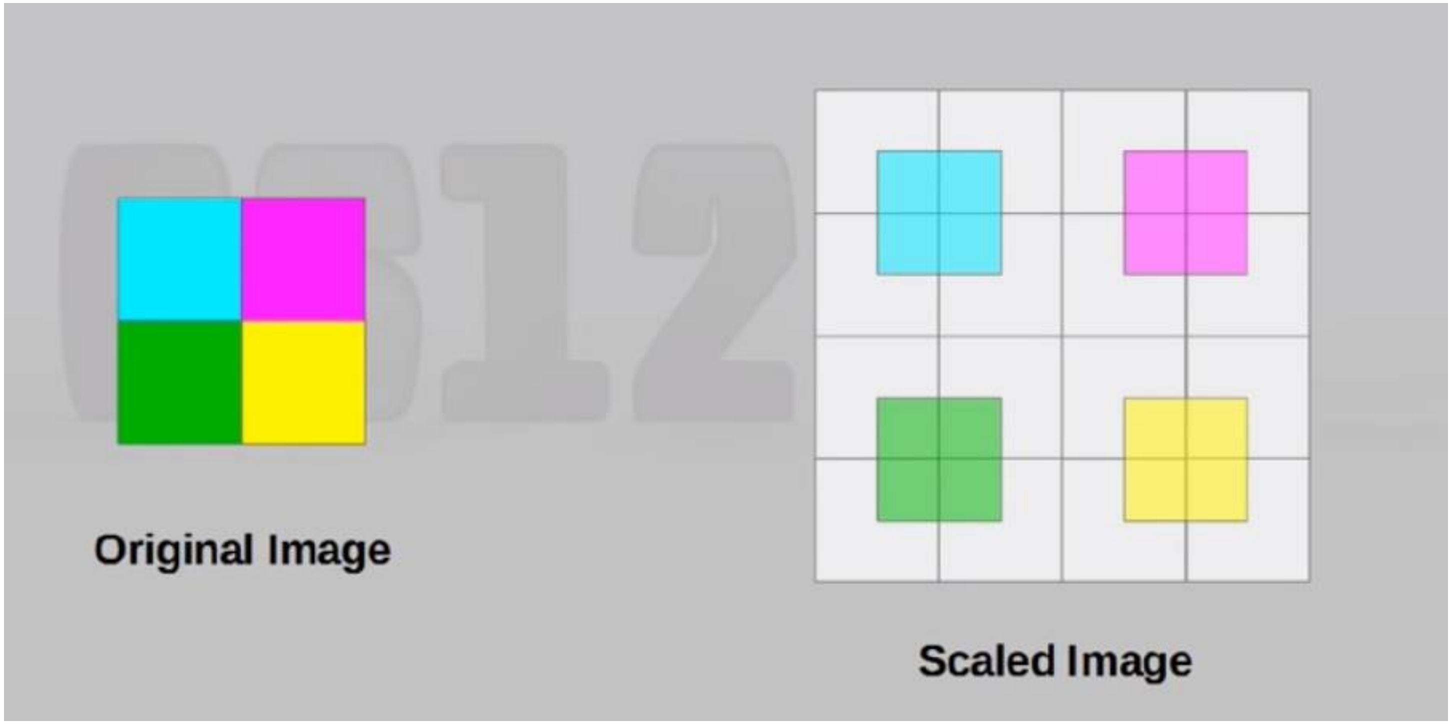
- Zooming requires two steps: the creation of new pixel locations, and the assignment of gray levels to those new locations.
- Zooming is laying an imaginary small grid over the original image.



Taken from https://slideplayer.com/slide/7801257/

Zooming digital images

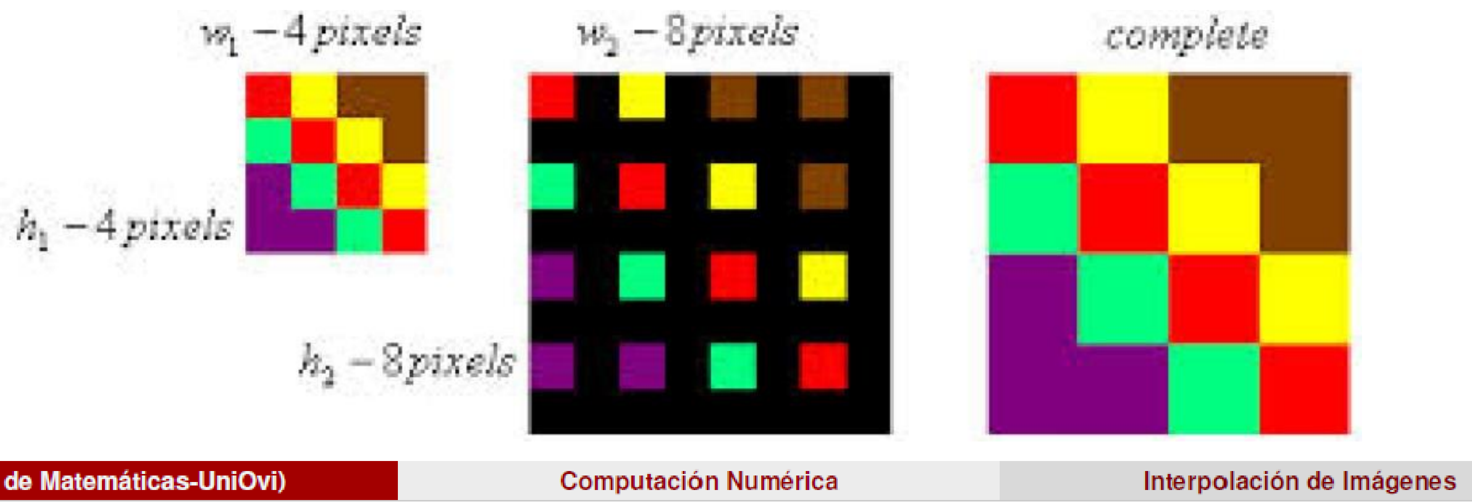
This method of gray level assignment is called *nearest neighbor interpolation.*



https://www.youtube.com/watch?v=fuTYqzWPHGg

Zooming digital images

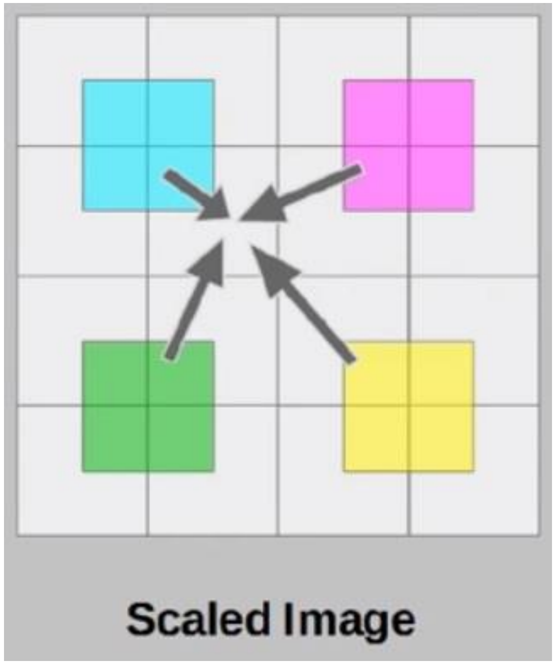
Pixel replication is applicable when we want to increase the size of an image an integer number of times.



This methods have the undesirable feature that it produces a checkerboard effect.

Zooming digital images

A more sophisticated way of accomplishing gray level assignments is *bilinear interpolation.*

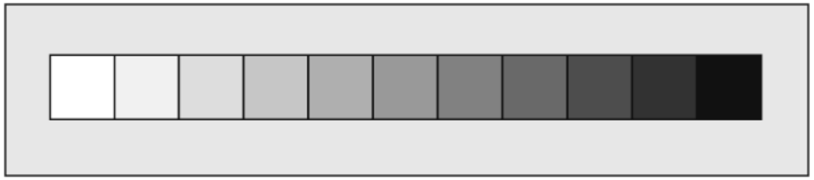


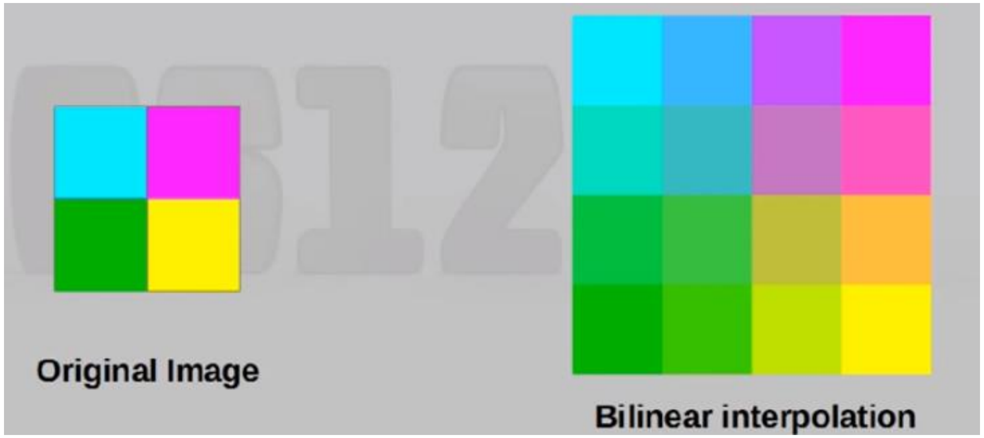
https://www.youtube.com/watch?v=fuTYqzWPHGg

Zooming digital images

- In order to perform gray level assignment for any point in the overlay, we look for the closest pixel in the original image and assign its gray level to the new pixel in the grid.

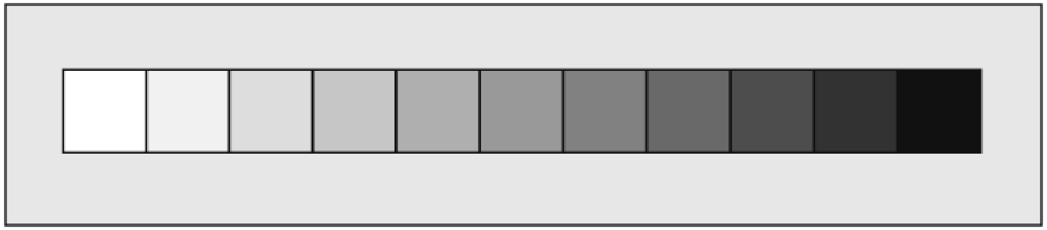
- Smooth transition.





https://www.youtube.com/watch?v=fuTYqzWPHGg

Interpolation

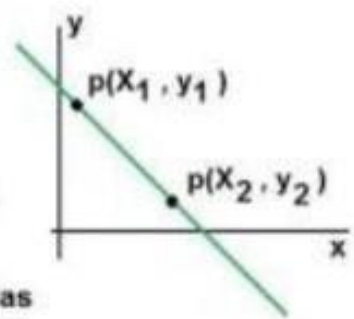


$$y - y_1 = \frac{y_2 - y_1}{x_2 - x_1}\left(x - x_1\right)$$

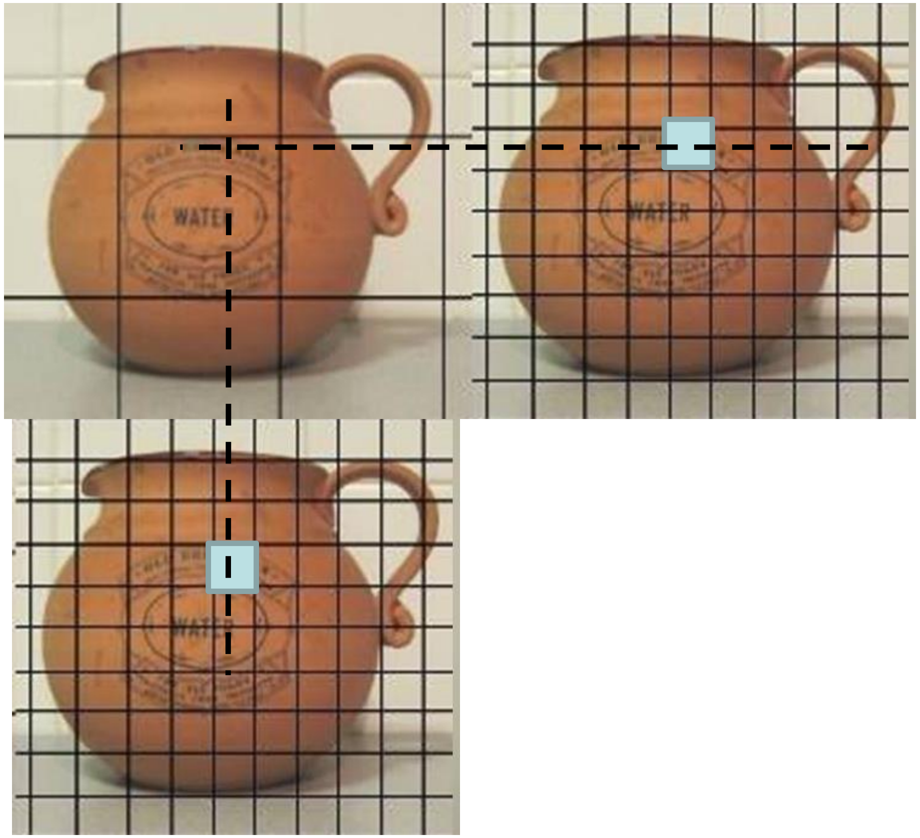
donde:

(x_1, y_1) son las coordenadas del primer punto

(x_2, y_2) son las coordenadas del segundo punto



Area sampling



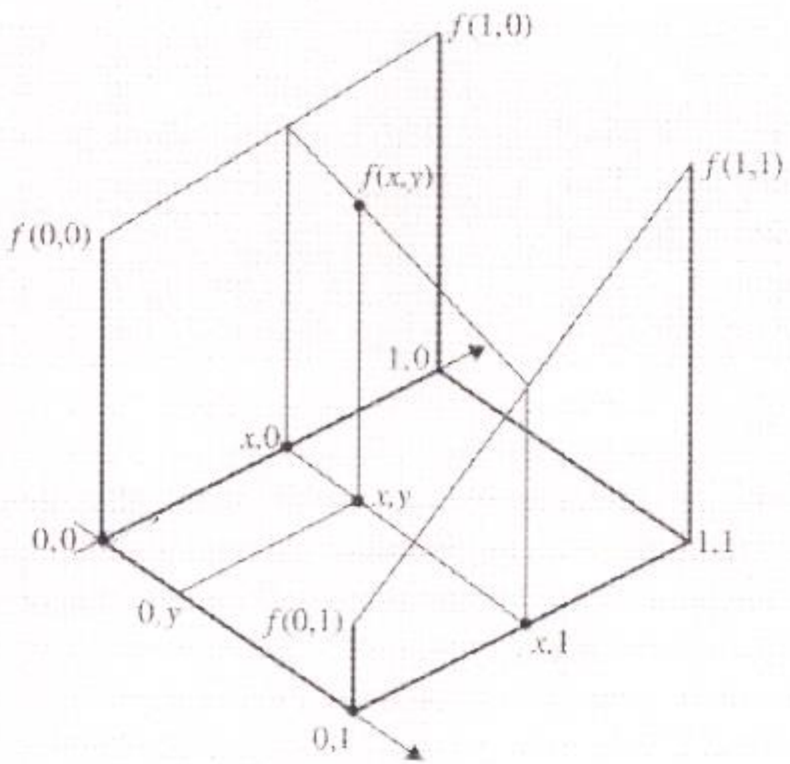
Modified from: https://slideplayer.com/slide/7801257/

Zooming digital images

- Position x=3.4 , y=16.8

Bilinear interpolation

Sea f(x, y) una función, que es conocida en los vértices de un **cuadrado unitario**. Supongamos que queremos conocer el valor de f(x, y) en cualquier punto dentro del cuadrado.



Bilinear interpolation

$$f(x,0) = f(0,0) + x[f(1,0) - f(0,0)] \qquad (3)$$

De la misma manera se interpola para los puntos inferiores:

$$f(x,1) = f(0,1) + x[f(1,1) - f(0,1)] \qquad (4)$$

Finalmente interpolamos verticalmente para determinar el valor de:

$$f(x,y) = f(x,0) + y[f(x,1) - f(x,0)] \qquad (5)$$

Substituyendo las ecuaciones (3) y (4) 3n (5)

$$f(x,y) = [f(1,0) - f(0,0)]\, x + [f(0,1) - f(0,0)]y + [f(1,1)-f(0,0) - f(0,1) - f(1,0)]xy + f(0,0) \qquad (6)$$

La cual es de la forma de la ecuación (1) y es bilineal.

Bilinear Interpolation

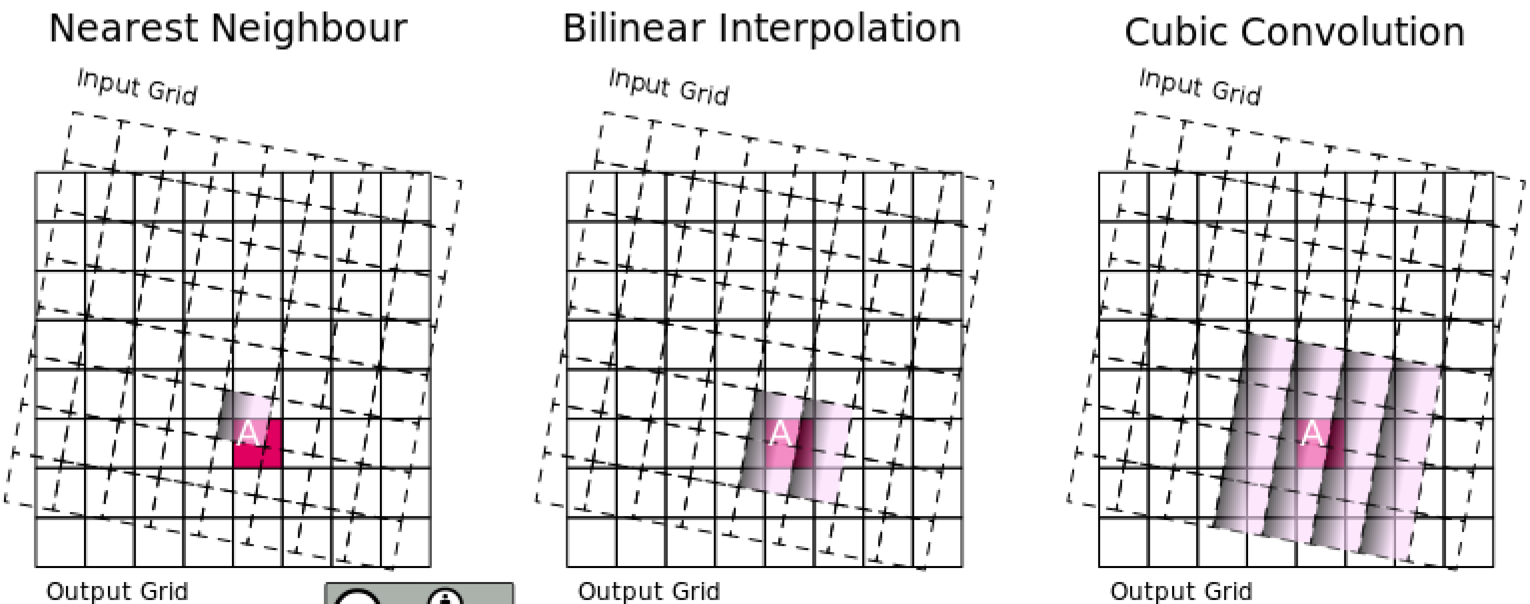
Notese que si se fija el valor de x o de y, la ecuación (2) se vuelve lineal en alguna de las dos variables.

La interpolación bilineal puede ser obtenida a partir de la ecuación (6) o bien realizando las tres interpolaciones lineales indicadas por las ecuaciones (3), (4) y (5).

Puesto que la ecuación 6 involucra 4 multiplicaciones y 8 adiciones o subtracciones, las transformaciónes geométricas casi siempre usan el calculo de las ecuaciones (3), (4) y (5), ya que ese solo requiere de 3 multiplicaciones y 6 sumas o substracciones.

Este algoritmo puede ser generalizado para el caso en que x, e y son fraccionales.

Interpolation



Zooming and shrinking digital images

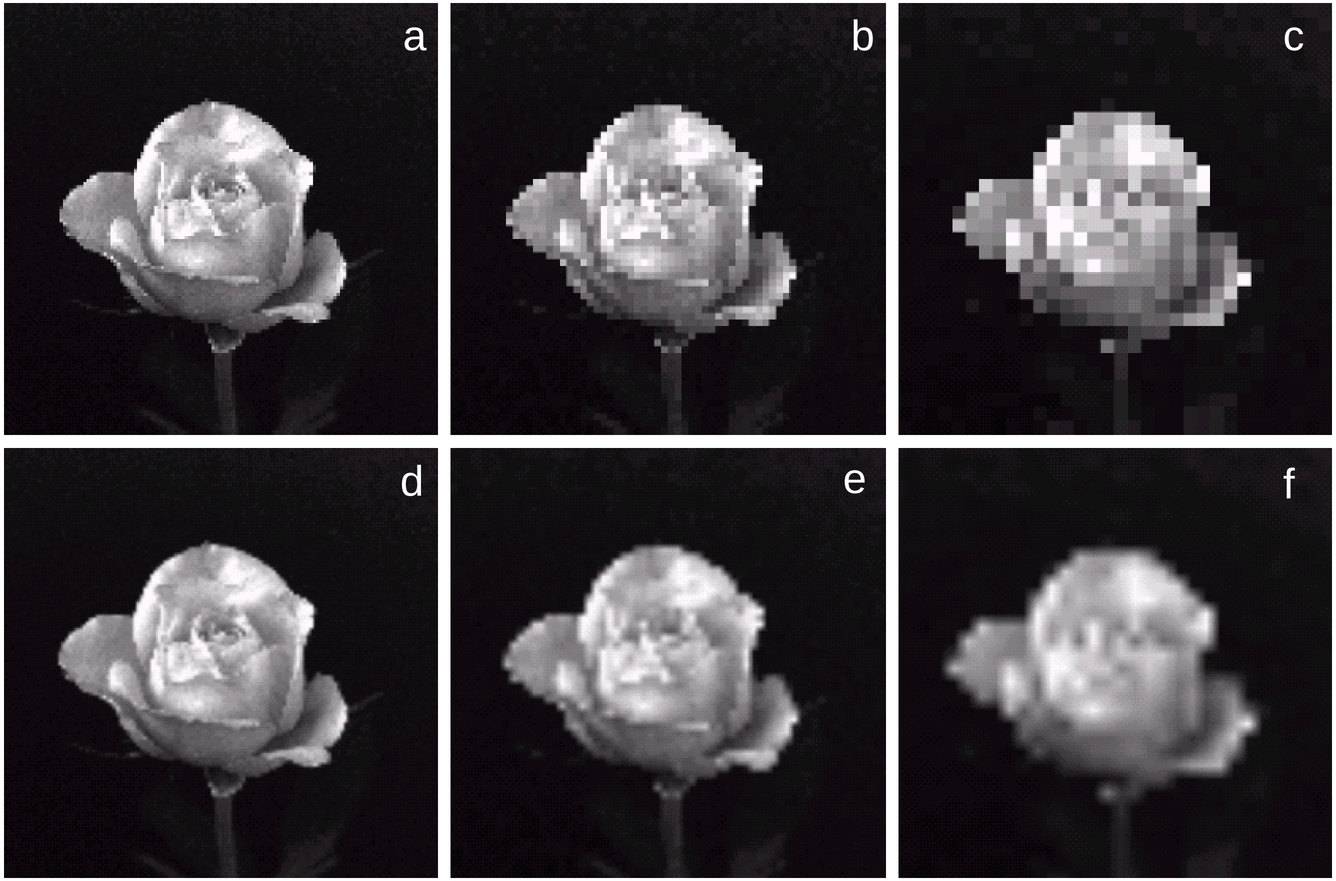


Figure 2.25 Top row: images zoomed from 128×128, 64×64, and 32×32 pixels to 1024×1024 pixels, using nearest neighbor gray level interpolation. Bottom row: same sequence, but using bilinear interpolation

Replicate

MATLAB: imresize

To control the interpolation method used by imresize, add a METHOD
argument to any of the syntaxes above, like this:

```
imresize(A, SCALE, METHOD)
imresize(A, [NUMROWS NUMCOLS], METHOD),
```

METHOD can be a string naming a general interpolation method:

```
'nearest'   - nearest-neighbor interpolation
'bilinear'  - bilinear interpolation
'bicubic'   - cubic interpolation; the default method
```

Project 02-04

Zooming and shrinking images by bilinear interpolation

(a) Write a computer program capable of zooming and shrinking an image by nearest neighbor and bilinear interpolation. The input to your program is the desired size of the resulting image in the horizontal and vertical direction. You may ignore aliasing effects.

(b) Download figure 2.19 (a) and use your program to shrink this image from 1024×1024 to 256×256 pixels.

(c) Use your program to zoom the image in (b) back to 1024×1024. Explain the reasons for their differences.

Basic relationships between pixels

A pixel p at coordinates (x,y) has four *horizontal* and *vertical* neighbors whose coordinates are given by

$$(x+1, y), (x-1, y), (x, y+1), (x, y-1)$$

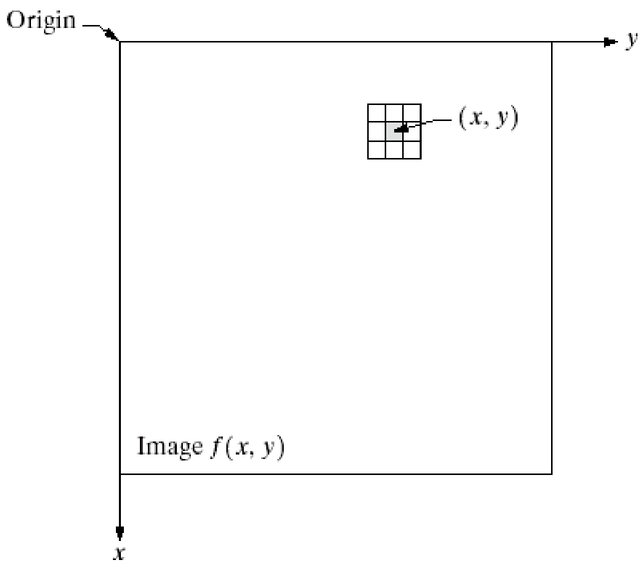
This set of pixels, called the 4-*neighbors* of p, is denoted by $N_4(p)$. The four *diagonal* neighbors of p have coordinates

$$(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1, y-1)$$

and are denoted by $N_D(p)$. These points, together with the 4-neighbors, are called the 8-*neighbors* de p, denoted by $N_8(p)$.

Background

The principal approach in defining a neighborhood is to use a square or rectangular subimage area centered at (x,y)



Adjacency, connectivity, regions, and boudaries

To establish if two pixels are connected, it must be determined if they are neighbors and if their gray levels satisfy a specified criterion of similarity. We consider two types of adjancency:

(a) 4-*adjacency*. Two pixels p and q with values from V are 4-adjancent if q is in the set $N_4(p)$

(b) 8-*adjacency*. Two pixels p and q with values from V are 8-adjacent if q is in the set $N_8(p)$

Adjacency, connectivity, regions, and boudaries

A (*digital*) *path* (or *curve*) from pixel p with coordinates (x,y) to pixel q with coordinates (s,t) is a sequence of distinct pixels with coordinates

$$(x_0, y_0), (x_1, y_1), \cdots (x_n, y_n)$$

where $(x_0,y_0) = (x,y) \cdot (x_n,y_n) = (s,t)$ and pixels (x_i,y_i) and (x_{i-1},y_{i-1}) are adjacent for $1 \leq i \leq n$

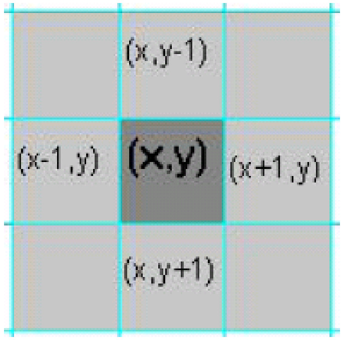
Two pixels p and q are said to be *connected* in S if there exist a path between them consisting entirely of pixels in S.

Adjacency, connectivity, regions, and boudaries

Let R be a subset of pixels in an image. We call R a *region* of the image if R is a connected set (*conex components*).

Conex components

Sea S una imagen digital binaria en un mallado cuadrado. Supongamos que queremos localizar las componentes conexas en nego de la imagen con la 4-adyacencia. Si un píxel se encuentra en posición (x,y), recordemos que sus vecinos pueden ser:



Recorremos la imagen de izquierda a derecha y de arriba a abajo.

Durante el primer rastreo, para cada punto que tenga valor 1, examinamos a los vecinos de arriba y a mano izquierda de *P*, nótese que si existen, acaban de ser visitados por el rastreo, así que si son 1's, ya han sido etiquetados.

- Si ambos son 0's, damos a *P* una nueva etiqueta;
- si tan sólo uno es 0, le damos a *P* la etiqueta del otro;

Conex components

- y si ninguno es 0's, le damos a P (por ejemplo) la etiqueta del de la izquierda, y si sus etiquetas son diferentes, registramos el hecho de que son equivalentes, i.e., pertenecen a la misma componente.

Cuando se completa este rastreo, cada 1 tiene una etiqueta, pero puede que se asignen muchas etiquetas diferentes a puntos en el mismo componente. Ahora ordenamos las parejas equivalentes en clases de equivalencia, y escogemos una etiqueta para representar cada clase. Finalmente, realizamos un segundo rastreo de la imagen y sustituimos cada etiqueta por el representante de cada clase; cada componente ha sido ahora etiquetada de forma única.

Consideremos la siguiente imagen:

				1	1	1			1	1		1	1
		1	1	1		1	1	1	1				1
1	1			1	1	1		1	1	1	1	1	

Conex components

El resultado del primer rastreo, usando 4-adyacencia es :

```
        a a a     b b   c c
    d d d   a a a a       c   d = a, b = a
e e     d d d   a a a a a     d = a
```

Resultado del segundo rastreo, reemplazando todas las etiquetas equivalentes por una representativa:

```
        a a a     a a   c c
    a a a   a a a a       c
e e     a a a   a a a a a
```

Si consideramos al 8-adyacencia para negro, para cada píxel P(x,y) en negro, examinamos los píxeles superiores A(x+1,y-1), B(x,y-1), C(x-1,y-1) y D(x-1,y).

Conex components

- Si todos sus vecinos son blancos, damos al actual píxel una etiqueta nueva.
- En otro caso, si uno de sus vecinos es negro, al píxel P le damos la misma etiqueta que la del píxel negro.
- Si dos o más píxeles son negros, le damos a P una de las etiquetas y anotamos que dichas etiquetas son equivalentes.

El proceso de equivalencia y re-etiquetado se realiza como en el caso de 4-adyacencia.

Resultados del primer rastreo usando el algoritmo de 8-adyacencia:

				a	a	a			b	b		c	c	
		d	a	a		a	a	b	b				c	d = a, b = a
e	d			a	a	a		b	b	b	b	c		e = d, b = a, c = b

Conex components

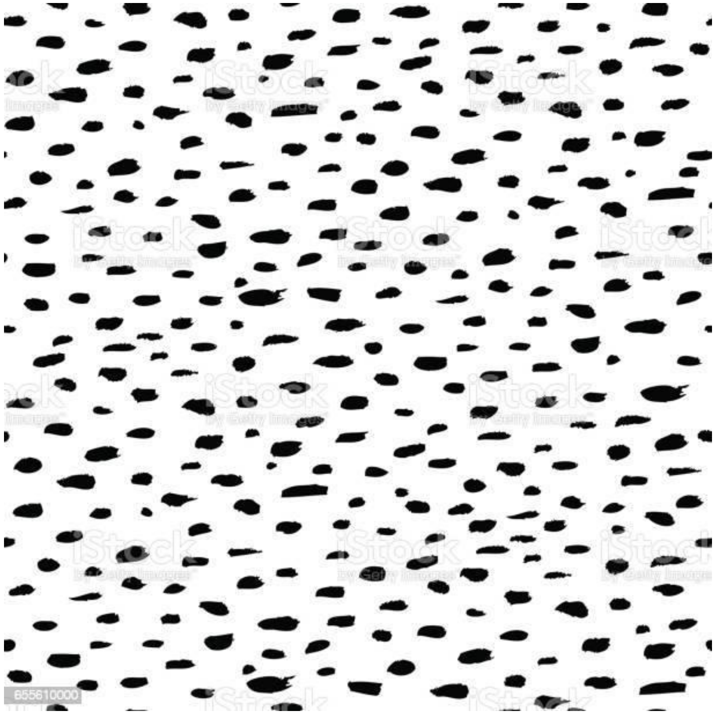
Alternativamente, para cada píxel P(x,y) de la imagen, sea blanco o negro, podemos examinar a los vecinos B(x,y-1), C(x-1,y-1) y D(x-1,y). Procedemos como antes si P es 1. Si *P* es 0, pero dos o más de sus vecinos B, C ó D son 1, anotamos la equivalencia de sus etiquetas.

Resultados del primer rastreo usando la segunda versión del algoritmo de 8-adyacencia:

				a	a	a			b	b		c	c	
		d	d	a		a	a	a	b				c	d = a, b = a
e	e			a	a	a		a	b	b	b	b		e = d, b = a, c = b

Conex components

- img=imread('Spots.jpg');
- img2=rgb2gray(img);
- BW=img2 < 25;
- CC = bwconncomp(BW,4);



CC = struct with fields:

Connectivity: 8
ImageSize: [612 612]
NumObjects: 396
PixelIdxList: {1×396 cell}

- L = bwlabel(BW,4);

Try with BWfigs, DB0255, BW01

Adjacency, connectivity, regions, and boudaries

Think…

The *boundary* of a region R is the set of pixels in the region that have one or more neighbors that are not in R.

Project 02-05

Regions and Boundaries in binary images

(a) Implement the conex components algorithm to identify the number of objects present in a given binary image. The output of the algorithm is the number of objects.

(b) Implement a boundary detector (binary image).

Bit-plane slicing

Highlighting the contribution made to total image appearance by specific bits might be that each pixel in an image is represented by 8 bits.

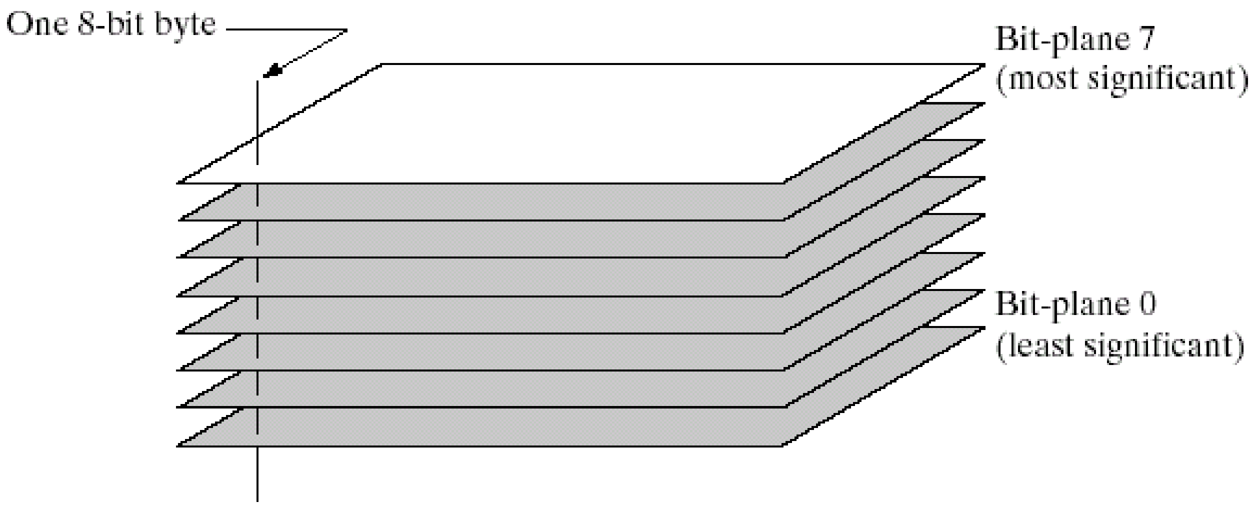


Figure 3.12 Bit-plane representation of an 8-bit image

Bit-plane slicing

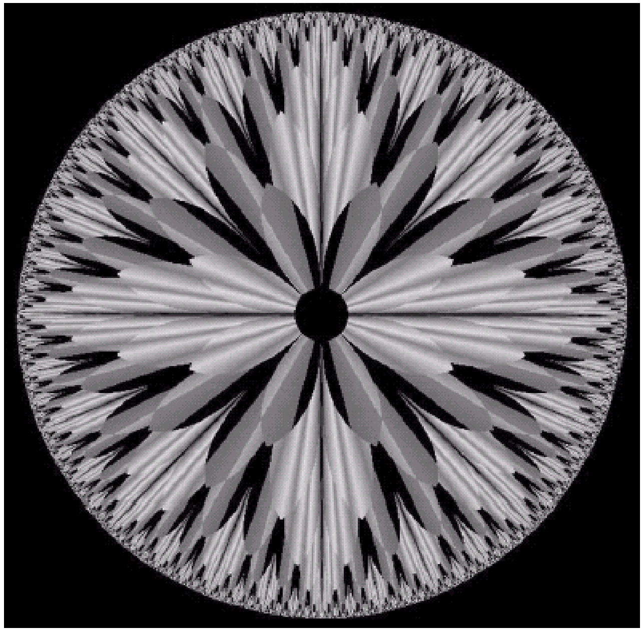


Figure 3.13 An 8-bit fractal image (A fractal is an image generated from mathematical expressions)

Bit-plane slicing

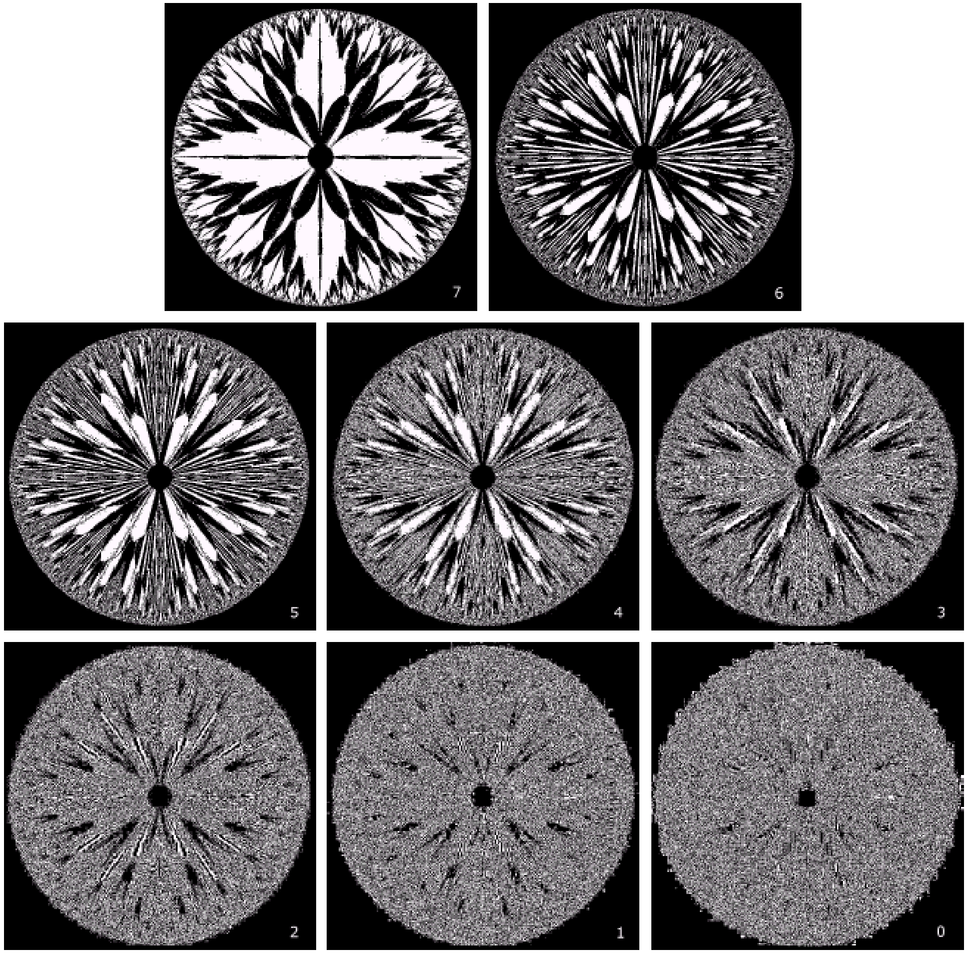


Figure 3.14 The eight bit planes of the image in figure 3.13. The number at the bottom, right of each image identifies the bit plane.

Homework: Problem 3.3

Make a program to create a set of gray-level slicing transformations capable of producing all the individual bit planes of an 8-bit monochrome image. (For example, a transformation function with the property $T(r) = 0$ for r in the range [0,127], and $T(r) = 255$ for r in the range [128,255] produces an image of the 7th bit plane in an 8-bit image).

Problem 3.3

The transformations required to produce the individual bit planes are nothing more than mappings of the truth table for eight binary variables. In this truth table, the values of the 7th bit are 0 for byte values 0 to 127, and 1 for byte values 128 to 255, thus giving the transformation mentioned in the problem statement. Note that the given transformed values of either 0 or 255 simply indicate a binary image for the 7th bit plane. Any other two values would have been equally valid, though less conventional.

Problem 3.3

(*continuation*)

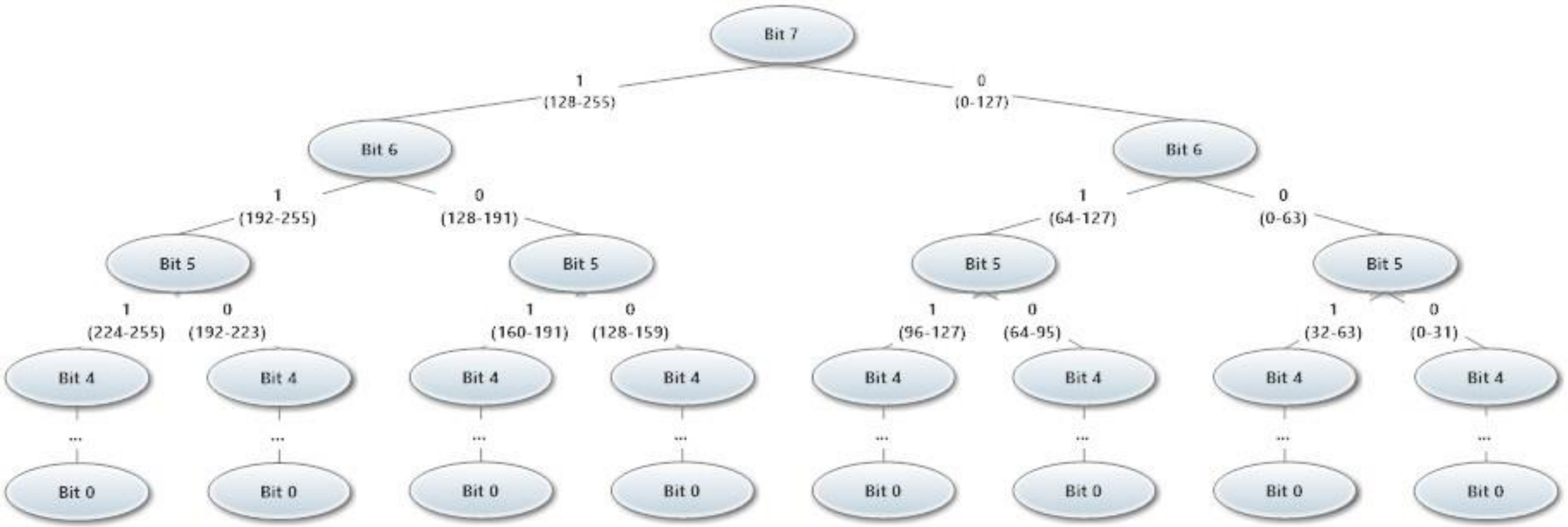
Continuing with the truth table concept, the transformation requerid to produce an image of the 6th bit plane outputs a 0 for byte values in the range [0,63], a 1 for byte values in the range [64,127], a 0 for byte values in the range [128,191], and a 1 for byte values in the range [192,255]. Similarly, the transformation for the 5th bit plane alternates between eight ranges of byte values, the transformation for the 4th bit plane alternates between 16 ranges, and so on.

Problem 3.3

(*continuation*)

Finally, the output of the transformation for the 0th bit plane alternates between 0 and 255 depending as the byte values are even or odd. Thus, this transformation alternates between 128 byte value ranges, which explains why an image of the 0th bit plane is usually the busiest looking of all the bit plane images.

Problem 3.3



Bit 7
1
(128-255)
0
(0-127)
Bit 6
Bit 6
1
(192-255)
0
(128-191)
1
(64-127)
0
(0-63)
Bit 5
Bit 5
Bit 5
Bit 5
1
(224-255)
0
(192-223)
1
(160-191)
0
(128-159)
1
(96-127)
0
(64-95)
1
(32-63)
0
(0-31)
Bit 4
Bit 4
Bit 4
Bit 4
Bit 4
Bit 4
Bit 4
Bit 4
...
Bit 0
Bit 0
Bit 0
Bit 0
Bit 0
Bit 0
Bit 0
Bit 0

Bit-plane slicing

Separating a digital image into its bit planes is useful for analyzing the relative importance played by each bit of the image. This type of decomposition is useful for image compression. Reproduce results shown in figure 3.14.

Minimum Description Length (MDL)

- Cost(Model,Data) = Cost(Model) + Cost(Data|Model)
  - Cost is the number of bits needed for encoding.
  - Search for the least costly model.
- Cost(Data|Model) encodes the misclassification errors.
- Cost(Model) uses node encoding (number of children) plus splitting condition encoding.

- **Run-length encoding**. La compresión RLE es una forma muy simple de compresión de datos en la que secuencias de datos con el mismo valor consecutivas son almacenadas como un único valor más su recuento.