ISLAMIC UNIVERSITY OF TECHNOLOGY (IUT)

ORGANISATION OF ISLAMIC COOPERATION (OIC)
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 4872: Biomedical Signal Processing

Lab – 04: Biomedical Image Processing

Biomedical images can be obtained through various imaging modalities such as X-ray, MRI (Magnetic Resonance Imaging), CT (Computed Tomography), ultrasound, PET (Positron Emission Tomography), and microscopy. The images collected from these modalities are sometimes passed through some preprocessing technique before they are utilized for predictive modeling or analysis. Sometimes the collected images may not be up to the quality, some measurement noise can be there, and not enough illumination may result in poor visualization. In addition, in some cases, we might want to highlight particular regions such as tumors in the image to assist clinicians in diagnosis.

Biomedical image processing includes different set of tasks such as –

- Image Enhancement: Improving the quality of acquired images through techniques like noise reduction, contrast enhancement, and sharpening.
- Image Restoration: Removing noise, artifacts, or other imperfections from images to restore them to a more accurate representation of the underlying biological structures.
- Image Segmentation: Identifying regions of interest within images, such as organs, tissues, tumors, or cells. This can involve techniques like thresholding, clustering, or edge detection.
- Feature Extraction: Extracting quantitative or qualitative features from segmented regions to aid in diagnosis.
- Image Registration: Aligning images from different modalities, time points, or subjects to enable comparison or fusion of information.

Prepared by
Asif Newaz
Lecturer, EEE, IUT

Biomedical image processing plays a crucial role in modern healthcare, enabling clinicians and researchers to visualize and analyze internal structures and processes, leading to improved diagnostics, treatments, and patient outcomes. These techniques vary with the imaging modality utilized, the purpose, and the application. In this lab, we are going to discuss and demonstrate some common image-processing techniques.

4.1 Image Enhancement

Image enhancement in biomedical image processing refers to the process of improving the quality of acquired images to make them more suitable for analysis, interpretation, and visualization. This enhancement can involve various techniques aimed at reducing noise, improving contrast, sharpening edges, and highlighting important features within the image.

Contrast Adjustment

Contrast adjustment remaps image intensity values to the full display range of the data type. An image with good contrast has sharp differences between black and white.

To understand it, take a look at the image in Fig. 1. The image on the left side has poor contrast and is barely visible. Compared to that, the image on the right is clearer.

Fig. 1

Prepared by
Asif Newaz
Lecturer, EEE, IUT

The reason becomes apparent if you look at their histogram (Fig. 2 and 3). For the image on the left side of Fig. 1, the associated histogram is provided in Fig. 2. You can see that the most of the pixel values are in the range [75, 160]. One way to enhance the contrast of the image is to use the entire grayscale range [0,255]. This is what has been done for the image on the right side of Fig. 1 and its associated histogram is provided in Fig. 3.

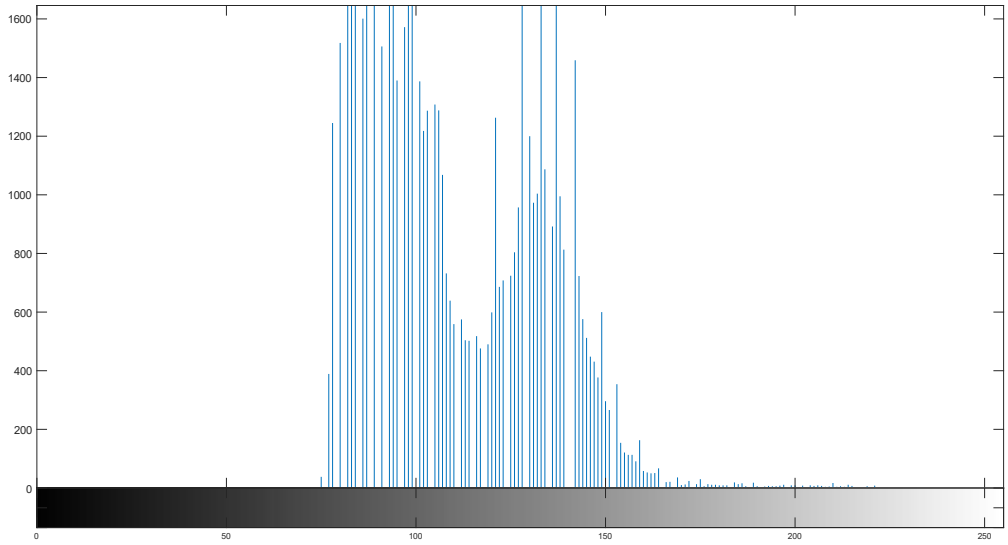


Fig. 2

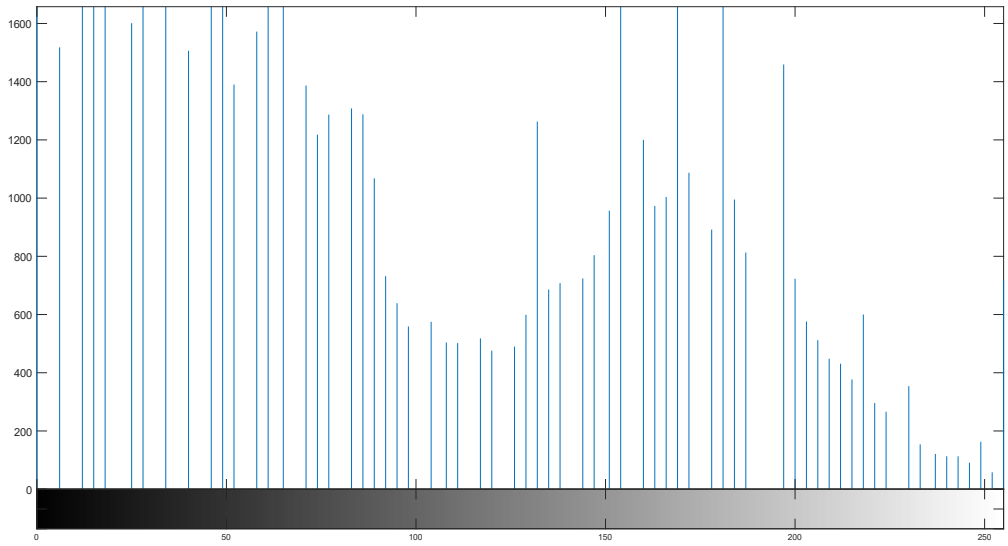


Fig. 3

Prepared by
Asif Newaz
Lecturer, EEE, IUT

This is a quite simple but effective image adjustment technique and can be done very easily in MATLAB using the 'imadjust' function. You can see the histogram of the image using the 'imhist' function. Use the 'imtool' function to open the image in the 'Image viewer app' and investigate different regions. Use the 'montage' function to compare the difference between multiple images side-by-side. We usually perform such operations on the grayscale image. You can convert an RGB image to grayscale using the 'rgb2gray' function.

```
img= imread("Tr-gl_0011.jpg");
img_gray= rgb2gray(img);
imshow(img_gray)

imtool(img_gray)

img_adj= imadjust(img_gray);
montage({img_gray, img_adj})

imhist(img_gray)
imhist(img_adj)
```

See the results in Fig. 4. On the right side, you have the enhanced version of the MRI image that provides better visualization.

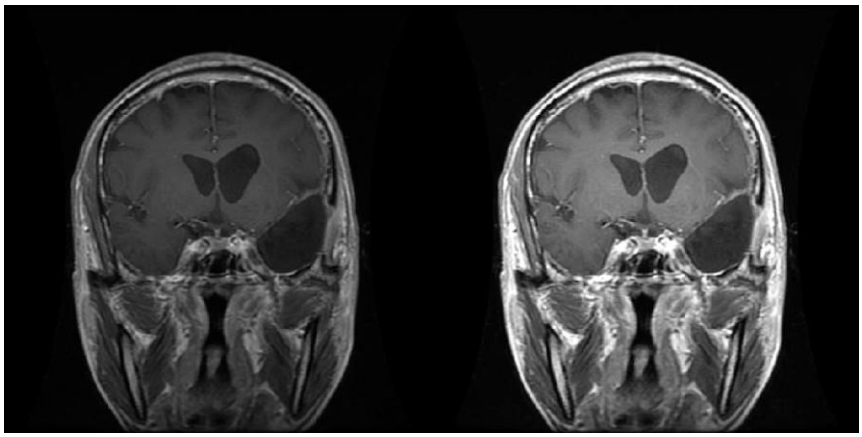


Fig. 4

Prepared by
Asif Newaz
Lecturer, EEE, IUT

Gamma Correction

This is another image enhancement/contrast adjustment technique. “imadjust” function maps low to bottom, and high to top. The values between low and high are mapped linearly to values between bottom and top. Gamma correction provides a way to insert non-linearity in mapping.

Gamma can be any value between 0 and infinity. If gamma is 1 (the default), the mapping is linear. If gamma is less than 1, the mapping is weighted toward higher (brighter) output values. If gamma is greater than 1, the mapping is weighted toward lower (darker) output values.

Use the following code to apply gamma correction to your image. Use a numeric slider to test the effect of different values of gamma.

```
img_gamma= imadjust(img_gray, [], [], gamma);
montage({img_gray, img_gamma})
```

Fig. 5 shows the change in the image for a gamma value of 0.6.

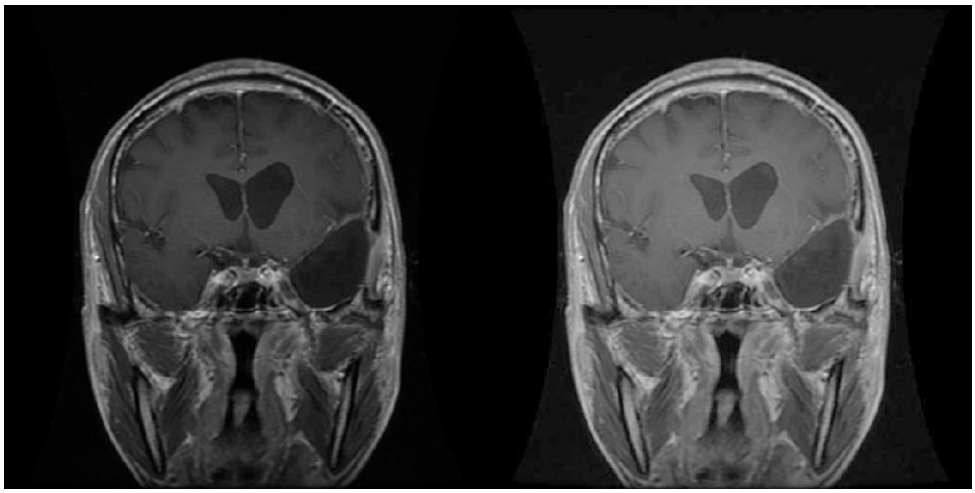


Fig. 5

Prepared by
Asif Newaz
Lecturer, EEE, IUT

Histogram Equalization

The basic idea behind histogram equalization is to spread out the intensity values over the entire dynamic range of the image histogram, thereby maximizing the usage of available intensity levels. This results in a more uniform distribution of intensities, leading to enhanced contrast and improved visual appearance of the image. Histogram equalization is widely used in biomedical image processing to improve the visual quality of medical images. However, it's important to note that histogram equalization may not always be suitable for all types of images.

```
img_hist=histeq(img_gray);
img_hist_ada= adapthisteq(img_gray);
montage({img_gray,  img_adj, img_hist, img_hist_ada,}, 'Size', [1,4])
```

You have two functions in MATLAB to perform this operation – 'histeq' and 'adaphisteq'. Check the histogram of the resultant images to see the difference in their operation.

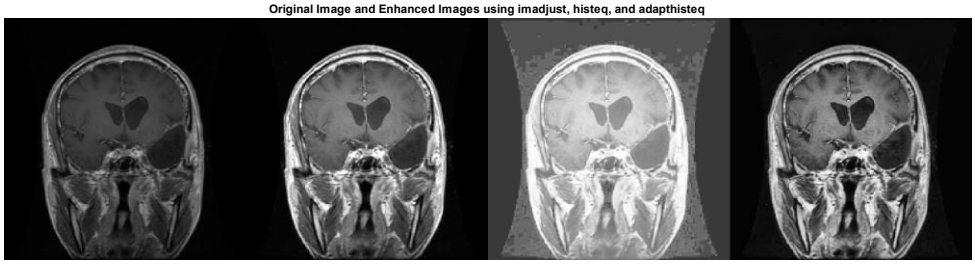


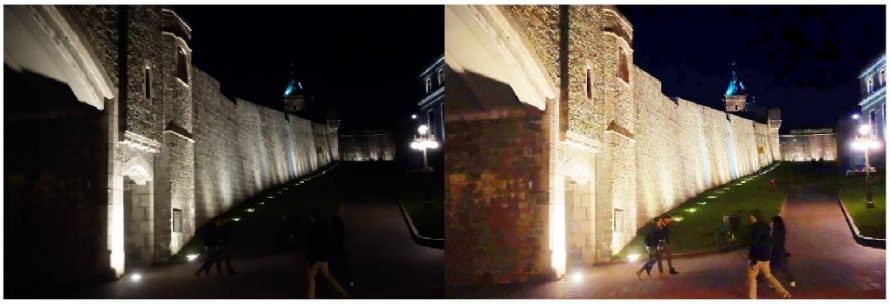
Fig. 6

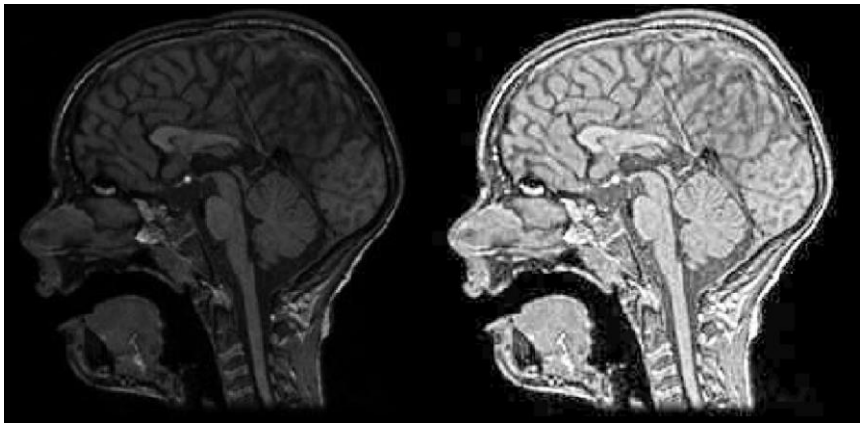
Prepared by
Asif Newaz
Lecturer, EEE, IUT

Brightening

The 'imlocalbrighten' function in MATLAB brightens the low-light areas in an image. You can specify the amount with the 2nd input argument of the function. You can see the result from the following figures.

```
A = imread('lowlight_2.jpg');
B = imlocalbrighten(A, 0.8);
montage({A,B})
```





Prepared by
Asif Newaz
Lecturer, EEE, IUT

Other functions you may look at –

i. imsharpen
ii. imreducehaze

4.2 Adding noise

You can add noise to the image to simulate different real-world scenarios. Gaussian, Poisson, and salt-and-pepper noise are some common types of noise that can corrupt digital images.

Gaussian noise:

This is characterized by random variations in pixel intensity values that resemble the bell-shaped curve of a Gaussian distribution. It typically occurs due to electronic and sensor noise during image acquisition or transmission. It appears as a smooth, continuous variation in pixel intensity across the image. Gaussian noise can be effectively reduced using linear filters such as Gaussian smoothing or averaging filters.

Use the following code to add Gaussian noise to your image. The 'imnoise' function has other input arguments for other types of noise.

```
ign= imnoise(ig,"gaussian");
montage({ig,ign})

igp= imnoise(ig,"poisson")

igs= imnoise(ig,"salt & pepper", 0.005)
```

Poisson Noise:

This follows a Poisson distribution. It occurs in images where the number of photons detected by the sensor varies randomly, such as in low-light conditions or in imaging techniques like fluorescence microscopy. It can be challenging to remove Poisson noise without compromising image details, as it is inherently associated with the photon statistics of the imaging process.

Salt and Pepper Noise:

This is a type of impulse noise that manifests as randomly occurring bright (salt) and dark (pepper) pixels in an image. It typically occurs due to defects in the image sensor, transmission errors, or

corruption during storage. Salt-and-pepper noise appears as isolated bright or dark pixels scattered throughout the image, resembling grains of salt and pepper. It can severely degrade image quality and hinder subsequent image-processing tasks. Median filtering is commonly used to effectively remove salt-and-pepper noise while preserving image details.

Fig. 9 shows the effect of different noises in the chest x-ray image.

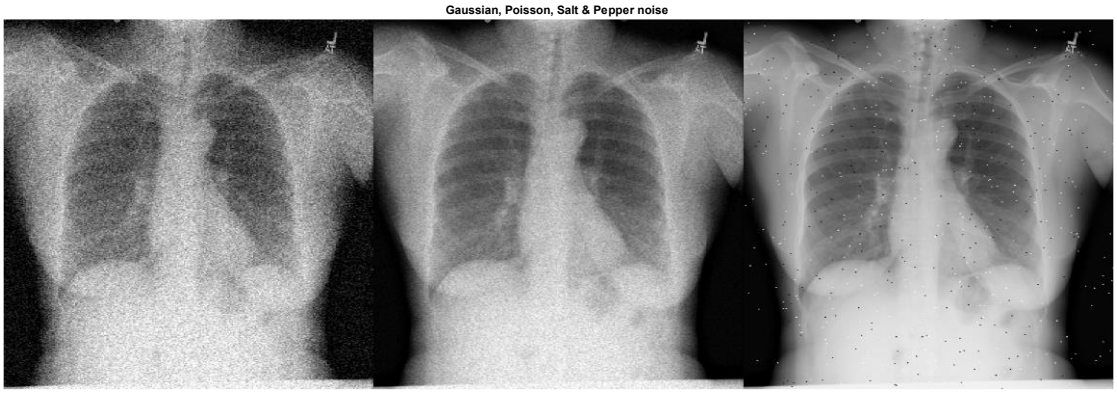


Fig. 9

4.3 Image Filtering

Image filtering is a fundamental operation in image processing that involves modifying the intensity values of pixels in an image based on their surrounding neighbors. Filtering is commonly used for various purposes such as noise reduction, edge detection, smoothing, sharpening, and feature enhancement. There are different types of image filtering techniques, including linear filters, non-linear filters, and frequency domain filters.

Linear Filters:

Linear filters operate by computing a weighted sum of pixel values in a neighborhood around each pixel and replacing the central pixel's value with the result. These filters are characterized by their linear operation, meaning that the output pixel value is a linear combination of the input pixel

values. Examples include the averaging filter, Sobel filter, Laplacian filter, etc. Different filters serve different purposes, for instance, the Sobel filter is used in edge detection, while low pass filters are used in blurring the image.

Non-linear Filters:

Non-linear filters differ from linear filters in that they apply non-linear operations to the pixel values in the neighborhood. These filters are often used for tasks like noise removal, where simple averaging or weighted summing may not be effective. Examples include the median filter.

Frequency Domain Filters:

Frequency domain filters operate on the Fourier-transformed representation of the image, allowing for operations in the frequency domain, such as filtering specific frequency components. For example, low-pass filter, high-pass filter, notch filter, etc.

In this section, we will look at spatial filtering, where a weight mask is used to express the effect of the filter on each pixel of the image.

Low-pass filter

A low-pass filter attenuates high-frequency components (edges, texture, sharp details) of an image. They are used in smoothing, blurring, and noise reduction. The disadvantage of such filters is that they remove/attenuate edges or sharp details from an image which might be important in many applications.

Two typical weight masks for a low-pass filter are given below –

	1	1	1
(1/9) *	1	1	1
	1	1	1

	1	2	1
(1/16) *	2	4	2
	1	2	1

The mask on the left will produce a smoother version of the image by averaging each pixel based on its surrounding pixels. The larger the mask, the more attenuation will be there. The mask on the right, however, will partially reduce the blurring effect as it assigns a higher value to the center pixel.

Prepared by
Asif Newaz
Lecturer, EEE, IUT

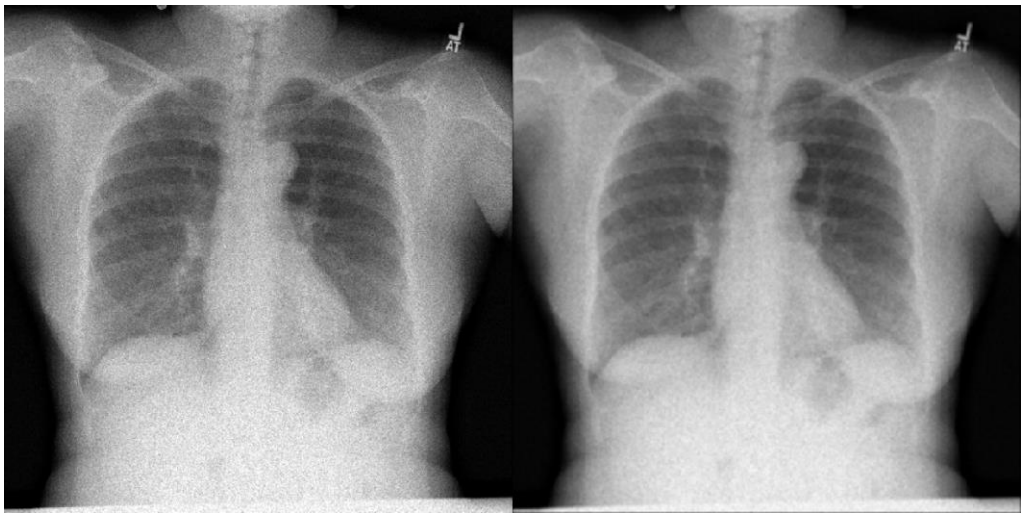


Fig. 10

You can observe the effect of using LPF on a noisy (Poisson noise) chest x-ray image in Fig. 10. As you can see, the LPF smoothens the image, reducing the noise. A kernel of size 3 was utilized here. Try other-sized kernels and see the effect using MATLAB.

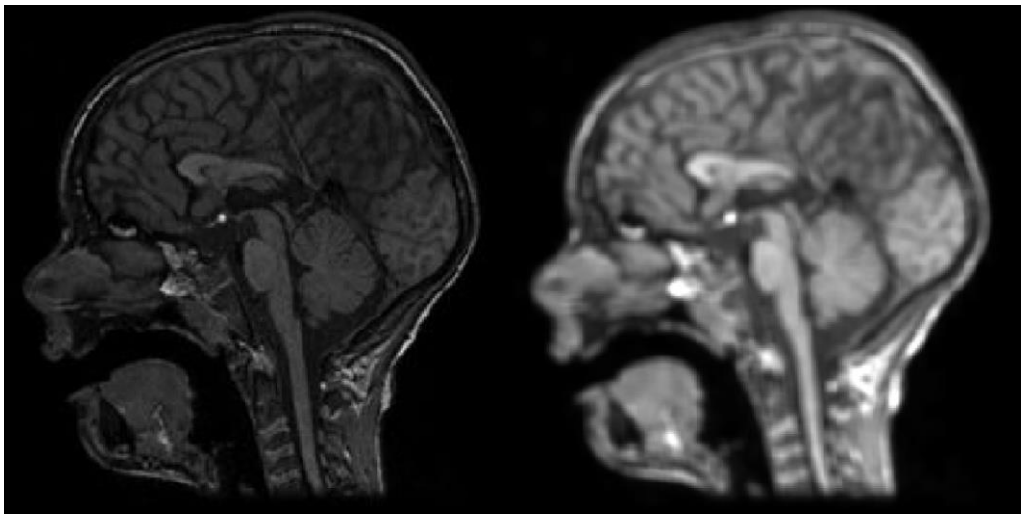


Fig. 11

You can observe the blurring effect of LPF in Fig. 11. A kernel of size 5 was utilized here.

Prepared by
Asif Newaz
Lecturer, EEE, IUT

Use the following code (‘imfilter’ function) to create an LPF in MATLAB.

```
mri= imread('Tr-no_0044.jpg');
h=(1/9)*ones(6);

ig_lpf= imfilter (mri, h);
montage({mri, ig_lpf})
```

Median filters

Median filters reduce noise without eliminating the edges and other high-frequency components. They are particularly useful in removing isolated spikes/salt and pepper noise (Fig. 12). Here, the center pixel value is replaced by the median of all surrounding pixels. Use the ‘medfilt2’ function to apply median filters to your image.

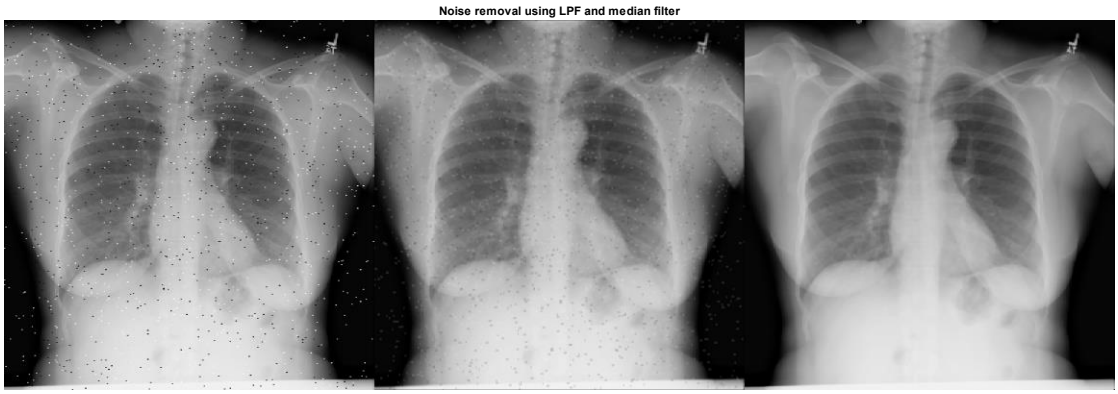


Fig. 12

High-pass filter

This type of filter preserves the high-frequency components such as edges and textures and removes other information from the image. The disadvantage of this type of filter is that it eliminates all the low-frequency components. The weight mask for HPF is given below –

Prepared by
Asif Newaz
Lecturer, EEE, IUT

	-1	-1	-1
(1/9) *	-1	8	-1
	-1	-1	-1

```
hpf= -(1/9)*ones(3);
hpf(2,2)= 8;
ig_hpf= imfilter(ig, hpf);
```

Other filters include a High-Boost Filter (which is a modification to the high-pass filter with an added all-pass filter), a Wiener filter ('wiener2' function), etc. Fig. 13 shows the effect of different filters on an image.

Fig. 13

Prepared by
Asif Newaz
Lecturer, EEE, IUT

4.4 Edge Detection

Edge detection is an important technique in biomedical image processing used to identify boundaries and edges within images, which can be essential for various analyses such as segmentation, feature extraction, and object recognition. There are different edge detection algorithms which include the Sobel filter, Laplacian of Gaussian (LoG), Canny, Prewitt, etc. MATLAB's 'edge' function provides the tool to perform such operations on an image. Edge detection approaches are usually followed by Morphological Operations such as Erosion, dilation, opening, and closing to refine edge maps and remove small artifacts.

Use the following code to check the output from different edge detection techniques. Fig. 14 demonstrates the results.

```
cell= imread('cell.png');
cell= rgb2gray(cell);

e2 = edge(cell, "sobel");
e22 = edge(cell, "log");
e222 = edge(cell, "canny");
```

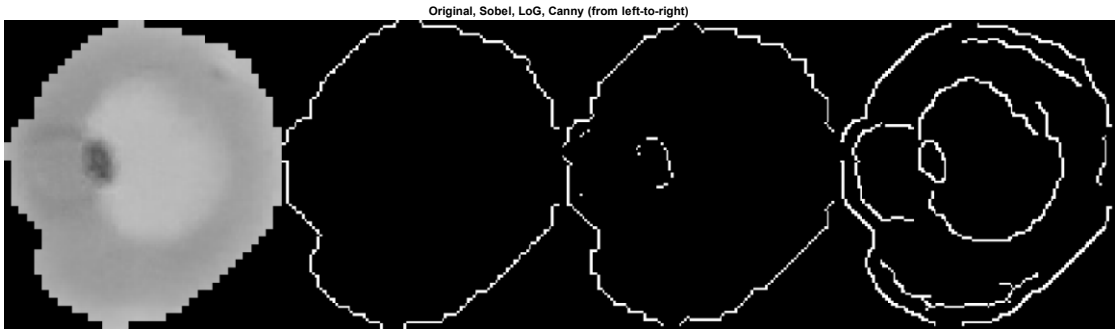


Fig. 14

Prepared by
Asif Newaz
Lecturer, EEE, IUT

You can see that only the LoG approach is able to detect the ROI, although not perfectly. To obtain better results, you can also extract and see the threshold value used by the algorithms. Then you can change the threshold to get better results. Fig. 15 shows the effect. The image in the middle is the default result from the LoG approach (threshold value = 0.0052). The threshold is reduced to 0.003 and the result is much better (image on the right).

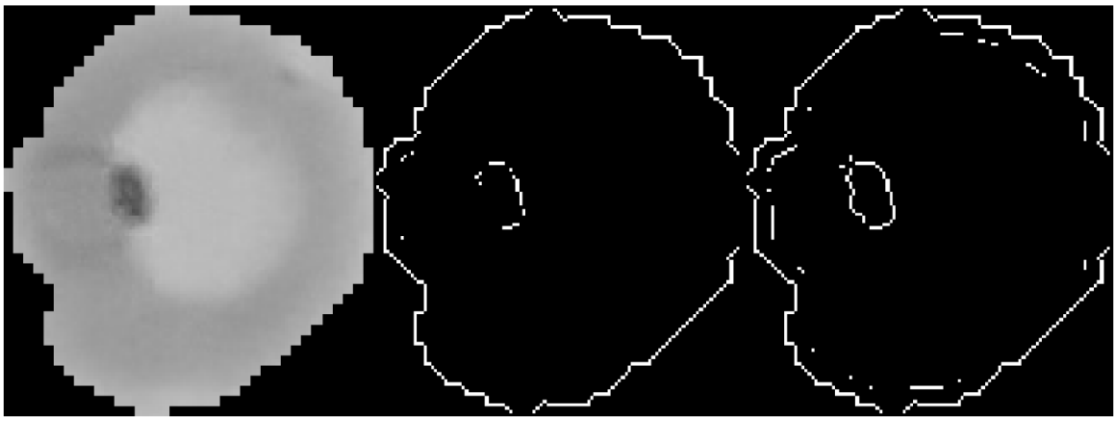


Fig. 15

You can see some small artifacts present in the resultant image which can be removed through other post-processing techniques such as morphological operations. You can do that using MATLAB's image segmenter app.

Code Files

All necessary files are available in this repository:

https://github.com/newaz-aa/Biomedical_Image_Processing

Prepared by
Asif Newaz
Lecturer, EEE, IUT