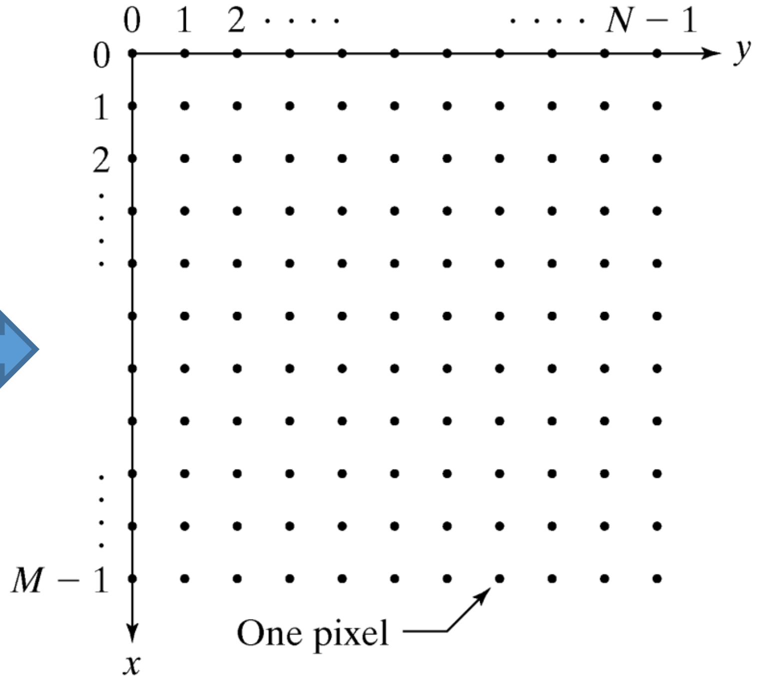
Procesamiento de Imágenes

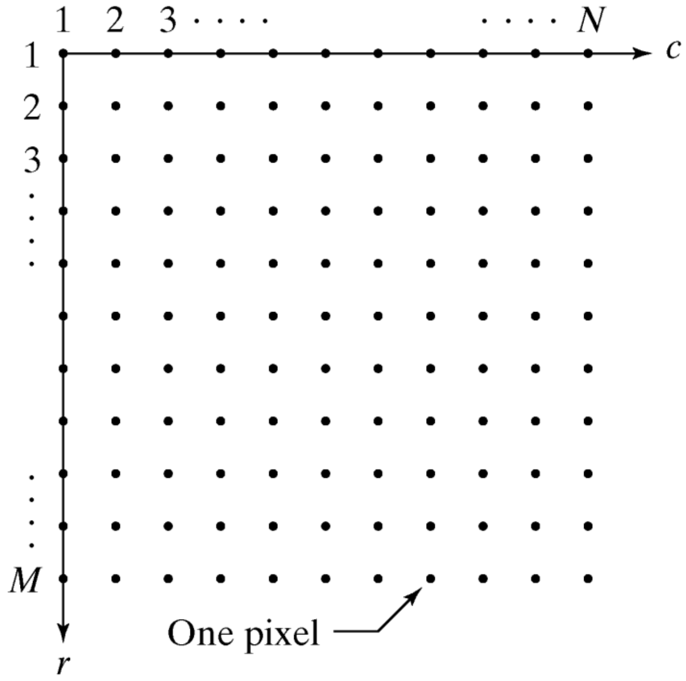
UNIDAD 1 – Fundamentos

Sistema de Coordenadas

Imagen digital

Sistema de coordenadas





Sistema de Coordenadas

Imagen digital $\Rightarrow$

$$f = \begin{bmatrix} f(1,1) & f(1,2) & \cdots & f(1,N) \\ f(2,1) & f(2,2) & \cdots & f(2,N) \\ \vdots & \vdots & & \vdots \\ f(M,1) & f(M,2) & \cdots & f(M,N) \end{bmatrix}$$

Imagen Digital $\Rightarrow$ Matriz M x N

M: nro. de filas

N: nro. de columnas

$f(m,n)$ $\Rightarrow$ pixel

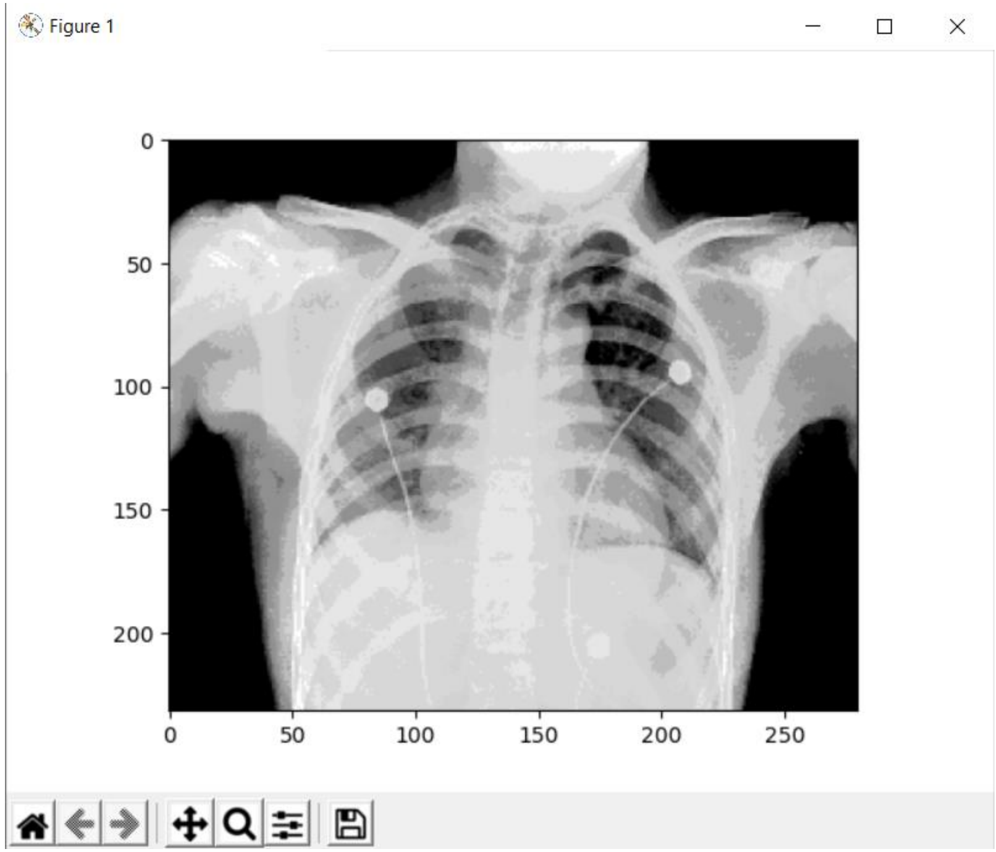
Formatos de Imagen

Format Name	Description	Recognized Extensions
TIFF	Tagged Image File Format	.tif, .tiff
JPEG	Joint Photographic Experts Group	.jpg, .jpeg
GIF	Graphics Interchange Format†	.gif
BMP	Windows Bitmap	.bmp
PNG	Portable Network Graphics	.png
XWD	X Window Dump	.xwd

Lectura y Visualización

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

img = cv2.imread('xray-chest.png',cv2.IMREAD_GRAYSCALE)
plt.imshow(img, cmap='gray')
plt.show()
```



Lectura y Visualización

```
plt.imshow(img, cmap='gray', vmin=low, vmax=high)
```

Muestra en negro los valores de intensidad menores o iguales que `low`, y en blanco los valores mayores o iguales que `high`.

```
plt.imshow(img, cmap='gray')
```

Setea como `low` al menor valor en `img`, y como `high` al máximo valor → **Mejora el rango dinámico!**

Lectura y Visualización

```
plt.imshow(img, cmap='gray')
```

```
plt.imshow(img, cmap='gray', vmin=0, vmax=255)
```

Escritura de Imágenes

```
img = cv2.imread('cameraman.tif',cv2.IMREAD_GRAYSCALE)
cv2.imwrite("cameraman.png", img)
cv2.imwrite("cameraman90.jpeg", img, [cv2.IMWRITE_JPEG_QUALITY, 90])
```

Donde “90” (q: quality) determina el grado de compresión jpeg (0 < q < 100).
Los detalles de una imagen pueden verse con el comando:

```
from PIL import Image
from PIL.ExifTags import TAGS

image = Image.open('cameraman.tif')
exifdata = image.getexif()
for tag_id in exifdata:
    tag = TAGS.get(tag_id, tag_id)
    data = exifdata.get(tag_id)
    print(f"{tag:25}: {data}")
```

```
ImageWidth               : 256
ImageLength              : 256
BitsPerSample            : 8
Compression              : 32773
PhotometricInterpretation: 1
ImageDescription         : This image is distributed by The MathWorks, Inc. with
permission from the Massachusetts Institute of Technology.
StripOffsets             : (8, 8262, 16426, 24578, 32492, 40499, 48599, 56637)
Orientation              : 1

                             ...
```

Escritura de Imágenes

Un factor de compresión alto (q bajo) introduce artefactos en la imagen.

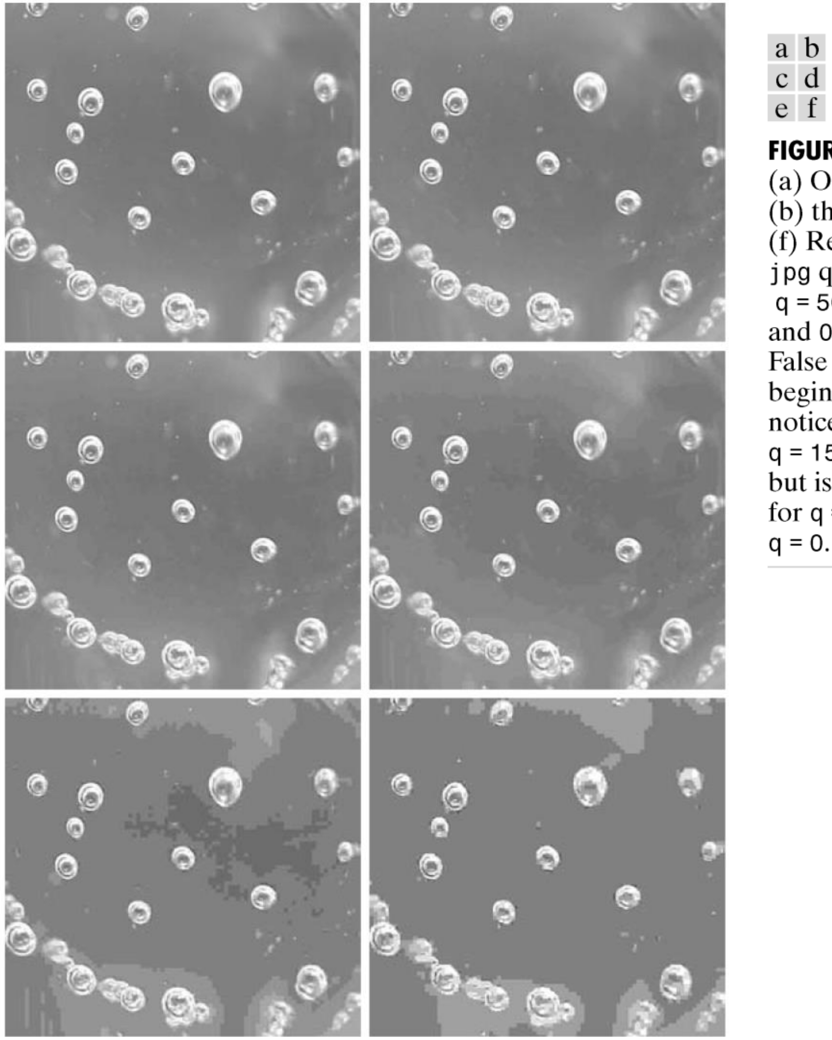


FIGURE 2.4 (a) Original image. (b) through (f) Results of using jpg quality values q = 50, 25, 15, 5, and 0, respectively. False contouring begins to be barely noticeable for q = 15 [image (d)] but is quite visible for q = 5 and q = 0.

Métricas de distorsión

$A_{M\times N}$ imagen original

$\tilde{A}_{M\times N}$ imagen comprimida

$$RMSE = \sqrt{\frac{1}{MN}\sum_{i=1}^{M}\sum_{j=1}^{N}\left[\tilde{A}(i,j) - A(i,j)\right]^2}$$

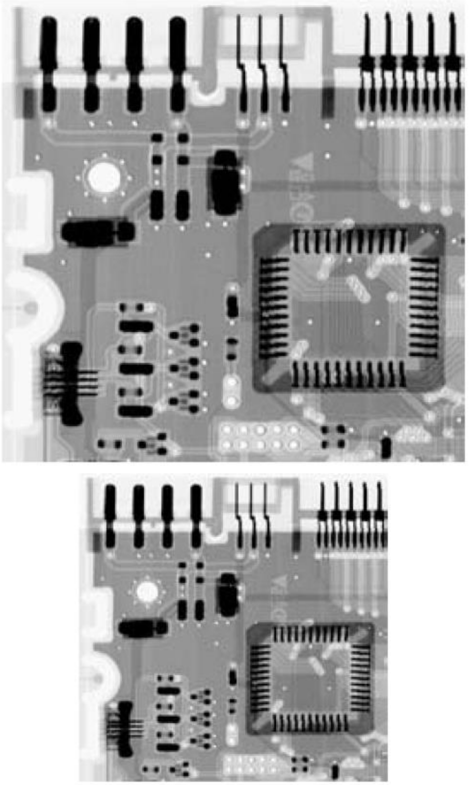
Root Mean Square Error (Métrica **no perceptual**)

Resolución

dpi: dots per inches

Si X es una imagen en formato jpg de 450 x 450 pixeles, con resolución 200 dpi, resulta en una imagen con dimensiones de 2.25 x 2.25 inches (pulgadas).

Manteniendo el número de pixeles pero ahora con resolución de 300 dpi, resulta en una imagen con dimensiones 1.5 x 1.5 inches.



a
b

FIGURE 2.5 Effects of changing the dpi resolution while keeping the number of pixels constant. (a) A 450 × 450 image at 200 dpi (size = 2.25 × 2.25 inches). (b) The same 450 × 450 image, but at 300 dpi (size = 1.5 × 1.5 inches). (Original image courtesy of Lixi, Inc.)

```
from PIL import Image
image = Image.open('cameraman.tif')
image.save('cameraman_dpi100.tif', dpi=(100,100))
```

Transformación de Imágenes mediante Indexado de arreglos

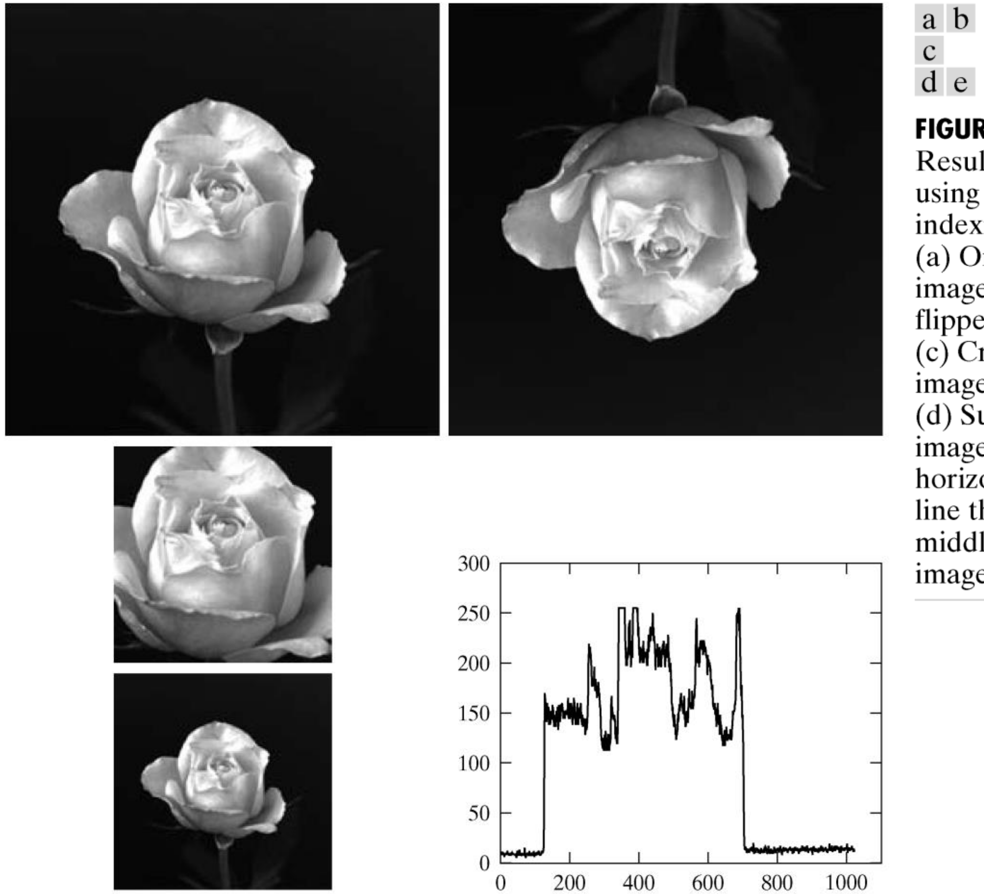


FIGURE 2.6 Results obtained using array indexing. (a) Original image. (b) Image flipped vertically. (c) Cropped image. (d) Subsampled image. (e) A horizontal scan line through the middle of the image in (a).

Tipo de datos

Name	Description
`double`	Double-precision, floating-point numbers in the approximate range -10^{308} to 10^{308} (8 bytes per element).
`uint8`	Unsigned 8-bit integers in the range [0, 255] (1 byte per element).
`uint16`	Unsigned 16-bit integers in the range [0, 65535] (2 bytes per element).
`uint32`	Unsigned 32-bit integers in the range [0, 4294967295] (4 bytes per element).
`int8`	Signed 8-bit integers in the range [−128, 127] (1 byte per element).
`int16`	Signed 16-bit integers in the range [−32768, 32767] (2 bytes per element).
`int32`	Signed 32-bit integers in the range [−2147483648, 2147483647] (4 bytes per element).
`single`	Single-precision floating-point numbers with values in the approximate range -10^{38} to 10^{38} (4 bytes per element).
`char`	Characters (2 bytes per element).
`logical`	Values are 0 or 1 (1 byte per element).

Tipo de datos - OpenCV

Nombre	Tipo de dato	Rango	cv::MAT
Unsigned 8 bits	uchar	0 ~ 255	CV_8UC1 | CV_8UC2 | CV_8UC3 | CV_8UC4
Signed 8 bits	char	-128 ~ 127	CV_8SC1 | CV_8SC2 | CV_8SC3 | CV_8SC4
Unsigned 16 bits	ushort	0 ~ 65.535	CV_16UC1 | CV_16UC2 | CV_16UC3 | CV_16UC4
Signed 16 bits	short	-32.768 ~ 32.767	CV_16SC1 | CV_16SC2 | CV_16SC3 | CV_16SC4
Signed 32 bits	int	-2147483648 ~ 2147483647	CV_32SC1 | CV_32SC2 | CV_32SC3 | CV_32SC4
Float 32 bits	float	-1.18e-38 ~ 3.40e-38	CV_32FC1 | CV_32FC2 | CV_32FC3 | CV_32FC4
Double 64 bits	double	-1.7e+308 ~ +1.7e+308	CV_64FC1 | CV_64FC2 | CV_64FC3 | CV_64FC4

C representa el número de canales →

- C1: 1 canal
- C2: 2 canales
- C3: 3 canales
- C4: 4 canales

Tipo de datos - Conversión

- Numpy:

```
x = np.array([1,2,3],dtype="uint8")
x16 = x.astype("int16")
```

- OpenCV:

  `convertScaleAbs(src,dst,alpha,beta)`

  Escala, calcula valor absoluto y convierte el resultado a 8 bits:

  ```
  dst = saturate_cast<uchar>(|src*alpha + beta|)
  ```

  `normalize(src,dst,alpha,beta,norm_type,dtype,mask)`

  Normaliza la escala y desplaza los datos de entrada.

Visualización de funciones de 2 variables

Función sinusoidal en 2D:

$$f(x, y) = A \sin\left(u_0 x + v_0 y\right)$$

$$x = 0, 1, 2, \cdots, M - 1$$

$$y = 0, 1, 2, \cdots, N - 1$$

```
nx, ny = (100, 100)
x = np.linspace(0, 8*np.pi, nx)
y = np.linspace(0, 8*np.pi, ny)
xv, yv = np.meshgrid(x, y)
z = np.sin(1*xv + 1*yv)
```