Procesamiento de imágenes con MATLAB

MATLAB (abreviatura de *MATrix LABoratory*, "laboratorio de matrices") es un software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas de Unix, Windows y Apple. Entre sus prestaciones básicas se hallan: la manipulación de matrices, la representación de datos y funciones, la implementación de algoritmos, la creación de interfaces de usuario (GUI) y la comunicación con programas en otros lenguajes. Además, se pueden ampliar las capacidades de MATLAB con las *cajas de herramientas* (*toolboxes*). Para el caso de manipulación de imágenes se emplea el *toolbox "Image Processing"*.

1. Introducción:

Leer una imagen:

>> f = imread ('imagen1.jpg')

>> f = imread ('C:\imagenes\imagen1.jpg')

La imagen a leer debe encontrarse en la carpeta de trabajo de Matlab. Los formatos de imagen soportados por Matlab son: TIFF, JPEG, GIF, BMP, PNG, XWD.

Una imagen de color RGB se representa por tres matrices bidimensionales, correspondientes a los planos R, G y B. Para obtener los planos RGB se ejecutan los comandos:

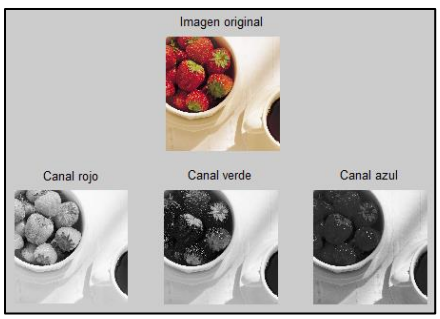
>> Im_R = f(:,:,1)

>> Im_G = f(:,:,2)

>> Im_B = f(:,:,3)

Por ejemplo:

```
>> f=imread('fresas.tif');
>> f_R=f(:,:,1);
>> f_G=f(:,:,2);
>> f_B=f(:,:,3);
```



>> [M,N] = size (f)

M = número de filas, N = número de columnas

>> imshow (f, G)

Muestra la imagen f en pantalla donde G especifica el rango de intensidades. Si omitimos G, se muestra la imagen con 256 niveles de gris por defecto.

>> impixel(f,i,j)

Acceder al pixel de coordenadas (i,j) de la imagen f.

>> whos f

Información adicional de la imagen: nombre, tamaño, bytes y clase.

```
>> whos f
  Name        Size                Bytes  Class    Attributes

  f         300x300x3            270000  uint8
```

>> imwrite (f, 'C:\imagenes\imagen2.tif')

Guarda la imagen f con nombre imagen2.tif en C:\imagenes.

El tipo de dato matriz, que contendrá una imagen puede ser de varios tipos (según el tipo de dato de cada pixel):

- **double**: Doble precisión, números en punto flotante que varían en un rango aproximado de -10308 a 10308 (8 bytes por elemento)
- **uint8**: Enteros de 8 bits en el rango de [0,255] (1 byte por elemento)
- **uint16**: Enteros de 16 bits en el rango de [0, 65535] (2 bytes por elemento)
- **uint32**: Enteros de 32 bits en el rango de [0, 4294967295] (4 bytes por elemento)
- **int8**: Enteros de 8 bits en el rango de [-128, 127] (1 byte por elemento)
- **int16**: Enteros de 16 bits en el rango de [-32768, 32767] (2 bytes por elemento)
- **int32**: Enteros de 32 bits en el rango de [-2147483648,2147483647] (4 bytes por elemento)
- **logical:** Los valores son 0 ó 1 (1 bit por elemento)

Conversión entre tipos de datos: Para ciertas operaciones es necesario convertir una imagen de su tipo original a otro tipo de imagen que facilite su procesamiento. Algunos métodos:

Comando	Descripción
gray2ind	Crea una imagen indexada a partir de una imagen de intensidad en escala de gris.
im2bw	Crea una imagen binaria a partir de una imagen de intensidad, imagen indexada o RGB basado en un umbral de luminancia.
ind2rgb	Crea una imagen RGB a partir de una imagen indexada
rgb2gray	Crea una imagen de intensidad en escala de gris a partir de una imagen RGB
rgb2ind	Crea una imagen indexada a partir de una imagen RGB

Comandos de conversión de imágenes en Matlab

Por ejemplo:

```
>> f=imread('baby.tif');
>> g=rgb2gray(f);
>> imshow(g)
```

Algunos comandos que pueden utilizarse para determinar el tipo de imagen con que se está trabajando:

Comando	Descripción
isbw	Regresa un valor verdadero (1) si la imagen es binaria
isgray	Regresa un valor verdadero (1) si la imagen es de intensidad
isind	Regresa un valor verdadero (1) si la imagen es indexada
isrgb	Regresa un valor verdadero (1) si la imagen es RGB
imfinfo	Regresa información sobre la imagen

Comandos Informativos sobre imágenes

Selección de una sección de una imagen:

>> g=imcrop(f)

Para seleccionar la región que se va a cortar, simplemente arrastre el ratón y forme un rectángulo sobre la región deseada. Cuando se suelta el botón del ratón, el comando regresa la sección seleccionada al argumento de salida especificado (g en este caso). También es posible seleccionar la sección de interés de forma no interactiva. En este caso se debe especificar el rectángulo de la siguiente forma:

>> g=imcrop(f,[xmin ymin ancho alto])

donde xmin y ymin forman el punto de la esquina superior izquierda de la región a seleccionar.

Por ejemplo:

```
>> f=imread('baby.tif');
g=imcrop(f);
imshow(g)
```

Manejo de ventanas:

MATLAB dispone de algunas funciones básicas para crear y manipular ventanas, entre las principales tenemos:

SUBPLOT

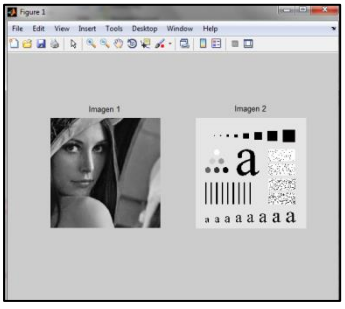
Divide la ventana gráfica en varias subventanas. Su sintaxis es:

>> subplot(m,n,p)

donde la ventana se divide en m filas y n columnas y hace que la subventana p sea la actual. Las ventanas se numeran de izquierda a derecha y de arriba hacia abajo.

Por ejemplo:

```
>> I=imread('Lenna.tif');
subplot(1,2,1);imshow(I);
title('Imagen 1');
J=imread('test.tif');
subplot(1,2,2);imshow(J);
title('Imagen 2');
```



2. Transformaciones de intensidad:

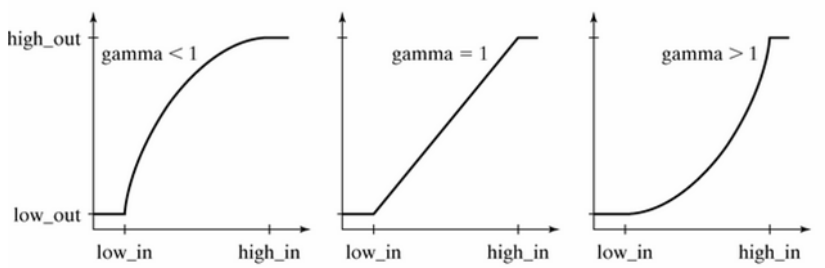
La sintáxis general para aplicar una transformación de intensidad a una imagen f es:

>> g = imadjust (f, [low_in high_in], [low_out high_out])

Aplica los valores de intensidad de la imagen f (uint8, uint16 o doublé) que están en el intervalo [low_in high_in] en valores del intervalo, [low_out high_out] (considerando valores entre 0 y 1).

Transformación gamma:

>> g = imadjust (f, [low_in high_in], [low_out high_out], gamma)



Por ejemplo:

```
>> g = imadjust (f, [0 1], [1 0]);
```

Transformación log:

>> g = c * log (1 + double(f))

Por ejemplo:

```
>> f=imread('Lenna.tif'); %Imagen uint8
>> f1=double(f); % Imagen double
>> g=log(1+f1);% Imagen double
>> g1=mat2gray(g); % Imagen double con rango [0,1]
>> g2=im2uint8(g1); % Imagen uint8
>> subplot(1,2,1);imshow(f);title('Imagn original');
subplot(1,2,2);imshow(g2);title('Transformación log');
```

Histograma de la imagen:

>> imhist (f,n)

Donde n es el número de intervalos (subdivisiones de la escala de intensidad) usados para formar el histograma, cuyo valor por defecto es 256.

>> imhist (f,n)/numel(f)

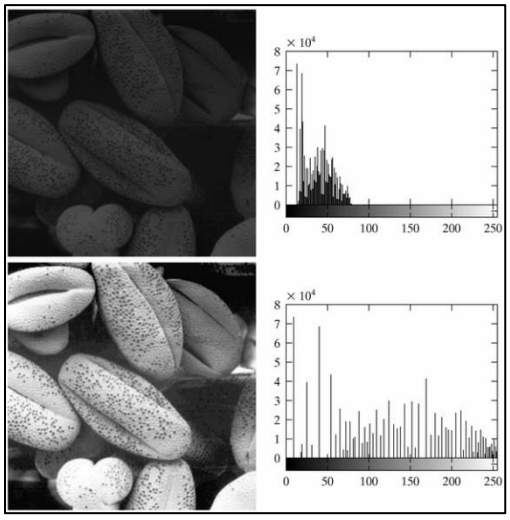
Histograma normalizado

>> g = histeq(f, n)

Histograma ecualizado, donde n indica el número de niveles de intensidad específico para la imagen de salida, por ejemplo, n=256.

Por ejemplo:

```
>> f=imread('imagen.tif');
imhist(f);
histeq(f,256);
```



3. Filtros espaciales:

>> g = imfilter(f, w, 'replicate')

- f es la imagen de entrada
- w es la máscara
- 'replicate': el tamaño de la imagen se aumenta duplicando los valores del borde. Hay más opciones para tratar el borde.

Las máscaras se pueden definir manualmente (como matriz). Por ejemplo:

>> w = 1/9*[1 1 1;1 1 1;1 1 1]; o bien

>> w = 1/9*ones(3);

Máscaras implementadas en Matlab:

TIPO	SINTÁSIS Y PARÁMETROS
'average'	**>> fspecial('average',[r c])** filtro de la media r x c. Por defecto, aplica el filtro 3 x 3. Si ponemos un solo número r indica que es una máscara cuadrara r x r.
'gaussian'	**>> fspecial('gaussian',[r c],sigma)** filtro gaussiano r x c con desviación típica sigma. Por defecto, aplica filtro gaussiano 3 x 3 con sigma=0,5.
'prewitt'	**>> fspecial('prewitt')** filtro gradiente Prewitt 3 x 3. Devuelve la máscara mx que aplica el gradiente vertical. La máscara que aplica el gradiente horizontal se obtiene con la traspuesta de la anterior, es decir, **>> my=mx'**.
'sobel'	**>> fspecial('sobel')** filtro gradiente Sobel 3 x 3. Devuelve la máscara mx que aplica el gradiente vertical. La máscara que aplica el gradiente horizontal se obtiene con la traspuesta de la anterior, es decir, **>> my=mx'**.
'laplacian'	**>> fspecial('laplacian', alpha)** filtro laplaciano 3 x 3 cuya forma viene dada por alpha (valor entre 0 y 1). Por defecto, alpha = 0,5.

Filtro de la mediana:

>> g = medfilt2(f, [m n]);

[m n] = tamaño de la máscara (por defecto 3x3).

Añadir **ruido** a una imagen:

>> g = imnoise(f, tipo de ruido, parámetros)

Valor	Descripción
'gaussian'	Ruido Gaussiano
'poisson'	Ruido de Poisson
'salt & pepper'	Sal y pimienta
'speckle'	Ruido Multiplicativo

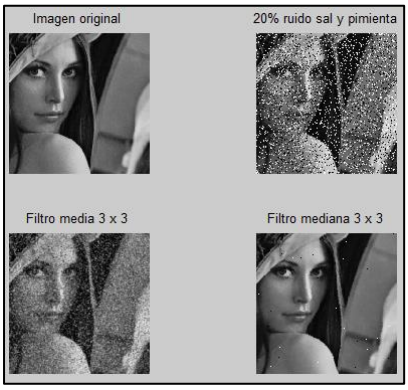
Tipos de ruidos en Matlab

Por ejemplo:

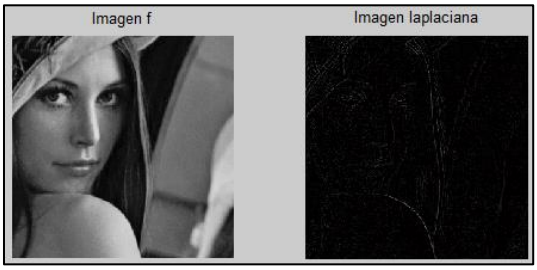
>> g = imnoise(f, 'salt & pepper',0.2)

Añade un 20% de ruido sal y pimiento.

```
>> f=imread('Lenna.tif');
>> g=imnoise(f,'salt & pepper',0.2);
>> m1=fspecial('average',3); g);
>> f_media=imfilter(g,m1);   '(g);
>> f_mediana = medfilt2(g, [3 3] );
```




```
>> f=imread('Lenna.tif');
>> m=fspecial('laplacian',0);
>> g=imfilter(f,m,'replicate');
```



4. Transformada Discreta de Fourier:

>> F=fft2 (f)

Transformada discreta de Fourier de la imagen f, luego es una matriz de valores complejos.

>> abs (F)

Magnitud de los valores complejos de la TDF (espectro de Fourier).

>> fftshift (F)

TDF desplazada (F(0,0) en el centro de frecuencias).

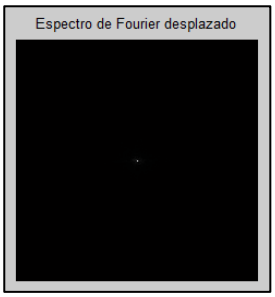
>> f=ifft2 (F)

TDF inversa.

Ejemplo:

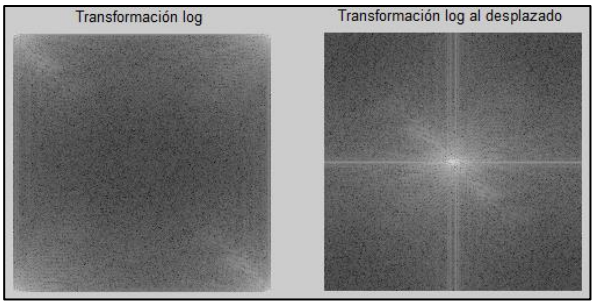
```
>> F=fft2(f);
>> E=abs(F);
>> E1=mat2gray(E);
>> E2=im2uint8(E1);
```


```
>> FT=fftshift(F);
>> ET=abs(FT);
>> E1T=mat2gray(ET);
>> E2T=im2uint8(E1T);
```




```
>> g=ifft2(F);
>> g1=mat2gray(g);
>> g2=im2uint8(g1);
```


```
>> L=log(1+E);
>> L1=mat2gray(L);
>> L2=im2uint8(L1);
>> LT=log(1+double(ET));
>> L1T=mat2gray(LT);
>> L2T=im2uint8(L1T);
```



5. Morfología:

Dilatación:

>> imdilate (f,SE)

Erosión:

>> imdilate (f,SE)

Apertura:

>> imopen (f,SE)

Clausura:

>> imclose (f,SE)

SE indica el elemento estructural que puede definirse a partir de la función **STREL**:

>> strel('diamond',R)

>> strel('disk',R)

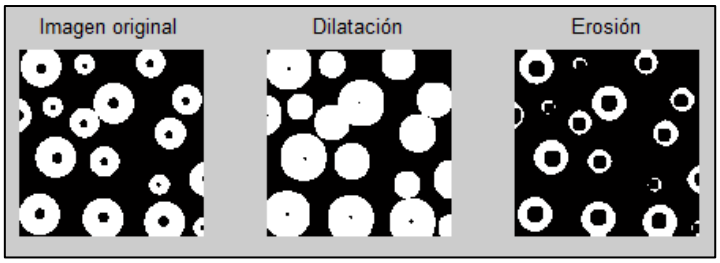
>> strel('line',length, angle)

>> strel('rectangle',MN), MN=vector de dos elementos.

>> strel('square',W)

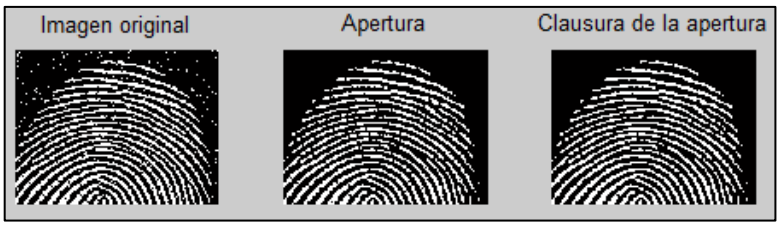
Ejemplo 1: Dilatación y erosión

```
>> f=imread('circulos.tif');
>> SE=strel('square',15);
>> fdil=imdilate(f,SE);
>> fero=imerode(f,SE);
```



Ejemplo 2: Apertura y clausura

```
>> f=imread('huella.tif');
>> SE=strel('square',3);
>> fo=imopen(f,SE);
>> fc=imclose(fo,SE);
```



Transformada hit-or-miss:

>> bwhitmiss(f,B)

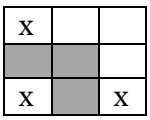
donde f es la imagen de entrada y B es una matriz conteniendo los valores -1, 0 y 1, tales que si B=(J,K),entonces:

vale 1 en los píxeles negros de J

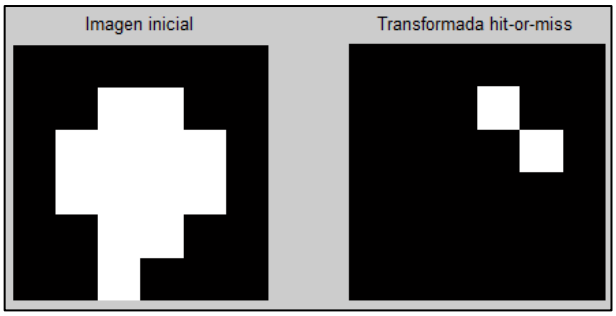
vale -1 en los píxeles negros de K

vale 0 cuando no importa el valor del pixel.

Por ejemplo: Sea B la siguiente configuración




```
>> f=[0 0 0 0 0 0;0 0 1 1 0 0;0 1 1 1 1 0;
0 1 1 1 1 0;0 0 1 1 0 0;0 0 1 0 0 0];
>> B=[0 -1 -1;1 1 -1;0 1 0];
>> g = bwhitmiss(f,B);
```

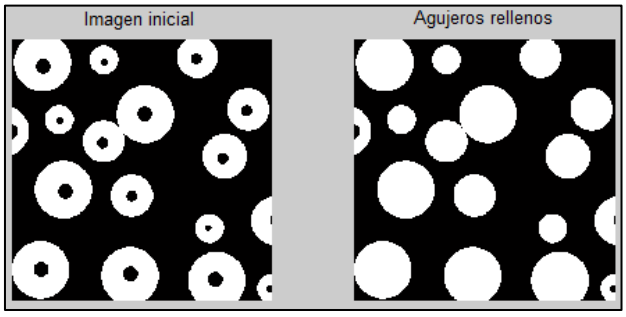


Relleno de agujeros:

>> imfill(f,'holes')

Ejemplo:

```
>> f=imread('circulos.tif');
>> g=imfill(f,'holes');
```

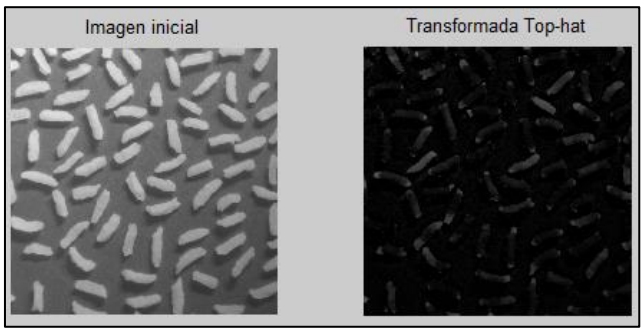


Transformada de top-hat:

>> imtophat(f,SE)

Ejemplo:

```
>> f=imread('arroz.tif');
>> SE=strel('disk',10);
>> g=imtophat(f,SE);
```



6. Segmentación:

>> [g, t] = edge(f, 'método', parámetros)

La salida g es una matriz logical con 1 en los píxeles que el método detecta como borde y 0 en caso contrario. La salida T es opcional, indica que el umbral que ha usado el método para obtener el borde.

Sobel:

>> [g, t] = edge(f, 'sobel', T, dir)

T umbral específico, en cuyo caso t=T.

dir = 'horizaontal', 'vertical' o 'both'.

Prewitt:

>> [g, t] = edge(f, 'prewitt', T, dir)

Roberts:

>> [g, t] = edge(f, ' roberts', T, dir)

LoG:

>> [g, t] = edge(f, 'log', T, sigma)

sigma = desviación estándar (por defecto, sigma=2).

Canny:

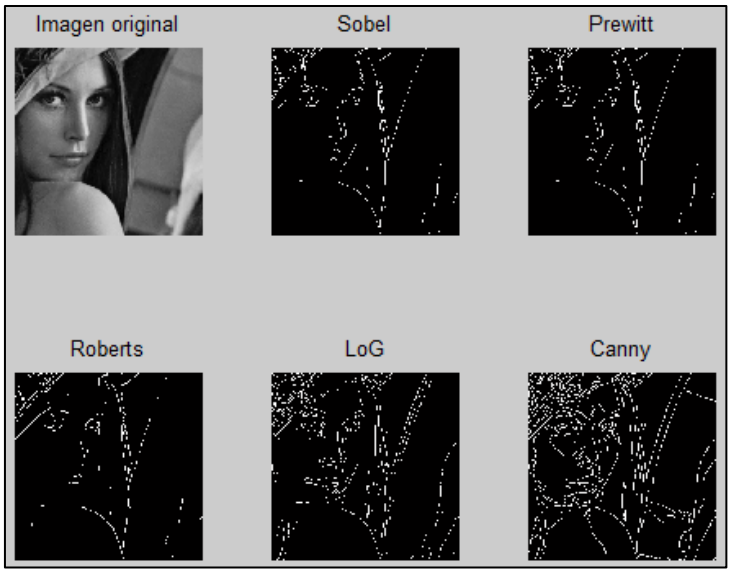
>> [g, t] = edge(f, 'canny', T, sigma)

T=[T1, T2] umbrales específicos.

sigma = desviación estándar del filtro de suavizado.

Ejemplo:

```
>> g=imread('Lenna.tif');
[h1,t1]=edge(g,'sobel');
[h2,t2]=edge(g,'prewitt');
[h3,t3]=edge(g,'roberts');
[h4,t4]=edge(g,'log');
[h5,t5]=edge(g,'canny');
```



Umbralización:

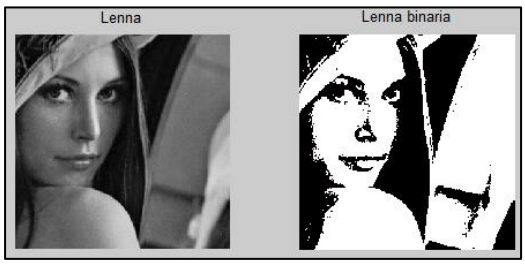
>> im2bw(f, T)

Donde T es el umbral tal que todos los niveles de intensidad por debajo de T los hace 0 y todos los que son mayores que T los hace 1. Dicho umbral se puede obtener con la siguiente función la cual aplica el método de Otsu:

>> T=graythresh(f)

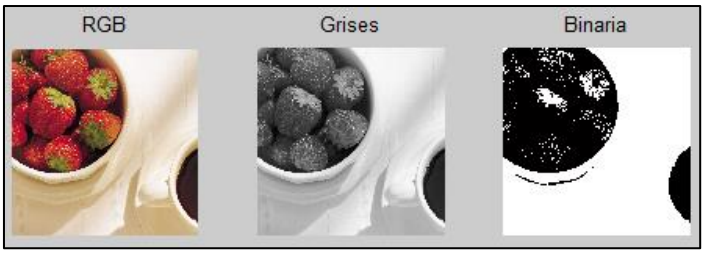
Ejemplo 1:

```
>> f=imread('Lenna.tif');
>> T=graythresh(f);
>> fbinaria=im2bw(f,T);
```



Ejemplo 2:

```
>>  f=imread('fresas.tif');
fgris=rgb2gray(f);
T=graythresh(fgris);
fbinaria=im2bw(fgris,T);
```



7. Representación y descripción:

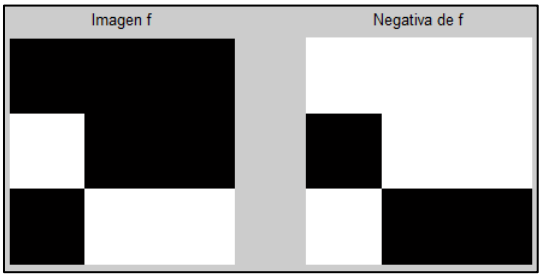
Número de componentes conexas:

>> g = bwlabel(f, n)

donde n indica la adyacencia usada, es decir, n=4 o n=8. Esta función cuenta el número de componentes conexas blancas luego, si consideramos que el objeto es lo negro, debemos aplicar bwlabel a la imagen negativa de f.

Ejemplo:

```
>> f=[0 0 0;1 0 0;0 1 1];
>> [L,n]=bwlabel(f,4);
>> L

L =

     0     0     0
     1     0     0
     0     2     2

>> fnegativa=not(f);
>> [Lneg,m]=bwlabel(fnegativa,4);
>> Lneg

Lneg =

     1     1     1
     0     1     1
     2     0     0
```



Función regionprops:

>> D = regionprops(L, properties)

donde L es una matriz de etiquetas de componentes conexas y D es una estructura de array conteniendo propiedades de las distintas regiones de la imagen binaria correspondiente a L.

Ejemplo:

```
>> f=imread('figuras.png');
>> T=graythresh(f);
>> fbin=im2bw(f,T);
>> [L,n]=bwlabel(fbin,4);
>> D=regionprops(L,'all');
>> w1=[D.Area]

w1 =

  Columns 1 through 7

       3805         353         573         623         746         778         436

  Columns 8 through 9

        697         570

>> w2=[D.Centroid]

w2 =

  Columns 1 through 9

   28.1398  107.1269   66.5722   81.3683   98.8901  108.4311   68.0000  137.0385   82.4035

  Columns 10 through 18
```

Shape Measurements

'Area'	'EulerNumber'	'Orientation'
'BoundingBox'	'Extent'	'Perimeter'
'Centroid'	'Extrema'	'PixelIdxList'
'ConvexArea'	'FilledArea'	'PixelList'
'ConvexHull'	'FilledImage'	'Solidity'
'ConvexImage'	'Image'	'SubarrayIdx'
'Eccentricity'	'MajorAxisLength'	
'EquivDiameter'	'MinorAxisLength'	

Pixel Value Measurements

'MaxIntensity'	'MinIntensity'	'WeightedCentroid'
'MeanIntensity'	'PixelValues'	

8. Transformaciones geométricas:

>> g=padarray(f, [m n])

Rellena la imagen f añadiendo m filas de ceros en la parte superior e inferior de la imagen y n columnas de ceros en a derecha e izquierda de la imagen.

Ejemplo:

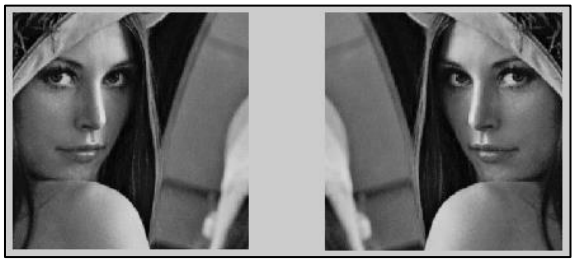
```
>> A=[1 2 3;4 5 6]

A =

     1     2     3
     4     5     6

>> B=padarray(A,[1 2])

B =

     0     0     0     0     0     0     0
     0     0     1     2     3     0     0
     0     0     4     5     6     0     0
     0     0     0     0     0     0     0

>> f=imread('Lenna.tif');
g=padarray(f,[100 100]);
>> whos f
  Name         Size              Bytes  Class

  f          300x300             90000  uint8

>> whos g
  Name         Size               Bytes  Class

  g          500x500             250000  uint8
```

Imagen reflejada:

>> g=fliplr(f)

Ejemplo:

```
>> h=fliplr(f);
```



Rotación de imágenes:

>> g=imrotate(f, angle, 'crop')

El tamaño de la imagen es aumentado automáticamente rellenando la imagen hasta adaptarse a la rotación. Si incluimos el argumento 'crop', observar qué ocurre con el tamaño de la imagen en el siguiente ejemplo:

Ejemplo:

```
>> g1=imrotate(f,45);
>> g2=imrotate(f,45,'crop');
>> whos g1
  Name         Size                Bytes  Class

  g1         425x425             180625  uint8

>> whos g2
  Name         Size                Bytes  Class

  g2         300x300             90000  uint8
```

Cambio de tamaño:

>> g=imresize(f,escala)

Ejemplo:

```
>> g1=imresize(f,0.2);
>> g2=imresize(f,0.5);
>> g3=imresize(f,0.8);
```