Máster Oficial en Visión Artificial
Tratamiento Digital de Imágenes
Aplicación de la transformada de Fourier 2D al tratamiento de imágenes

En esta práctica utilizaremos la transformada de Fourier 2D para el tratamiento digital de imágenes. Cada tarea tiene asociada una guía con algunas de las funciones necesarias para realizarla, así como indicaciones en caso necesario.

1.- Genera imágenes cuyos niveles de intensidad varíen como funciones sinusoidales. Para ello, puedes programar tu propio código o utilizar la función `imGenerator` que se proporciona con esta práctica.
Indicaciones: La función `imGenerator` tiene la siguiente interfaz
`[imOut] = imGenerator (imSize, functType, frec, orientation)`
donde:

- `imOut` es la imagen que devuelve la función
- `imSize` es un valor entero que indica el tamaño de la imagen (las imágenes generadas son cuadradas).
- `functType` es el tipo de función y puede tomar los valores '`sin`' y '`cos`'.
- `frec` es un valor entero que indica el número de veces que oscila la función sinusoidal a lo largo de la imagen.
- `orientation` es la orientación de los patrones sinusoidales respecto a los bordes de la imagen y puede tomar los valores enteros `-45`, `0`, `45` y `90`.

2.- Construye una función en Octave/Matlab llamada `imfft` tal que, dada una imagen que se pasa como parámetro, devuelva la transformada de Fourier centrada.
Indicaciones: Son de utilidad las siguientes funciones: `fft2`, `fftshift`. Consultar en la ayuda de Octave/Matlab para saber qué son y cómo se usan.

3.- Construye una función en Octave/Matlab llamada `imifft` tal que, dada la transformada de Fourier centrada de una imagen que se pasa como parámetro devuelva la transformada inversa de Fourier.
Indicaciones: Son de utilidad las siguientes funciones: `ifft2`, `ifftshift`. Consultar en la ayuda de Octave/Matlab para saber qué son y cómo se usan.

4.- Ya puedes probar a hacer la transformada de Fourier de una imagen, luego la transformada inversa y mostrarla.
Indicaciones: Puede ser de utilidad la función `mat2gray`, para convertir el resultado de la llamada a `imifft` a una imagen en escala de grises, que pueda ser mostrada con `imshow`.

5.- Construye una función en Octave/Matlab llamada `immagphase` tal que, dada la transformada de Fourier centrada de una imagen que se pasa como parámetro, represente la información de la magnitud y la fase de dicha transformada en forma de imagen. Prueba con las imágenes mostradas en el ejercicio 1.
Indicaciones:

- Para el cálculo del módulo y la fase, son de utilidad las siguientes funciones: `abs`, `angle`.
- Normalmente, en lugar de la magnitud se representa el logaritmo de la misma. Como el logaritmo no está definido para el valor 0, suele añadirse a la magnitud un offset (p. ej., 1e-7) antes de calcular el logaritmo.
- Recuerda que puede ser útil el uso de `mat2gray` previo a representar el resultado de una operación.

6.- Construye una función en Octave/Matlab llamada `lowpassfilter` que construya un filtro paso bajo de Butterworth de orden *n* en 2D.

Indicaciones:

- La cabecera de esta función será: `filtro = lowpassfilter(tamaño, frecCorte, n)`, donde `tamaño` hace referencia al tamaño del filtro en número de filas y número de columnas (usualmente coincide con el tamaño de la transformada de la imagen), `frecCorte` es la frecuencia de corte, que puede tomar valores entre 0 y 0.5, y `n` es el orden del filtro.
- Recordar la expresión del filtro pasobajo de Butterworth visto en clase:

$$H(u,v)=\frac{1}{1+\left(\frac{D(u,v)}{D_0}\right)^{2n}}$$

  donde D_0 es la frecuencia de corte y $D(u,v)=\sqrt{(u-N/2)^2+(v-M/2)^2}$, siendo N y M el tamaño de los datos (es decir, D(u,v) es la distancia al centro de la matriz de frecuencias).

7.- Ahora puedes calcular de forma sencilla el resultado de filtrar una imagen en el dominio de la frecuencia utilizando este filtro. Si calculas la transformada inversa de la imagen después del filtrado, observarás que el resultado es otra imagen con los bordes menos definidos.

8.- Construye una función en Octave/Matlab llamada `highpassfilter` que construya un filtro paso alto de Butterworth de orden *n* en 2D.

Indicaciones:

- La cabecera de esta función será: `filtro = highpassfilter(tamaño, frecCorte, n)`, donde `tamaño` hace referencia al tamaño del filtro en número de filas y número de columnas (usualmente coincide con el tamaño de la transformada de la imagen), `frecCorte` es la frecuencia de corte, que puede tomar valores entre 0 y 0.5, y `n` es el orden del filtro.
- Recordar la expresión del filtro pasoalto en función del pasobajo:

$$H_{PA}(u,v)=1-H_{PB}(u,v)$$

  donde H_{PA} es el filtro paso alto y H_{PB} es el filtro paso bajo correspondiente.

9.- Ahora puedes calcular de forma sencilla el resultado de filtrar una imagen en el dominio de la frecuencia utilizando este filtro. Si calculas la transformada inversa de la imagen después del filtrado, observarás que el resultado es una imagen en la que sobreviven los bordes.

10.- Hagamos una comprobación cualitativa del funcionamiento del teorema de convolución. Tomando las máscaras de detección de bordes horizontales y verticales de *Sobel*, aplicarlo a una imagen en el dominio de la imagen y en el dominio de frecuencias. Comparar los resultados obtenidos.

Indicaciones: antes de aplicar la transformada a la máscara, es necesario hacer un rellenado con ceros hasta el tamaño de la imagen a la que se desea aplicar el detector de bordes.