Pontificia Universidad Javeriana
Departamento de Ingeniería de Sistemas
Imágenes Médicas
Ejercicios: Manejo de histogramas, 2023-30

Objetivos

- Utilizar MATLAB para interactuar de forma práctica con conceptos de histogramas, umbralización y binarización.
- Explorar el uso de la librería de manipulación de imágenes (médicas) en la manipulación de histogramas 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.

1. Normalización del histograma

El siguiente código (adaptado de https://www.geeksforgeeks.org/how-to-normalize-a-histogram-in-matlab/) presenta un ejemplo simple de normalización lineal del histograma en una imagen en escala de grises en MATLAB:

```
% cargar la imagen
img=imread("butterfly.png");
% convertir a escala de grises
grayimg=im2gray(img);
% visualizar la imagen y su histograma
figure(1)
subplot(1,2,1)
imshow(grayimg);
subplot(1,2,2)
imhist(grayimg);

% definir las intensidades en los extremos del rango actual de representacion
% es decir, el rango dentro del cual se quiere normalizar
% (se especifican manualmente)
min=45;
max=100;

% trabajar sobre una copia de la imagen con tipo de dato real
% para utilizar mayor precision en los calculos
realimg=double(grayimg);

% aplicar la ecuacion de normalizacion del histograma
normimg=(realimg-min)./(max-min);
```

```
% multiplicar por el valor maximo del rango de representacion
normimg=normimg.*255;

% convertir de nuevo a tipo de dato entero para facilitar la visualizacion
entimg=uint8(normimg);

% visualizar la imagen normalizada y su histograma
figure(2)
subplot(1,2,1)
imshow(entimg);
subplot(1,2,2)
imhist(entimg);

% guardar la imagen normalizada en disco
imwrite(entimg,"butterfly_norm.png");
```

- **Ejercicio 1**
  Ejecute el código para obtener la normalización del histograma de la imagen `butterfly.png` (adjunta a este enunciado), e incorpore el resultado obtenido a su reporte.

2. Ecualización del histograma (manual)

El siguiente código (adaptado de https://www.geeksforgeeks.org/how-to-perform-contrast-enhancement-using-histogram-equalization-in-matlab/) presenta un ejemplo simple de ecualización del histograma de una imagen en escala de grises en MATLAB:

```
% cargar la imagen
img=imread("butterfly.png");
% convertir a escala de grises
grayimg=im2gray(img);
% visualizar la imagen y su histograma
figure(1)
subplot(1,2,1)
imshow(grayimg);
subplot(1,2,2)
imhist(grayimg);

% calcular la frecuencia de cada valor de intensidad
[x,y]=size(grayimg);
Freq=zeros(1,256);
for i=1:x
    for j=1:y
        Freq(grayimg(i,j)+1)=Freq(grayimg(i,j)+1)+1;
    end
end

% calcular la funcion de densidad de probabilidad para cada intensidad
PDF=zeros(1,256);
Total=x*y;
for i=1:256
    PDF(i)=Freq(i)/Total;
end

% calcular la funcion de densidad acumulada para cada intensidad
```

```
CDF=zeros(1,256);
CDF(1)=PDF(1);
for i=2:256
    CDF(i)=CDF(i-1)+PDF(i);
end

% multiplicar por el maximo valor de intensidad del rango
Result=zeros(1,256);
for i=1:256
    Result(i)=CDF(i)*(255);
end

% generar la imagen ecualizada
eqimg=uint8(zeros(size(grayimg)));
for i=1:x
    for j=1:y
        eqimg(i,j)=uint8(Result(grayimg(i,j)+1));
    end
end

% visualizar la imagen ecualizada y su histograma
figure(2)
subplot(1,2,1)
imshow(eqimg);
subplot(1,2,2)
imhist(eqimg);

% guardar la imagen ecualizada en disco
imwrite(eqimg,"butterfly_eq.png");
```

- **Ejercicio 2**
  Ejecute el código para obtener la ecualización del histograma de la imagen `butterfly.png` (adjunta a este enunciado), e incorpore el resultado obtenido a su reporte.

- **Ejercicio 3**
  Compare visualmente los resultados obtenidos con la imagen `butterfly.png` (adjunta a este enunciado), y comente en su reporte las diferencias observadas tanto sobre la imagen como sobre el histograma, después de los procesos de normalización y ecualización. En particular, para esta imagen, ¿cuál operación genera el mejor resultado (visualmente hablando)?

3. Ecualización del histograma (función en MATLAB)

El siguiente código presenta un ejemplo simple de utilización de la función nativa de MATLAB para la ecualización del histograma de una imagen en escala de grises:

```
% cargar la imagen
img=imread("butterfly.png");
% convertir a escala de grises
grayimg=im2gray(img);
% visualizar la imagen y su histograma
figure(1)
subplot(1,2,1)
imshow(grayimg);
```

```
subplot(1,2,2)
imhist(grayimg);

% ecualizar el histograma de la imagen usando la funcion propia de MATLAB
eqimg = histeq(grayimg);

% visualizar la imagen ecualizada y su histograma
figure(2)
subplot(1,2,1)
imshow(eqimg);
subplot(1,2,2)
imhist(eqimg);

% guardar la imagen ecualizada en disco
imwrite(eqimg,"butterfly_eqM.png");
```

- **Ejercicio 4**
  Ejecute el código para obtener la ecualización del histograma de la imagen `butterfly.png` (adjunta a este enunciado), e incorpore el resultado obtenido a su reporte.

- **Ejercicio 5**
  Compare visualmente los resultados obtenidos con la imagen `butterfly.png` (adjunta a este enunciado), y comente en su reporte las diferencias observadas tanto sobre la imagen como sobre el histograma, después de los procesos de ecualización manual y con la función nativa de MATLAB.

4. Binarización y umbralización usando el método de Otsu

El siguiente código (adaptado de https://www.geeksforgeeks.org/binarization-of-digital-images-using-otsu-method-in-matlab/) presenta un ejemplo simple de estimación automática del umbral que permite posteriormente la binarización y umbralización de una imagen en escala de grises en MATLAB:

```
% cargar la imagen
img=imread("mushrooms.jpg");
% convertir a escala de grises
grayimg=im2gray(img);
% visualizar la imagen y su histograma
figure(1)
subplot(1,2,1)
imshow(grayimg);
subplot(1,2,2)
imhist(grayimg);

% calcular automaticamente el umbral utilizando el metodo de Otsu
level = graythresh(grayimg);

% binarizar utilizando el umbral calculado
binimg = imbinarize(grayimg,level);

% multiplicar la imagen original por la imagen binarizada para
% obtener la imagen umbralizada
umbimg = uint8(binimg) .* grayimg;

% visualizar y guardar la imagen binarizada
figure(2)
```

```
imshow(binimg);
imwrite(binimg,"mushrooms_bin.jpg");

% visualizar y guardar la imagen umbralizada
figure(3)
imshow(umbimg);
imwrite(umbimg,"mushrooms_umb.jpg");
```

- **Ejercicio 6**
  Ejecute el código para obtener la versión binarizada y umbralizada de la imagen `mushrooms.jpg` (adjunta a este enunciado), e incorpore los resultados obtenidos a su reporte (no olvidar el valor de umbral calculado).

5. Binarización local usando el método de Otsu

El siguiente código (adaptado de https://www.geeksforgeeks.org/binarization-of-digital-images-using-otsu-method-in-matlab/) presenta un ejemplo simple de estimación automática de los umbrales locales (con diferentes tamaños de ventana) que permiten la binarización local de una imagen en escala de grises en MATLAB:

```
% cargar la imagen
img=imread("page.jpeg");
% convertir a escala de grises
grayimg=im2gray(img);
% visualizar la imagen y su histograma
figure(1)
subplot(1,2,1)
imshow(grayimg);
subplot(1,2,2)
imhist(grayimg);

% encontrar los umbrales locales por ventanas de 50x50
loc50img=nlfilter(grayimg,[50 50], @ibimage);

% visualizar y guardar la imagen binarizada localmente
figure(2)
imshow(loc50img);
imwrite(loc50img,"page_bin50.jpeg");

% encontrar los umbrales locales por ventanas de 30x30
loc30img=nlfilter(grayimg,[30 30], @ibimage);

% visualizar y guardar la imagen binarizada localmente
figure(3)
imshow(loc30img);
imwrite(loc30img,"page_bin30.jpeg");

% funcion ibimage
% encuentra el umbral en cada ventana de la imagen
function f=ibimage(img)
  [x, y]=size(img);
  level=graythresh(img);
  bw=imbinarize(img,level);
  x1=round(x/2);
```

```
  y1=round(y/2);
  f=bw(x1, y1);
end
```

- **Ejercicio 7**
  Ejecute el código para obtener las versiones binarizadas localmente (con diferentes tamaños de ventana) de la imagen `page.jpeg` (adjunta a este enunciado), e incorpore los resultados obtenidos a su reporte.

- **Ejercicio 8**
  Utilizando el código del ejercicio 6 (binarización con único umbral), ejecútelo sobre la imagen `page.jpeg` (adjunta a este enunciado) y compare los resultados obtenidos con un único umbral y con umbrales locales. ¿Qué conclusiones pueden extraerse de esa comparación?

- **Ejercicio 9**
  Replique el proceso completo con dos imágenes médicas de diferentes modalidades:
  1. Normalización del histograma (ejercicio 1)
  2. Ecualización del histograma (ejercicios 2, 3, 4 y 5)
  3. Binarización y umbralización usando el método de Otsu (ejercicio 6)
  4. Binarización local usando el método de Otsu (ejercicios 7 y 8)

  ¿Estas operaciones permiten mejorar las imágenes utilizadas? ¿Permiten resaltar información que pueda ser usada posteriormente para el diagnóstico?

6. Binarización usando el método de Otsu (librería ITK)

El siguiente código en Python presenta un programa simple utilizando la librería ITK, en el cual se carga una imagen médica (volumen) y se binariza utilizando el método de Otsu, almacenando el resultado obtenido en disco, e imprimiendo en pantalla el valor del umbral calculado automáticamente:

```
import itk

# definir la imagen como volumen (3 dimensiones)
ImageType = itk.Image[itk.UC, 3]
# definir un lector para la imagen de entrada e inicializarlo
reader = itk.ImageFileReader[ImageType].New()
# definir el algoritmo de Otsu que binarizara la imagen
otsuFilter = itk.OtsuThresholdImageFilter[ImageType,ImageType].New()
# definir un escritor para la imagen de salida e inicializarlo
writer = itk.ImageFileWriter[ImageType].New()
# asignar al lector el nombre del archivo de entrada
reader.SetFileName("MRLiverTumor.nii.gz")
# asignar al escritor el nombre del archivo de salida
writer.SetFileName("MRLiverTumor_bin.nii.gz")
# conectar el algoritmo al lector para tomar la imagen de entrada
otsuFilter.SetInput(reader.GetOutput())
# conectar el escritor al algoritmo para tomar la imagen de salida
writer.SetInput(otsuFilter.GetOutput())
# ejecutar la linea de procesamiento completa
writer.Update()
# imprimir en pantalla el valor del umbral automatico
print(otsuFilter.GetThreshold())
```

- **Ejercicio 10**
  Ejecute el código para obtener la versión binarizada de la imagen médica `MRLiverTumor.nii.gz` (adjunta a este enunciado), e incorpore el resultado obtenido a su reporte (no olvidar el valor de umbral calculado). ¿La operación permitió generar información de interés diagnóstico o interpretativo de la imagen?

- **Ejercicio 11**
  Ejecute el mismo código, ahora para obtener la versión binarizada de las imágenes médicas `MRBrainTumor.nii.gz` y `MRBreastCancer.nii.gz` (adjuntas a este enunciado), e incorpore los resultados obtenidos a su reporte (no olvidar en cada caso el valor de umbral calculado). ¿La operación permitió generar información de interés diagnóstico o interpretativo de las imágenes?

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 16 de octubre 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.