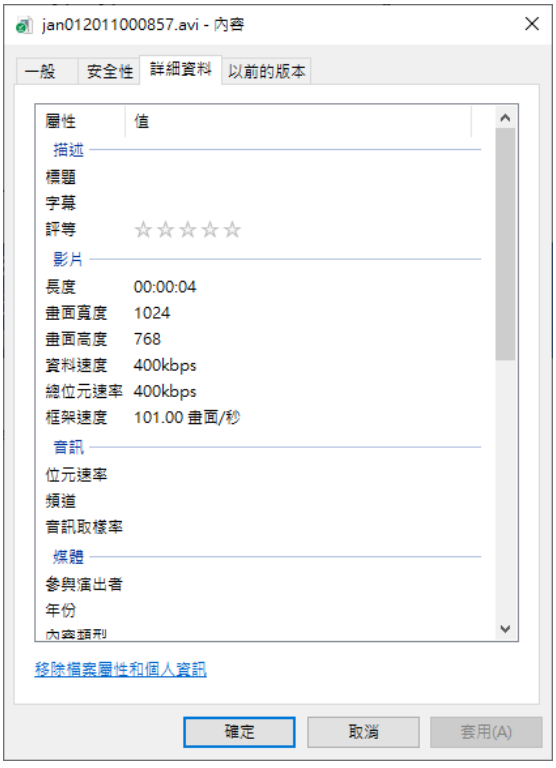
2022 fall 生醫影像處理系統 Homework4

Ultrasound Imaging and Carotid Artery Stiffness Estimation

學號：611410083 姓名：劉宇廷

一、 擷取超音波影像

超音波影像在儀器中收集完成後，可選擇儲存至隨身碟，再複製到電腦上作處理，檔名是以拍攝日期為開頭，因使用前未先將時間標準化，故出現日期錯誤的情形發生，預設的副檔名為.avi，影片長度為 4 秒，解析度為 1024 * 768，詳細資料如下圖所示。



二、 Convert Video into Image Frames

目標是希望從 4 秒的影片中產生 500 張連續影像，搜尋 Google 後發現不論是 Python 或是 MATLAB 都有相關的作法，比較兩者方式所得之影像，並無明顯差異，詳細程式碼如下所示。

1. Python

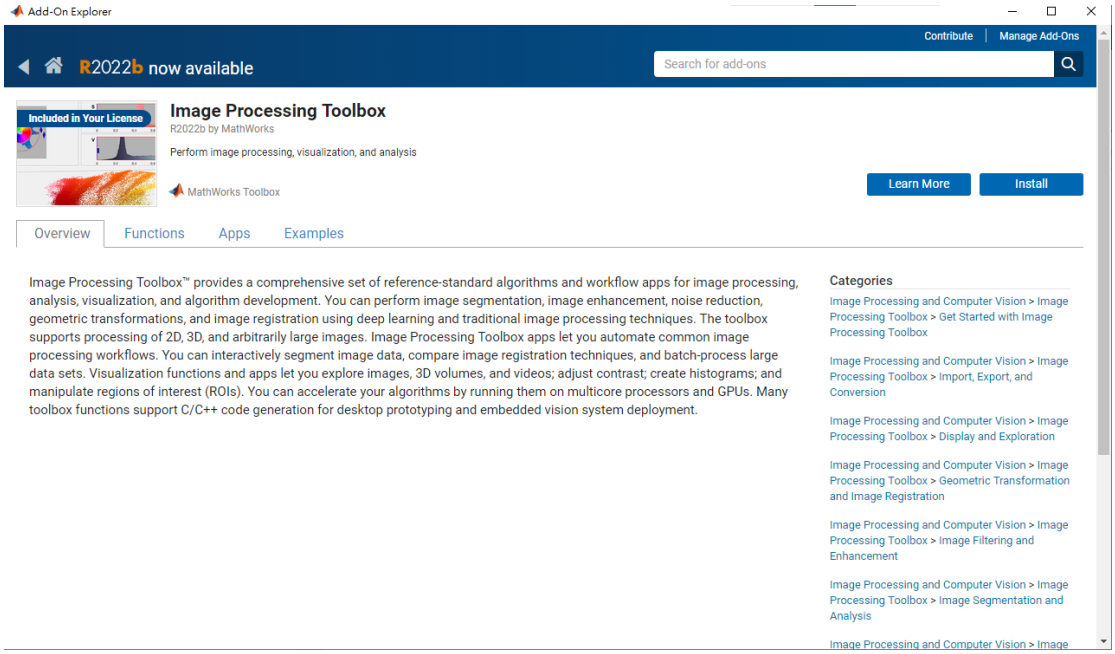
```
import cv2

vidcap = cv2.VideoCapture( "/data/2TSSD/ytl0623/1.avi" )

def getFrame( sec ) :
  vidcap.set( cv2.CAP_PROP_POS_MSEC, sec * 1000 )
  hasFrames, image = vidcap.read()

  if hasFrames :
      cv2.imwrite( "image" + str( count ) + ".jpg",
image )  # save frame as JPG file
  return hasFrames

sec = 0
frameRate = 0.008  # capture image in each 0.008 second
count = 1
success = getFrame( sec )

while success :
  count = count + 1
  sec = sec + frameRate
  sec = round( sec, 2 )
  success = getFrame( sec )
```

2. MATLAB

```
vid = VideoReader( "C:\Users\User\Desktop\1.avi" ) ;
numFrames = vid.NumberOfFrames ;
n = numFrames ;

for i = 1:1:n
  frames = read( vid, i ) ;
  imwrite( frames, [ 'Image' int2str(i), '.jpg' ] ) ;
  im( i ) = image( frames ) ;
end
```

三、 Image Processing: Filter

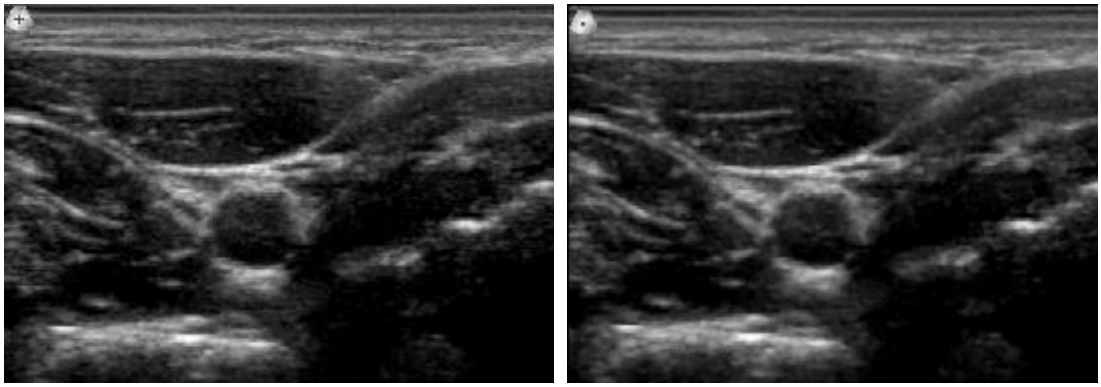
在進行影像處理之前，可以在 MATLAB 中下載影像處理工具箱 Image Processing Toolbox，有助於我們進行影像處理。



1. Median Filter [1]

中值濾波 (median filtering) 是一個非線性的運算，它是將目前要處理的像素及其週邊共 $(2m+1)^2$ 個像素 (要處理的像素位於最中央) 根據灰階值大小排序，再以排在最中間的灰階取代被處理像素的灰階，適用影像為灰階影像，特色是影像比較不會模糊，但執行速度偏慢，詳細程式碼如下所示。

```
# Read image into workspace.
I = imread('C:\Users\ytlWin\Desktop\1.jpg');

# Convert RGB image to grayscale image.
I = rgb2gray(I);

# Use a median filter to filter out the noise.
K = medfilt2(I);

# Display results, side-by-side.
imshowpair(I,K,'montage')
```

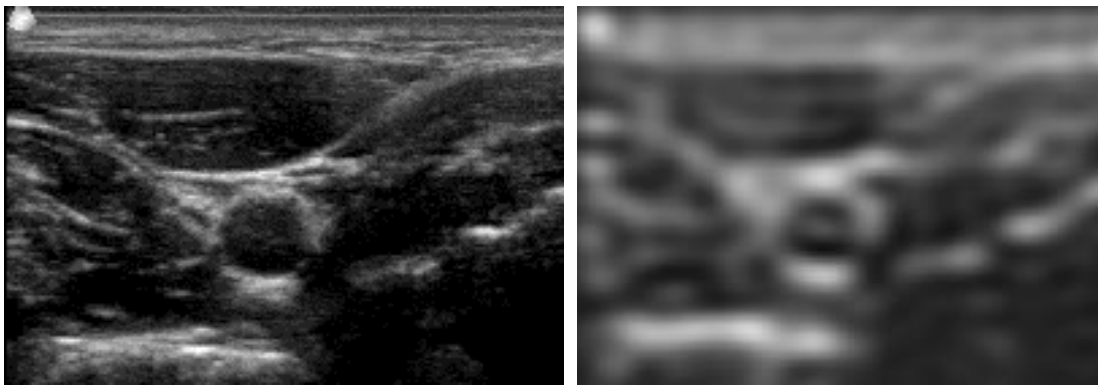


2. Lowpass Filter [2]

Step 1: Input – Read an image
Step 2: Saving the size of the input image in pixels
Step 3: Get the Fourier Transform of the input_image
Step 4: Assign the Cut-off Frequency D_{0}
Step 5: Designing filter: Ideal Low Pass Filter
Step 6: Convolution between the Fourier Transformed input image and the filtering mask
Step 7: Take Inverse Fourier Transform of the convoluted image
Step 8: Display the resultant image as output

```
# Reading input image
input_image = imread('C:\Users\ytlWin\Desktop\1.jpg');

# Convert RGB image to grayscale image.
input_image = rgb2gray(input_image);

# Saving the size of the input_image
[M, N] = size(input_image);

# Getting Fourier Transform of the input_image
FT_img = fft2(double(input_image));

# Assign Cut-off Frequency
D0 = 30;

# Designing filter
u = 0:(M-1);
idx = find(u>M/2);
u(idx) = u(idx)-M;
v = 0:(N-1);
idy = find(v>N/2);
v(idy) = v(idy)-N;
```

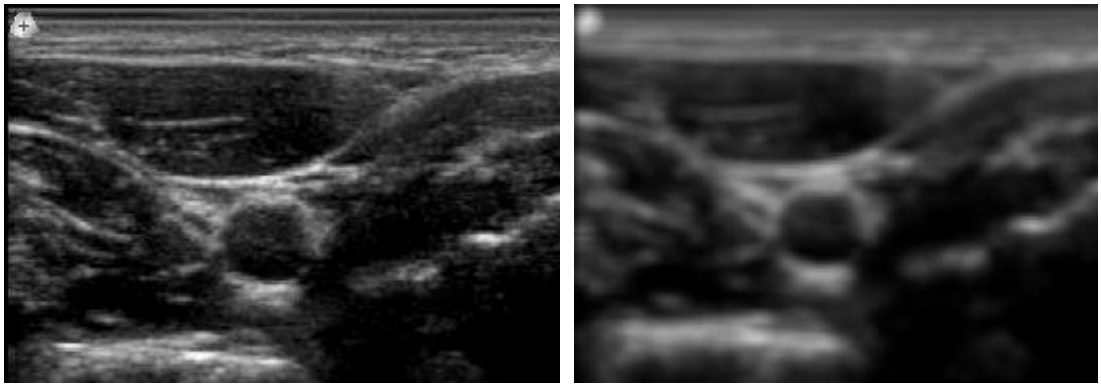
```
[V, U] = meshgrid(v, u);

# Calculating Euclidean Distance
D = sqrt(U.^2+V.^2);

# Comparing with the cut-off frequency and determining
the filtering mask
H = double(D <= D0);

# Convolution between the Fourier Transformed image and
the mask
G = H.*FT_img;

# Getting the resultant image by Inverse Fourier
Transform of the convoluted image
output_image = real(ifft2(double(G)));

# Displaying Input Image and Output Image
subplot(2, 1, 1), imshow(input_image),
subplot(2, 1, 2), imshow(output_image, [ ]);
```



依實驗結果我們可以看到，畫面變得非常的模糊，並不利於我們做之後的影像處理。

3. Box Filter [3]

```
# Read image into workspace.
I = imread('C:\Users\ytlWin\Desktop\1.jpg');

# Convert RGB image to grayscale image.
I = rgb2gray(I);

# Perform the mean filtering using an 11-by-11 filter.
localMean = imboxfilt(I,11);

# Display the original image and the filtered image,
side-by-side.
imshowpair(I,localMean,'montage')
```



如上圖所示，雖然 Box Filter 比 LowPass Filter 清楚一點，但相較於原圖還是模糊不少，因此該方法無法有效去除途中部必要細節。

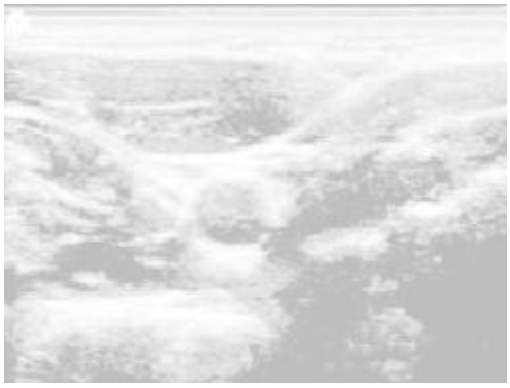
四、 Image Processing: Enhance Contrast

imadjust() 將輸入圖像的值映射到新值，對輸入資料中強度最低和最高的 1%（預設值）資料進行飽和處理，進而提高圖像的對比度。histeq() 執行長條圖均衡化，它可以變換圖像中的值，使得輸出圖像的長條圖近似匹配指定的長條圖（預設情況下為均勻分佈），進而增強圖像的對比度。adapthisteq() 執行對比度受限的自我調整長條圖均衡化，與 histeq() 不同，它對小資料區域（圖塊）而不是整個圖像執行運算。它會增強每個圖塊的對比度，使得每個輸出區域的長條圖近似匹配指定的長條圖（預設情況下為均勻分佈），可以限制對比度增強，以避免放大圖像中可能存在的雜訊。 [4]

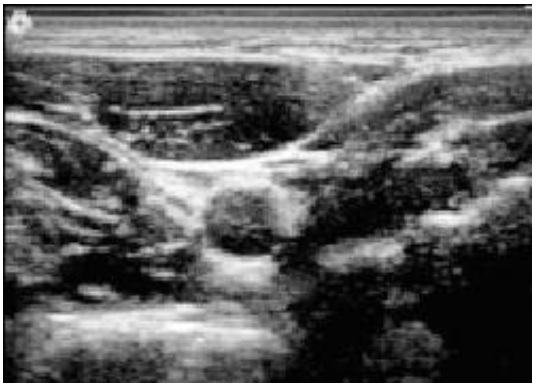
```
# Read image into workspace.
pout = imread('C:\Users\ytlWin\Desktop\1.jpg');

# Convert RGB image to grayscale image.
pout = rgb2gray(pout);

# Enhance the image using the three contrast adjustment
techniques with default settings.
pout_imadjust = imadjust(pout);
pout_histeq = histeq(pout);
pout_adapthisteq = adapthisteq(pout);

# Display the original image and the three contrast
adjusted images as a montage.
montage({pout,pout_imadjust,pout_histeq,pout_adapthisteq}
,"Size",[1 4])
```

Original Image

Enhanced by imadjust()

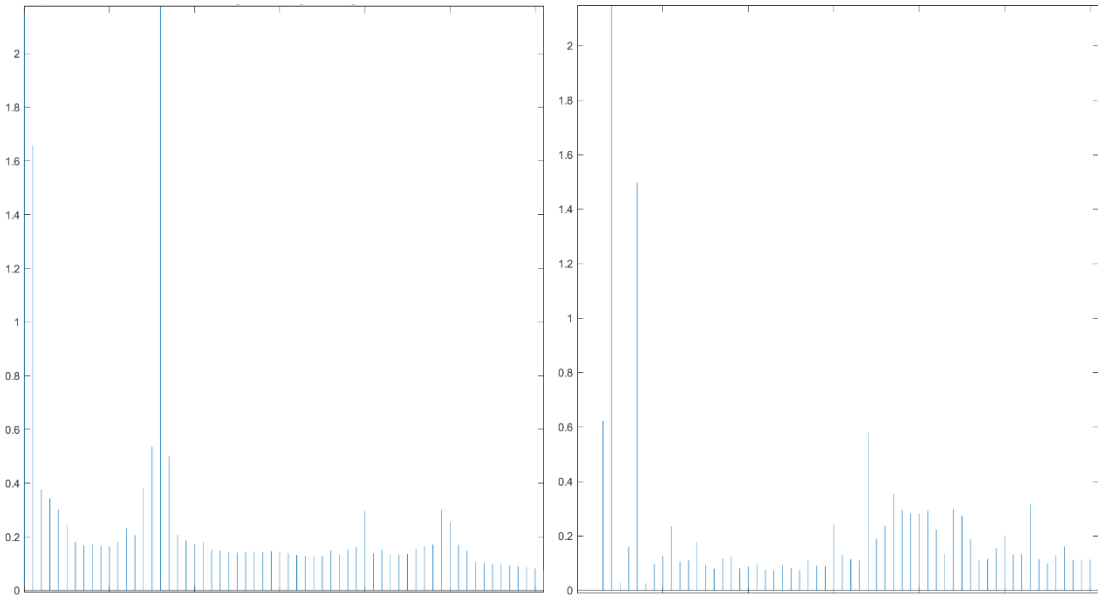


Enhanced by histeq()



Enhanced by adapthisteq()

Original Image and Enhanced Images using
imadjust(), histeq(), and adapthisteq()



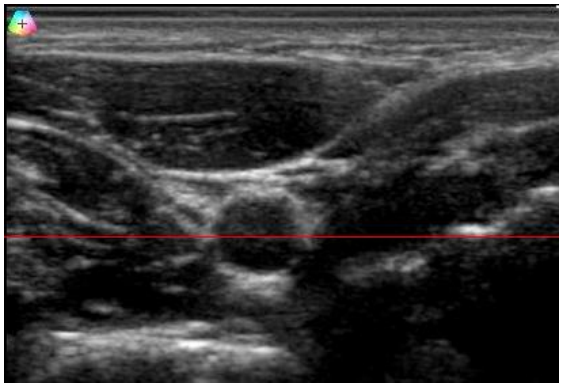
Histogram of Original Image and Enhanced Images using adapthisteq()

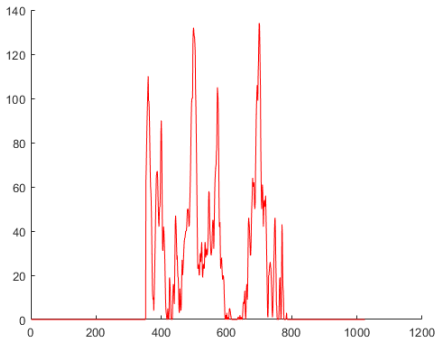
五、 Image Processing: Image Segmentation

1. 二值化 Image Binarization [5]

以特定顏色數值為閾值 (Threshold)，把影像轉為二值圖 (Binary Image)，超過閾值為白色，不超過為黑色。這是最簡單的一種圖像分割法，缺點是閾值要人為決定。

a. 我們可以沿著某條線將圖像中的像素提取出來，觀察像素值變化最大的地方。





b. 觀察像素可以看出在頸動脈邊緣處的像素值都大於 100，而數值變化處即為邊緣，這裡可以把閾值設為 100，並把圖像中數值小於閾值的部分篩選出來。

```
# Read image into workspace.
I = imread('C:\Users\ytlWin\Desktop\1.jpg');
imshow(I)
row = 250;
hold on
h = plot([0 size(I,2)],[row, row],'r-');

figure
hold on
plot(I(row,:,1),'r-')
plot(I(row,:,2),'g-')
plot(I(row,:,3),'b-')

threshold = 100;
binary = I(:,:,1) > threshold;
imshow(binary)
```

2. 可適性二值化 Adaptive Thresholding [6]

可以根據鄰近像素決定區域閾值，優點是不用人為決定閾值，且能適應光線不佳的影像，可惜的是這個方法的閾值適應範圍非常局部。

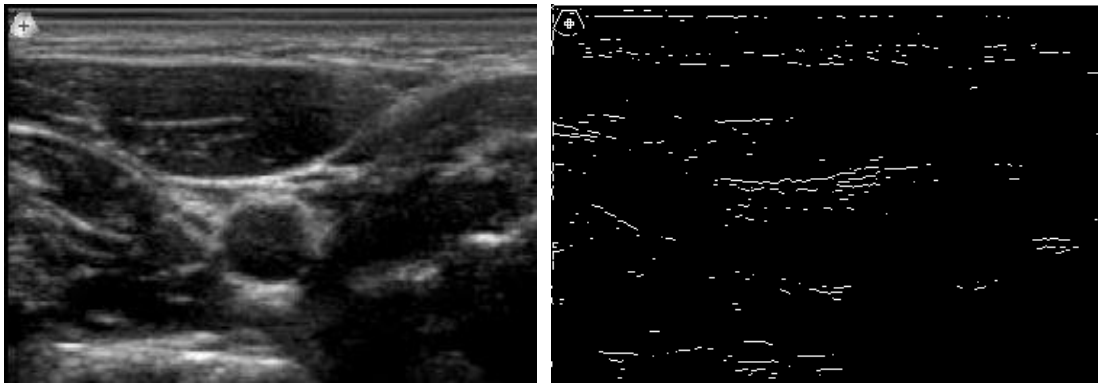
```
I = imread('C:\Users\ytlWin\Desktop\1.jpg');
gray = rgb2gray(I);
adaptive = imbinarize(gray,'adaptive');
imshow(adaptive)
```

3. 邊緣偵測 Edge Detection [7]

計算影像梯度 (Gradient)，再選一個適當的閾值針對梯度把圖像二值化，圖像梯度大的地方即為邊緣。

a. Sobel 算法

```
I = imread('C:\Users\ytlWin\Desktop\1.jpg');
gray = rgb2gray(I);
sobel = edge(gray);
imshow(sobel)
```



Edge Detection using Sobel

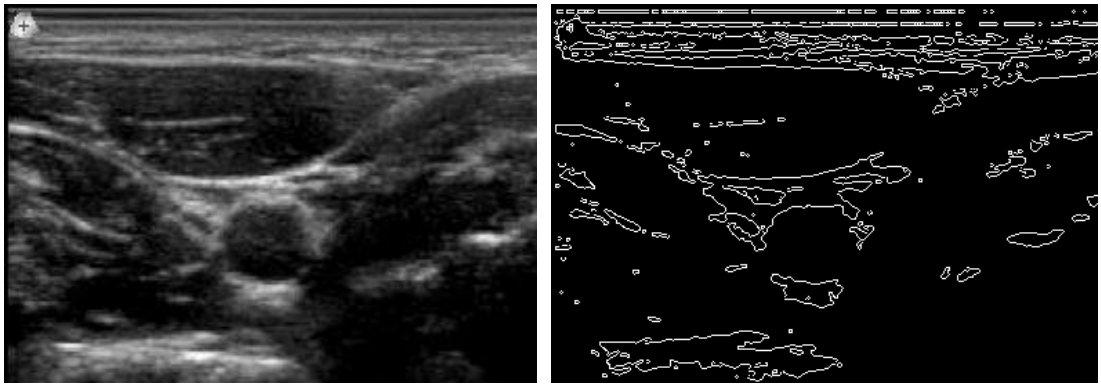
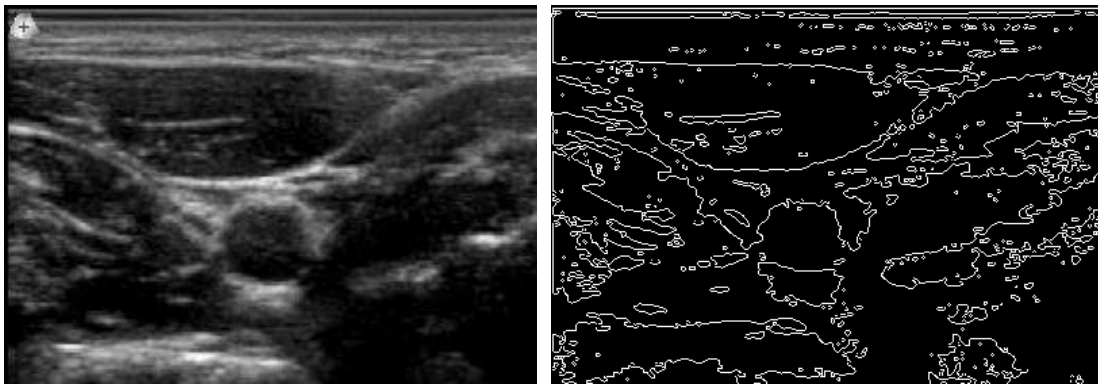


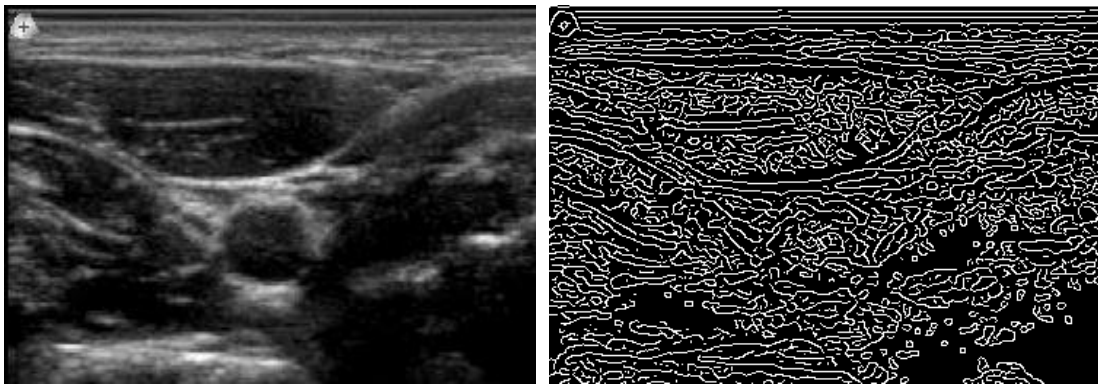
Image Binarization + Edge Detection using Sobel



Adaptive Thresholding + Edge Detection using Sobel

b. Canny 算法

```
I = imread('C:\Users\ytlWin\Desktop\1.jpg');
gray = rgb2gray(I);
[canny, thres] = edge(gray,'canny');
imshow(canny)
```



Edge Detection using Canny

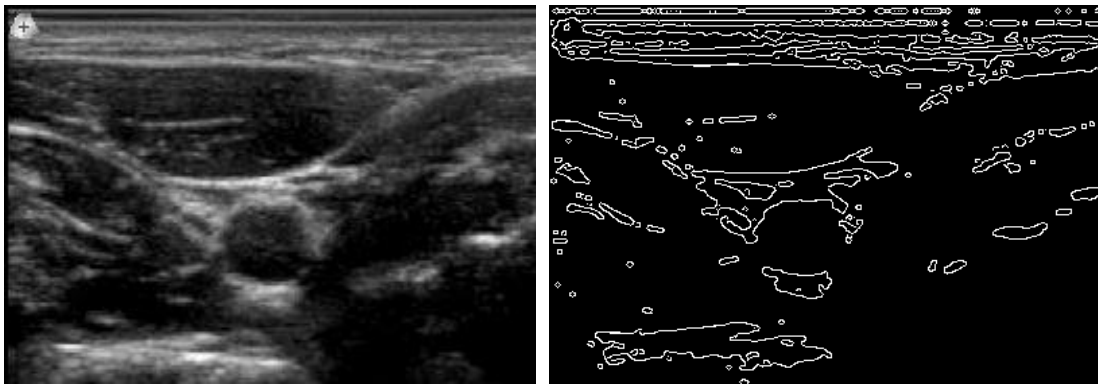
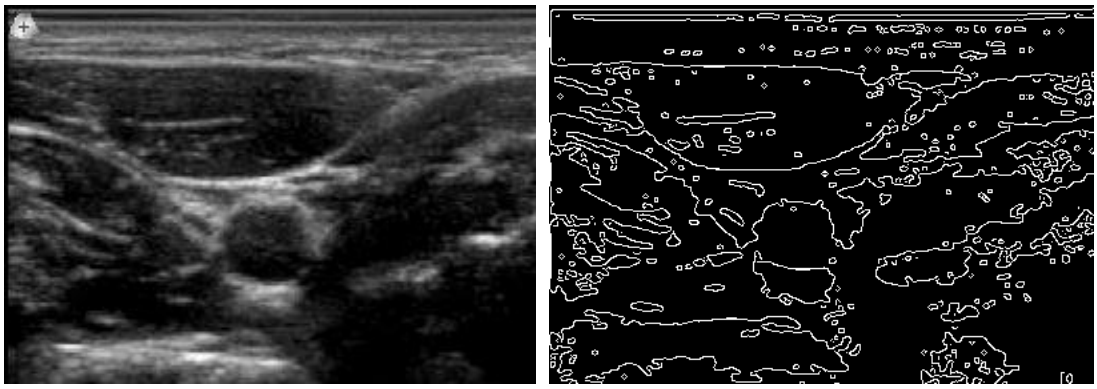


Image Binarization + Edge Detection using Canny



Adaptive Thresholding + Edge Detection using Canny

如上圖結果所示，偵測出來的邊界還是有雜訊 (Noise)，我們可以利用均值濾波器 (Average Filter) 影像平滑化，讓影像對它進行卷積運算，所有像素點的數值都會跟周圍的像素平均，另外 edge() 方法會回傳一個 thres 代表篩選邊緣的閾值，我們可以更改這個數值，再把它作為參數輸入到 edge() 裡面來得到理想的邊緣。

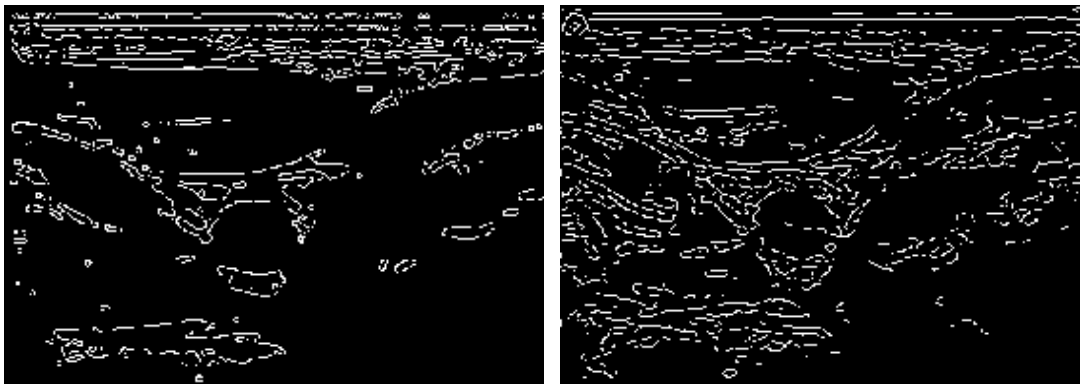
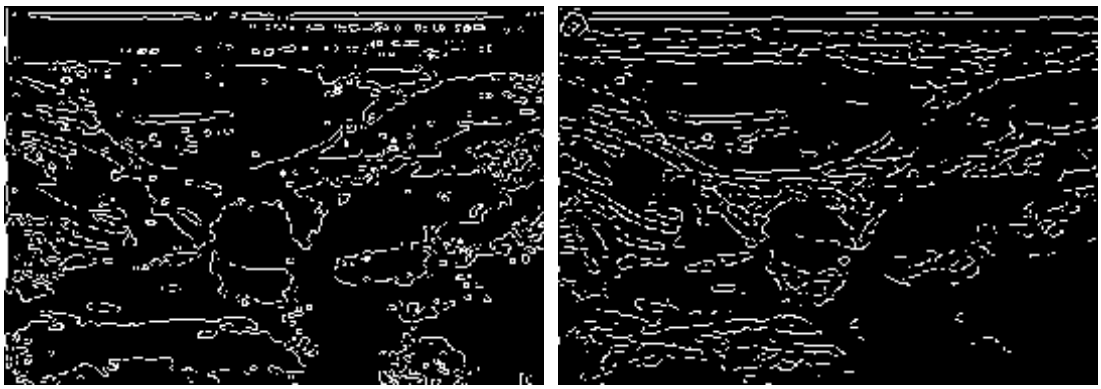


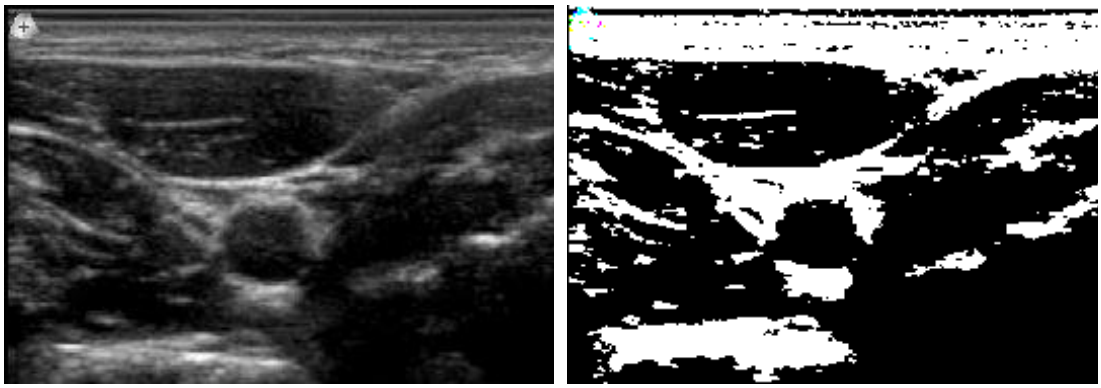
Image Binarization + Edge Detection using Canny (thres + 0.1)



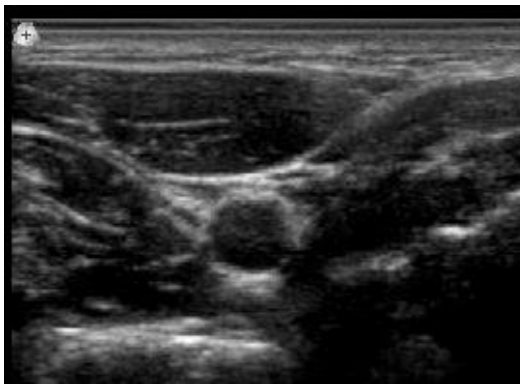
Adaptive Thresholding + Edge Detection using Canny (thres + 0.1)

4. Otsu's method Segmentation [9]

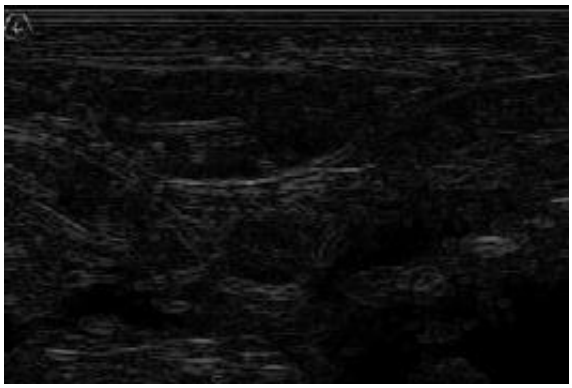
```
I = imread('C:\Users\ytlWin\Desktop\1.jpg');
gray = rgb2gray(I);
level = graythresh(I)
BW = imbinarize(I,level);
imshowpair(I,BW,'montage')
```



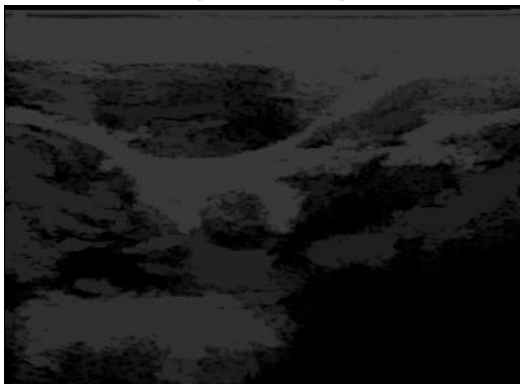
5. Watershed Segmentation



Original Image



Gradient Magnitude



Opening-by-Reconstruction

Opening-Closing by Reconstruction

六、 參考資料

1. 2-D median filtering。

   https://www.mathworks.com/help/images/ref/medfilt2.html

2. Lowpass Filter。https://www.geeksforgeeks.org/matlab-ideal-lowpass-filter-in-image-processing/

3. 2-D box filtering of images。

   https://www.mathworks.com/help/images/ref/imboxfilt.html

4. Enhance Grayscale Images。

   https://www.mathworks.com/help/images/contrast-enhancement-techniques.html

5. Image Binarization。

   https://yuchungchuang.wordpress.com/2021/01/21/matlab-%E5%BD%B1%E5%83%8F%E8%99%95%E7%90%86-image-segmentation/

6. Adaptive Thresholding。

   https://yuchungchuang.wordpress.com/2021/01/21/matlab-%E5%BD%B1%E5%83%8F%E8%99%95%E7%90%86-image-segmentation/

7. Edge Detection。

   https://yuchungchuang.wordpress.com/2021/01/21/matlab-%E5%BD%B1%E5%83%8F%E8%99%95%E7%90%86-image-segmentation/

8. Image Smoothing。

   https://yuchungchuang.wordpress.com/2021/01/21/matlab-%E5%BD%B1%E5%83%8F%E8%99%95%E7%90%86-image-segmentation/

9. Global image threshold using Otsu's method。

   https://www.mathworks.com/help/images/ref/graythresh.html