B-TECH PROJECT REPORT ON X-RAY IMAGES BONES SEGMENTATION AND FRACTURE LOCATION

Under the Supervision

Of

Dr. Rachna Asthana, Professor, HOD,

Electronics Engineering Department,

HBTU, Kanpur

Submitted by:

Shivam Gupta (209/14)
Mayank Agrawal (195/14)
Ravi Mittal (203/14)
Mohd. Yasar Khalid Ansari(544/15)

CERTIFICATE

Certified that this is a progress record of the project entitled

"X-RAY IMAGES BONES SEGMENTATION AND FRACTURE LOCATION"

Done by "**Shivam Gupta, Mayank Agrawal, Ravi Mittal, Mohd. Yasar Khalid Ansari** "of the VIII semester, Electronics Engineering in the year 2018 in partial fulfillment of the requirements to the award of Degree of Bachelor of Technology in Electronics Engineering of Harcourt Butler Technical University.

(Dr. RACHNA ASTHANA)
Professor
Electronics Engg. Department
HOD,HBTU, Kanpur
Project Supervisor

DECLARATION

We, **Shivam Gupta, Mayank Agrawal, Ravi Mittal, Mohd. Yasar Khalid Ansari** studying in the final semester(8th) of Bachelor of Technology in Electronics Engineering at Harcourt Butler Technical University, Kanpur, hereby declare that this project work entitled **"X-RAY IMAGES BONES SEGMENTATION AND FRACTURE LOCATION"** which is being submitted by us in the partial fulfillment for the award of the degree of Bachelor of Technology in Electronics Engineering at Harcourt Butler Technical University, Kanpur is an authentic record of our carried out during the academic year 2017-2018, under the guidance of **Dr. Rachna Asthana,** Professor, Department of Electronics Engineering at Harcourt Butler Technical University, Kanpur.

We further undertake that the matter embodied in the dissertation has not been submitted previously for the award of any degree or diploma by us to any other university or institution.

Place: Kanpur

Shivam Gupta (209/14)

Mayank Agrawal (195/14)

Ravi Mittal (203/14)

Mohd. Yasar Khalid Ansari (544/15)

ACKNOWLEDGEMENT

We are thankful to our Project Guide **Dr. Rachna Asthana (Professor, ET Dept.)** Harcourt Butler Technical University, Kanpur, for his valuable guidance, encouragement and co-operation during the course of this project and its presentation. It was his guidance and support, which resulted in the successful presentation of project within the specified time. Their unflinching help and encouragement was a constant source of inspiration to us.

We are very graceful to **Dr. Rachna Asthana (Professor, Head of Department, ET Dept.)** and **Dr. Ashutosh Singh (Associate Professor,ET Dept.)** For giving opportunity to us to present this project. They took personal interest in project so that we could utilize our potential.

A Project owes its success from commencement to completion, to people involved with seminar at various stages. We avail this opportunity to convey our sincere thanks to all the individuals who have helped and assisted us in carrying and bringing out this seminar. Last but not the least, the co-operation and help received from teachers and friends Dept. of ET is gratefully acknowledged.

SHIVAM GUPTA (209/14)

MAYANK AGRAWAL (195/14)

RAVI MITTAL (203/14)

MOHD. YASAR KHALID ANSARI (544/15)

(Final B.Tech Electronics Eng.)

ABSTRACT

Image segmentation operation has a great importance in most medical imaging applications, by extracting anatomical structures from medical images. There are many image segmentation techniques available in the literature, each of them having advantages and disadvantages. The extraction of bone contours from X-ray images has received a considerable amount of attention in the literature recently, because they represent a vital step in the computer analysis of this kind of images. The aim of X-ray segmentation is to subdivide the image in various portions, so that it can help doctors during the study of the bone structure, for the detection of fractures in bones, or for planning the treatment before surgery. The goal of this paper is to review the most important image segmentation methods starting from a data base composed by real X-ray images. We will discuss the principle and the mathematical model for each method, highlighting the strengths and weaknesses. In this project We have tested on many images. We have applied different Thresholding and morphological algorithms for the segmentation of the bones. We have used the Hough Transform for the detection of the joints and then checking for the break-site is done for the location of the fracture.

TABLE OF CONTENTS

1. INTRODUCTION

Image segmentation operation has a great importance in most medical imaging applications, by extracting anatomical structures from medical images [2]. There are many image segmentation techniques available in the literature, each of them having advantages and disadvantages. The extraction of bone contours from X-ray images has received a considerable amount of attention in the literature recently, because they represent a vital step in the computer analysis of this kind of images. The aim of X-ray segmentation is to subdivide the image in various portions, so that it can help doctors during the study of the bone structure, for the detection of fractures in bones, or for planning the treatment before surgery.

When analyzing objects in images, it is necessary to distinguish the objects of interest from the background. This task can be realized through segmentation. Medical imaging began with the discovery of Roentgen rays (X-Rays). Then, various image modalities have appeared over the years, each with their own advantages and disadvantages. These are: Magnetic Resonance Imaging (MRI), Ultrasound (US), Computed Tomography (CT), Nuclear Imaging, including Single Photon Emission Computed Tomography (SPECT) and Position Emission Tomography (PET). Figure 1 shows the Block diagram of bone fracture system in X-Ray images.
Among the applications of segmentation in medical imaging we mention the anatomical localization, whose main purpose is to describe anatomic regions of interest.

They may be also affected by noise, sampling artifacts or spatial aliasing so that the boundaries of the regions of interest to become indistinct or disconnected. X-rays may have
various orientations, resolutions, or luminous intensities, depending on the X-ray equipment, that could influence the quality of the segmentation result.

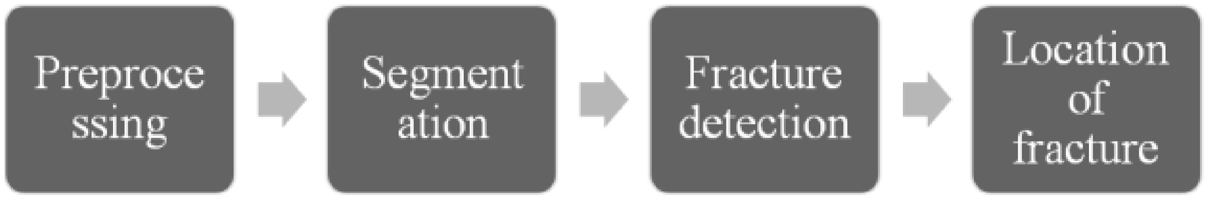


Fig 1: Block diagram of bone fracture system

2. MEDICAL IMAGE SEGMENTATION TECHNIQUES

General medical image segmentation methods can be categorized into the following classes: classical image segmentation methods (thresholding, regions-based, and edges-based), pattern recognition-based, deformable models, wavelets-based methods, and atlas-based Techniques. To exemplify some of the segmentation technique, we will consider a real X-ray image shown in Figure 2.

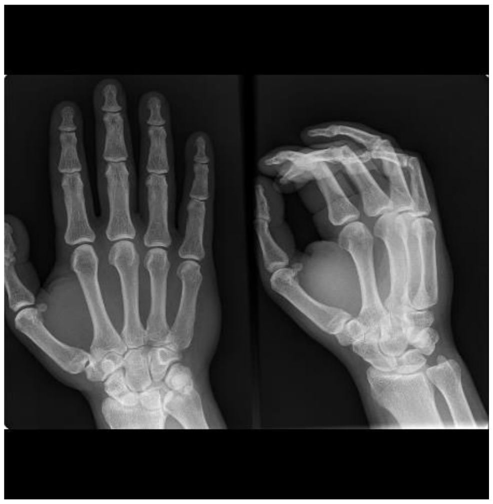


Fig. 2 An X-ray image test.

3. Classical image segmentation methods

Classical methods include the following segmentation techniques: **thresholding, region-based, and edge-based methods.**

3.1. Thresholding is one of the most simple segmentation techniques and involves thresholding the image intensity. There are two classes of thresholding methods: global methods and adaptive methods. In the case of global thresholding, only one threshold is selected for the entire image, while in the case of adaptive thresholding, the local thresholds are selected independently for each pixel (groups of pixels).The figure 3 shows an example of thresholding.

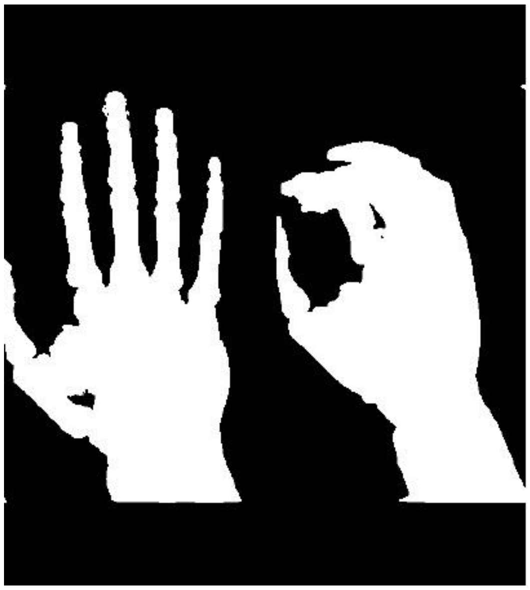


Figure 3. X-ray image segmentation using thresholding.

3.2. Global methods are based on the fact that the image has a bimodal histogram . The object of interest can be separated from the background by comparing the intensity of each pixel in the image with a threshold. Some pixels, whose intensity values are greater than the threshold, are classified as being part of group A - object of interest (with an intensity value of 1), and the rest of the pixels as being part of group B-background (with an intensity value of 0.

3.3. Adaptive methods are based on the fact that a given image is split into a series of sub-images and, for each subimage, some thresholds are computed. A different approach, called local adaptive thresholding, consists in analyzing the image intensities around each pixel and selecting an individual threshold for each pixel, taking in consideration the degree of the] intensity values in its local neighborhood.

3.4. Region-based methods have the purpose of grouping pixels having similar intensities. The most important region-based segmentation algorithms are: region-growing segmentation, and watershed algorithms.
3.5. Region-growing algorithm is a simple pixel-based image segmentation method, which involves the selection of pixels (the seeds), and then growing regions around these seeds, using a homogeneity criteria. If the joining pixels have similar image features as the seed, they are integrated into the region.

3.6. Edge-based segmentation methods use edge detectors to find edges in the image. Edge detection has an important role in image processing and computer vision, especially in feature detection and extraction domain. Edges can be viewed as image points, where the luminous intensity of the image changes distinctly along particular orientation. If the intensity of the images has a strong change, then there is a high probability for an edge at that image position.

The classical operators for edge detection are the following: **Prewitt, Sobel, Roberts and Laplacian of Gaussian (LoG) operator**. Most classical edge detectors are based on the local gradient (the first order derivatives) of the image function. Practically, the difference between these operators is that they use different types of filters for estimating the gradient components and a different way for combining these components.

3.6.1. Prewitt operator is a discrete operator which estimates the gradient of the image intensity function. It computes the approximations of the derivatives using two 3×3 kernels (masks), in order to find the localized orientation of each pixel in an image. Prewitt differs from Sobel operator only in the filters they use. Prewitt operator used the following filters:

$$H_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad (1)$$

and

$$H_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}, \qquad (2)$$

The local gradient components are obtained from the filter by scaling:

$$\nabla I(u,v) \approx \frac{1}{6} \cdot \begin{bmatrix} (I * H_x^P)(u,v) \\ (I * H_y^P)(u,v) \end{bmatrix} \qquad (3)$$

3.6.2. Sobel operator computes the approximation of gradients along the horizontal (x) and the vertical (y) directions (2D spatial) of the image intensity function, at each pixel, and highlights regions corresponding to edges. Sobel edge detection is implemented using two 3x3 convolution masks or kernels, one for horizontal direction, and the other for vertical direction in an image, that approximate the derivative along the two directions. Sobel operator uses the following filters:

$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad (4)$$

and

$$H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \qquad (5)$$

The two filters are almost identical with the filters used by Prewitt operator, excepting the weighting of the middle row (for horizontal kernel) and column (for vertical kernel): Sobel uses a weighting of 2 and -2, while Prewitt uses a weighting of 1 and-1.

The local gradient components are computed as follows:

$$\nabla I(u, v) \approx \frac{1}{8} \cdot \begin{bmatrix} (I * H_x^S)(u, v) \\ (I * H_y^S)(u, v) \end{bmatrix} \qquad (6)$$

3.6.3. Laplacian of Gaussian operator computes the second-order derivatives of the intensity function for a given image. The image is smoothed using a Gaussian smoothing filter, to reduce its sensitivity to noise, and then the Laplacian filter is applied. The edges obtained using LoG operator, have a more precise localization than the ones detected by applying Prewitt or Sobel

More advanced edge detectors have been proposed in the computer vision literature such as **Harris detector or Canny edge detector**. Harris finds the edges based on the eigenvalues of the Hessian matrix [2]. Canny is a very effective edge detecting technique. It detects faint edges, even when the image is noisy, because it is used after a series of preprocessing procedures, such as edge enhancement (Gaussian filtering). Next, the edge strength (magnitude) of the image must be found. This procedure implies the approximation of the image gradient in the xdirection (G_x) and in the in the y-direction (G_y), using Sobel operator. The gradient magnitudes are determined using the formula:

$$|G| = \sqrt{G_x^2 + G_y^2} \qquad (9)$$

The next step consists in finding the edge direction, as shown in the following equation:

$$\theta = \text{arctag}\left(\frac{|G_y|}{|G_x|}\right) \qquad (10)$$

3.7. Pattern recognition-based

Segmentation implies pixels classification, so it is frequently handled as a pattern recognition issue. Pattern recognition techniques include **unsupervised** methods (clustering) and **supervised** methods (classification) of **Machine Learning**.

Clustering or cluster analysis is an unsupervised method and refers to a class of algorithms extensively used for image segmentation. It is a technique for grouping a set of objects into groups (clusters), so that similar objects belong to the same cluster, while dissimilar object belong to different clusters. Various clustering algorithms have been proposed in the literature. Between them we mention: the **K-means algorithm**, hierarchical clustering or the Gaussian mixture approach.

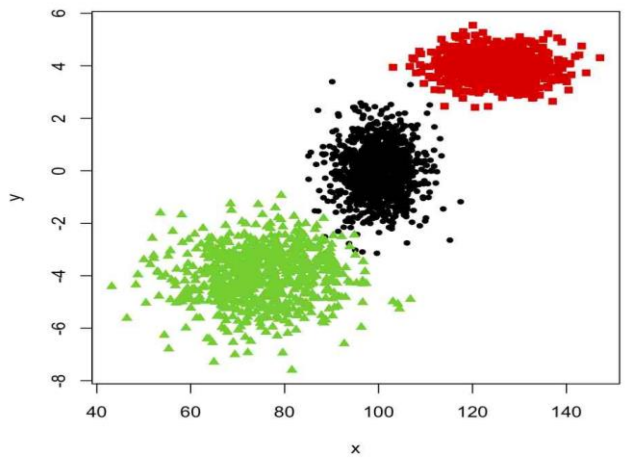


Figure 4. Formation of Clusters in K-Means Clustering

A particular algorithm that can be included in this category is the mean-shift algorithm which was introduced and searches modes or local maxima of the density function in the features space, defining the clusters. The next step is grouping data in these clusters. The two steps of the mean shift algorithm are: the filtering step, in which the original image is filtered in the feature space, and the clustering step, in which the filtered data points are grouped, using linkage clustering or edge-directed clustering. In the filtering step, the probability density function (pdf) of the image is analyzed in the feature space. The density function at point x is estimated using the next equation:

$$f(x) = \frac{c}{nh^d} \sum_{i=1}^{n} K \left\| \frac{x - x_i}{h} \right\|^2 , \qquad (11)$$

with n being the number of points, x_i(i =1,..,n) the pixels in the image, h the bandwidth, d the data dimensionality, a constant, and K(-) the density estimation kernel. The iterations for the centroids of clusters locations are evaluated as shown in figure 4.

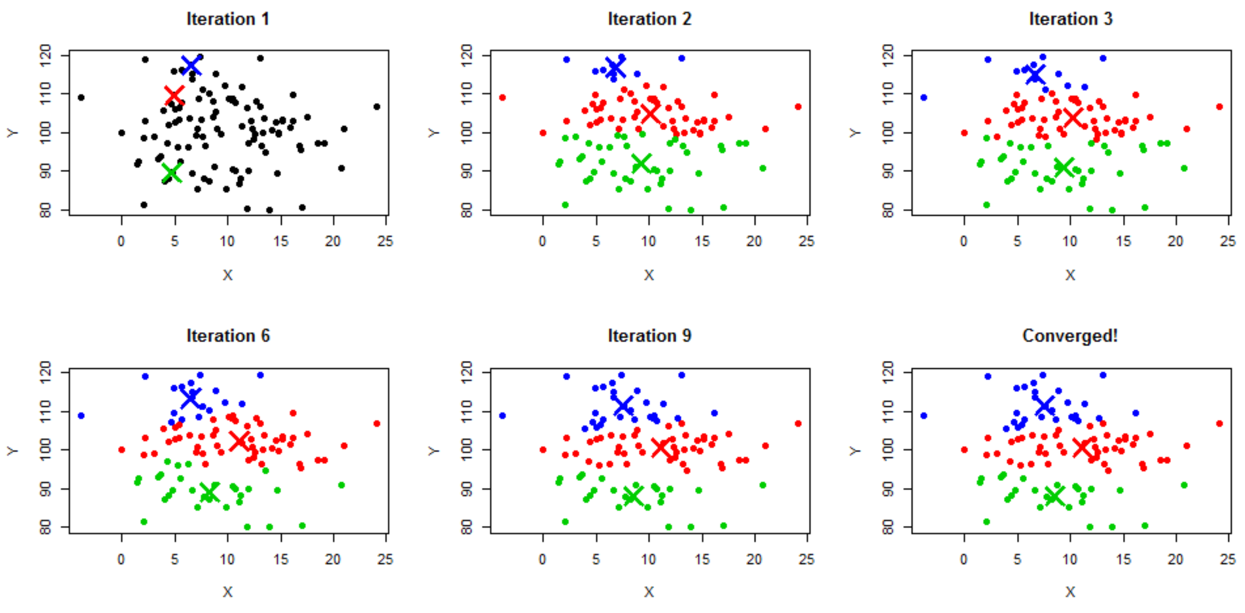


Fig 5. Iterations for Centroid estimation in K-Means Clustering

4. MATLAB IMPLEMENTATION FOR X-RAY BONE SEGMENTATION

General medical image segmentation methods can be categorized into the following classes: thresholding, regions-based, and edges-based. To exemplify some of the segmentation technique, we will consider a real X-ray image shown in Figure 2.

Image is read using the command :

```
I = imread('Abdul.jpg');
```

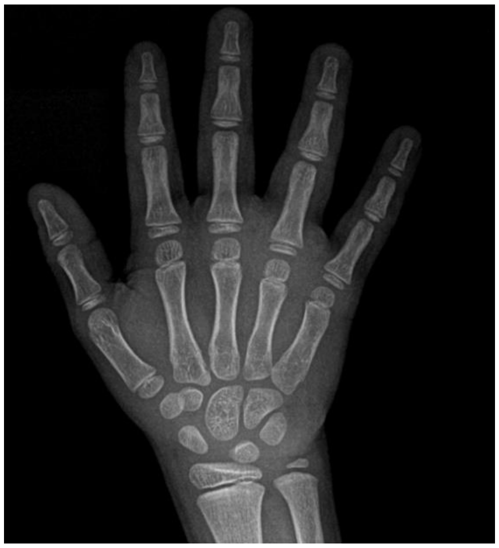


Fig. 2 An X-ray image test.

After that we converted the 3-D image to 2-D image by : "I = I(:,:,1);"

```
[cx, cy, c] = improfile(I,xi, yi);
```

improfile retrieves the intensity values of pixels along a line or a multiline path in the grayscale, binary, or RGB image in the current axes and displays a plot of the intensity values. If the specified path consists of a single line segment, improfile creates a two-dimensional plot of intensity values versus the distance along the line segment. If the path consists of two or more line segments, improfile creates a three-dimensional plot of the intensity values versus their x- and y-coordinates.

[cx,cy,c] = improfile(I,xi,yi,n) additionally returns the spatial coordinates of the pixels, cx and cy, of length n.

Then we cropped the image upto an extent where we can detect the bones which we need to segment with the help of the command :

```
I2 = imcrop(I, [x1, yi(1), x2-x1, y2-yi(1)-4]);
```

- I2 = imcrop(I,rect) crops the image I. rect is a four-element position vector of the form [xmin ymin width height] that specifies the size and position of the crop rectangle. imcrop returns the cropped image, I2.

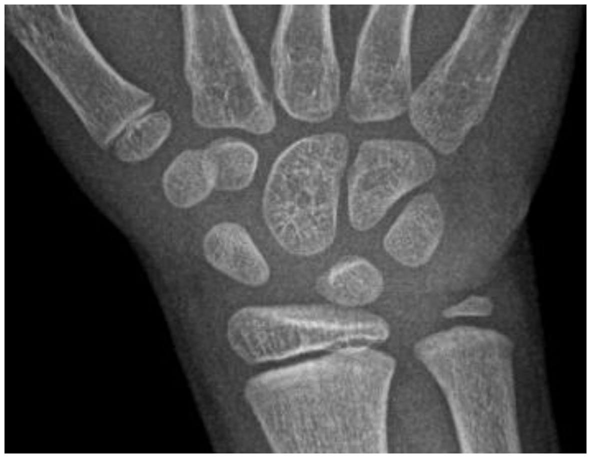


Fig. 3 Cropped image.

The above theory of thresholding is applied by using "*for-loop & if-else/break command* "

Edge-based segmentation methods use edge detectors to find edges in the image [2]. Edge detection has an important role in image processing and computer vision, especially in feature detection and extraction domain. Edges can be viewed as image points, where the luminous intensity of the image changes distinctly along particular orientation. If the intensity of the images has a strong change, then there is a high probability for an edge at that image position.

The classical operator used for edge detection is **Sobel operator [2]**. Sobel operator computes the approximation of gradients along the horizontal (x) and the vertical (y) directions (2D spatial) of the image intensity function, at each pixel, and highlights regions corresponding to edges. Sobel edge detection is implemented using two 3x3 convolution masks or kernels, one for horizontal

direction, and the other for vertical direction in an image, that approximate the derivative along the two directions.
Sobel operator is used by the following command:

```
[~, threshold] = edge(I2, 'sobel');
```

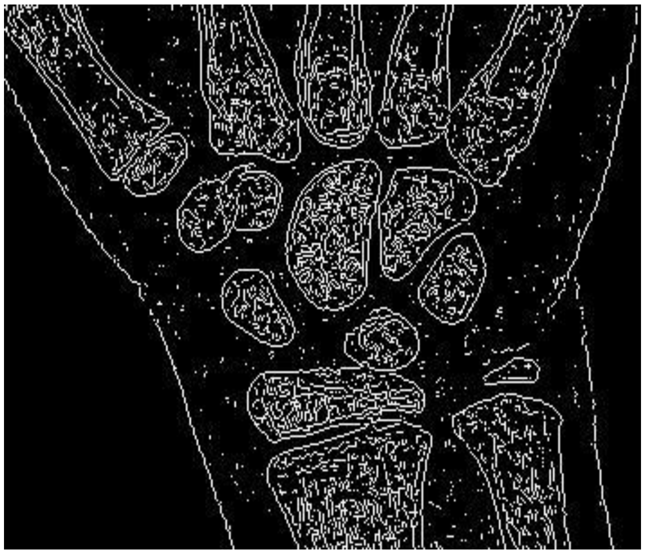


Fig. 4 Binay gradient mask image

Sobel operator uses the following filters:

$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad (4)$$

and

$$H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \qquad (5)$$

The two filters are almost identical with the filters used by Prewitt operator, excepting the weighting of the middle row (for horizontal kernel) and column (for vertical kernel): Sobel uses a weighting of 2 and -2, while Prewitt uses a weighting of 1 and-1.

The local gradient components are computed as follows:

$$\nabla I(u,v) \approx \frac{1}{8} \cdot \begin{bmatrix} (I * H_x^S)(u,v) \\ (I * H_y^S)(u,v) \end{bmatrix} \qquad (6)$$

Now we use **morphological operations** on the image to remove noise by using different types of structuring elements with the help of dilation and erosion methods required accordingly.

```
se90 = strel('line', 2, 90);
se0 = strel('line', 2, 0);
BWsdil = imdilate(BWs, [se90 se0]);
```

SE = strel('line',len,deg) creates a linear structuring element that is symmetric with respect to the neighborhood center. deg specifies the angle (in degrees) of the line as measured in a counterclockwise direction from the horizontal axis. len is approximately the distance between the centers of the structuring element members at opposite ends of the line.

IM2 = imdilate(IM,SE) dilates the grayscale, binary, or packed binary image IM, returning the dilated image, IM2. The argument SE is a structuring element object, or array of structuring element objects, returned by the strel.

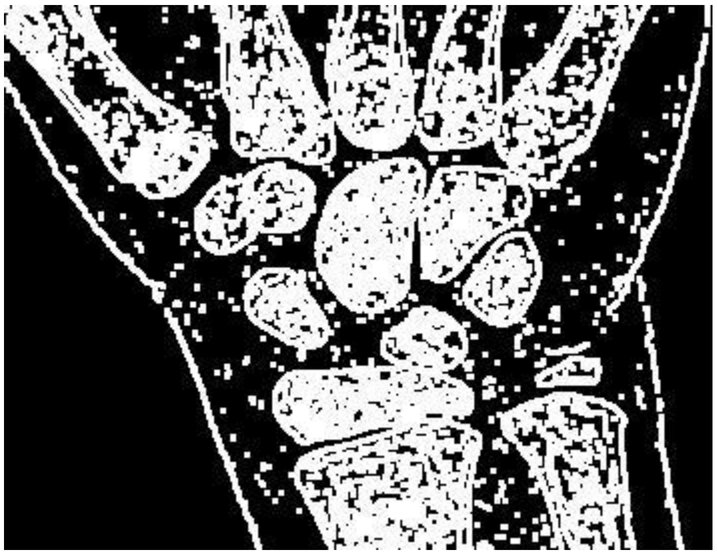


Fig. 5 Dilated Image

Now the bone's regions with boundaries are filled and extracted from the background using the command :

```
bw = imfill(BWsdil, 'holes');
```

BW2 = imfill(BW,'holes') fills holes in the input binary image BW. In this syntax, a hole is a set of background pixels that cannot be reached by filling in the background from the edge of the image.

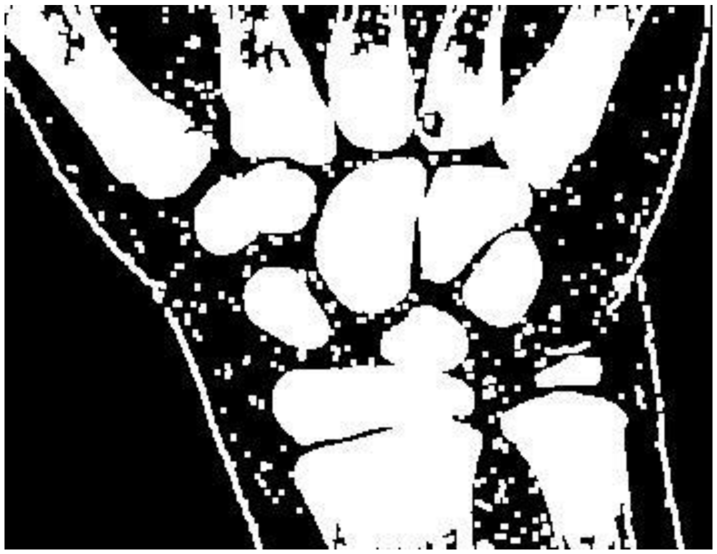


Fig. 6 Binary image filled with holes

After using dilation and filling methods, we will now try to remove these small white dots noise present with the help of the command :

```
bw2 = ~bwareaopen(~bw, 10);
```

BW2 = bwareaopen(BW,P) removes all connected components (objects) that have fewer than P pixels from the binary image BW, producing another binary image, BW2.

Now we will calculate Euclidean distance from each pixel to a non-zero pixel.

```
D = -bwdist(~bw);
```

D = bwdist(BW) computes the Euclidean distance transform of the binary image BW. For each pixel in BW, the distance transform assigns a number that is the distance between that pixel and the nearest nonzero pixel of BW.

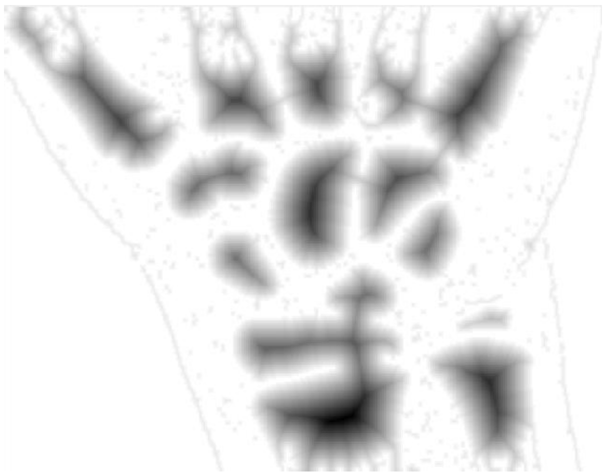


Fig. 7 Image after calculating the Euclidean distance transform

Now we apply **watershed algorithm** which is a type of region based thresholding. The direct command for applying this algorithm is:

```
Ld = watershed(D);
```

L = watershed(A) returns a label matrix L that identifies the watershed regions of the input matrix A, which can have any dimension. The watershed transform finds "catchment basins" or "watershed ridge lines" in an image by treating it as a surface where light pixels represent high elevations and dark pixels represent low elevations.

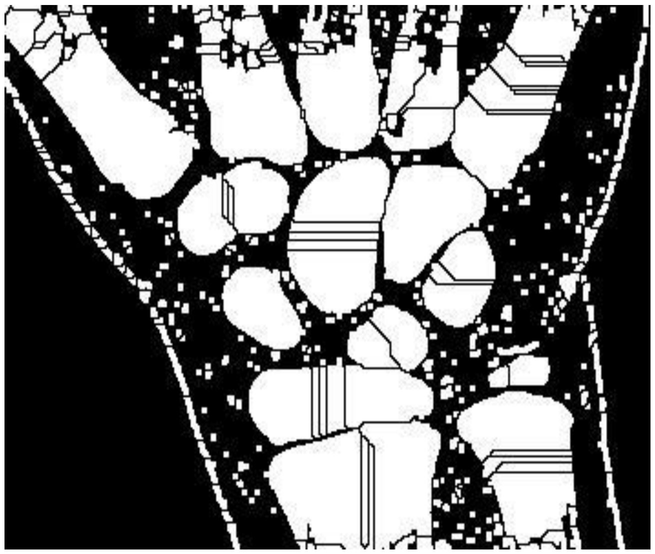


Fig. 8 Image after applying Watershed

Now we apply the second type of region based thresholding called as **region-growing algorithm** by using the command:

```
mask = imextendedmin(D,2);
```

BW = imextendedmin(I,h) computes the extended-minima transform, which is the regional minima of the H-minima transform. Regional minima are connected components of pixels with a constant intensity value, and whose external boundary pixels all have a higher value.

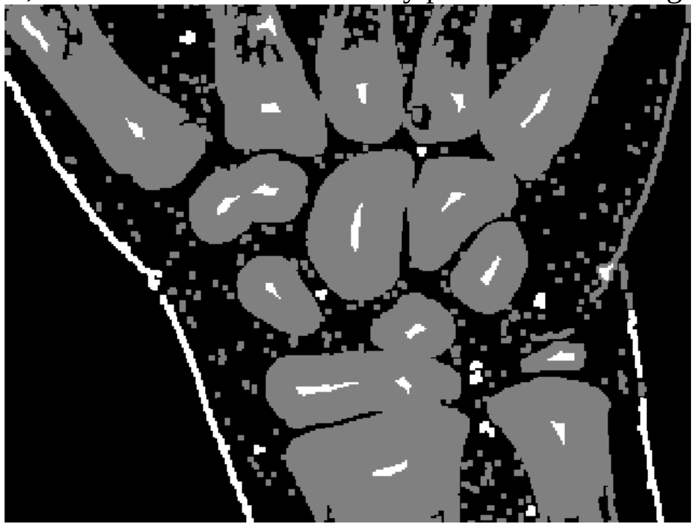


Fig. 9

After applying thresholding, we will impose them on one another by using the command:

```
D2 = imimposemin(D,mask);
```

I2 = imimposemin(I,BW) modifies the intensity image I using morphological reconstruction so it only has regional minima wherever BW is nonzero. BW is a binary image the same size as I.

Fig. 10 Superimposed Image

Now again watershed algorithm is applied to clearly extract the bones and removes noise then we use clear the borders of unwanted things or noise present with the help of the command :
BWnobord = imclearborder(BWdfill, 4);

IM2 = imclearborder(IM) suppresses structures that are lighter than their surroundings and that are connected to the image border. Use this function to clear the image border. IM can be a grayscale or binary image.

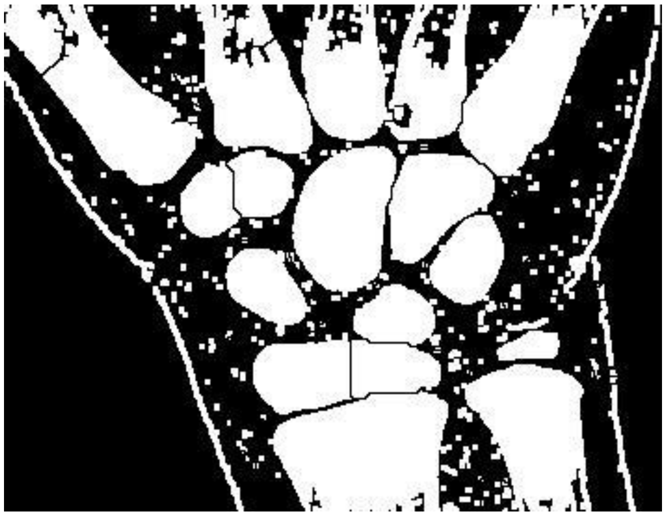


Fig.11 watershed image

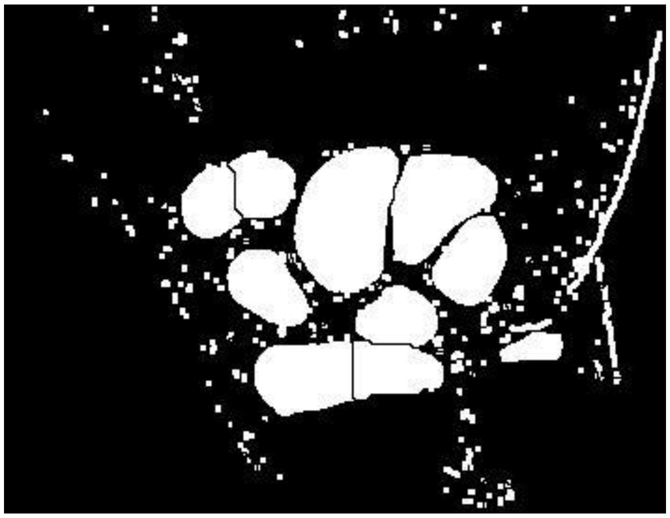


Fig.12 cleared border image

Now again we need to use morphological operations so we use a modified command for erosion and dilation which it follows the former one by using the command:
BWfinal = imopen(BWnobord, se);

The morphological "*open*" operation is an erosion followed by a dilation, using the same structuring element for both operations.

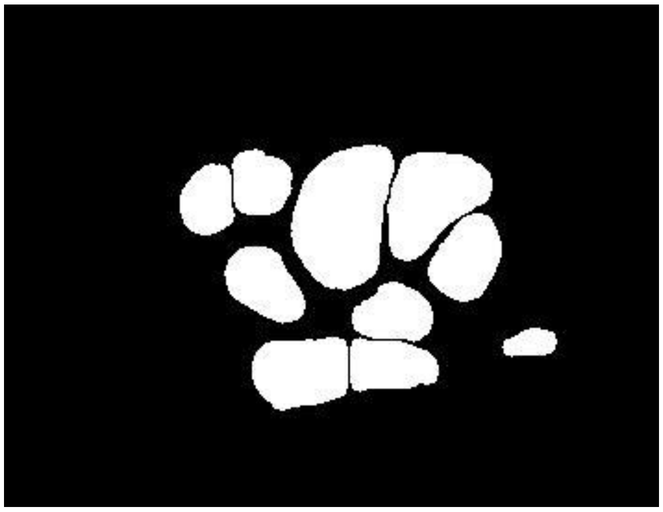


Fig.13 Final segmented image

Last time we implemented different types of image segmentation methods on the X-ray leg bone images and found out that if a fracture or any type of joint is present on the X-ray image then that many peaks will be formed in the Hough transform plot diagram which is between the Max Hough transform and different theta values [4].

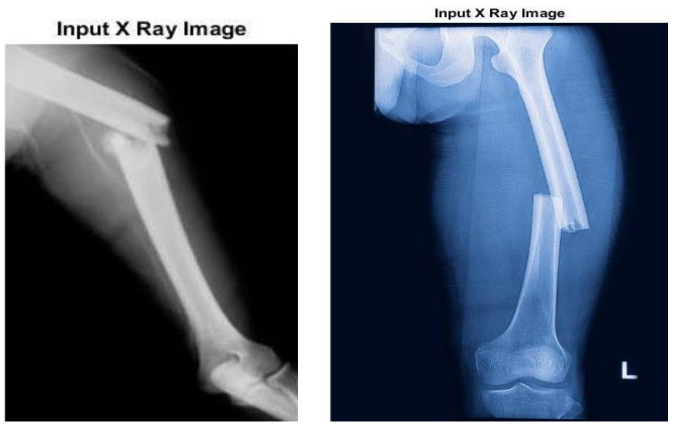


Fig 14 X-ray Images of leg

First we used Canny edge detection using mathematical formulae which detected the edges of bones [5].

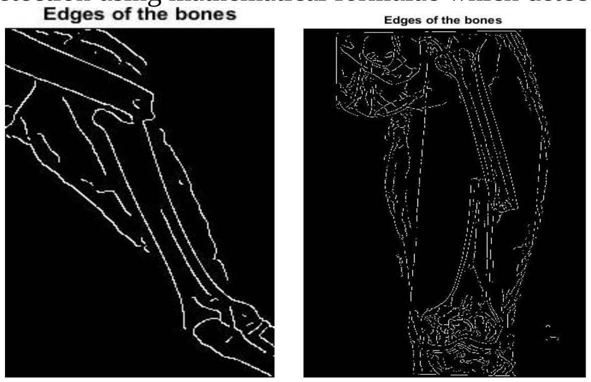


Fig 15 Canny edge detected Images

After using some morphological operations and Hough transform, we finally got the peaks at different theta values for the joints or fracture present in the X-ray image.[3]

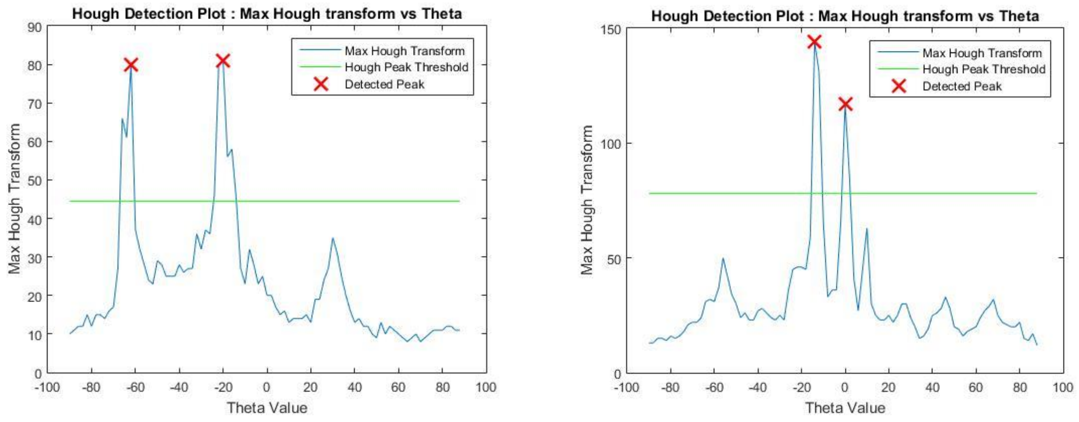


Fig 16 Hough transform plot

5. MATLAB IMPLEMENTATION FOR FRACTURE LOCATION

General medical image segmentation methods can be categorized into the following classes: thresholding, regions-based, and edges-based. To exemplify some of the segmentation technique, we will consider a real X-ray image shown in Figure 2.

Image is read using the command :

```
img = imread('leg_XRay.jpg');
```

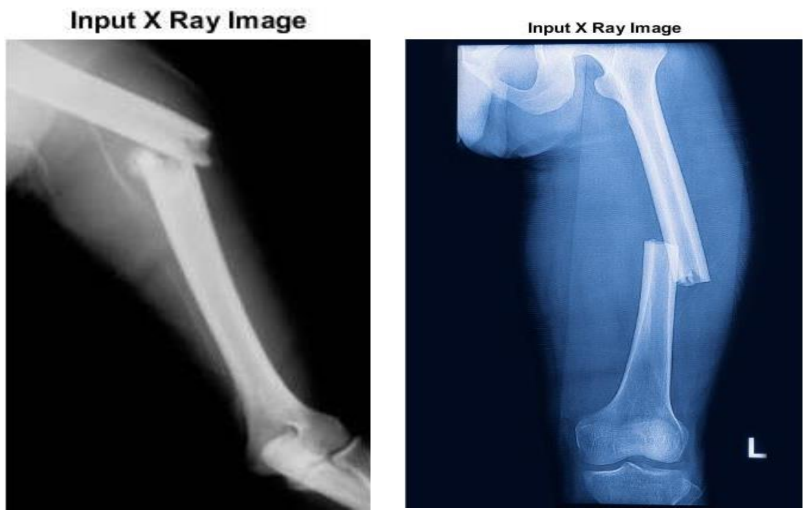


Fig. 17 An X-ray image test.

We used image blur parameter to denoise the x ray image and minimum Hough peak distance which is the Distance between peaks in Hough transform angle detection. All these parameters will be initialized first.

5.1. GRAY SCALE CONVERSION

```
img_gray = (rgb2gray(img));
```

I = rgb2gray(RGB) converts the true colour image RGB to the grayscale intensity image I. The rgb2gray function converts RGB images to grayscale by eliminating the hue and saturation information while retaining the luminance.

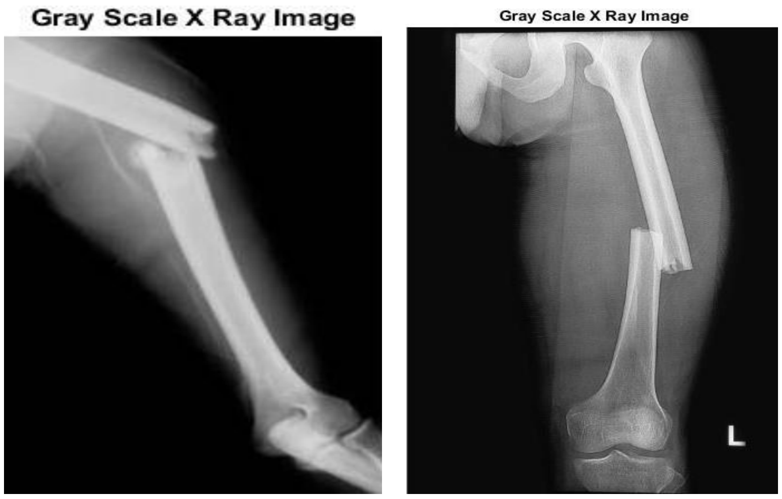


Fig.18 Denoising the RGB Image

```
img_filtered = imfilter(img_gray, fspecial('gaussian', 10, ImgBlurSigma), 'symmetric');
```

B = imfilter(A,h) filters the multidimensional array A with the multidimensional filter h. The array A can be logical or a nonsparse numeric array of any class and dimension. The result B has the same size and class as A.

imfilter computes each element of the output, B, using double-precision floating point. If A is an integer or logical array, imfilter truncates output elements that exceed the range of the given type, and rounds fractional values.

h = fspecial(type) creates a two-dimensional filter h of the specified type. Some of the filter types have optional additional parameters, shown in the following syntaxes. fspecial returns h as a correlation kernel, which is the appropriate form to use with imfilter.

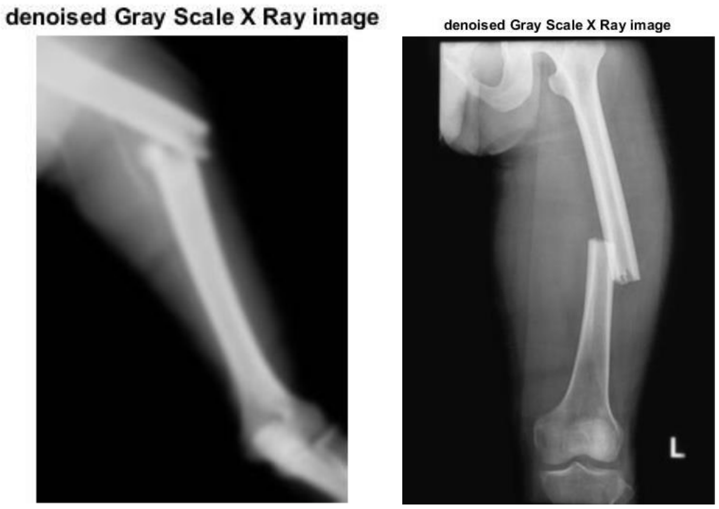

denoised Gray Scale X Ray image
denoised Gray Scale X Ray image
L

Fig 19. Gray scale Image

5.2. Canny Edge Detection

Edges are considered to be most important image attributes that provide valuable information for human image perception. Edge detection is a very complex process affected by deterioration due to different level of noise . An edge is the boundary between an object and the background. Edge detection is identifying points in a digital image at which the image brightness changes sharply or

more formally has discontinuities. The purpose of detecting sharp changes in image brightness is to capture important events and changes in properties of the world .
Edge detection is used for identification of blurred frame broad classification among smooth and rough surface classification of cement and asphalt. The Canny edge detection is performed on the frames with the sensitive threshold values (upper threshold 10000 and lower threshold 4900) and again it is performed with the insensitive threshold values (upper threshold 50000 and lower threshold 9800). If a pixel has a gradient greater than the upper threshold, then it is an edge pixel. If a pixel has a gradient lower than the lower threshold, it is not an edge pixel. If the pixel's gradient is between the upper and the lower thresholds, then it is considered as an edge, only if it is connected to a pixel that is above the high threshold value as given in
Canny is one of modern edge detection method that founded by Marrdan Hildreth, who is doing research in modeling human visual perception.
There are several criteria on edge detecting that can be fulfilled by Canny Edge Detection:
1. Canny has better detection (for detection criteria). Canny method capable to marks all existing edges matching with user determined parameter's threshold. Also giving high flexibility on determining thickness level of edge detection according to the required conditions.
2. Canny has better localizing way (localize criteria). Canny capable on producing minimum gap between detected edge and the real image edge.
3. Obvious response (response criteria). Only one response for every edge. This make less confusion on edge detection for the next image. Choosing parameters on Canny Edge Detection will giving effect on every result and edge detection. The parameters are :
a. Gaussian Deviation Standard Value.
b. Threshold Value.

The following is the steps to do Canny Edge Detection.[5]
1. Remove all noise on the image by implementing Gaussian Filter. The result is an image with less blur. It is intended to obtain the real edges of the image. If we did not apply the Gaussian Filter before, sometimes the noise itself will be detected as an edge.
2. Detect the edge with one of these detection operators, like Roberts, Perwit, or Sobel by do horizontal searching (Gx) and vertical searching (Gy). The following is the sample of edge detection operator (Sobel operators).

$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \tag{4}$$

and

$$H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \tag{5}$$

The two filters are almost identical with the filters used by Prewitt operator, excepting the weighting of the middle row (for horizontal kernel) and column (for vertical kernel): Sobel uses a weighting of 2 and -2, while Prewitt uses a weighting of 1 and-1.
The local gradient components are computed as follows:

$$\nabla I(u,v) \approx \frac{1}{8} \cdot \begin{bmatrix} (I * H_x^S)(u,v) \\ (I * H_y^S)(u,v) \end{bmatrix} \tag{6}$$

The result from both operators combined to obtain the summary of vertical edge and horizontal edge with this formula:

$$[G] = [Gx] + [Gy]$$

3. Determining direction of the edge by using the following formula:

$$|G| = \sqrt{G_x^2 + G_y^2} \quad (9)$$

The next step consists in finding the edge direction, as shown in the following equation:

$$\theta = \text{arctag}\left(\frac{|G_y|}{|G_x|}\right) \quad (10)$$

Canny Edge Detection [5] using two thresholds (maximum threshold and minimum threshold). If pixel gradient higher than maximum threshold, pixel will be marked as an edge. If the pixel gradient lower than minimum threshold, the pixel will be denied as background image. If the pixel gradient between maximum threshold and minimum threshold, the pixel will be accepted as an edge if it is connected with other edge pixel that higher than maximum threshold.
4. Minimize the emerging edge line by applying non maximum suppression. This process will give slimmer edge line.
5. The last step is binarizing the image pixels by applying two threshold value.

- Do edge detection to find bone edges in image
- Filter out all but the two longest lines
- This feature may need to be changed if break is not in middle of bone

```
boneEdges = edge(img_filtered, 'canny');
```

BW = edge(I,'Canny') detect edges using the Canny method. The Canny method finds edges by looking for local maxima of the gradient of I. The edge function calculates the gradient using the derivative of a Gaussian filter. This method uses two thresholds to detect strong and weak edges, including weak edges in the output if they are connected to strong edges. By using two thresholds, the Canny method is less likely than the other methods to be fooled by noise, and more likely to detect true weak edges.

The Canny method is not supported on a GPU.

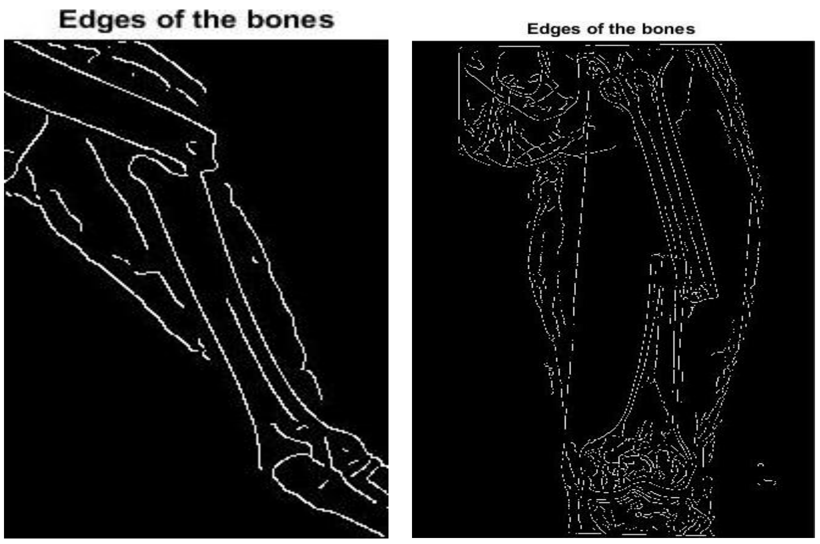


Fig 20. Canny edge detected image

5.3. Morphological Operations

Now we use **morphological operations** on the image to remove noise by using different types of structuring elements with the help of dilation and erosion methods required accordingly.

```
se90 = strel('line', 2, 90);
se0 = strel('line', 2, 0);
BWsdil = imdilate(BWs, [se90 se0]);
```

SE = strel('line',len,deg) creates a linear structuring element that is symmetric with respect to the neighborhood center. deg specifies the angle (in degrees) of the line as measured in a counterclockwise direction from the horizontal axis. len is approximately the distance between the centers of the structuring element members at opposite ends of the line.

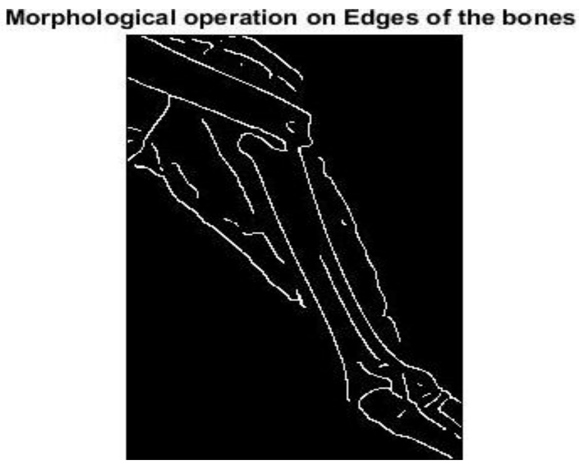
IM2 = imdilate(IM,SE) dilates the grayscale, binary, or packed binary image IM, returning the dilated image, IM2. The argument SE is a structuring element object, or array of structuring element objects, returned by the strel.

```
BWfinal = imopen(BWnobord, se);
```

The morphological "*open*" operation is an erosion followed by a dilation, using the same structuring element for both operations.

```
boneEdges1 = bwmorph(boneEdges, 'close');
```

W2 = bwmorph(BW,operation) applies a specific morphological operation to the binary image BW.Over here we are performing close operations



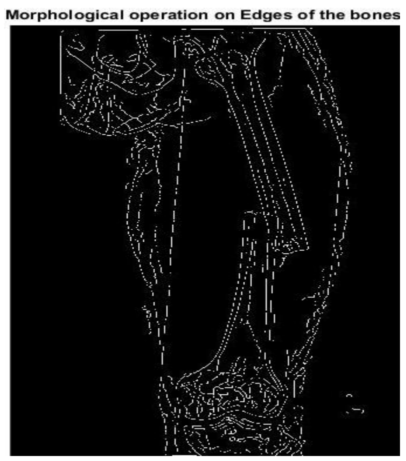


Fig 21. Image after morphological operation

```
edgeRegs = regionprops(boneEdges1, 'Area', 'PixelIdxList');
```

stats = regionprops(CC,properties) returns measurements for the set of properties specified by properties for each connected component (object) in CC. CC is a structure returned by bwconncomp.

```
AreaList = sort(vertcat(edgeRegs.Area), 'descend');
```

B = sort(___,direction) returns sorted elements of A in the order specified by direction using any of the previous syntaxes. 'ascend' indicates ascending order (the default) and 'descend' indicates descending order.

```
edgeRegs(~ismember(vertcat(edgeRegs.Area), AreaList(1:2))) = [];
```

Lia = ismember(A,B) returns an array containing logical 1 (true) where the data in A is found in B. Elsewhere, the array contains logical 0 (false).

```
edgeImg = zeros(size(img_filtered, 1), size(img_filtered,2));
```

X = zeros(sz) returns an array of zeros where size vector sz defines size(X)

```
edgeImg(vertcat(edgeRegs.PixelIdxList)) = 1;
```

The above theory of thresholding is applied by using “for-loop & if-else/break command ”

5.4. HOUGH TRANSFORM

The Hough transform is a feature extraction technique used in image analysis, computer vision, and digital image processing.[1] The purpose of the technique is to find imperfect instances of objects within a certain class of shapes by a voting procedure. This voting procedure is carried out in a parameter space, from which object candidates are obtained as local maxima in a so-called accumulator space that is explicitly constructed by the algorithm for computing the Hough transform.

The classical Hough transform was concerned with the identification of lines in the image, but later the Hough transform has been extended to identifying positions of arbitrary shapes, most commonly circles or ellipses.

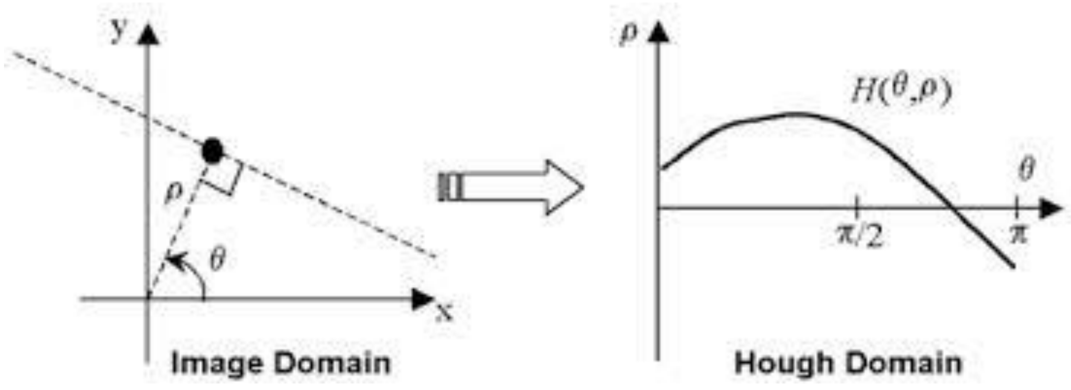


Fig 22. Graph of Hough domain resulted from a line

[H,T,R] = hough(edgeImg,'RhoResolution',1,'Theta',-90:2:89.5);
[H,theta,rho] = hough(BW) computes the Standard Hough Transform (SHT) of the binary image BW. The hough function is designed to detect lines. The function uses the parametric representation of a line: rho = x*cos(theta) + y*sin(theta). The function returns rho, the distance from the origin to the line along a vector perpendicular to the line, and theta, the angle in degrees between the x-axis and this vector. The function also returns the Standard Hough Transform, H, which is a parameter space matrix whose rows and columns correspond to rho and theta values respectively.
maxHough = max(H, [], 1);
It fetched the maximum values in every column.

HoughThresh = (max(maxHough) - min(maxHough))/2 + min(maxHough);
This is the Threshold value for the Maximum hough transforms.

[~, HoughPeaks] = findpeaks(maxHough,'MINPEAKHEIGHT',HoughThresh, 'MinPeakDistance', MinHoughPeakDistance);
pks = findpeaks(data) returns a vector with the local maxima (peaks) of the input signal vector, data. A local peak is a data sample that is either larger than its two neighboring samples or is equal to Inf. Non-Inf signal endpoints are excluded. If a peak is flat, the function returns only the point with the lowest index.

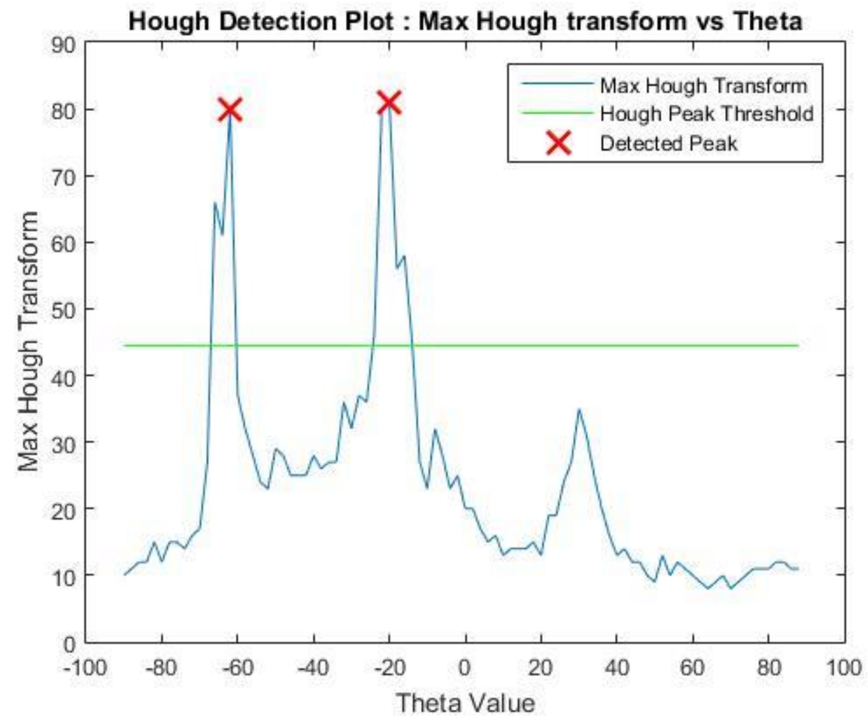


Fig 23. Hough transform v/s theta values

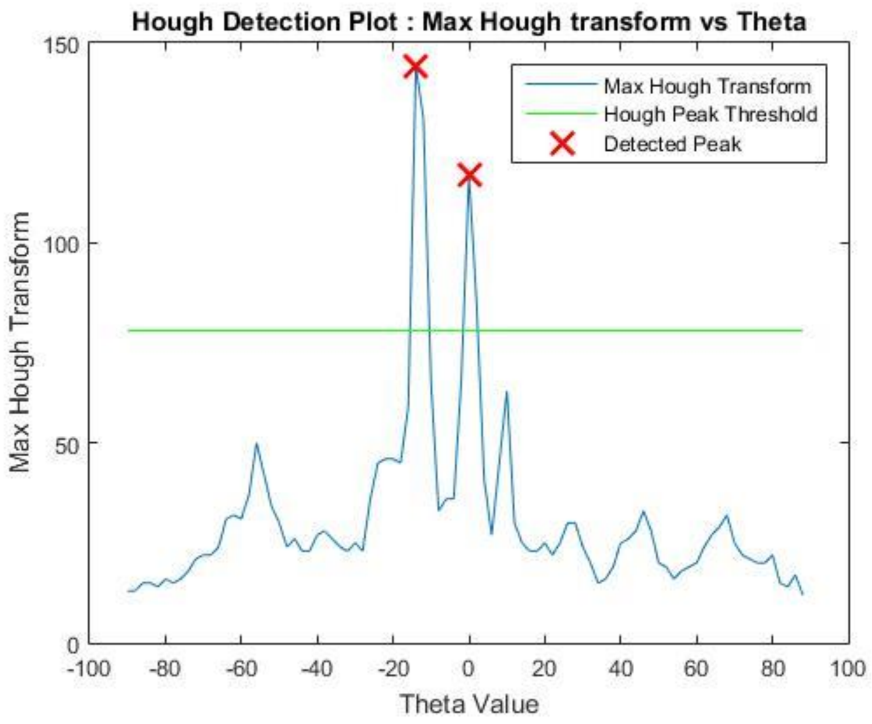


Fig 24. Hough transform v/s theta values

5.5. LOCATING SITE OF BREAK

General method to detect number of joints or fractures present in any image is by using function [4]:

Aa= numel(HoughPeaks); [6]
This is the number of joints.

n = numel(A) returns the number of elements, n, in array A, equivSalent to prod(size(A)).

With the help of this function, we can know the number of joints or fractures present in X-ray image. After that we create a zero matrix of size of filtered image taken before as 300x200x2. We used 2 because there are 2 joints present in the image given, Well there could be more than depending upon the input X Ray images

BreakStack = zeros(size(img_filtered, 1), size(img_filtered, 2), numel(HoughPeaks));

We have created a zero matrix with the number of layers equal to the number of joints

X = zeros(sz1,...,szN)

It returns an sz1-by-...-by-szN array of zeros where sz1,...,szN indicate the size of each dimension. For example, zeros(2,3) returns a 2-by-3 matrix.

5.6. CONVOLUTION OF EDGE IMAGE

Now we will be convolving edge image with line of detected angle from Hough transform using filter and morphological operations.

boneKernel = strel('line', HoughConvolutionLength, T(HoughPeaks(m)));[6]

HoughConvolutionLength = 40; % Length of line to use to detect bone regions

A strel object represents a flat morphological *structuring element*, which is an essential part of morphological dilation and erosion operations.

A flat structuring element is a binary valued neighborhood, either 2-D or multidimensional, in which the true pixels are included in the morphological computation, and the false pixels are not. The center pixel of the structuring element, called the *origin*, identifies the pixel in the image being processed. Use the strel function (described below) to create a flat structuring element. You can use flat structuring elements with both binary and grayscale images.

SE = strel('line',len,deg) creates a linear structuring element that is symmetric with respect to the neighborhood center. deg specifies the angle (in degrees) of the line as measured in a counterclockwise direction from the horizontal axis. len is approximately the distance between the centers of the structuring element members at opposite ends of the line.

kern = double(bwmorph(boneKernel.getnhood(), 'dilate', HoughConvolutionDilate));
HoughConvolutionDilate = 2; Amount to dilate kernel for bone detection

BW2 = bwmorph(BW,operation,n) applies the operation n times. n can be Inf, in which case the operation is repeated until the image no longer changes.[6]

NHOOD = getnhood(SE) returns the neighborhood associated with the structuring element SE.
SE is a STREL object. NHOOD is a logical array.

double(s) converts the symbolic value s to double precision. Converting symbolic values to double precision is useful when a MATLAB® function does not accept symbolic values.

BreakStack(:,:,m) = imfilter(edgeImg, kern).*edgeImg;

B = imfilter(A,h) filters the multidimensional array A with the multidimensional filter h. The array A can be logical or a nonsparse numeric array of any class and dimension. The result B has the same size and class as A.[1]

Here we convolved edgeImg which is first filtered with kern which is a double precision image with the edgeImg only which gives clear view of joints present in the image. [1]

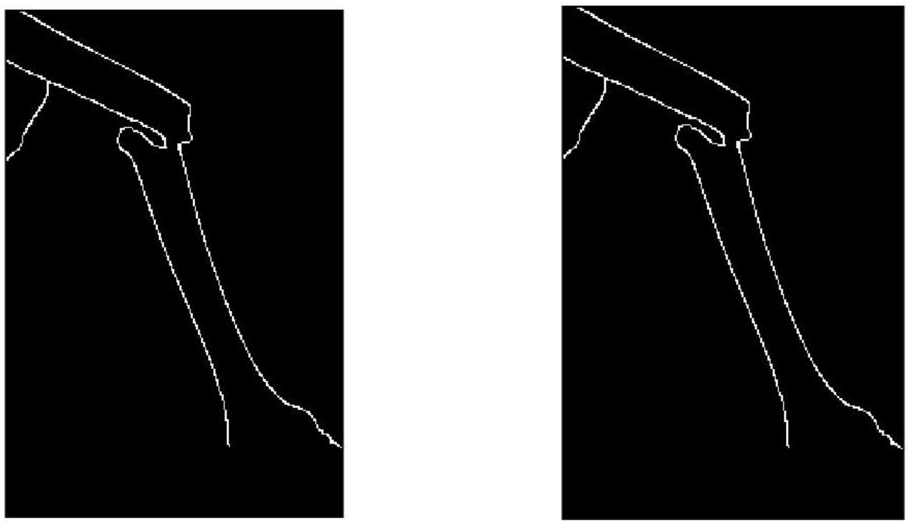


Fig 25 Filtered Images after Convolution

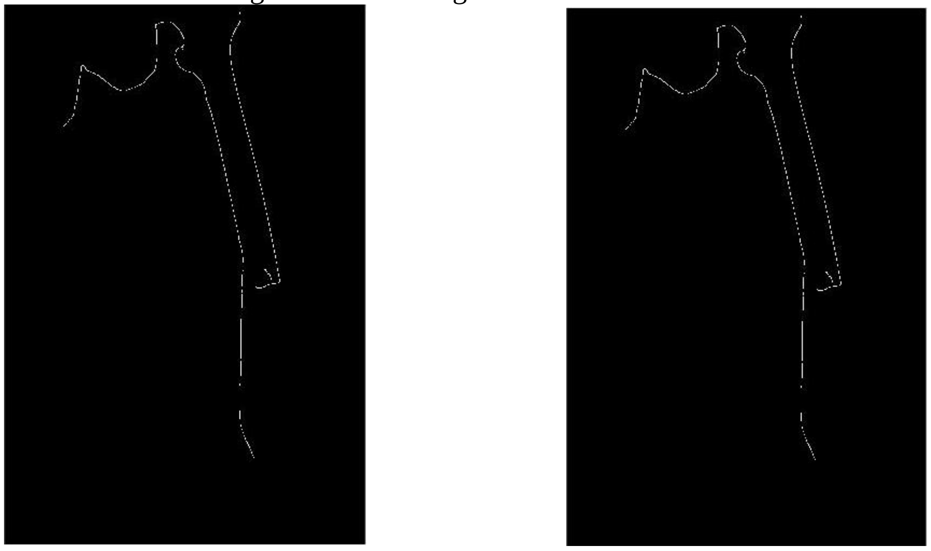


Fig 26 Filtered Images after Convolution

5.7. DIFFERENCE BETWEEN CONVOLUTION IMAGES

Now we will take difference between the convolution images generated above. Locations where these images will cross zero (within tolerance) should be the position where break is. And after that we have to filter out the regions elsewhere where the bone simply ends.[1]

brImg = abs(diff(BreakStack, 1, 3)) < BreakLineTolerance*max(BreakStack(:)) & edgeImg > 0;

Y = diff(X,n,dim) is the nth difference calculated along the dimension specified by dim. The dim input is a positive integer scalar.

Y = abs(X) returns the absolute value of each element in array X.

[BpY, BpX] = find(abs(diff(BreakStack, 1, 3)) < BreakLineTolerance*max(BreakStack(:)) & edgeImg > 0);
BreakLineTolerance = 0.25; Tolerance for bone end detection

[row,col] = find(___) returns the row and column subscripts of each nonzero element in array X using any of the input arguments.

brImg = bwmorph(brImg, 'dilate', breakPointDilate);

breakPointDilate = 6; Amount to dilate detected bone end points

Now we will use morphological operation i.e. dilation on the image which is generated after taking differences. This operation will help us to identify those points only where a joint is present in the image.
brReg = regionprops(brImg, 'Area', 'MajorAxisLength', 'MinorAxisLength',
            'Orientation', 'Centroid');

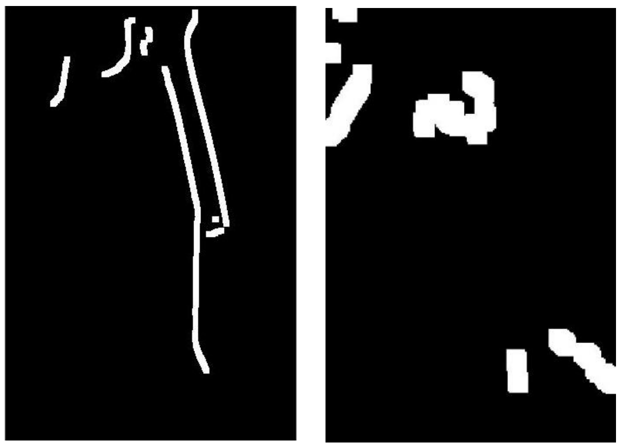


Fig 27 Dilated Images of the Break Points

stats = regionprops(BW,properties) returns measurements for the set of properties specified by properties for each 8-connected component (object) in the binary image, BW. stats is struct array containing a struct for each object in the image. You can use regionprops on contiguous regions and discontiguous regions [1]

brReg(vertcat(brReg.Area) ~= max(vertcat(brReg.Area))) = [];

C = vertcat(A1,...,AN) vertically concatenates arrays A1,...,AN. All arrays in the argument list must have the same number of columns.

This will help in finding the actual fracture in the X-ray image and not of joints present there. Now we will try to locate this fracture in the image which can provide some sort of circle or anything else to highlight the fracture present.[1]

5.8. CALCULATING AND DRAWING BOUNDING ELLIPSE

Now we will calculate coordinates of ellipse which will be used to encircle the fracture found above. And that will show us the location of fracture present.

brReg.EllipseCoords = zeros(100, 2); [6]

here we created a zero matrix of size 100x2 in the Ellipse coordinates matrix and which will be filled with the coordinates of fracture location.

t = linspace(0, 2*pi, 100);

y = linspace(x1,x2,n) generates n points. The spacing between the points is (x2-x1)/(n-1).

linspace is similar to the colon operator, ":", but gives direct control over the number of points and always includes the endpoints. "lin" in the name "linspace" refers to generating linearly spaced values as opposed to the sibling function logspace, which generates logarithmically spaced values.

brReg.EllipseCoords(:,1) = brReg.Centroid(1) + brReg.MajorAxisLength/2*cos(t - brReg.Orientation);
brReg.EllipseCoords(:,2) = brReg.Centroid(2) + brReg.MinorAxisLength/2*sin(t - brReg.Orientation);

Now these both above formulae are the parametric form of ellipse equation whose centroid is shifted to other coordinates so this forms the ellipse on the fracture as we have shifted the centroid to the fracture centroid location. [4]

5.9. PLOTTING OF ELLIPSE

Now we will plot ellipse whose coordinates are calculated above and will highlight the location of bone fracture present in X-ray image.

figure(7)
imshow(img)
hold on
colormap('gray')

colormap map sets the colormap for the current figure to one of the predefined colormaps. If you set the colormap for the figure, then axes and charts in the figure use the same colormap. The new colormap is the same length (number of colors) as the current colormap.

plot(brReg.EllipseCoords(:,1), brReg.EllipseCoords(:,2), 'r');[6]

plot(X,Y,LineSpec) sets the line style, marker symbol, and color.

plot(X,Y) creates a 2-D line plot of the data in Y versus the corresponding values in X.

- If X and Y are both vectors, then they must have equal length. The plot function plots Y versus X.

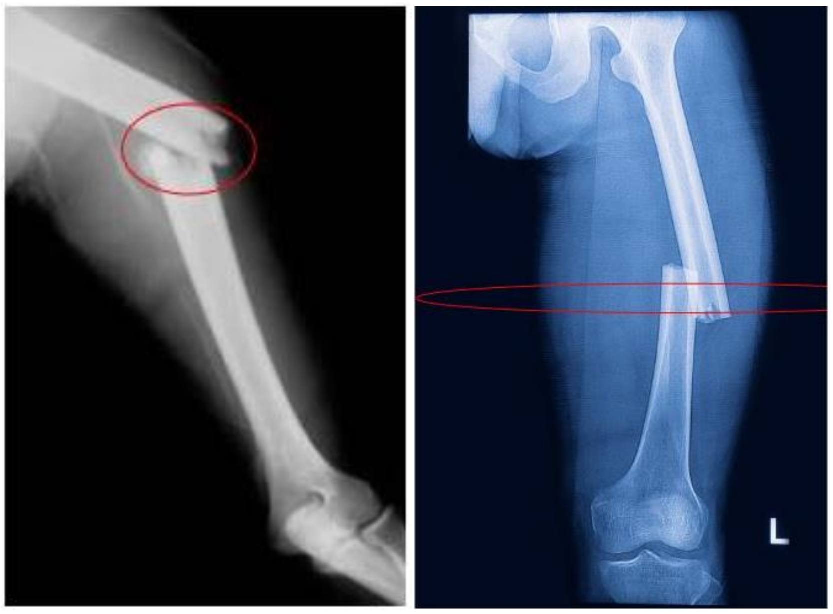


Fig 28 Images showing location of fracture in X-rays

6. PERFORMANCE EVALUATION

Image Dataset	No. of actual Fractures	No. of Detected Fractures	Accuracy (in %)
Xray1.jpeg	3	3	100
Xray2.jpeg	3	2	66.67
Xray3.jpeg	2	2	100
Xray4.jpeg	2	2	100
Xray5.jpeg	2	1	50
Xray6.jpeg	1	1	100
Xray7.jpeg	1	1	100

Table 1: Performance Evaluation

7. RESULT ANALYSIS

The performance analysis of the Fracture Detection is done using slandered measurement parameters of True Positive Rate (TPR), False Positive Rate (FPR), Precision and Recall.

However recall is nothing but TPR and FPR is measure for Fall-out.

TPR = TP/(TP + FN)

FPR = FP/(FP + TN)

Precision = TP/(TP + FP)

Where TP, FP, TN & FN are defined in compliance with problem as:

• TP is number of Fractures in which it is detected when actually it is present

• FP is number of Fractures in which it is detected when actually it is not present

• TN is number of Fractures in which it is not detected when actually it is not present

• FN is number of Fractures in which it is not detected when actually it is present

7.1. TPR, FPR AND PRECISION CALCULATION

In our cases, We have calculated these values for the Fractured X-Rays as:

- ▶ TP= 58
- ▶ FP= 6
- ▶ TN= 31
- ▶ FN= 8
- ▶ TPR(%)=58/(58+8)=87.88%
- ▶ FPR(%)=6/(6+31)=16.21%
- ▶ Precision=58/(58+6)=90.63%

8. CONCLUSION AND FUTURE SCOPE

- ▶ In the end we would like to conclude that we have done with segmentation of the bones of the interest in the X-Ray Images by applying certain algorithms like thresholding, morphological operations etc. initially till the 7th semester.
- ▶ Now we have completed fracture detection and their location in the X-Ray images by using Hough Transforms and convolutions
- ▶ We have tried on many X-Ray images as well getting the proper results for the fracture in the bones.
- ▶ We have located the break site and created the bounding ellipse on that break site for the location of fracture.
- ▶ We are getting the precision of upto 93.33% on certain fracture datasets.
- ▶ This precision could be increased with the Deep Learning Algorithms (CNN,RNN,etc)[4].
- ▶ We are also writing a Research Paper based on this Project as It is quite an innovative research for Bones segmentation and fracture Location

9. REFERENCES

- [1] R.Aishwariya , M.Kalaiselvi Geetha , M.Archana, Computer- Aided Fracture Detection of X-Ray Images. IOSR Journal of Computer Engineering (IOSR-JCE) e-ISSN: 2278-0661, p-ISSN: 2278-8727 PP 44-51.
- [2] Cristina STOLOJESCU-CRIŞAN, Ştefan HOLBANA, Comparison of X-Ray Image Segmentation Techniques. Advances in Electrical and Computer Engineering.
- [3] Nathanael .E. Jacob, M.V. Wyawahare, Tibia Bone Segmentation in X-ray Images - A Comparative Analysis. International Journal of Computer Applications (0975 – 8887).
- [4] S.K.Mahendran and S.Santhosh Baboo, Automatic Fracture Detection Using Classifiers- A Review IJCSI International Journal of Computer Science Issues, Vol. 8, Issue 6, No 1, November 2011 ISSN (Online): 1694-0814.
- [5] SAMUEL FEBRIANTO KURNIAWAN,,I KETUT GEDE DARMA PUTRA,A.A KOMPIANG OKA SUDANA, BONE FRACTURE DETECTION USING OPENCV, Journal of Theoretical and Applied Information Technology, ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195.
- [6] www.mathworks.com

10. APPENDIX

A. Matlab code for the segmentation of Bones:

```
clc;
tic;
clear all;
close all;
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% X-ray image segmentation
I = imread('xray1.jpg');
figure
imshow(I)

I = I(:,:,1);

[yl xl] = size(I);
xi = [0, xl];
yi = [yl*0.60, yl*0.60];
[cx, cy, c] = improfile(I,xi, yi);
flag1 = 1;
for i=1:xl
    if c(i) > 60 && flag1
        x1 = cx(i);
        flag1 = 0;
    end
    if c(i) < 20 && flag1 == 0
        x2 = cx(i);
        break;
    end
end
xi = [0, xl];
yi2 = [yl*0.98, yl*0.98];
[cx, cy, c] = improfile(I,xi, yi2);

flag1 = 1;
up = 0;
down = 0;
for i=1:yl
    if c(i) < 130
        y2 = cy(i);
        break;
    end
end


I2 = imcrop(I, [x1, yi(1), x2-x1, y2-yi(1)-4]);
figure(2);
imshow(I2)
title('Cropped focused image');
[~, threshold] = edge(I2, 'sobel');
fudgeFactor = .5;
BWs = edge(I2,'sobel', threshold * fudgeFactor);
figure(3)
imshow(BWs)
title('binary gradient mask');
imwrite(BWs, 'exp.jpg');
se90 = strel('line', 2, 90);
se0 = strel('line', 2, 0);
BWsdil = imdilate(BWs, [se90 se0]);
figure(4)
imshow(BWsdil)
title('dilated gradient mask');

sedisk = strel('disk', 4);
bw = imfill(BWsdil, 'holes');
bw2 = ~bwareaopen(~bw, 10);
imshow(bw2)
D = -bwdist(~bw);
imshow(D,[])
Ld = watershed(D);
imshow(label2rgb(Ld))
bw2 = bw;
bw2(Ld == 0) = 0;
```

```
imshow(bw2)
mask = imextendedmin(D,2);
imshowpair(bw,mask,'blend')
D2 = imimposemin(D,mask);
Ld2 = watershed(D2);
bw3 = bw;
bw3(Ld2 == 0) = 0;
imshow(bw3)
BWdfill = bw3;
figure(5)
imshow(BWdfill);
title('binary image with filled holes');
BWnobord = imclearborder(BWdfill, 4);
figure(6)
imshow(BWnobord)
title('cleared border image');

radius = 5;
decomposition = 0;
se = strel('disk', radius, decomposition);
BWfinal = imopen(BWnobord, se);
figure(7)
imshow(BWfinal)
 title('final segmented image');
```

B. Matlab code for the Fracture of Bones:

```
clear all;
close all;
%img = imread('http://i.stack.imgur.com/mHo7s.jpg');
%img = imread('frac3.jpg');
img = imread('leg_XRay.jpg');
figure(1)
imshow(img);
title('Input X Ray Image');

%% Important parameters

ImgBlurSigma = 2; % Amount to denoise input image
MinHoughPeakDistance = 5; % Distance between peaks in Hough
transform angle detection
HoughConvolutionLength = 40; % Length of line to use to detect
bone regions
HoughConvolutionDilate = 2; % Amount to dilate kernel for bone
detection
BreakLineTolerance = 0.25; % Tolerance for bone end detection
breakPointDilate = 6; % Amount to dilate detected bone end
points
```

```
%%%%%%%%%%%%%%%%%%%%%%%

img_gray = (rgb2gray(img)); % Load image
figure(2)
imshow(img_gray);
title('Gray Scale X Ray Image');
img_filtered = imfilter(img_gray, fspecial('gaussian', 10,
ImgBlurSigma), 'symmetric'); % Denoise
figure(3)
imshow(img_filtered);
title('denoised Gray Scale X Ray image');

% Do edge detection to find bone edges in image
% Filter out all but the two longest lines
% This feature may need to be changed if break is not in middle
of bone
boneEdges = edge(img_filtered, 'canny');
figure(4)
imshow(boneEdges);
title('Edges of the bones');

boneEdges1 = bwmorph(boneEdges, 'close');
figure(5)
imshow(boneEdges1);
title('Morphological operation on Edges of the bones ');

edgeRegs = regionprops(boneEdges1, 'Area', 'PixelIdxList');
AreaList = sort(vertcat(edgeRegs.Area), 'descend');
edgeRegs(~ismember(vertcat(edgeRegs.Area), AreaList(1:2))) = [];
edgeImg = zeros(size(img_filtered, 1), size(img_filtered,2));
edgeImg(vertcat(edgeRegs.PixelIdxList)) = 1;

% Do hough transform on edge image to find angles at which bone
pieces are
% found
% Use max value of Hough transform vs angle to find angles at
which lines
% are oriented.  If there is more than one major angle
contribution there
% will be two peaks detected but only one peak if there is only
one major
% angle contribution (ie peaks here = number of located bones =
Number of
% breaks + 1)
[H,T,R] = hough(edgeImg,'RhoResolution',1,'Theta',-90:2:89.5);
maxHough = max(H, [], 1);
HoughThresh = (max(maxHough) - min(maxHough))/2 + min(maxHough);
```

```
[~, HoughPeaks] =
findpeaks(maxHough,'MINPEAKHEIGHT',HoughThresh,
'MinPeakDistance', MinHoughPeakDistance);

% Plot Hough detection results
figure(6)
plot(T, maxHough);
hold on
plot([min(T) max(T)], [HoughThresh, HoughThresh], 'g');
plot(T(HoughPeaks), maxHough(HoughPeaks), 'rx', 'MarkerSize',
12, 'LineWidth', 2);
hold off
xlabel('Theta Value'); ylabel('Max Hough Transform');
legend({'Max Hough Transform', 'Hough Peak Threshold', 'Detected
Peak'});
title('Hough Detection Plot : Max Hough transform vs Theta');




% Locate site of break
if numel(HoughPeaks) > 1;
    BreakStack = zeros(size(img_filtered, 1), size(img_filtered,
2), numel(HoughPeaks));
    % Convolute edge image with line of detected angle from
hough transform
    for m = 1:numel(HoughPeaks);

        boneKernel = strel('line', HoughConvolutionLength,
T(HoughPeaks(m)));
        kern = double(bwmorph(boneKernel.getnhood(), 'dilate',
HoughConvolutionDilate));
        BreakStack(:,:,m) = imfilter(edgeImg, kern).*edgeImg;
        figure()
        imshow(BreakStack(:,:,m));

    end

    % Take difference between convolution images.  Where this
crosses zero
    % (within tolerance) should be where the break is.  Have to
filter out
    % regions elsewhere where the bone simply ends.


    brImg = abs(diff(BreakStack, 1, 3)) <
BreakLineTolerance*max(BreakStack(:)) & edgeImg > 0;
    [BpY, BpX] = find(abs(diff(BreakStack, 1, 3)) <
BreakLineTolerance*max(BreakStack(:)) & edgeImg > 0);
```

```
    brImg = bwmorph(brImg, 'dilate', breakPointDilate);
    figure(9);
    imshow(brImg);
    brReg = regionprops(brImg, 'Area', 'MajorAxisLength',
'MinorAxisLength', ...
        'Orientation', 'Centroid');
    brReg(vertcat(brReg.Area) ~= max(vertcat(brReg.Area))) = [];

    % Calculate bounding ellipse
    brReg.EllipseCoords = zeros(100, 2);
    t = linspace(0, 2*pi, 100);
    brReg.EllipseCoords(:,1) = brReg.Centroid(1) +
brReg.MajorAxisLength/2*cos(t - brReg.Orientation);
    brReg.EllipseCoords(:,2) = brReg.Centroid(2) +
brReg.MinorAxisLength/2*sin(t - brReg.Orientation);

else
    brReg = [];      %% No Fracture points are there

end

% Draw ellipse around break location
figure(10)
imshow(img)
hold on
colormap('gray')
if ~isempty(brReg)
    plot(brReg.EllipseCoords(:,1), brReg.EllipseCoords(:,2),
'r');
end
hold off
```

C. Abbreviations and Algorithms Used:

- Thresholding
- Morphological Operations
- K-Means Clustering
- Prewitt and Sobel Operators
- Hough Transform
- Canny edge detection
- Support Vector Machines(SVM)
- Convolutional Neural Networks(CNN)
- Recurrent Neural Networks(RNN)