Tarea 1
Transformaciones de intensidad y filtrado espacial
INFORME

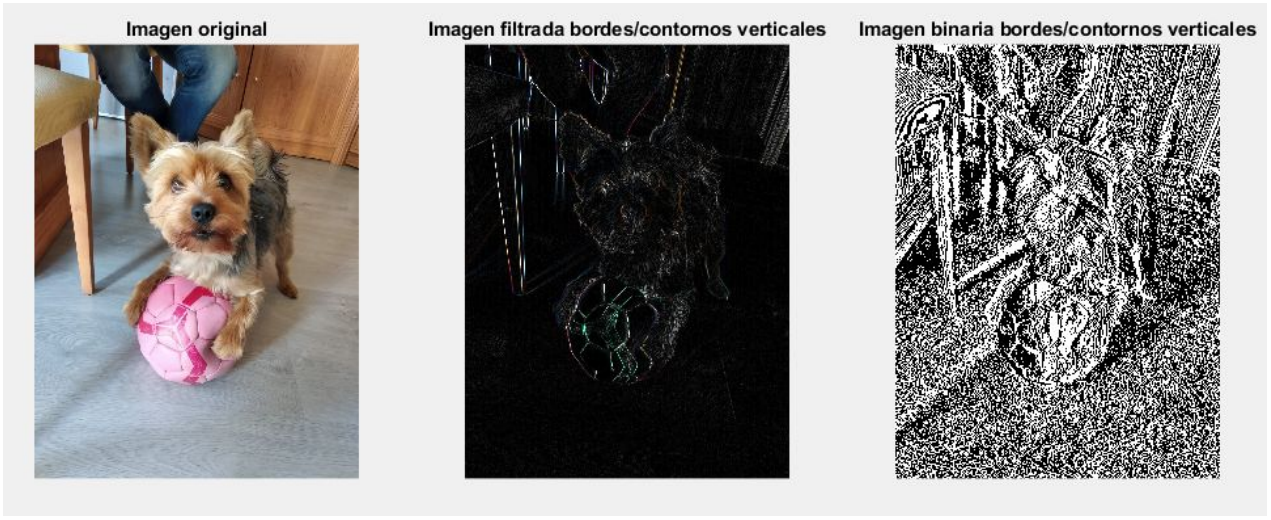
Integrante 1: Ana Poveda García Herrero.
Integrante 2: Cristina Taboada Mayo.

Objetivo de la parte obligatoria: Obtener una imagen binaria, del mismo tamaño (filas*columnas) que la imagen original, en la que sólo aparezcan como primer plano los bordes/contornos verticales de mayor "intensidad" de la imagen seleccionada. Es decir, sólo son de interés aquellos contornos verticales asociados a un mayor valor absoluto del gradiente. En caso de que la imagen seleccionada apenas presente "bordes verticales", puede rotar la imagen 90 grados para conseguirlos. La instrucción de Matlab que permite rotar una imagen es *imrotate*.

Los objetivos que se perseguían para esta parte, era conseguir una imagen, con las mismas dimensiones que la original pero que apareciese como primer plano los bordes/contornos verticales con mayor intensidad.
Para conseguir este objetivo, se han utilizado técnicas vistas en las práctica 1 y 2.
Primero de todo, se carga la imagen seleccionada previamente utilizando el comando *'imread()'.*
Tras tener la imagen, se debía de filtrar para obtener los bordes/contornos verticales con mayor intensidad. En este caso, se hace con un filtro no lineal, concretamente el *'Prewitt'*.
Para ello se crea el filtro a utilizar, que ha sido posible con el comando *'fspecial()'*, ya que es el que se usa para filtros no lineales, pasando como argumento el tipo de máscara, en este caso el del filtro 'prewitt'.
Con ese filtro se obtiene la imagen filtrada en los que se destacan los contornos/bordes horizontales, y como se requiere que sean los verticales,lo que se hace es la transpuesta del filtro, utilizando el comando Hprew2 = Hprew';
Tras esto, se filtra la imagen con el comando '*imfilter()'*, pasando como parámetros:

- La imagen original de tipo double.
- El filtro prewitt que obtendrá la imagen nueva
- El tipo de padding utilizado por la máscara, en este caso *'Symmetric' (Mirror padding)*

Se obtiene la imagen filtrada, y a continuación se realiza la conversión de la imagen RGB a Binario, mediante el comando *'Im2bw()'* pasando como argumento la imagen filtrada, ya que el umbral por defecto es 0.5, y no se ha modificado.(ver anexo 1.1)
Finalmente, visualizamos los resultados con el comando *'imshow()'.*



Con este proceso nos hemos dado cuenta de que la imagen filtrada con los bordes/contornos en verticales se apreciaba bien el contorno de los objetos, en este caso del perro, en cambio, al transformar la imagen filtrada a blanco y negro, obteniendo así una imagen en la que es más difícil de diferenciar los diferentes contornos de los objetos, ya que los niveles de intensidad de los contornos verticales pasan a primer plano (negro) y el resto al fondo (blanco).

Parte creativa: Obtener la imagen RGB en negativo, del mismo tamaño (filas*columnas) que la imagen original.

Los objetivos que se perseguían para esta parte, era conseguir una imagen, con las mismas dimensiones que la original pero que apareciese con sus colores invertidos, es decir, la imagen en negativo.

Para conseguir este objetivo, se han utilizado técnicas vistas en las práctica 1 y 2.

Primero de todo, se carga la imagen seleccionada previamente, utilizando el comando *'imread()'.*

Tras tener la imagen, se descompone en sus componentes RGB, de manera que todas las filas y columnas de cada componente se asignan a una variable para cada una, y además se ha visualizado con el comando *'imshow()'.*

Después, teniendo los componentes individuales de cada imagen, se obtiene el negativo de cada una de ellas usando la ecuación de la recta (**Y = -x + 255,** en intensidad), invirtiendo así los niveles de intensidad de cada componente, ya que no se habla de 'color' porque el color lo componen las tres, RGB juntas. Se ha visualizado también con el comando *'imshow()'* cada una.(ver Anexo 1.2)

Finalmente, teniendo las nuevas componentes, se ha vuelto a componer la imagen con las componentes negativo, obteniendo así una imagen RGB en negativo, y visualizando los resultados con *'imshow()'.*

Negativo de la componente roja

Negativo de la componente verde

Negativo de la componente azul

Imagen original

Imagen negativo

Anexo 1

- 1.1:

```
%% -------------------- PARTE OBLIGATORIA --------------------
%clear all, close all, clc
% Cargamos la imagen que vamos a utilizar.
I = imread('G15.jpeg');

% Primero, realizo un filtrado de la imagen para obtener los
% bordes/contornos verticales de mayor intensidad.

% Definimos el filtro.
% Utilizando directamente este filtro, obtendría los contornos horizontales.
Hprew = fspecial('prewitt');
% Definimos la transpuesta de filtro previamente definido para obtener en
% el filtrado los bordes/contornos verticales.
Hprew2 = Hprew';
% Realizamos el filtrado.
I_Hprew2 = imfilter(double(I),Hprew2,'symmetric');

% Convierto la imagen resultante en una imagen binaria.
BW = im2bw(I_Hprew2);

% Mostramos las imagenes resultantes.
figure,subplot(1,3,1),imshow(I),title('Imagen original')
subplot(1,3,2),imshow(uint8(abs(I_Hprew2))), title('Imagen filtrada contornos verticales')
subplot(1,3,3),imshow(BW),title('Imagen binaria bordes/contornos verticales')
```

- 1.2:

```
%% -------------------- PARTE CREATIVA --------------------

% Vamos a visualizar el negativo de I

% Cargamos la imagen que vamos a utilizar.
I = imread('G15.jpeg');

% Separamos sus componentes RGB y las visualizamos.
R = I(:,:,1);
G = I(:,:,2);
B = I(:,:,3);

figure,subplot(1,3,1),imshow(R),title('Componente roja')
subplot(1,3,2),imshow(G), title('Componente verde')
subplot(1,3,3),imshow(B),title('Componente azul')

% Hallamos el negativo de cada componente y las visualizamos.
NR_R = 255 - R;
NR_G = 255 - G;
NR_B = 255 - B;
figure,subplot(1,3,1),imshow(NR_R),title('Negativo de la componente roja')
subplot(1,3,2),imshow(NR_G), title('Negativo de la componente verde')
subplot(1,3,3),imshow(NR_B),title('Negativo de la componente azul')

% Componenmos la imagen de nuevo con los negativos, para obtener la imagen
% RGB en negativo.
I_Negativo(:,:,1) = NR_R;
I_Negativo(:,:,2) = NR_G;
I_Negativo(:,:,3) = NR_B;


imwrite(I_Negativo, 'I_Negativo.tif');
figure, subplot(1,2,1), imshow(I), title('Imagen original')
subplot(1,2,2), imshow(I_Negativo), title('Imagen negativo')
```

- **1.3: Observaciones en la parte obligatoria:**

A la hora de obtener la imagen en blanco y negro de los bordes/contornos verticales nos hemos dado cuenta de que si cambiamos los pasos a seguir en el procedimiento no obtenemos los resultados deseados.

La opción que nos habíamos planteado era:

- Obtener la imagen original.
- Convertir la imagen RGB a una imagen BW.
- Filtrar la imagen y obtener las componentes verticales.

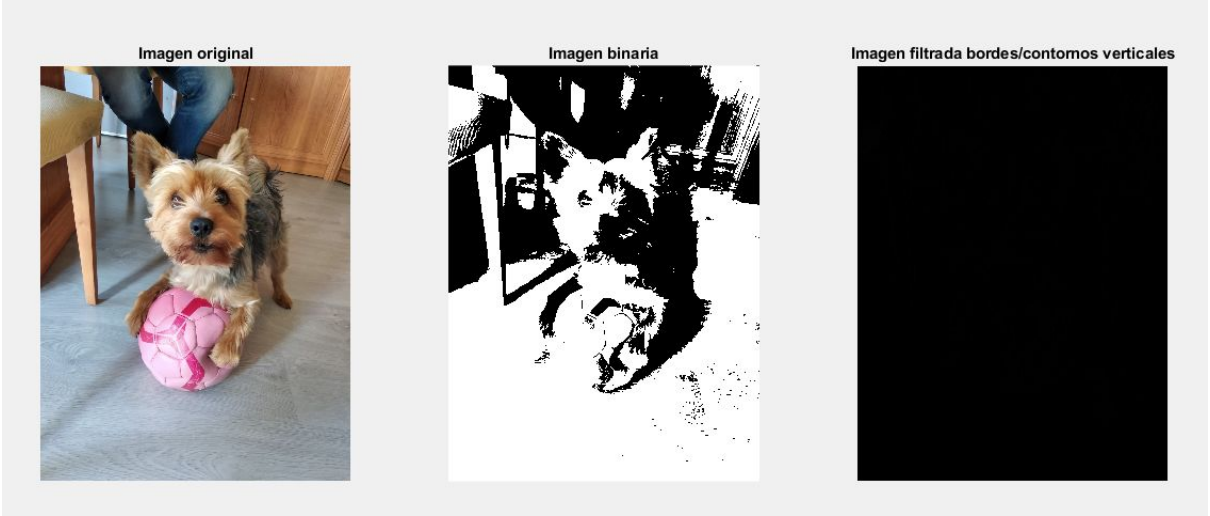
```
%% Otra forma de resolverlo. (El resultado de esta transformacion no es bueno.
% Cargamos la imagen que vamos a utilizar.
I = imread('G15.jpeg');

% Convierto la imagen en una imagen binaria.
BW = im2bw(I);

% Definimos el filtro.
Hprew = fspecial('prewitt'); % Utilizando directamente este filtro, obtendría los bordes/contornos horizontales.
% Definimos la transpuesta de filtro previamente definido para obtener en
% el filtrado los bordes/contornos verticales.
Hprew2 = Hprew';
% Realizamos el filtrado.
I_Hprew2 = imfilter(double(BW),Hprew2,'symmetric');

% Mostramos las imagenes resultantes.
figure,subplot(1,3,1),imshow(I),title('Imagen original')
subplot(1,3,2),imshow(BW),title('Imagen binaria')
subplot(1,3,3),imshow(uint8(abs(I_Hprew2))), title('Imagen filtrada bordes/contornos verticales')
```

Este es el resultado visual que hemos obtenido:



Como se puede apreciar, al pasar la imagen a blanco y negro perdemos tanto las componentes verticales como la horizontales, y cuando hacemos el filtrado no obtenemos nada.