Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas
Imágenes Médicas
Ejercicios: Conceptos básicos, 2023-30

Objetivos

1. Utilizar MATLAB para interactuar de forma práctica con conceptos básicos como convolución, relación señal a ruido y transformada de Fourier.
2. Explorar el uso de una librería de manipulación de imágenes (médicas) desde Python.

Desarrollo del ejercicio

El desarrollo del ejercicio práctico consistirá en generar un informe escrito (en formato PDF), el cual debe enviarse a través de la respectiva asignación de BrightSpace. En el informe debe incluirse, como respuesta a cada ejercicio, el código fuente solicitado e impresiones de pantalla del resultado obtenido con ese código.

Parte 1: Conceptos básicos de imágenes ilustrados en MATLAB

Convolución de señales

El siguiente código (adaptado de https://www.youtube.com/watch?v=hcyy144Gu60) presenta un ejemplo simple de convolución de dos señales unidimensionales en MATLAB:

```
% definir el tamano del paso
dt = 0.1;
% definir la primera senal f(t), respuesta al impulso
t1 = 0:dt:2;
ft = 2-t1;
% definir la segunda senal g(t), entrada del sistema
t2 = -2:dt:2;
gt = 3*ones(1,length(t2));
% generar la respuesta del sistema a la entrada con la convolucion
yt = dt*conv(ft,gt);
yt_x = (1:length(yt))*dt + t2(1) + t1(1);

% graficar todas las senales
figure(1)
subplot(2,2,1), plot(t1,ft,'r'), title('f(t) respuesta al impulso'),
grid on, xlim([-2 2]), xlabel('t'), ylabel('amplitud')
subplot(2,2,3), plot(t2,gt,'b'), title('g(t) entrada del sistema'),
grid on, xlabel('t'), ylabel('amplitud'), ylim([0 3.5])
subplot(2,2,[2;4]), plot(yt_x,yt,'k'),
title('f(t)*g(t) respuesta del sistema'),
grid on, xlabel('t'), ylabel('amplitud')
```

- **Ejercicio 1**
  Modifique el código anterior para que la señal de entrada ya no sea una función tipo rampa, sino

más bien una función tipo triángulo. Genere la gráfica de las señales en donde se muestre el cambio de la función de entrada y de la respuesta al sistema usando esa nueva señal.

Siguiendo con los ejemplos de la convolución, el siguiente código en MATLAB (adaptado también de https://www.youtube.com/watch?v=hcyy144Gu60) muestra cómo una señal aleatoria (ruidosa) puede ser simplificada a partir de su convolución con otras señales más simples y suaves:

```
% definicion de la senal aleatoria
data = rand(1,1000) – 0.5;
% definicion de las senales suaves
curv_rect = 1/61*ones(1,61);
t = 0:60;
curv_ham = -1/(61*2)*cos(1/61*2*pi*t) + 1/(61*2);
% aplicar la convolucion sobre la senal aleatoria
rec_smooth = conv(data,curv_rect,'same');
ham_smooth = conv(data,curv_ham,'same');

% graficar todas las senales
figure(1)
subplot(2,2,1)
plot(data,'k'), grid on, title('Senal aleatoria')
xlabel('x'), ylabel('y')
subplot(2,2,2)
plot(curv_rect,'b','LineWidth',1), title('Senales suaves'), grid on
hold on
plot(curv_ham,'r','LineWidth',2)
legend('Rectangular','Campana'), xlabel('x'), ylabel('y')
subplot(2,2,[3,4])
plot(rec_smooth,'b','LineWidth',1)
hold on
plot(ham_smooth,'r','LineWidth',2)
title('Senal simplificada por convolucion'), grid on
```

- **Ejercicio 2**
  Modifique el código anterior para agregar una nueva señal suave que genere un efecto intermedio entre la función rectangular y la función campana. Genere la gráfica de las señales en donde se muestre la nueva señal agregada y la respuesta del sistema (simplificación) usando esa nueva señal. Es importante no eliminar las funciones rectangular y campana y sus resultados, para verificar el efecto de la nueva señal en comparación con las que ya se tenían.

Relación señal a ruido

El siguiente código en MATLAB (adaptado de https://www.youtube.com/watch?v=QJOnkSU20pY) presenta un ejemplo simple de una señal y una estimación de ruido, que permite calcular el valor de la relación señal a ruido en decibeles:

```
% definicion de la longitud de las senales
length=linspace(-2*pi,2*pi);
% definicion de la senal
signal1=sin(length);
% definicion del ruido
noise1=0.5*rand(size(signal1));
% calculo de la relacion senal a ruido
snr1=snr(signal1,noise1)
```

```
% graficar la senal y el ruido estimado
figure(1)
subplot(2,1,1)
plot(signal1), title('Senal')
subplot(2,1,2)
plot(noise1), title('Ruido')
```

- **Ejercicio 3**
  ¿Cuál será el efecto de variar la amplitud del ruido estimado para la señal? Genere otras dos estimaciones de ruido con amplitudes menores y mayores a la presentada en el ejemplo, y obtenga el valor de la relación señal a ruido para cada una de ellas. Analice qué pasa con los valores, y grafique también los diferentes ruidos estimados.

Transformada de Fourier

El siguiente código en MATLAB (adaptado de https://www.youtube.com/watch?v=nWGirrTlpnk) presenta un ejemplo simple de aplicación de la transformada de Fourier a una imagen en escala de grises, permitiendo visualizar el espectro de frecuencias y orientaciones, así como la reconstrucción de la imagen usando la transformada inversa de Fourier (en su versión discreta y rápida):

```
% cargar una imagen de ejemplo de MATLAB
imdata = imread('street2.jpg');
% obtener la version en escala de grises de la imagen
g_imdata = rgb2gray(imdata);
% obtener la transformada de Fourier de la imagen
F = fft2(g_imdata);
% obtener el espectro centrado de frecuencias y orientaciones
Fsh = fftshift(F);
% aplicar una transformacion logaritmica para facilitar la visualizacion
S2 = log(1+abs(Fsh));
% finalmente, reconstruir la imagen original
F2 = ifftshift(Fsh);
f = ifft2(F2);

% visualizar las imagenes y el espectro de la transformada
figure(1)
subplot(2,2,1)
imshow(g_imdata), title('Imagen original (en escala de grises)')
subplot(2,2,3)
imshow(S2,[]), title('Espectro de la transformada de Fourier')
subplot(2,2,4)
imshow(f,[]), title('Imagen reconstruida')
```

- **Ejercicio 4**
  ¿Cuál será el efecto en el espectro de variar el contenido frecuencial de la imagen utilizada? Utilice el código ejemplo para visualizar el espectro de dos imágenes adicionales: una con poca información de bordes o detalles (una figura geométrica plana, por ejemplo) y otra con mucha información de altas frecuencias (una imagen con pixeles aleatorios, por ejemplo). Analice qué pasa con los espectros de las transformadas de Fourier de esas imágenes, incorporando los resultados obtenidos.

Parte 2: Instalación y uso básico de la librería ITK en Python

ITK (Insight Toolkit, www.itk.org) es una librería de código abierto y multiplataforma que provee una gran cantidad de herramientas para el procesamiento y análisis de imágenes. Desarrollada originalmente en C++, y orientada principalmente en sus inicios al manejo de imágenes médicas, provee interfaces para trabajar con imágenes de diferentes formatos y con otros lenguajes de programación, como Python. Es el resultado de un trabajo colaborativo de diferentes investigadores y programadores en todo el mundo, que contribuyen continuamente con el mejoramiento y ajuste de los componentes de la librería.

Es importante aclarar que esta librería no provee capacidades de visualización de las imágenes, tan solo incluye las herramientas de procesamiento y análisis que se pueden aplicar sobre las imágenes. Para visualizar los resultados, es necesario utilizar una aplicación o software de visualización de imágenes, en nuestro caso, específicamente de imágenes médicas.

Para utilizar la librería, es necesario tener instalado Python (y opcionalmente Anaconda, o alguna otra plataforma para Python). Primero, debe instalarse el módulo de ITK dentro de Python:

```
python –m pip install --upgrade pip
python –m pip install itk
```

Y una vez hecho esto, ya se puede escribir un programa que pueda utilizar la librería, teniendo en cuenta siempre importar el módulo respectivo al iniciar el programa:

```
import itk
```

El siguiente código en Python presenta un programa simple utilizando la librería ITK, en el cual se carga una imagen médica en formato DICOM (.dcm, se asume que ya existe almacenada en disco) y se reescribe en el disco utilizando el formato Analyze (.hdr + .img):

```
# definir la imagen con tipo de dato y dos dimensiones
ImageType = itk.Image[itk.UC, 2]
# definir un lector para la imagen e inicializarlo
reader = itk.ImageFileReader[ImageType].New()
# definir un escritor para la imagen e inicializarlo
writer = itk.ImageFileWriter[ImageType].New()

# asignar al lector el nombre del archivo de entrada
reader.SetFileName(“inputImage.dcm”)
# asignar al escritor el nombre del archivo de salida
writer.SetFileName(“outputImage.hdr”)

# conectar el lector y el escritor
writer.SetInput( reader.GetOutput() )
# ejecutar el proceso completo
writer.Update()
```

Como segundo ejemplo, el siguiente código en Python muestra el código necesario para cargar una imagen (no necesariamente médica) en color (se asume que ya existe almacenada en disco) y luego generar su versión en escala de grises:

```
# definir la imagen de entrada con 3 canales (color) y dos dimensiones
InputImageType = itk.Image[itk.RGBPixel[itk.UC], 2]
# definir la imagen de salida con 1 canal (gris) y dos dimensiones
OutputImageType = itk.Image[itk.UC, 2]
```

```
# definir un lector para la imagen de entrada e inicializarlo
reader = itk.ImageFileReader[InputImageType].New()
# definir un proceso para cambiar los canales de la imagen (color a gris)
rgbFilter = itk.RGBToLuminanceImageFilter.New()
# definir un escritor para la imagen de salida e inicializarlo
writer = itk.ImageFileWriter[OutputImageType].New()
# asignar al lector el nombre del archivo de entrada
reader.SetFileName("street2.jpg")
# asignar al escritor el nombre del archivo de salida
writer.SetFileName("g_street2.jpg")
# conectar el proceso al lector para tomar la imagen de entrada
rgbFilter.SetInput(reader.GetOutput())
# conectar el escritor al proceso para tomar la imagen de salida
writer.SetInput(rgbFilter.GetOutput())
# ejecutar la linea de procesamiento completa
writer.Update()
```

- **Ejercicio 5**
  Utilice el código del segundo ejemplo para generar versiones en escala de grises de 3 imágenes, con diferentes contenidos de color. Incluya en el informe el código utilizado para cada imagen, así como las imágenes de entrada y salida en cada caso.

Entrega del ejercicio

La entrega del ejercicio práctico consistirá únicamente en el archivo de reporte en formato PDF, nombrado con los apellidos de los integrantes del grupo. Este archivo deberá enviarse a través de la correspondiente asignación en BrightSpace antes de la medianoche del lunes 7 de agosto de 2023. El envío del archivo comprimido en otro formato diferente a los especificados resultará en una calificación de (0.0/5.0) para el taller.

La escala de evaluación es la siguiente, para cada ejercicio individual:

- **Excelente (5.0/5.0)**: El estudiante propone un código que realiza todos los pasos sugeridos y presenta evidencias de los resultados obtenidos.
- **Bueno (3.5/5.0)**: El estudiante propone un código que realiza la mayoría de los pasos sugeridos, y presenta al menos una evidencia de los resultados obtenidos.
- **Aceptable (2.5/5.0)**: El estudiante propone un código que realiza solo algunos de los pasos sugeridos, y presenta evidencia(s) de los resultados obtenidos, no todos correctos.
- **Malo (1.0/5.0)**: El código propuesto por el estudiante no es correcto en su sintaxis (no puede interpretarse en el entorno de ejecución adecuadamente).
- **No entregó (0.0/5.0)**: El estudiante no entrega los resultados solicitados.