Práctica 1. Mejoramiento de la Imagen

Andrés González Flores, Facultad de Ingeniería

—————————— ◆ ——————————

# 1 Introducción

E

L preprocecesado en el reconocimiento de patrones es una tarea muy importante, ya que siempre necesitaremos técnicas que nos permitan tener la información lo más libre de ruido posible.

Los métodos más comunes para lograr esto es el uso de filtros, por ejemplo, el uso de filtros paso bajas para eliminar ruido de tipo Gaussiano o el uso de filtros de mediana, los cuales son eficaces para deshacerse de ruidos tipo sal y pimienta.

Figura 1. Imagen preprocesada con filtro Gaussiano.

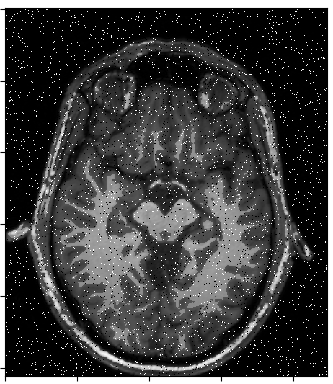
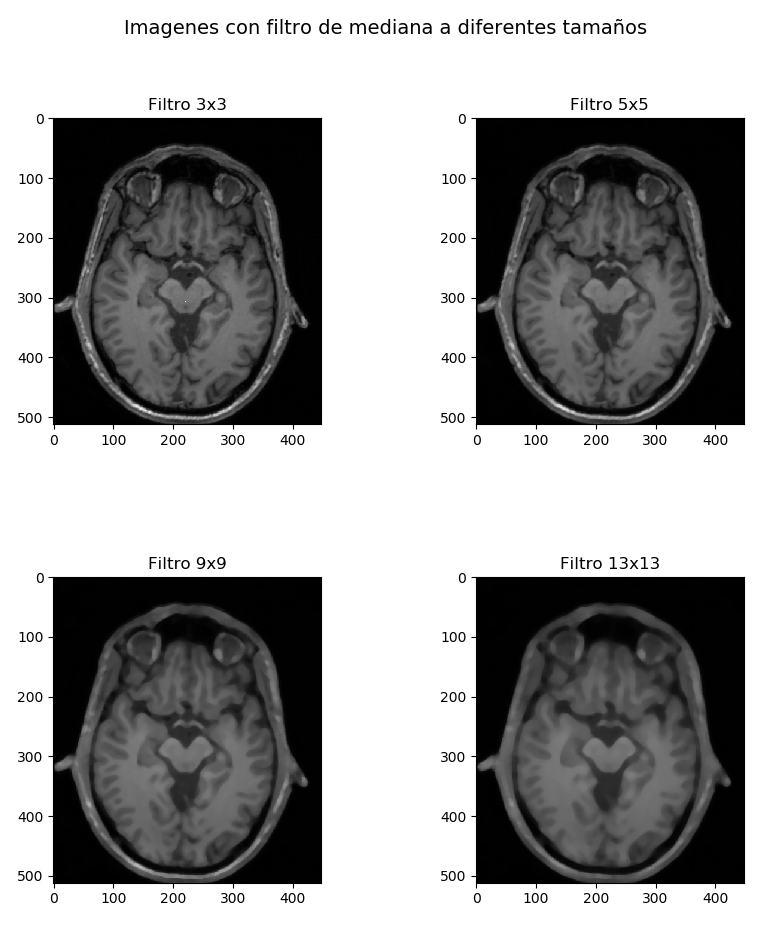
También existen filtros que resaltan características en la imagen, por ejemplox los filtros paso altas, que pueden ser utilizados para detectar cambios en el contraste y con ello, bordes en una imagen.

# 2 Objetivos

El alumno:

* Aprenderá a mejorar la imagen de acuerdo con las modificaciones que se le realizan originalmente.
* Entenderá aumentar las características de contraste, perfilado que apoyen a una mejor medición.

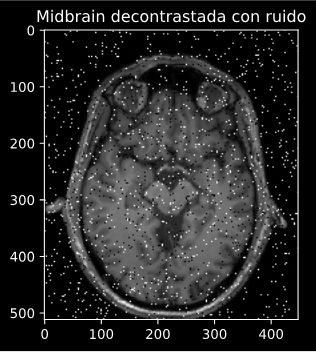
# 3 Desarrollo

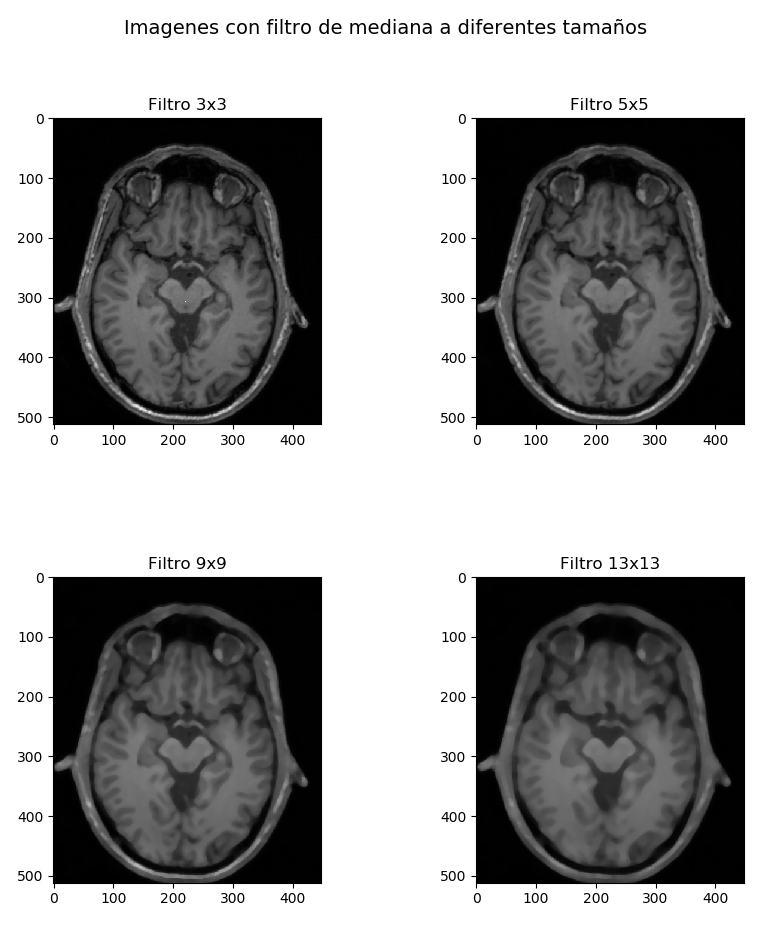
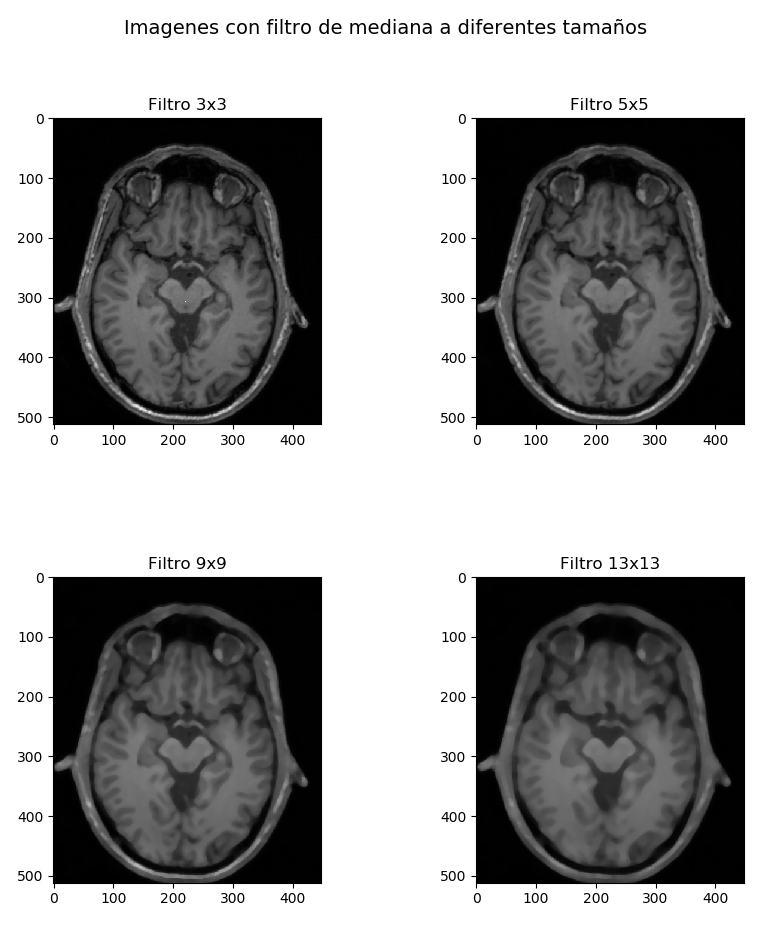
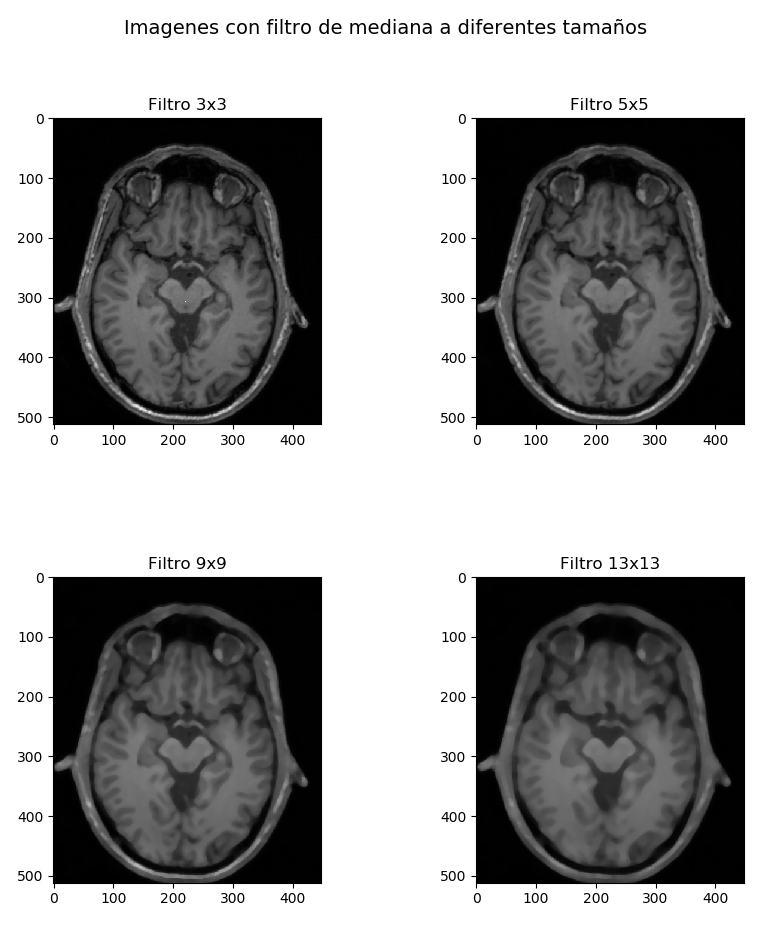
A. Se tiene en el archivo de Midbrain.mat 2 imágenes a cargar. La primera es midbrain que es la imagen original. A esta imagen se le realizó normalización entre 0 y 1, y se le añadió ruido sal y pimienta. Luego, se le modificó el contraste con la siguiente función

En realidad, no servía la imagen así, por lo que hay que aplicar el paso contrario a la misma.

1. Regresar a los valores iniciales de contraste.

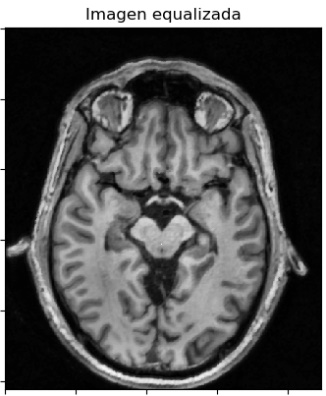
Para esto se usa la función inversa de contraste, quedando como:

Después de esta función la imagen queda de esta manera:

1. Luego, se le quitó el ruido. Usando un filtro de mediana de diferentes tamaños.

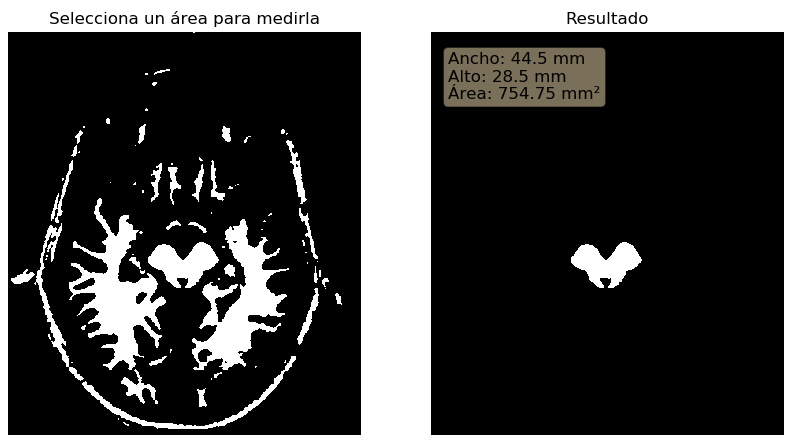
B. Ahora realice un filtrado para mejorar el procesamiento de la imagen. Y compare contra su imagen anterior para ver si mejoró.

Aquí utilicé una ecualización de histograma local, con tamaño de ventana de 8x8. Esto resalta el contraste en toda la imagen.

Después apliqué una umbralización. Con ello se logra descartar niveles de gris que no son de interés para la segmentación.

C. Luego se generó un clasificador para la imágen. El clasificador que escogí fue “region growing”, que consiste en los vecinos inmediatos del un determinado pixel y compararlos con éste, si su valor de gris es similar, los toma en cuenta para la región. Con esto puedo segmentar el mesencéfalo.

D. A partir de la imagen binaria, podemos medir longitudinal y transversalmente el mecensécalo. También se puede obtener el área que ocupa. La resolución de la imagen es de 0.5 mm por pixel.



Dando esto como resultado un ancho de 44.5 mm y de alto 28.5 mm. Con un área aproximada de 754.74 mm2.

# Conclusiones

Con el desarrollo de está practica comprendí la importancia del preprocesado para el buen funcionamiento de algoritmos de segmentación en imágenes.

En el apartado B del desarrollo, pude notar de manera práctica el desempeño de un filtro de mediana para disminuir drásticamente el ruido tipo sal y pimienta.

En el apartado C, el algorimo que usé para implementar region growing no fue el más óptimo, lo que afectó directamente al tiempo que tomó al programa segmentar y medir la región de interés.

# Código

import cv2

import numpy as np

import scipy.io as spio

import matplotlib.pyplot as plt

midbrain = spio.loadmat('./Midbrain.mat')

img\_midbrain = np.array(midbrain['midbrain'])

img\_midbrain\_thresh = np.array(midbrain['midbrainthesh'])

brain\_shape = img\_midbrain\_thresh.shape

plt.title('Midbrain original')

plt.imshow(img\_midbrain, cmap='gray')

plt.show()

plt.title('Midbrain threshold + noise')

plt.imshow(img\_midbrain\_thresh, cmap='gray')

plt.show()

# Generando la operación inversa

# x = (8/3)y si 0<y<0.25

# x = y si 0.4<y<0.6

# x = (8/3)y-1 si 0.6<y<0.75

img\_decontrastada = np.zeros(shape = brain\_shape)

for ii in range(brain\_shape[0]):

for jj in range(brain\_shape[1]):

if(img\_midbrain\_thresh[ii,jj] < 0.25):

img\_decontrastada[ii,jj] = (8/5)\*img\_midbrain\_thresh[ii,jj]

elif (img\_midbrain\_thresh[ii,jj] >= 0.25 and img\_midbrain\_thresh[ii,jj] <0.6):

img\_decontrastada[ii,jj] = img\_midbrain\_thresh[ii,jj]

elif img\_midbrain\_thresh[ii,jj] >= 0.6 and img\_midbrain\_thresh[ii,jj]<=0.75:

img\_decontrastada[ii,jj] = (8/3)\*img\_midbrain\_thresh[ii,jj]-1

# Quitar valores negativos

img\_decontrastada = np.add(img\_decontrastada, -np.amin(img\_decontrastada))

plt.title('Midbrain decontrastada con ruido')

plt.imshow(img\_decontrastada, cmap='gray', vmin='0', vmax='1')

plt.show()

# Pasar todo a enteros de 8 bits

img\_decontrastada = np.rint(np.multiply(img\_decontrastada, 255)).astype(np.uint8)

# Filtrar imágenes con diferentes filtros

img\_filtrada3 = cv2.medianBlur(img\_decontrastada, 3)

img\_filtrada5 = cv2.medianBlur(img\_decontrastada, 5)

img\_filtrada9 = cv2.medianBlur(img\_decontrastada, 9)

img\_filtrada13 = cv2.medianBlur(img\_decontrastada, 13)

fig = plt.figure(figsize = (10,10))

fig.suptitle('Imagenes con filtro de mediana a diferentes tamaños', fontsize=14)

ax1 = fig.add\_subplot(2, 2, 1)

ax1.set\_title('Filtro 3x3')

ax1.imshow(img\_filtrada3, cmap='gray', vmin='0', vmax='255')

ax2 = fig.add\_subplot(2, 2, 2)

ax2.set\_title('Filtro 5x5')

ax2.imshow(img\_filtrada5, cmap='gray', vmin='0', vmax='255')

ax3 = fig.add\_subplot(2, 2, 3)

ax3.set\_title('Filtro 9x9')

ax3.imshow(img\_filtrada9, cmap='gray', vmin='0', vmax='255')

ax4 = fig.add\_subplot(2, 2, 4)

ax4.set\_title('Filtro 13x13')

ax4.imshow(img\_filtrada13, cmap='gray', vmin='0', vmax='255')

plt.subplots\_adjust(hspace=0.5)

plt.show()

plt.title('Imagen a segmentar')

plt.imshow(img\_filtrada3, cmap='gray', vmin='0', vmax='255')

plt.show()

# Equalización local del histograma, con ventanas de 8x8

clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))

plt.title('Imagen equalizada')

img\_eq = clahe.apply(img\_filtrada3)

plt.imshow(img\_eq, cmap='gray')

plt.show()

# Aplicar umbralización para quitar zonas que no necesito

ret,img\_bin = cv2.threshold(img\_filtrada3,100,255,cv2.THRESH\_BINARY)

plt.title('Imagen umbralizada')

plt.imshow(img\_bin, cmap='gray')

plt.show()

cv2.imwrite('./imagen\_preprocesada.bmp', img\_bin)

def obtener\_8\_vecinos(x, y, shape):

out = []

maxx = shape[1]-1

maxy = shape[0]-1

#top left

outx = min(max(x-1,0),maxx)

outy = min(max(y-1,0),maxy)

out.append((outx,outy))

#top center

outx = x

outy = min(max(y-1,0),maxy)

out.append((outx,outy))

#top right

outx = min(max(x+1,0),maxx)

outy = min(max(y-1,0),maxy)

out.append((outx,outy))

#left

outx = min(max(x-1,0),maxx)

outy = y

out.append((outx,outy))

#right

outx = min(max(x+1,0),maxx)

outy = y

out.append((outx,outy))

#bottom left

outx = min(max(x-1,0),maxx)

outy = min(max(y+1,0),maxy)

out.append((outx,outy))

#bottom center

outx = x

outy = min(max(y+1,0),maxy)

out.append((outx,outy))

#bottom right

outx = min(max(x+1,0),maxx)

outy = min(max(y+1,0),maxy)

out.append((outx,outy))

return out

def region\_growing(img, seed):

print('Inicia el proceso. Espere por favor...')

minx = 100000

miny = 100000

maxx = 0

maxy = 0

total\_px = 0

lista = []

outimg = np.zeros\_like(img)

lista.append((seed[0], seed[1]))

processed = []

while(len(lista) > 0):

pix = lista[0]

outimg[pix[0], pix[1]] = 255

for coord in obtener\_8\_vecinos(pix[0], pix[1], img.shape):

if img[coord[0], coord[1]] != 0:

outimg[coord[0], coord[1]] = 255

#Encontrar las dimensiones y los pixeles que ocupa

minx = coord[0] if coord[0]<minx else minx

miny = coord[1] if coord[1]<miny else miny

maxx = coord[0] if coord[0]>maxx else maxx

maxy = coord[1] if coord[1]>maxy else maxy

if not coord in processed:

total\_px += 1

lista.append(coord)

processed.append(coord)

lista.pop(0)

print('Termina el proceso...')

dimx = maxx-minx

dimy = maxy-miny

return outimg, [dimx, dimy], total\_px

def onmouseclick(event):

if(event.inaxes == ax1):

seed = [np.round(event.ydata).astype(np.int), np.round(event.xdata).astype(np.int)]

img\_rg, dim\_img, area\_img = region\_growing(img\_bin, seed)

ax2.imshow(img\_rg, cmap='gray')

textstr = '\n'.join((

'Ancho: %.1f mm' % (dim\_img[1]\*0.5),

'Alto: %.1f mm' % (dim\_img[0]\*0.5),

'Área: %.2f mm\u00b2' % (area\_img\*0.25)))

text\_box.set\_text(textstr)

fig1.canvas.draw()

img\_bin = cv2.imread('./imagen\_preprocesada.bmp', 0)

fig1 = plt.figure(figsize=(10, 7))

fig1.canvas.mpl\_connect('button\_press\_event', onmouseclick)

ax1 = fig1.add\_subplot(121)

ax1.set\_title('Selecciona un área para medirla')

ax1.imshow(img\_bin, cmap='gray')

ax1.axis('off')

ax2 = fig1.add\_subplot(122)

ax2.set\_title('Resultado')

ax2.axis('off')

ax2.imshow(np.zeros([img\_bin.shape[0], img\_bin.shape[1]]).astype(np.uint8), cmap='gray')

textstr = '\n'.join((

'Ancho: 0 mm',

'Alto: 0 mm',

'Área: 0 mm\u00b2'))

props = dict(boxstyle='round', facecolor='wheat', alpha=0.5)

text\_box = ax2.text(0.05, 0.95, textstr, transform=ax2.transAxes, fontsize=12,

verticalalignment='top', bbox=props)

plt.show()

**Referencias**

1. Gonzalez Woods Digital Image Processing, 2004
2. <http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/MARBLE/medium/segment/recurse.htm>
3. <https://stackoverflow.com/questions/43923648/region-growing-python>
4. <https://docs.opencv.org/3.4.0/d7/d4d/tutorial_py_thresholding.html>