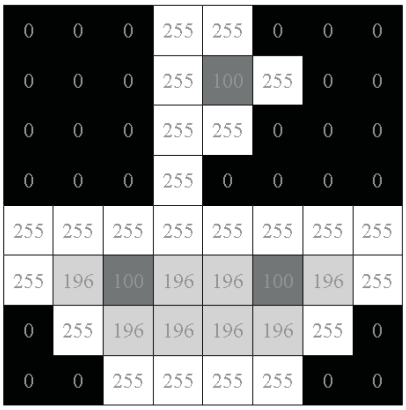
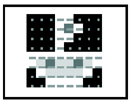
Señales y Sistemas Discretos

Tarea 4 - Entregar 28 de Octubre

Imágenes digitales

Una imágen digital es una señal discreta $f[x, y]$ en dos dimensiones, cuyos valores representan la intensidad observada en el pixel (x, y). La siguiente figura muestra un ejemplo de una imagen digital y su respectiva matriz de intensidades.





En Octave, las imágenes se representan mediante matrices. Existen varias funciones para cargar y guardar imágenes en distintos formatos de archivo. Entre estas se incluyen `jpgread` y `pngread`, las cuales leen imágenes de archivos JPG y PNG, respectivamente; así mismo, se tienen las funciones `jpgwrite` y `pngwrite` para guardar imágenes en estos tipos de archivo. Para visualizar una imagen en Octave, se puede usar la función `imshow()`, como se muestra en el siguiente ejemplo:

```
m = pngread("lena.png");
imshow(m);
```

Hay que recordar que para indexar los elementos de una matriz, se especifica primero el número de renglón, y luego la columna, por lo que si la matriz `m` representa una imágen, entonces el pixel (x, y) de la imágen corresponde al elemento `m[y,x]` de la matriz.

Podemos implementar un sistema lineal e invariante al desplazamiento mediante la convolución en dos dimensiones:

$$g[x, y] = f[x, y] * h[x, y] = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} f[k, l]\ h[x - k, y - l],$$

donde $M \times N$ es el tamaño de las imágenes y $h[x, y]$ es la respuesta al impulso del sistema. En Octave se puede utilizar la función `conv2` para convolucionar dos imágenes (revise la ayuda para esta función - el parámetro `shape` es particularmente útil).

Ejercicio 1.- Un filtro Gaussiano en dos dimensiones tiene una respuesta al impulso dada por

$$g_\sigma(x, y) = \frac{1}{\sqrt{2\pi}\ \sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}.$$

Este kernel se utiliza principalmente para el suavizamiento de imágenes y, en algunos casos, para la reducción de ruido.

1. Cargue y visualice los archivos `lena.png` y `lena_ruido.png`.

2. Escriba una función que devuelva la matriz correspondiente a un kernel Gaussiano con parámetro σ, truncando en $|x|, |y| \approx 4\sigma$. Tenga en cuenta que los subíndices de las matrices en Octave comienzan a partir de 1, por lo que será necesario desplazar al centro (c_x, c_y) de la imagen como sigue:

   $$g_\sigma[x + c_x, y + c_y] = \frac{1}{\sqrt{2\pi}\ \sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}.$$

   Utilice esta función para calcular y visualizar un kernel Gaussiano con $\sigma = 10$ (utilice el parámetro `limits` de la función `imshow`).

3. La imagen `lena_ruido.png` es una copia de `lena.png` a la cual se le ha agregado ruido *sal y pimienta*. Este modelo de ruido consiste en seleccionar pixeles de manera aleatoria y cambiar su valor a cero o el valor máximo de intensidad. Utilice un filtro Gaussiano para eliminar el ruido de `lena_ruido.png`. Encuentre de manera empírica un valor de σ que produzca buenos resultados, comparando con la imágen original.

4. Realice el mismo ejercicio que en el inciso anterior, pero ahora utilizando un filtro de caja (promedio móvil en 2D) de tamaño $M \times M$. Determine el tamaño del filtro que produce los mejores resultados.

5. Un kernel $h[x, y]$ es *separable* cuando puede expresarse como el producto de dos kernels unidimensionales h_1 y h_2:

   $$h[x, y] = h_1[x]h_2[y].$$

   Demuestre que el filtro Gaussiano es separable.

6. Demuestre que si $h[x, y]$ es separable, de manera que $h[x, y] = h_1[x]h_2[y]$, entonces la convolución en 2D $f * h$ puede realizarse convolucionando (en 1D) primero todos los renglones de f con h_1, y posteriormente convolucionando las columnas con h_2.

7. Si f y g son imágenes de $N \times N$ pixeles, cuál es el orden de complejidad de la convolución $f * g$ en términos de N. Cuál sería el orden de complejidad si g es separable y se utiliza el algoritmo del inciso anterior?

Ejercicio 2.- El kernel $d = [1, -1]$ funciona como un operador de diferenciación (derivación); sin embargo, es muy sensible al ruido, y lo amplifica. Esto puede verificarse fácilmente observando la derivada de una cuadrática a la cual se le ha agregado ruido Gaussiano:

```
d = [1, -1];
x1 = (1:100).^2;
x2 = x1 + 100 * randn(1,100);
y1 = conv(x1, d)(1:100);
y2 = conv(x2, d)(1:100);
plot(y1, "r", y2, "b");
```

Una manera de hacer este operador de diferenciación más robusto (e.g., menos sensible) al ruido, consiste en suavizar primero la señal de entrada, utilizando, por ejemplo, un kernel Gaussiano g_σ:

$$y = (x * g_\sigma) * d.$$

Debido a que la convolución es asociativa, entonces la ecuación anterior equivale a:

$$y = x * d_\sigma,\ d_\sigma = g_\sigma * d,$$

donde d_σ es el operador robusto de diferenciación, el cual puede obtenerse derivando el kernel Gaussiano.

1. Calcule la derivada analítica del kernel Gaussiano en una dimensión.
2. Utilice el resultado del inciso anterior para implementar una función que devuelva el kernel de un diferenciador robusto d_σ.
3. Tome la señal `x2` del ejemplo anterior (la señal cuadrática con ruido Gaussiano) y obtenga su derivada utilizando el operador robusto de diferenciación d_σ con distintos valores para σ. Grafique los resultados y discuta.
4. Obtenga (en 2D) kernels robustos de diferenciación con respecto a x y con respecto a y (utilice la propiedad de separabilidad de la Gaussiana en 2D). Obtenga las derivadas de la imagen `barbara.png` con respecto a las dimensiones x y y, y grafíquelas.

Ejercicio 3.- En una imagen, los bordes de los objetos dividen regiones con valores de intensidad relativamente distintos; por lo tanto, una manera de detectar bordes en una imagen es encontrando aquellos puntos donde se observe un cambio significativo de intensidad; es decir, aquellos puntos donde la derivada de la señal sea máxima. Considere, por ejemplo, el siguiente ejemplo en una dimensión:

```
x = [zeros(1,50), 10 * ones(1,50)] + randn(1,100);
y = conv(x, [1, -1])(1:100);
plot(y);
```

Supongamos que $g(x)$ es la derivada de $f(x)$. Una forma de encontrar los máximos de $g(x)$ (y los bordes de f) es resolviendo la ecuación $g'(x) = 0$; es decir

$$g'(x) = f''(x) = \frac{d^2 f}{dx^2} = 0.$$

En dos dimensiones, el equivalente a la segunda derivada es el operador Laplaciano ∇^2, el cual se define como:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}.$$

Por lo tanto, una forma de encontrar los bordes de una imagen f es encontrando los cruces por cero de $\nabla^2 f$. Los cruces por cero se dan en aquellos puntos donde el valor de pixeles adyacentes cambia de signo.

1. El operador Laplaciano también es sensible al ruido. Al igual que con la derivada, podemos obtener un operador Laplaciano más robusto tomando el Laplaciano de una Gaussiana $\nabla^2 g_\sigma$. Demuestre que si $L_\sigma = \nabla^2 g_\sigma$, entonces
$$L_\sigma(x, y) = g_\sigma(x, y) \cdot \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}.$$

2. Escriba una función que devuelva el kernel de un operador Laplaciano robusto L_σ. Visualice el kernel correspondiente a $\sigma = 10$.

3. Una manera fácil de detectar los cruces por cero de una imagen $g(x, y)$, consiste en multiplicar g por una versión de g desplazada un pixel en el eje X. Aquellos pixeles donde el producto sea negativo, corresponden a un cruce por cero. De manera similar, se pueden detectar cruces por cero sobre el eje Y. En Octave, puede utilizarse el siguiente código:

```
B = ((g .* shift(g, 1, 1)) < 0) | ((g .* shift(g, 1, 2)) < 0);
```

   Utilice lo anterior para implementar una función que obtenga la imagen de bordes correspondiente a una imagen de entrada `f` con un parámetro de escala σ. El procedimiento es el siguiente:

   (a) Calcular un kernel Laplaciano robusto $L_\sigma = \nabla^2 g_\sigma$, donde g_σ es una Gaussiana en 2D.

   (b) Calcular $g = f * L_\sigma$.

   (c) Calcular la imagen de bordes B como se sugiere en el párrafo anterior.

4. Utilice la función del inciso anterior para encontrar los bordes de la imagen `mri_t2.png`, la cual corresponde a una resonancia magnética axial. Pruebe con distintos valores para σ y discuta el papel que tiene este parámetro en el resultado.