Imágenes

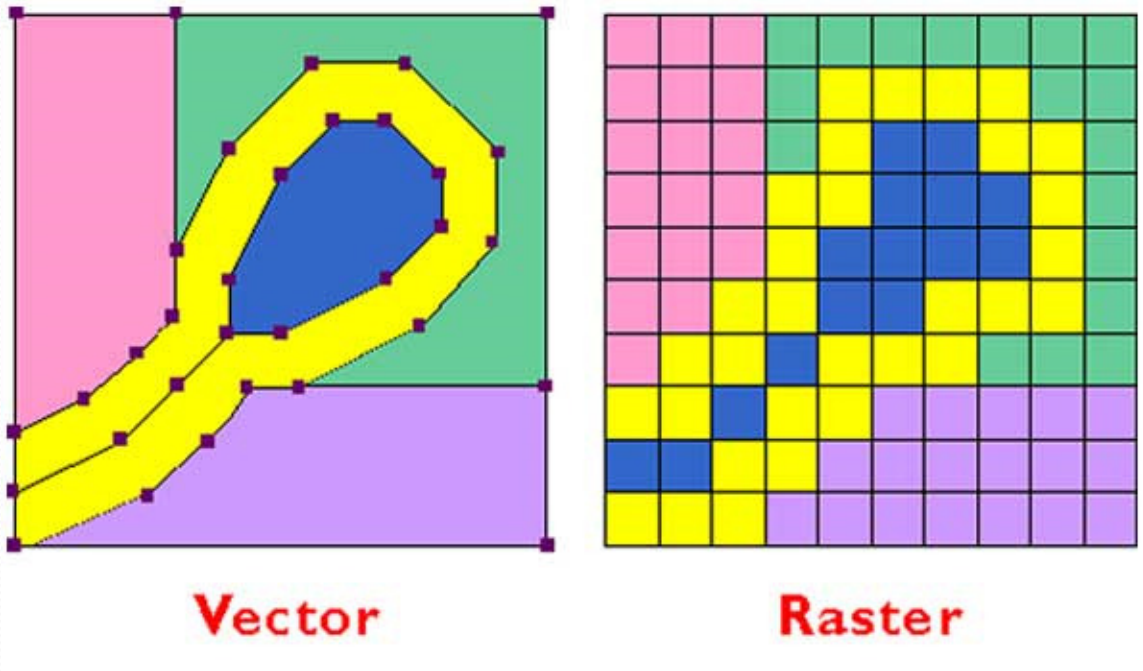
Tipos de imágenes

- **Imágenes raster.** Son imágenes compuestas de *elementos de imagen* (pixels, picture elements) dispuestas en una cuadrícula rectangular regular.
- También se les llama *mapas de bits* (*bitmaps*).
- **Imágenes vectoriales.** Son imágenes definidas en términos de puntos en un plano cartesiano, conectados por líneas y curvas para formar polígonos y otras formas.

Tipos de imágenes

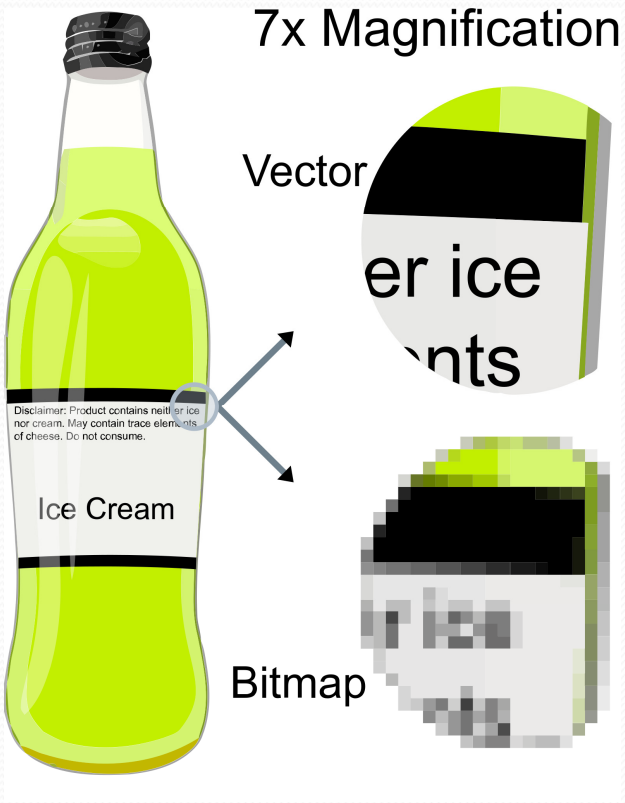
- Las imágenes vectoriales tienen la ventaja sobre las imágenes raster en que los puntos, líneas y curvas se pueden escalar a cualquier resolución sin alias.

Vector vs raster



Fuente: https://www.fastprint.co.uk/blog/raster-vs-vector-the-easy-to-understand-guide.html

Vector vs raster



- Fuente: https://en.wikipedia.org/wiki/File:VectorBitmapExample.svg

Vector vs raster

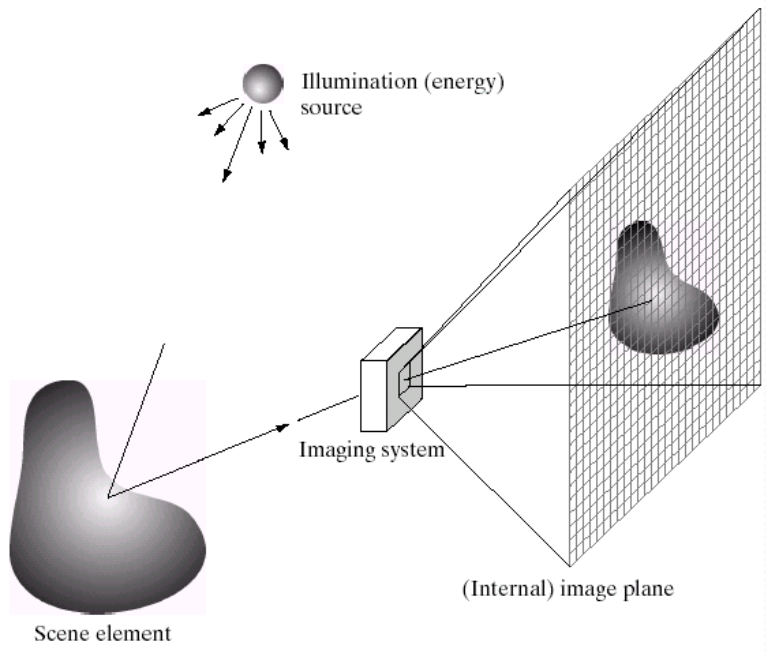
	Vector	Raster
Comprised of	multiple reference points and curves	pixels
Scalability	scalable without quality loss	loses quality when scaled
Convertibility	convertible to raster	not convertible to vector
File Formats	EPS, AI,SVG, PDF	BMP, JPG, GIF, PNG

Fuente: https://amadine.com/useful-articles/what-is-vector-graphics

Vector vs raster

- Ejemplos de formatos raster: JPEG, PNG, GIF y MPEG4.
- Ejemplos de formatos vectoriales: SVG, EPS, PDF y AI.
- Más información:
- https://en.wikipedia.org/wiki/Vector_image
- https://en.wikipedia.org/wiki/Raster_graphics
- https://en.wikipedia.org/wiki/Digital_image

Adquisición de imágenes



Fuente: Derek Hoiem (UIUC)

Cámaras digitales

- Utilizan un arreglo de sensores.
- Cada celda en el arreglo es un diodo sensible a la luz que convierte fotones a electrones.
- https://electronics.howstuffworks.com/cameras-photography/digital/digital-camera.htm

Imágenes raster

- Ejemplo de una imagen en tonos de gris de 8 bits.

$F(x, y)$

$\rightarrow$

148	123	52	107	123	162	172	123	64	89 ···
147	130	92	95	98	130	171	155	169	163 ···
141	118	121	148	117	107	144	137	136	134 ···
82	106	93	172	149	131	138	114	113	129 ···
57	101	72	54	109	111	104	135	106	125 ···
138	135	114	82	121	110	34	76	101	111 ···
138	102	128	159	168	147	116	129	124	117 ···
113	89	89	109	106	126	114	150	164	145 ···
120	121	123	87	85	70	119	64	79	127 ···
145	141	143	134	111	124	117	113	64	112 ···
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

$I(u, v)$

1.4 Image Acquisition

Fig. 1.5
The transformation of a continuous grayscale image $F(x, y)$ to a discrete digital image $I(u, v)$ (left), image detail (below).

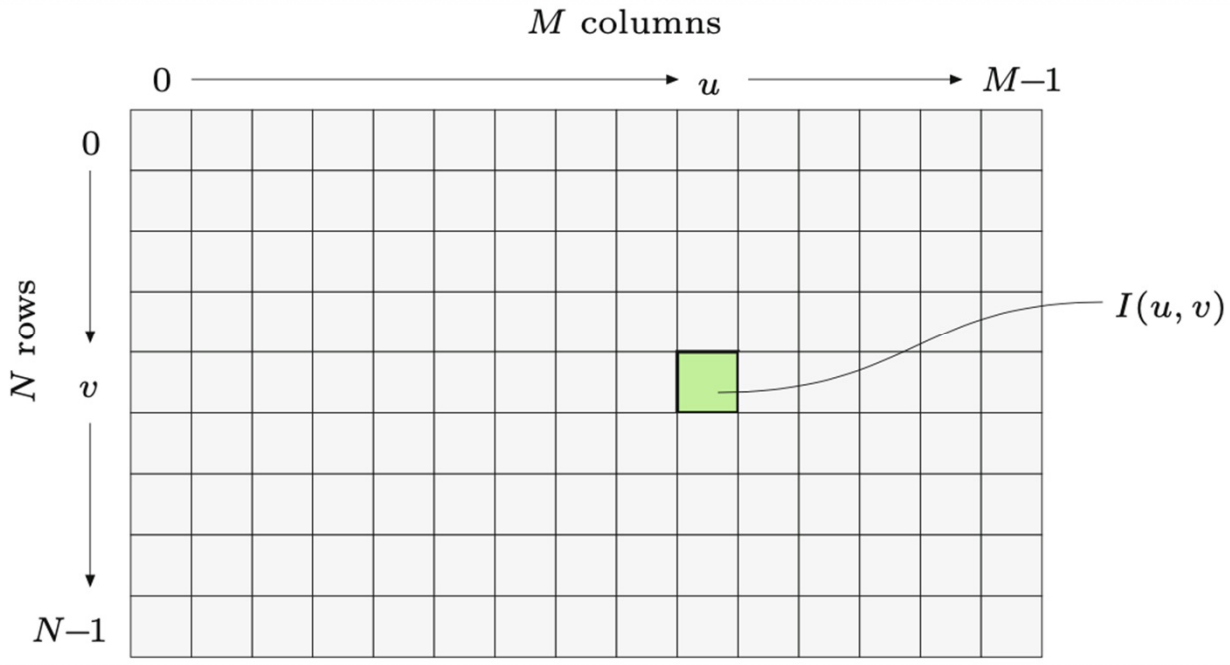
Sistema de coordenadas de imagen

- El origen está en la esquina superior izquierda.
- El eje x va de izquierda a derecha.
- El eje y va de arriba abajo.
- La numeración comienza en 0.

Sistema de coordenadas de imagen

1 Digital Images

Fig. 1.6
Image coordinates. In digital image processing, it is common to use a coordinate system where the origin ($u = 0$, $v = 0$) lies in the upper left corner. The coordinates u, v represent the columns and the rows of the image, respectively. For an image with dimensions $M \times N$, the maximum column number is $u_{max} = M - 1$ and the maximum row number is $v_{max} = N - 1$.



Imágenes como funciones discretas

- Una imagen digital I es una función 2D que mapea el dominio de coordenadas enteras $N \times N$ a un rango, $\mathbb{P}$, de posibles valores de píxel tal que
- $I(u, v) \in \mathbb{P}$ y $u, v \in \mathbb{N}$

Tamaño y resolución de imagen

- El **tamaño** de una imagen se determina directamente a partir del ancho M (número de columnas) y la altura N (número de renglones) de la matriz de imagen I.
- La **resolución** de una imagen especifica las dimensiones espaciales de la imagen en el mundo real.
- Se da como el número de pixeles por unidad de medición.
- Por ejemplo, puntos por pulgada (dpi), líneas por pulgada (lpi) para la producción de impresión, pixeles por kilómetro para imágenes de satélite.

Valores de pixel

- Los valores de pixel son números binarios de longitud k.
- Un pixel puede representar 2^k valores diferentes entre 0 y $2^k - 1$.
- El valor k se denomina profundidad de bit (o simplemente "profundidad") de la imagen.
- El valor k depende de si la imagen es a color o en tonos de gris.

Imágenes en escala de grises

- Tienen un solo canal que representa intensidad o brillo.
- Los valores de los pixeles son enteros entre 0 y $2^k - 1$.
- Las imágenes con $k = 8$ bits (1 byte) son comunes.
- Cada pixel ocupa un byte y tiene un valor entre 0 (negro) y 255 (blanco).
- Otras aplicaciones pueden usar una profundidad de $k = 12$ o hasta $k = 16$.

Imagen en tonos de gris de 8 bits

- Imagen en tonos de gris de 8 bits.

$F(x, y)$

$\rightarrow$

148	123	52	107	123	162	172	123	64	89 ···
147	130	92	95	98	130	171	155	169	163 ···
141	118	121	148	117	107	144	137	136	134 ···
82	106	93	172	149	131	138	114	113	129 ···
57	101	72	54	109	111	104	135	106	125 ···
138	135	114	82	121	110	34	76	101	111 ···
138	102	128	159	168	147	116	129	124	117 ···
113	89	89	109	106	126	114	150	164	145 ···
120	121	123	87	85	70	119	64	79	127 ···
145	141	143	134	111	124	117	113	64	112 ···
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮

$I(u, v)$

1.4 Image Acquisition

Fig. 1.5
The transformation of a continuous grayscale image $F(x, y)$ to a discrete digital image $I(u, v)$ (left), image detail (below).

Imágenes binarias

- Es un caso especial de una imagen en escala de grises con $k = 1$.
- Cada pixel solo puede ser negro o blanco.
- Es costumbre que el 0 sea blanco y el 1 negro.
- Se usa para archivar documentos, transmisiones fax e impresiones.

Imágenes a color

- La mayoría de las imágenes en color se basan en los colores primarios rojo, verde y azul (RGB).
- Normalmente utilizan 8 bits para cada componente de color.
- Cada pixel requiere 3 x 8 = 24 bits (3 bytes) para codificar los tres componentes.
- El rango de cada componente de color individual es [0, 255].
- En el modelo RGB un pixel puede valer desde (0, 0, 0) hasta (255, 255, 255), para un total de 256^3 = 16,777,216 colores.

Imágenes a color

- El modelo RGB se puede extender al agregar otro componente A (alpha) de 8 bits para representar la opacidad o transparencia del color
- El valor de $A = 0$ indica que el color es transparente, $A = 255$ indica que el color es opaco.
- En RGBA cada pixel requiere 32 bits (4 bytes).

Profundidades comunes

Table 1.1 Bit depths of common image types and typical application domains.

Grayscale (Intensity Images):

Chan.	*Bits/Pix.*	*Range*	*Use*
1	1	$[0, 1]$	Binary image: document, illustration, fax
1	8	$[0, 255]$	Universal: photo, scan, print
1	12	$[0, 4095]$	High quality: photo, scan, print
1	14	$[0, 16383]$	Professional: photo, scan, print
1	16	$[0, 65535]$	Highest quality: medicine, astronomy

Color Images:

Chan.	*Bits/Pix.*	*Range*	*Use*
3	24	$[0, 255]^3$	RGB, universal: photo, scan, print
3	36	$[0, 4095]^3$	RGB, high quality: photo, scan, print
3	42	$[0, 16383]^3$	RGB, professional: photo, scan, print
4	32	$[0, 255]^4$	CMYK, digital prepress

Special Images:

Chan.	*Bits/Pix.*	*Range*	*Use*
1	16	$[-32768, 32767]$	Integer values pos./neg., increased range
1	32	$\pm 3.4 \cdot 10^{38}$	Floating-point values: medicine, astronomy
1	64	$\pm 1.8 \cdot 10^{308}$	Floating-point values: internal processing

Ordenamiento de los pixeles

- Hay dos métodos para ordenar los componentes de color en imágenes de color: *ordenamiento planar* y *ordenamiento empacado*.

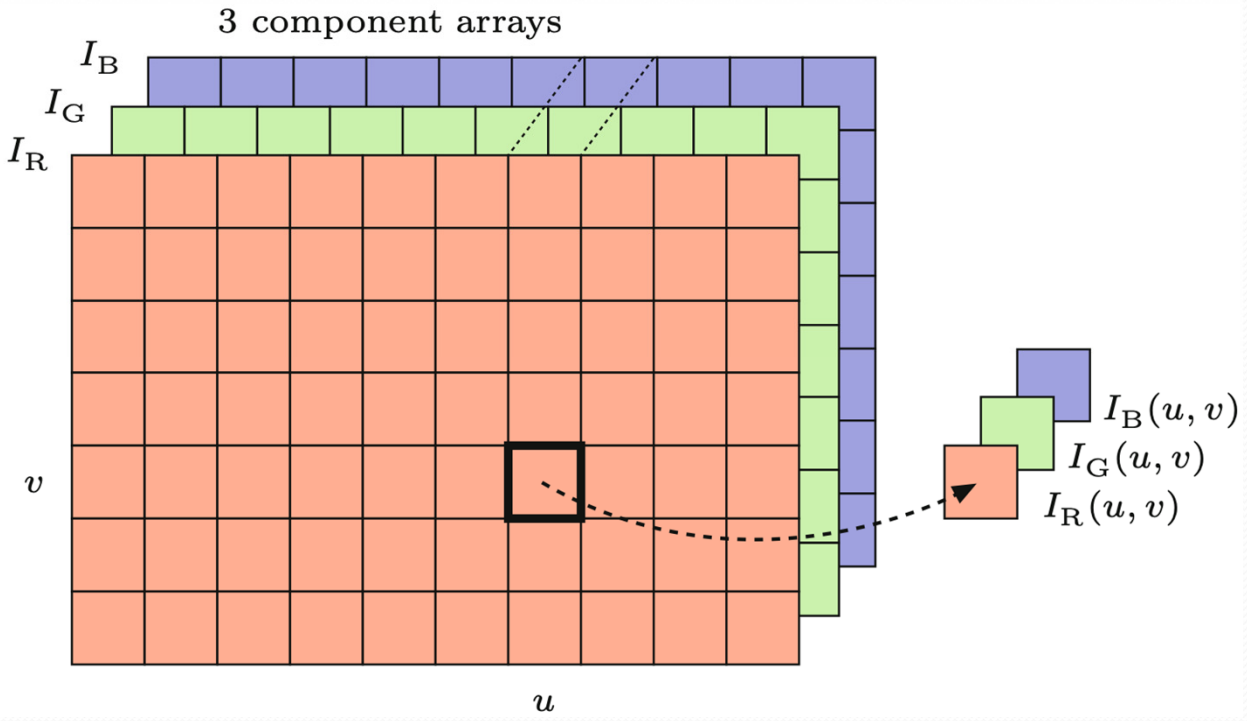
Ordenamiento planar

- Los componentes de color se guardan en arreglos separados con idénticas dimensiones.
- En Python, una imagen a color se guarda en una matriz $H \times W \times 3$.

Ordenamiento planar

12 COLOR IMAGES

Fig. 12.3
RGB color image in component ordering. The three color components are laid out in separate arrays I_R, I_G, I_B of the same size.



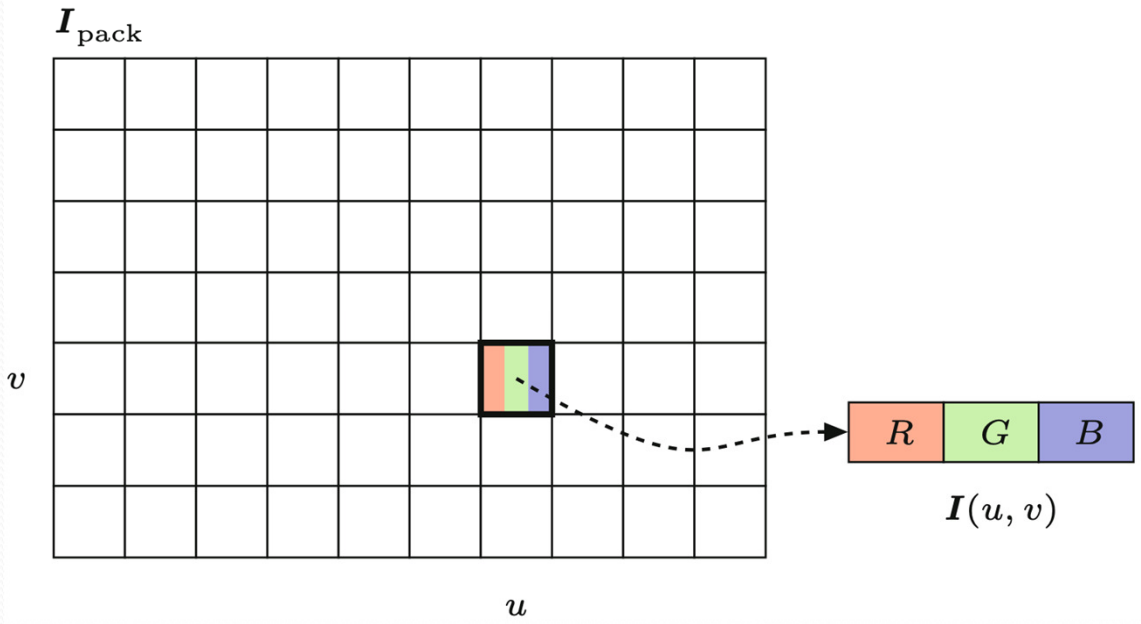
Imágenes en Python

```
image = cv2.imread(filename)                    # lee una imagen BGR
# image es una matriz H x W x 3 (numpy.ndarray)
image[0, 0, 0]   # valor del pixel superior izquierdo en el canal B
image[y, x, c]   # y + 1 pixeles abajo, x + 1 pixeles a la derecha en el canal c
image[H - 1, W - 1, 2]   # valor del pixel inferior derecho en el canal R
```

Ordenamiento empacado

- Los valores de los componentes que representan el color de un pixel se empacan en un solo elemento de la matriz.
- En Java, una imagen a color se guarda en una matriz $H \times W$.

Ordenamiento empacado

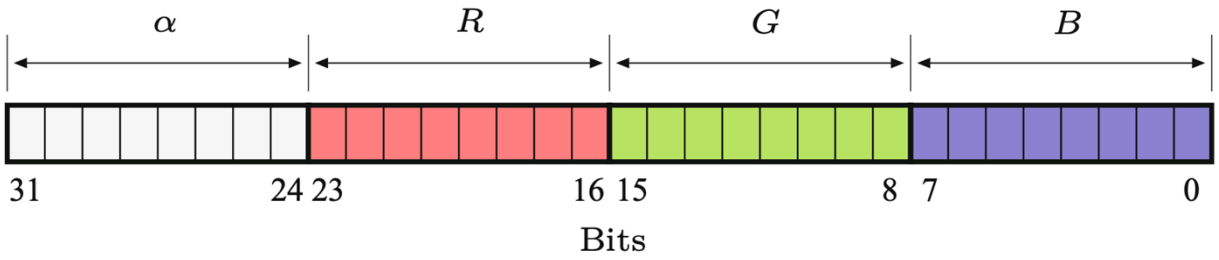


12.1 RGB Color Images

Fig. 12.4
RGB-color image using packed ordering. The three color components R, G, and B are placed together in a single array element.

RGB empacado en Java

Fig. 12.6
Structure of a packed RGB color pixel in Java. Within a 32-bit `int`, 8 bits are allocated, in the following order, for each of the color components R, G, B, and the transparency value α (unused in ImageJ).



RGB empacado en Java

- Dado un pixel, los canales de color individuales se accesan mediante operaciones AND (&) y corrimientos a la derecha (>>).

```
int color = getPixel(u, v);
int red = (c & 0xff0000) >> 16;          // red component
int green = (c & 0x00ff00) >> 8;         // green component
int blue = (c & 0x0000ff);               // blue component
```

RGB empacado en Java

- Dados los colores individuales, un pixel de 32 bits se empaca mediante operaciones AND (&), OR (|) y corrimientos a la izquierda (<<).

```
int color = ((red & 0xff) << 16 | ((green & 0xff) << 8) | (blue & 0xff);
putPixel(u, v, color);
```

Operaciones de pixel

- Las operaciones de pixel realizan una modificación de los valores de los pixeles sin cambiar el tamaño, la geometría o la estructura local de la imagen.
- Cada nuevo valor de pixel $b = I'(u, v)$ depende exclusivamente del valor anterior $a = I(u, v)$ en la misma posición.

Operaciones de pixel

- Operaciones de pixel básicas:

1. Modificar la intensidad de la imagen.
2. Invertir la imagen.
3. Operación umbral.

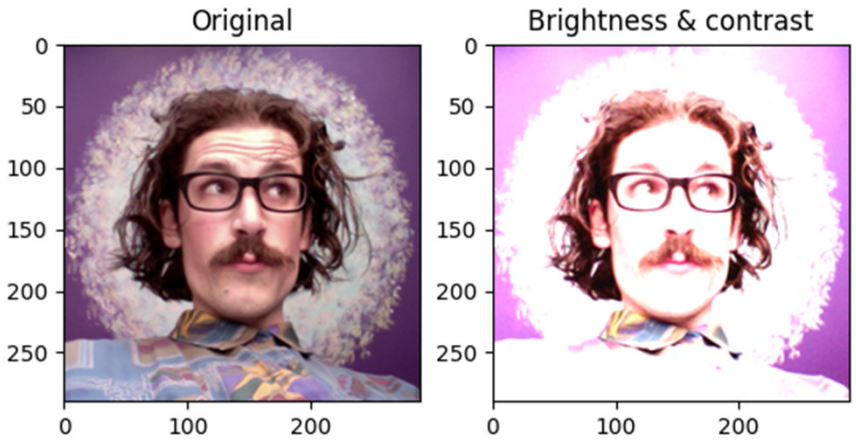
Modificar la intensidad de la imagen

- Consiste en cambiar los valores de los pixeles.
- Ejemplo: incrementar la intensidad en un 50% se expresa así:
- $f(a) = a * 1.5$
- Donde a es la intensidad original del pixel.
- También se puede incrementar la intensidad en, por ejemplo, 10 unidades:
- $f(a) = a + 10$

Ejemplo

```
# File: image_intensity.py
# Adjust the brightness and contrast
brightness = 10        # adds 10 to each pixel value
contrast = 2.3         # scales the pixel values by 2.3
image = cv.addWeighted(image, contrast,
                       np.zeros(image.shape, image.dtype), 0,
                       brightness)
```

Resultado



Invertir una imagen

- Para un valor de pixel $a = I(u, v)$ en el rango $[0, a_{max}]$ la operación de pixel es
- $f_{inv}(a) = a_{max} - a$
- Para una imagen en escala de grises de 8 bits $a_{max} = 255$.
- En una imagen a color, la inversión se hace para cada canal.

Ejemplo en tonos de grises

```
# File: invert_grayscale.py
import cv2 as cv

image = cv.imread('MyPic.png', cv.IMREAD_GRAYSCALE)
image_inv = cv.bitwise_not(image)
cv.imwrite('MyPicInvGray.png', image_inv)
```

Resultado

Ejemplo a colores

```
# File: invert_color.py
import cv2 as cv

image = cv.imread('MyPic.png')
image_inv = cv.bitwise_not(image)
cv.imwrite('MyPicInvColor.png', image_inv)
```

Resultado

Operación umbral

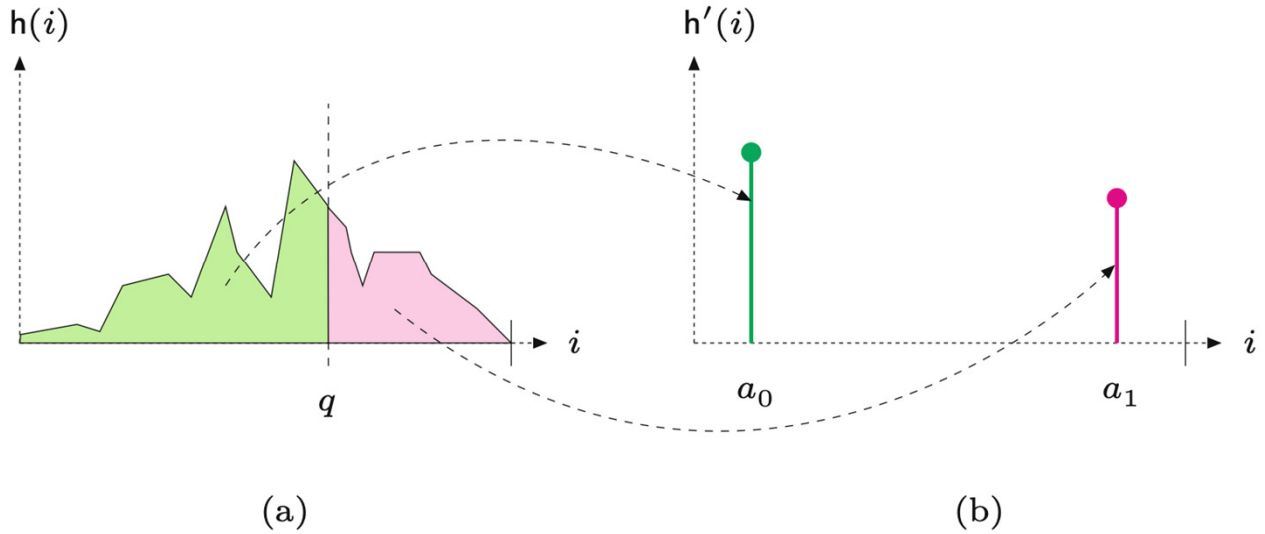
- La operación umbral separa los pixeles en dos clases, dependiendo de un umbral q que suele ser una constante.
- La operación asigna todos los pixeles a uno de dos valores de intensidad fijos a_0 o a_1, es decir,
- $$f_{\text{threshold}}(a) = \begin{cases} a_0 & \text{para } a < q \\ a_1 & \text{para } a \geq q \end{cases}$$
- Con $0 < q \leq a_{max}$.

Operación umbral

- Una aplicación común es binarizar una imagen con los valores $a_0 = 0$ y $a_1 = 255$.
- La operación umbral afecta al histograma al separar la distribución en dos entradas en las posiciones a_0 y a_1.

Operación umbral

Fig. 4.2
Effects of thresholding upon the histogram. The threshold value is a_{th}. The original distribution (a) is split and merged into two isolated entries at a_0 and a_1 in the resulting histogram (b).



Ejemplo

```
# File: threshold.py
import cv2 as cv

image = cv.imread('MyPic.png')
image_gray = cv.cvtColor(image, cv.COLOR_BGR2GRAY)
# above 150 to white; 150 and below to black
_, dst = cv.threshold(image_gray, 150, 255, cv.THRESH_BINARY)
cv.imwrite('MyPicBW.png', dst)
```

Resultado