Bloque 1: Operadores puntuales

MATERIAL EXPERTOS	Experto 1: Representación de imágenes y cálculo de energías

El objetivo de esta práctica es presentar al alumno las técnicas para realizar operadores puntuales sobre imágenes en escala de grises y en color. Adicionalmente, se presentará el histograma de una imagen como elemento clave para la visualización del efecto de los operadores puntuales.

Representación de imágenes

Esta sección es una guía para la representación de imágenes en Matlab que deberá recordarse a lo largo de la asignatura. Busca proporcionar una serie de reglas que ayuden al estudiante a decidir el tipo de representación a utilizar en función de la naturaleza de la imagen.

Distinguiremos cuatro casos principales:

i. La imagen tiene un mapa asociado (VLT): representaremos la imagen **`ima`** mediante **`subimage(ima,map)`** donde **`map`** es la VLT . Esta opción puede utilizarse independientemente del tipo de la imagen (**`uint8`** ó **`double`**).

ii. La imagen es de tipo **`uint8`** con valores en el rango $x \in [0,255], x \in \mathbb{N}^*$: podemos representar la imagen **`ima`** directamente mediante **`imshow(ima)`**

iii. La imagen es de tipo **`double`** y tiene valores comprendidos en el rango $x \in [0,255], x \in \mathbb{R}$: Podemos realizar un casting a **`uint8 ima_u8 = uint8(ima)`** asumiendo que perderemos todo el contenido decimal en **`ima`** por el redondeo. A continuación podremos representar la señal con **`imshow(ima_u8)`** como en el caso ii). Alternativamente, podemos utilizar **`imagesc(ima)`** y aplicar un mapa de color en escala de grises a continuación: **`colormap(gray)`**. **`imagesc(ima);colormap(gray);`**

iv. La imagen es de tipo **`double`** y tiene valores comprendidos en el rango $x \in [0,1], x \in \mathbb{R}$: Podemos representar esta imagen **`ima`** directamente con **`imshow(ima)`** o con **`imagesc(ima)`**. En este último caso, se recomienda aplicar un mapa de color en escala de grises a continuación: **`colormap(gray)`**.

En el caso de que se deseen representar varias imágenes, histogramas o gráficas en una misma figura (**`figure`**), puede utilizase **`subplot`**.

El uso de la función **`subplot`** es sencillo. La invocación a **`subplot`** se realiza generalmente mediante tres argumentos:

```
subplot(nf,nc,j)
```

El primer y el segundo argumento **nf y nc** definen la partición del **figure** que se realiza. Es decir, divide el **figure** en una cuadrícula, matriz o rejilla de **(nf*nc)** celdas donde **nf** es el número de filas de celdas y **nc** el número de columnas de celdas. El tercer argumento **j** indica la posición dentro de la partición donde se desea que aplique la representación. El argumento **j** indexa la cuadrícula considerando primero avances en columna.

Es decir, si dividimos el **figure** en 2x2 celdas y queremos representar una imagen **ima** de tipo **double** con valores comprendidos en el rango $x \in [0,1], x \in \mathbb{R}$ en la primera fila, segunda columna de la partición invocaremos:

```
subplot(2,2,2), imagesc(ima);colormap(gray);
```

Si queremos representar otra imagen imagen **ima** de tipo **double** con valores comprendidos en el rango $x \in [0,1], x \in \mathbb{R}$ en la segunda fila, primera columna, invocaremos:

```
subplot(2,2,3), imagesc(ima);colormap(gray);
```

Cuando utilice **subplot** ha de tener en cuenta dos factores:

a) No represente imágenes de distinto tipo en un mismo **figure**.
b) Intente no utilizar **subplot** para pintar imágenes mediante **subimage**.

Recuerde incluir etiquetas en los ejes de las figuras mediante los comandos **xlabel** e **ylabel** así como título en cada figura mediante el comando **title**. Para incluir un valor de manera dinámica en alguno de estos comandos puede utilizar la instrucción **sprintf**, por ejemplo, para incluir en el título de la figura un parámetro de configuración **N** entero puede hacer:

```
title(sprintf('Imagen resultado obtenida para N = %d',N));
```

Cálculo de energías

Durante estas prácticas utilizaremos la energía de la imagen como guía para saber si el resultado de cada ejercicio es correcto o no (ver la carpeta soluciones). Para calcular la energía **E** de una imagen **ima1**, independientemente del número de canales que la compongan puede hacer:

```
E = sum(double(ima1(:)).^2);
```

El valor de la energía suele incluirse en título de la figura bien en coma flotante con un número determinado de decimales (en el ejemplo 2):

```
title(sprintf('Imagen resultado con energía = %.2f', E));
```

También puede incluirse utilizando notación científica:

```
title(sprintf('Imagen resultado con energía = %g', E));
```

Experimente con las diferentes técnicas para la representación de imágenes y cálculo de energía, para ello puede hacer uso de las imágenes demo de Matlab:

```
ima1  = imread('pears.png');
ima2 = imread('rice.png');
```

Puede pasarlas a tipo double invocando al comando:

```
ima1d = im2double(ima1);
```

, y obtendrá el resultado escalado entre **0** y **1** ó:

```
ima1d = double(ima1);
```

, para mantenerse en el rango original de valores.

Este último es, en general, el caso seguido para el cálculo de energías durante estas prácticas.