UNITED INTERNATIONAL UNIVERSITY

Department of Electrical and Electronics Engineering

EEE 4331: Biomedical Engineering

SUMMER 2023

Assignment – 02

[MATLAB and Image-J demo and analysis]

Submitted by

Student Name	Ayman Zafar				
Student ID	021 191 058	**Dept.**	EEE	**Sec.**	A
Date of Submission	Aug 24, 2023				

Submitted To

Dr. Khawza Iftekhar Uddin Ahmed
Professor
Department of Electrical & Electronics Engineering

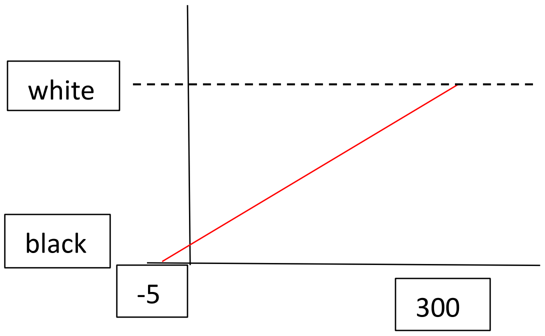
Part 1: Contrast Enhancement

Theory

Goal: To create a clear understanding of how we can map image intensities to screen brightness.

Contrast/Brightness

You have a choice of how to map image intensities to screen gray levels. Matlab, by default, maps min to black and max to white.



Min. intensity is -5

Max intensity in +300

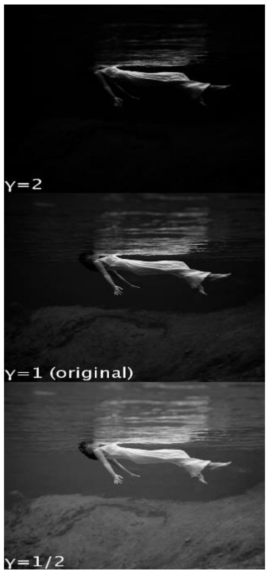
Gamma Correction

A common way to map the image intensities is called gamma correction.

Effect:

γ= ½; Enhances dark contrast at cost of bright contrast.

γ=2; Enhances bright contrast at cost of dark contrast.



Better contrast in dress

No background contrasts.

Better contrast in background(dark),

But less contrast in dress(bright).

Part 2: Denoising

Theory

Goal: To find out the types of compromises we use to try to remove noise from images. There are different types of noise in images.

In an MRI, the acquired (raw) data is complex-valued. Both the real & imaginary parts have Gaussian noise. When you look at the noise in the magnitude of the reconstructed images, its distribution is called Rician.

There are many approaches for noise remove such as,

Median filter = Speckle noise/ impulse noise

Look in a neighborhood around a pixel and assign the median value.

10	15	21
11	18	30
9	11	12

Middle=median=12

Assign the intensity of 12 where the 18 is.

Windowed Averaging

windowed averaging is equivalent to multiplying the Fourier coefficients by the sinc function. Thus, the low frequencies are maintained, but the high frequencies are largely dampened.

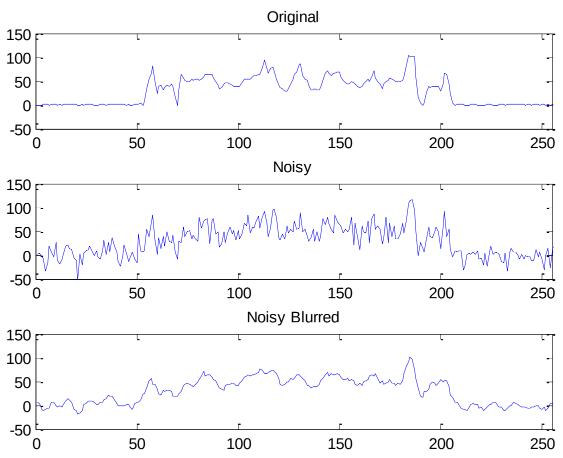
MATLAB code:

```
img = imread('t2.jpg');
img = double(img(:,:,1));
f = img(128,:); % extract only the
128th row
F = fftshift( fft( ifftshift(f) ) ); %
used later
```

```
figure(1);
subplot(3,1,1); plot(f);
title('Original');
axis([0 255 -50 150]); % set axis
limits same on all plots
```

```
% Add some Gaussian noise
fn = f + randn(1,256)*15;
Fn = fftshift( fft( ifftshift(fn) ) );
subplot(3,1,2); plot(fn);
title('Noisy');
axis([0 255 -50 150]); % set axis
limits same on all plots

% Window-averaging kernel (constant)
h = zeros(1,256);
radius = 2;
h((129-radius):(129+radius)) = 1;
h = h / sum(h(:)); % normalize (so they
add to 1)
H = fftshift( fft( ifftshift(h) ) );

% Convolve (in the frequency domain)
G = H .* Fn;
g = fftshift( ifft( ifftshift(G) ) );
subplot(3,1,3); plot(real(g));
title('Noisy Blurred');
axis([0 255 -50 150]); % set axis
limits same on all plots

%% Look at the situation in the
frequency domain
figure(2);

% Frequency domain: original and noisy
subplot(2,1,1);
plot(log(abs(Fn)+1),'r'); hold on;
plot(log(abs(F)+1),'b'); hold off;
title('FT of original (blue) and noisy
(red)');

% Frequency domain: effect of filter
subplot(2,1,2);
plot(10*log(abs(H)+1),'k'); hold on; %
scaled to see it better
plot(log(abs(G)+1),'r'); hold off;
title('FT of filter (green) and
denloised (red)');
```

Output:



Additive Gaussian with σ=5

Filtered with (+2 to -2)

Figure 1

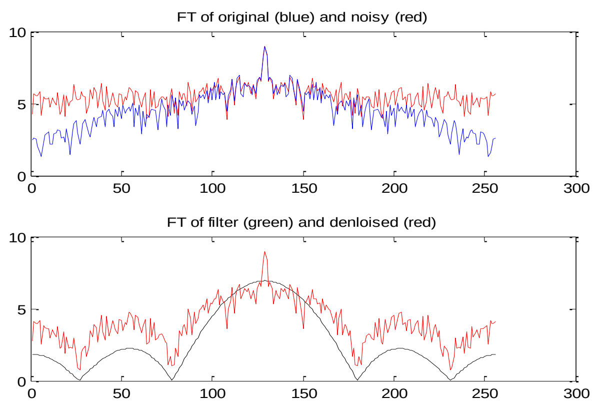


Figure 2

Analysis and discussion:

Fig-2:

The graph is in the log scale. The blue one is the FT of original signal and the Red one is the noisy signal.

The noise tense to disrupt the high frequencies quite a bit. The low frequencies are not affected very much but the high frequencies are. So, the noise often influences the high frequencies. Most information/images are low frequency information.

In the second graph, the black one is the FT of sinc. Red(noisy) multiplied by sinc function. The result is not that perfect compared to the blue signal.

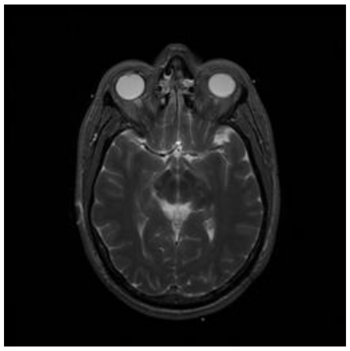
About Images:

MATLAB code:

```
f = imread('t2.jpg');
f = double(f(:,:,1));
figure(3);
imshow(f,[]); title('Original');

% Add some noise
fn = f + randn(size(f))*15;
figure(4);
imshow(fn,[]); title('Noisy');

% Create a convolution kernel

% Block filter
radius = 2;
h = zeros(size(f));
h((128-radius):(128+radius),(128-
radius):(128+radius)) = 1;

% Or Gaussian
%h = Gaussian(1.5, size(f));

% Normalize so sum is 1
h = h / sum(h(:));

% Convolve (in frequency domain)
% Notice I don't fftshift in the
frequency domain; no need to
% since I'm not visualizing it.
Fn = fft2( ifftshift(fn) );
H = fft2( ifftshift(h) );
G = H .* Fn;

% IDFT to get the result
g = fftshift( ifft2( G ) );

figure(5);
imshow(real(g), []);
title('Filtered Noisy Image');
```

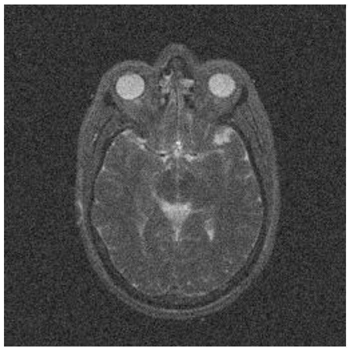
Output images:

Original



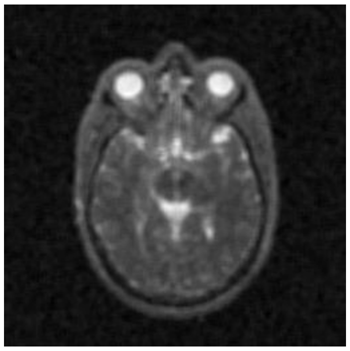
8 bit

Noisy



σ=15

Filtered Noisy Image



Filtered

Box filter with radius = 2

Gaussian Smoothing

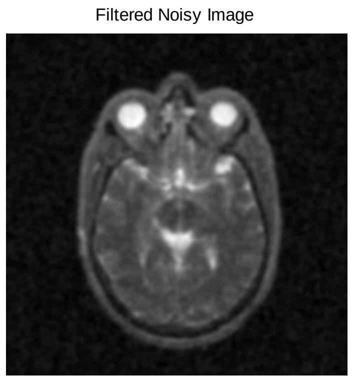
Similar to windowed averaging above, but using a Gaussian kernel. The FT of a Gaussian is a Gaussian, so the high frequencies are all but gone.

MATLAB code for Gaussian:

```
f = imread('t2.jpg');
f = double(f(:,:,1));
figure(3);
imshow(f,[]); title('Original');

% Add some noise
fn = f + randn(size(f))*15;
figure(4);
imshow(fn,[]); title('Noisy');

% Create a convolution kernel

% Block filter
radius = 2;
h = zeros(size(f));
h((128-radius):(128+radius),(128-
radius):(128+radius)) = 1;

% Or Gaussian
h = Gaussian(1.5, size(f));

% Normalize so sum is 1
h = h / sum(h(:));

% Convolve (in frequency domain)
% Notice I don't fftshift in the
frequency domain; no need to
% since I'm not visualizing it.
Fn = fft2( ifftshift(fn) );
H = fft2( ifftshift(h) );
G = H .* Fn;

% IDFT to get the result
g = fftshift( ifft2( G ) );

figure(5);
imshow(real(g), []);
title('Filtered Noisy Image');
```

[Note: the code is similar to the windowed averaging but to add an extra line for gaussian.]

Output:



Analysis and discussion:

It seems almost similar image to windowed averaging (filtered noisy image). But

if we look there is a bit difference.

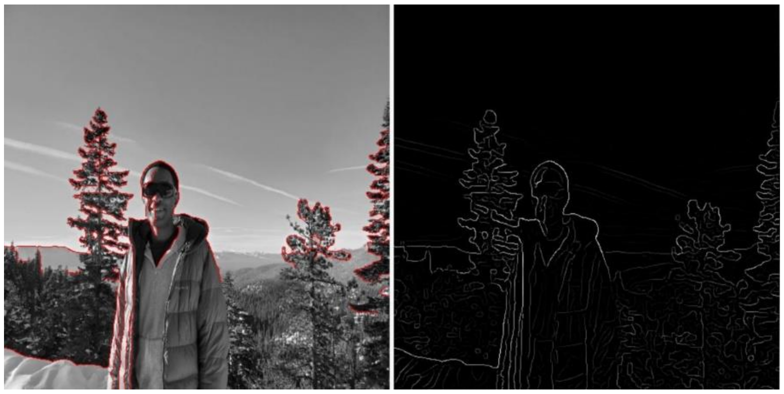
If we multiply out fourier coefficient with gaussian, means our fourier coefficient are going to be lower frequency information.

Part 3: Edge Detection

Theory

Goal: To lay the foundation for detecting and using the edges in image content.

Edges in images are boundaries between regions with different intensities.



There are many reasons to want to locate the edges.

- delineate organ boundaries

- align images.

Definition:

An edge is indicated by a rapid change in intensity.

Edge can be calculated by radiant vector with a large magnitude.

If we compute the gradient, we can get the edges.

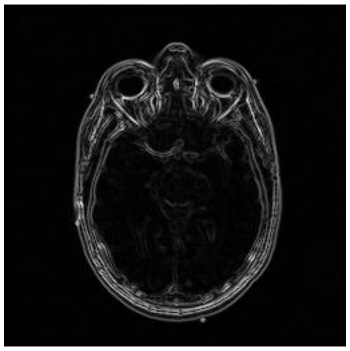
Image Gradient:

MATLAB code (from lecture note):

```
f = imread('t1.jpg');
f = double( f(:,:,1) );
imshow(f,[])
dfdr = ( circshift(f,[-1 0]) - circshift(f,[1 0]) ) / 2;
imshow(dfdr,[])
dfdc = ( circshift(f,[0 -1]) - circshift(f,[0 1]) ) / 2;
imshow(dfdc,[])
grad_mag = sqrt(dfdr.^2 + dfdc.^2);
imshow(grad_mag, [])
```

[Note: the 'circshift' is basically circular shift function. It applies shift in a circular way. In 180-degree shift.]

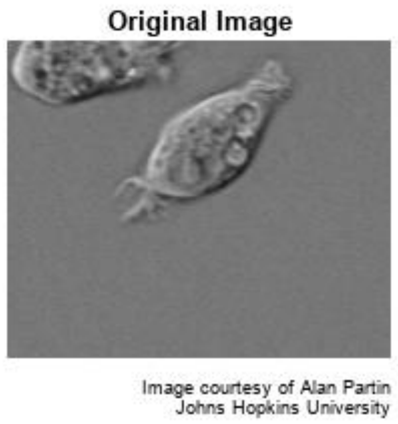
Result:



MATLAB Code (Internet):

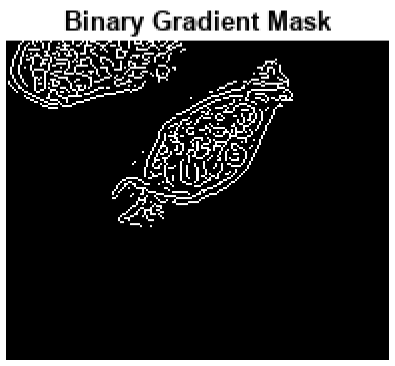
Step-1

```
I = imread('cell.tif');
imshow(I)
title('Original Image');
text(size(I,2),size(I,1)+15, ...
    'Image courtesy of Alan Partin', ...
    'FontSize',7,'HorizontalAlignment','right');
text(size(I,2),size(I,1)+25, ....
    'Johns Hopkins University', ...
    'FontSize',7,'HorizontalAlignment','right');
```



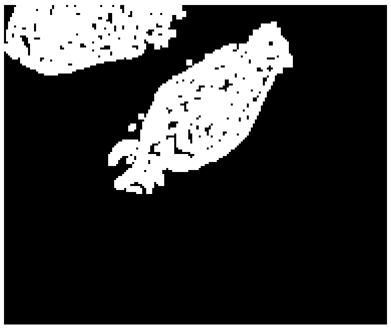
Step-2

```
[~,threshold] = edge(I,'sobel');
fudgeFactor = 0.5;
BWs = edge(I,'sobel',threshold * fudgeFactor);
imshow(BWs)
title('Binary Gradient Mask')
```



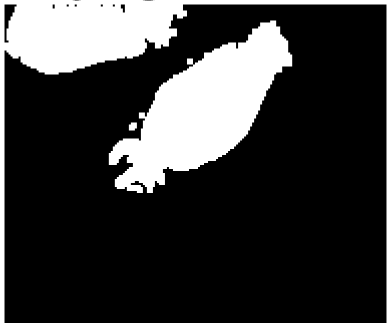
Step-3

```
se90 = strel('line',3,90);
se0 = strel('line',3,0);
BWsdil = imdilate(BWs,[se90 se0]);
imshow(BWsdil)
title('Dilated Gradient Mask')
```



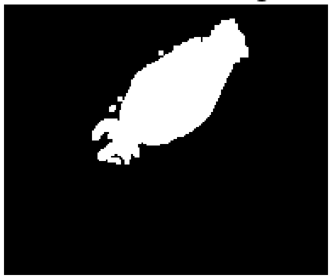
Spet-4

```
BWdfill = imfill(BWsdil,'holes');
imshow(BWdfill)
title('Binary Image with Filled Holes')
```



Step-5

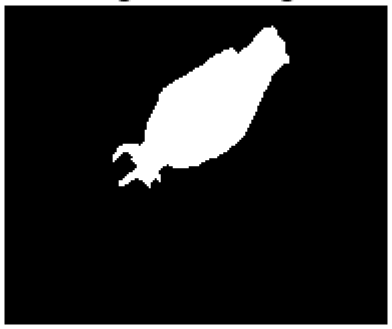
```
BWnobord = imclearborder(BWdfill,4);
imshow(BWnobord)
title('Cleared Border Image')
```



Step-6

```
seD = strel('diamond',1);
BWfinal = imerode(BWnobord,seD);
BWfinal = imerode(BWfinal,seD);
imshow(BWfinal)
title('Segmented Image');
```

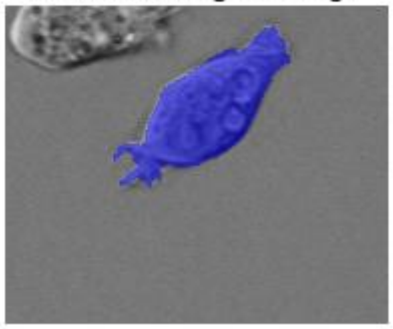
Segmented Image



Step-7

```
imshow(labeloverlay(I,BWfinal))
title('Mask Over Original Image')

BWoutline = bwperim(BWfinal);
Segout = I;
Segout(BWoutline) = 255;
imshow(Segout)
title('Outlined Original Image')
```

Mask Over Original Image



Analysis and discussion:

Step-1: Read Image

Read in the cell.tif image, which is an image of a prostate cancer cell. Two cells are present in this image, but only one cell can be seen in its entirety. The goal is to detect, or segment, the cell that is completely visible.

Step-2: Detect Entire Cell

The object to be segmented differs greatly in contrast from the background image. Changes in contrast can be detected by operators that calculate the gradient of an image. To create a binary mask containing the segmented cell, calculate the gradient image and apply a threshold.

Use edge and the Sobel operator to calculate the threshold value. Tune the threshold value and use edge again to obtain a binary mask that contains the segmented cell.

Step 3: Dilate the Image

The binary gradient mask shows lines of high contrast in the image. These lines do not quite delineate the outline of the object of interest. Compared to the original image, there are gaps in the lines surrounding the object in the gradient mask. These linear gaps will disappear if the Sobel image is dilated using linear structuring elements. Create two perpendicular linear structuring elements by using strel function.

Step 4: Fill Interior Gaps

The dilated gradient mask shows the outline of the cell quite nicely, but there are still holes in the interior of the cell. To fill these holes, use the imfill function

Step 5: Remove Connected Objects On Border

The cell of interest has been successfully segmented, but it is not the only object that has been found. Any objects that are connected to the border of the image can be removed using the imclearborder function. To remove diagonal connections, set the connectivity in the imclearborder function to 4.

Step 6: Smooth the Object

Finally, in order to make the segmented object look natural, smooth the object by eroding the image twice with a diamond structuring element. Create the diamond structuring element using the strel function.

Step 7: Visualize the Segmentation

You can use the labeloverlay function to display the mask over the original image.

Part 4: Image Registration

Theory

Image registration is an image processing technique used to align multiple scenes into a single integrated image. It helps overcome issues such as image rotation, scale, and skew that are common when overlaying images.

Image registration is often used in medical imagery to align images from different camera sources. Digital cameras use image registration to align and connect adjacent images into a single panoramic image.

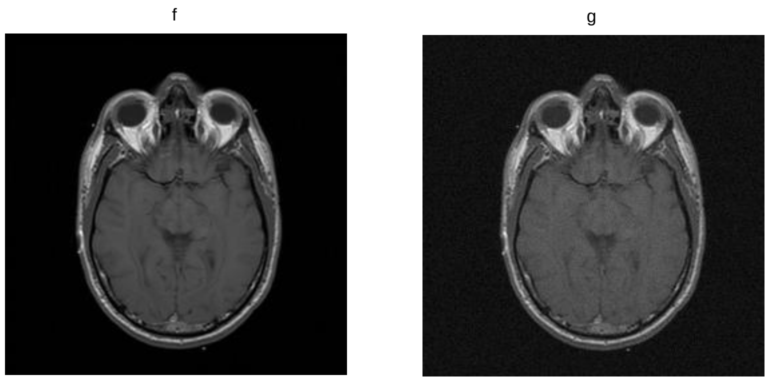
Sum of Absolute Differences (SAD) is the popular method for Image Registration.

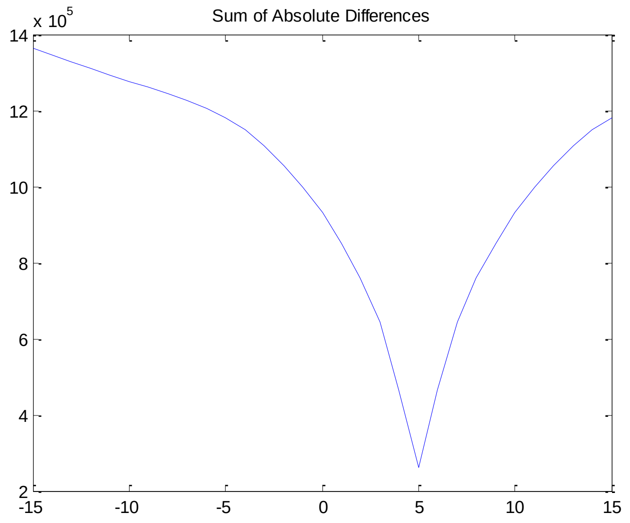
MATLAB Code

```
% Registration demo

warning off;

f = imread('t1.jpg');
f = double( f(:,:,1) );

% Choose an offset
offset = [0 0 0 0 5 0];
disp(['True params needed: (theta,r,c) = (' num2str(offset(3)) ','
num2str(offset(4)) ',' num2str(offset(5)) ')'])

% Shift f and add noise to get g
M = p2m(offset);
g = MyAffine(f, M, 'cubic', 'centred') + randn(size(f))*5;

figure(1); imshow(f,[]); title('f');
figure(2); imshow(g,[]); title('g');

%% Now try a bunch of different shifts on f

p = -15:1:15;
cost = zeros(size(p)); % to record the costs

for idx = 1:length(p)
    % Apply shift to f
    fs = MyAffine(f, p2m([0 0 0 0 p(idx) 0]), 'linear');

    % Compute cost function
    cost(idx) = sum( abs(fs(:)-g(:)) );
```

```
end

figure(3);
plot(p, cost); title('Sum of Absolute Differences');
```

Output:

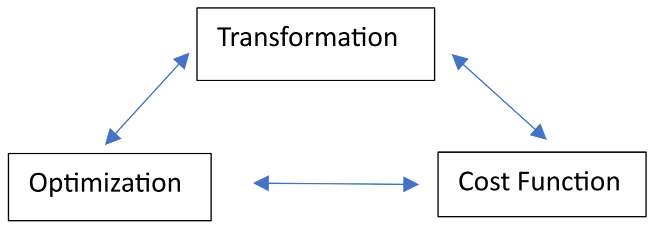




Analysis and discussion:

We have two images that we wish to align so that they correspond on a pixel-by-pixel basis. The images are f and g.

Registration Process: Each registration process involves an interplay between three main components.



The output of image registration in MATLAB is a registered image volume that represents the aligned images. This registered output image volume can be viewed and observed directly to assess the quality of registration. It provides a visual representation of the aligned images, allowing researchers and clinicians to analyze and interpret the results.

The SAD curve provides a quantitative measure of the similarity between the fixed and moving images as a function of the transformation parameters. The goal is to find the transformation parameters that minimize the SAD curve, indicating the best alignment between the images.

Part 5: Image Segmentation

Theory:

Segmentation is the process of classifying pixels into groups that correspond to the same tissue type. A good example is classifying pixels as "liver" or "non-liver" in this T1-weighted MRI. Or "lung" vs. "nonlung" pixels.

Segmentation has many uses:

- measure volume of an organ
- render a 3D view of an organ
- surface-based registration

MATLAB Code

```
f = imread('thorax_t1.jpg');
f = double(f(:,:,1));

figure(1);
imshow(f,[]);
```

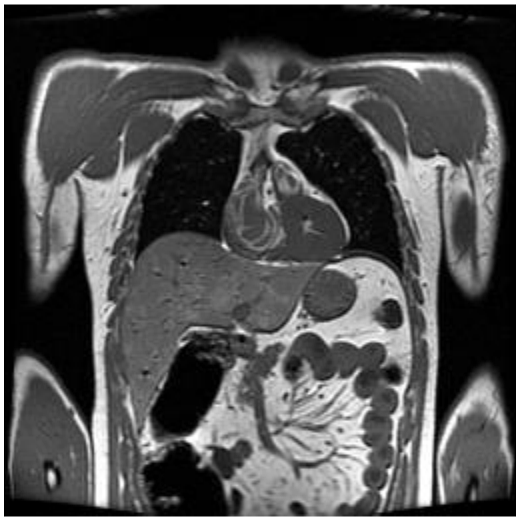
```
%% Create a mask to store the
selected pixels
mask = zeros(size(f));
```

```
%% Choose some params and seed
points
meth = 1;

switch meth
    case 1 % lungs
        low = 0;
        high = 50; % try 50 and 70
        re_thresh = 0; % recompute
thresholds?
        mask(96,88)  = 1; %
patient's right lung
        mask(77,177)  = 1; %
patient's left lung
    case 2 % liver
        low = 70;  % try 70 and 80
        high = 110;
        re_thresh = 0; % recompute
thresholds?
        mask(141,108)  = 1; % liver
    case 3 % user input
        s = ginput;
        c = round(s(:,1)); r =
round(s(:,2));
        val = zeros(n,1);
        for n = 1:length(r)
            val(n)  = f(r(n),c(n));
            mask(r(n),c(n))  = 1;
        end
        low = min(val)  - (max(val)-
min(val));
        high = max(val)  +
(max(val)-min(val));
        re_thresh = 1; % recompute
thresholds?
end

%% Flood-fill

% Close off image boundaries
because the floodfill can
% only handle interior points.
f(1,:)   = high+1;
f(end,:) = high+1;
f(:,1)   = high+1;
f(:,end) = high+1;

keep_going = 1;

temp_low = low;
temp_high = high;

while keep_going

    [rows cols] = find(mask==1);

    keep_going = 0;

    for n = 1:length(rows)

        r = rows(n);
        c = cols(n);

        % Up
        if f(r-1,c)>temp_low &&
f(r-1,c)<temp_high && mask(r-
1,c)~=1
            mask(r-1,c)  = 1;
            keep_going = 1;
        end

        % Down
        if f(r+1,c)>temp_low &&
f(r+1,c)<temp_high &&
mask(r+1,c)~=1
            mask(r+1,c)  = 1;
            keep_going = 1;
        end

        % Left
        if f(r,c-1)>temp_low &&
f(r,c-1)<temp_high && mask(r,c-
1)~=1
            mask(r,c-1)  = 1;
            keep_going = 1;
        end

        % Right
        if f(r,c+1)>temp_low &&
f(r,c+1)<temp_high &&
mask(r,c+1)~=1
            mask(r,c+1)  = 1;
            keep_going = 1;
        end


    end % end of for-loop

    if re_thresh
        seg = find(mask==1);
        new_low = mean(f(seg))  -
2*std(f(seg));
        new_high = mean(f(seg))  +
2*std(f(seg));

        temp_low = new_low;
        temp_high = new_high;

        %disp(['Low: '
num2str(temp_low) ',  High: '
num2str(temp_high)]);
```

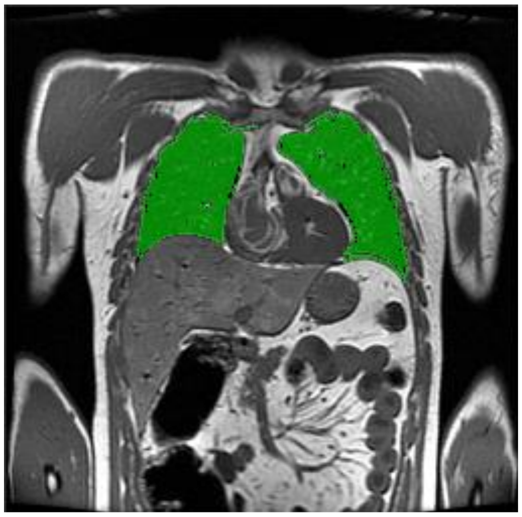
```
    end

    % Visualize as we go
```

```
        %figure(1);
        Overlay(f, mask);

end % end of while-loop
```

Output:



Original Image



Segmented Image

Analysis and discussion:

We used MATLAB code to demonstrate Image Segmentation that was provided. The difference between original image and the segmented image is clear.

Segment the bone in this CT scan. Bone is usually the brightest thing in a CT image.

The primary output of image segmentation in MATLAB is the segmented image itself. This is a result of the segmentation algorithm applied to the input image, where different regions or objects of interest are assigned unique labels. The segmented image highlights the boundaries or areas of the image that correspond to the segmented objects.