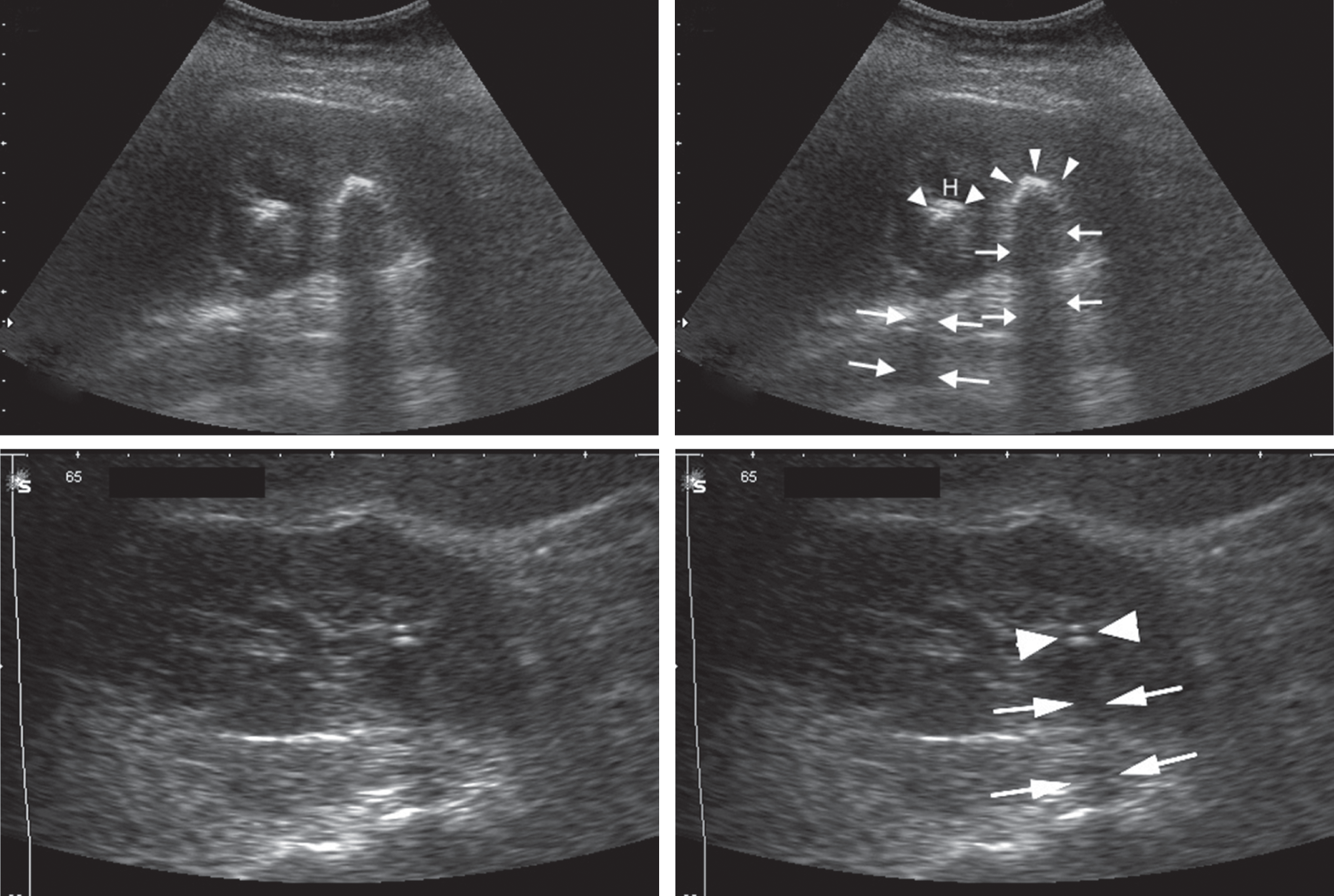
Kidney Stone Detection

Using Image Processing Techniques



**Arnab Sinha - 17ucs034**

**Anshul Kiyawat - 17ucs030**

**Shubhi Rustagi - 17ucs158**

**K Parkavi - 17dcs004**

**Aniket Poddar - 17ucs022**

# INTRODUCTION

There are various Kidney abnormalities such as the formation of stones, cysts, blockage of urine, congenital anomalies, and cancerous cells. Among these, kidney stone disease is when a solid piece of material occurs in the urinary tract. The small piece of stone may pass without causing symptoms. If a stone grows to more than 5 millimetres it can cause blockage of the ureter resulting in severe pain in the lower back or abdomen. Hence it's necessary to have an approach to detect the stone in the kidney to avoid further health issues.

The main objective of this project is to detect the kidney stone from a digital ultrasound image of the kidney by performing various image processing techniques. But, the image produced by the ultrasound techniques is not suitable for further processing due to low contrast and the presence of speckle noise. Hence, the study also examined the effectiveness of various denoising techniques on the ultrasound image to enhance the quality of the image.Further, the enhanced ultrasound image is used to locate the exact position of the stone.

The main motive of this project was to develop an elementary and straightforward technique to find the stone in the kidney. This detection can be done in any available PC’s and hence any normal being can check an ultrasound for a kidney stone and dissolve it in the starting stage. These techniques mainly help the doctor to further treat the patient based on the size and location of the stone.

# 

# APPROACH

To detect the kidney stone, we used the following approach:

The kidney stone produces an acoustic shadow in the ultrasound image. An acoustic shadow or sound shadow is an area through which the sound waves fail to propagate due to topographical obstructions. In this case, the kidney stone is the barrier that disrupts the sound waves. Once we detect the shadow, the stone can be located just above it in the image.

The ultrasound images contain speckle noise. To process the image and detect the location of the stone in the image, we need to remove the noise. Thus, the first step is to enhance the image by using various sharpening and smoothing filters.

After the image is enhanced, image segmentation is used to differentiate between the shadow and the stone by separating the foreground and the background

# WORK PERFORMED

The data were collected from various sources some were online, whose link is given below and also from some close contacts. After collecting the data, we proceeded towards our working.

The first step was to get a clear and better quality of the data so we worked on the **IMAGE ENHANCEMENT**:

1. The first one we applied was Frequency Domain + Gaussian Filter. Here we had to divide the image into two plains real plain and complex plain and then apply the Gaussian Filter in both.

2. The second attempt was with Gaussian Blur + Laplacian Filter. We had to apply Gaussian Blur to make Laplacian filter less sensitive to noise.

3. On further studies, we found in some research papers (link given below) that Ultrasounds give speckle noise. So, our third attempt was to replace Gaussian Blur with Median Blur as the latter is more effective for low-level noises like speckle noise, salt-and-pepper noise etc. followed by Laplacian filter.

4. In the application of the Gabor filter, the restored image is enriched with optimal resolution in both spatial and frequency domains (as stated in one of the research papers, whose link is given below). 2-D Gabor filter is easier to tune the direction and radial frequency bandwidth, and easier to tune centre frequency, so they can simultaneously get the best resolution in the spatial domain and frequency domain.

Now we moved to our next step **HISTOGRAM EQUALIZATION**.

It is a technique for adjusting image intensities to enhance contrast. It assigns the intensity values of pixels in the input image such that the output image contains a uniform distribution of intensities. It improves contrast and obtains a uniform histogram. This technique can be used on a whole image or just on a part of an image. This process leads to an increase in contrast of the shadow of the stone and the stone itself. Thus, it became more visible.

After this, we proceeded towards **IMAGE SEGMENTATION**.

It partitions an image into distinct regions containing each pixel with similar attributes. To be meaningful and useful for image analysis and interpretation, the regions should strongly relate to depicted objects or features of interest. Meaningful segmentation is the first step from low-level image processing transforming a greyscale or colour image into one or more other images to high-level image description in terms of features, objects, and scenes. Hereby performing this, we needed to partition our stone from the rest of the image. The type of Image Segmentation used here is Watersheds. Here we divide the image into peaks (high intensity) and valleys (low intensity). We fill the valleys (points of minima or background) with water of different colours (labels). As the water rises, depending on the peaks(gradients) nearby, water from different valleys, obviously with different colours will start to merge. To avoid this, we need to build barriers in the area where the water is mixing. Also, we continue the process of filling water until all the peaks are under water. The barriers, thus, created are the result of image segmentation. Now, the shadow gets separated from the stone giving a clear image of the stone.

The last step being **MARKING**.

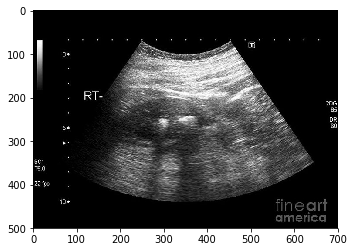
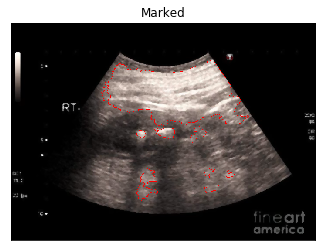
It provides you with a way to identify or label a spot within the processing pipeline. Adding a marker into the pipeline will allow you to refer to the image currently being processed at that point. It can also be used to specify a location in the pipeline for other modules.

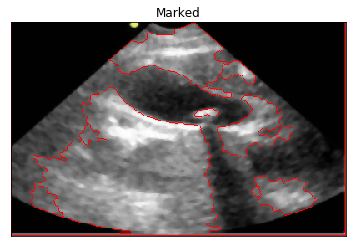
# RESULTS

Total of 4 approaches have been performed to enhance the quality of the image, namely: Gabor Filter, Median blur + Laplacian Filter, Gaussian blur + Laplacian Filter, Gaussian Filtering in Frequency domain. Out of the 4, we have noticed that Gabor Filter was effective on a larger number of images though not all. Nextly Median blur + Laplacian Filter and Gaussian blur + Laplacian Filter worked perfectly with other of the few images which are lesser compared to Gabor Filter. Lastly, Gaussian Filtering in Frequency domain, which didn’t work as expected and hence this technique was not employed to enhance any of the other images. We also could generalize the effectiveness of the techniques due to lack of data samples.

Resulting images of ultrasound are given below:

Original image Image with detection

# 

# 

# DISCUSSION

The effectiveness of various enhancement techniques

Out of the 4 approaches, we conclude that Gabor filter is more effective because it is a [linear filter](https://en.wikipedia.org/wiki/Linear_filter) used for [texture](https://en.wikipedia.org/wiki/Texture_mapping) analysis, which means that it basically analyzes whether there is any specific frequency content in the image in specific directions in a localized region around the point or region of analysis. Hence due to this, to locate a stone (i.e. we have to locate a region above the shadow formed by the stone) Gabor filter works more efficiently. Whereas in the case of median blur followed by Laplacian filter, it’s a nonlinear digital filtering technique, often used to remove noise from an image or signal with retaining the edges. Hence it’s an effective procedure when there are particulate noises such as salt and pepper. Nextly Gaussian blur followed by Laplacian filter is used to typically to reduce [image noise](https://en.wikipedia.org/wiki/Image_noise) and reduce detail. As details are also lost hence this technique doesn’t give a clear image so it’s followed by a Laplacian filter to increase the edges. Lastly, Gaussian in frequency domain failed because the real and imaginary plane were not separated and hence the inverse transformation did not take place properly.

Summary and Conclusion

We had an image of an ultrasound of a kidney containing a stone. We applied the Gabor Filter for the Image Enhancement followed by Histogram Equalization. The restored image went under Image Segmentation, namely, Watersheds after which Marking was done and the final image was produced. The final image showed a distinct location of the stone in the kidney. Hence the stone was detected.

# 

# WORKLOAD DISTRIBUTION

|  |  |
| --- | --- |
| TASK | TEAM MEMBER |
| Research | Arnab Sinha (35%)  Anshul Kiyawat (35%)  Aniket Poddar (10%)  Shubhi Rustagi (10%)  K. Parkavi (10%) |
| Coding | Arnab Sinha (25%)  Anshul Kiyawat (25%)  K. Parkavi (25%)  Shubhi Rustagi (25%) |
| Project Report | K. Parkavi (35%)  Aniket Poddar (35%)  Shubhi Rustagi (30%) |
| Collection of Dataset | Arnab Sinha (50%)  Anshul Kiyawat (50%) |
| PowerPoint Presentation | Arnab Sinha (20%)  Anshul Kiyawat (20%)  Aniket Poddar (20%)  Shubhi Rustagi (20%)  K. Parkavi (20%) |

# 

# 

# 

# REFERENCES

1. <http://imgsimon.blogspot.com/2016/05/python-image-filtering-in-python-and.html>
2. <https://github.com/tvganesh/weiner>
3. <https://www.sciencedirect.com/science/article/pii/S0169260717300639>
4. <http://matlabserver.cs.rug.nl/edgedetectionweb/web/edgedetection_params.html>
5. <http://tudr.thapar.edu:8080/jspui/bitstream/10266/969/1/Abhay_Thesis.pdf>
6. <https://www.ijser.org/researchpaper/Enhancement-of-Ultrasound-Images-Using-RADWT.pdf>
7. <https://www.hindawi.com/journals/cmmm/2014/758439/fig1/>
8. <https://www.hindawi.com/journals/cmmm/2014/758439/>
9. <https://wiseodd.github.io/techblog/2016/11/05/levelset-method/>
10. <https://medium.com/nanonets/how-to-use-deep-learning-when-you-have-limited-data-part-2-data-augmentation-c26971dc8ced>
11. <https://ieeexplore.ieee.org/document/6572601>
12. <https://onlinelibrary.wiley.com/doi/pdf/10.7863/jum.1984.3.3.123>
13. [https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=802756https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=802756https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=802756](https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=802756)
14. <https://www.youtube.com/watch?v=7Nxpa0cjOBE>
15. <https://www.hindawi.com/journals/vlsi/2015/581961/>
16. <https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_watershed/py_watershed.html>
17. <http://www.allsubjectjournal.com/download/2234/3-5-109-311.pdf>

# SOURCE CODE

import cv2

#from skimage.measure import compare\_ssim

import argparse

#import imutils

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

s = r'C:\Users\Arnab Sinha\Documents\GitHub\Kidney-Stone-Detection-IP\images'

image\_no = '\image2.jpg'

s = s + image\_no

img = cv2.imread(s,0)

def build\_filters():

#returns a list of kernels in several orientations

filters = []

ksize = 31

for theta in np.arange(0, np.pi, np.pi / 32):

params = {'ksize': (ksize, ksize), 'sigma': 0.0225, 'theta': theta, 'lambd': 15.0,

'gamma': 0.01, 'psi': 0, 'ktype': cv2.CV\_32F}

kern = cv2.getGaborKernel(\*\*params)

kern /= 1.5\*kern.sum()

filters.append((kern, params))

return filters

def process(img, filters):

#returns the img filtered by the filter list

accum = np.zeros\_like(img)

for kern, params in filters:

fimg = cv2.filter2D(img, cv2.CV\_8UC3, kern)

np.maximum(accum, fimg, accum)

return accum

def Histeq(img):

equ = cv2.equalizeHist(img)

return equ

def GaborFilter(img):

filters = build\_filters()

p = process(img, filters)

return p

def Laplacian(img,par):

lap = cv2.Laplacian(img,cv2.CV\_64F)

sharp = img - par\*lap

sharp = np.uint8(cv2.normalize(sharp, None, 0 , 255, cv2.NORM\_MINMAX))

return sharp

def Watershed(img):

ret, thresh = cv2.threshold(img,0,255,cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)

# noise removal

kernel = np.ones((3,3),np.uint8)

opening = cv2.morphologyEx(thresh,cv2.MORPH\_OPEN,kernel, iterations = 2)

# sure background area

sure\_bg = cv2.dilate(opening,kernel,iterations=3)

# Finding sure foreground area

dist\_transform = cv2.distanceTransform(opening,cv2.DIST\_L2,5)

ret, sure\_fg = cv2.threshold(dist\_transform,0.14\*dist\_transform.max(),255,0)

# Finding unknown region

sure\_fg = np.uint8(sure\_fg)

unknown = cv2.subtract(sure\_bg,sure\_fg)

# Marker labelling

ret, markers = cv2.connectedComponents(sure\_fg)

# Add one to all labels so that sure background is not 0, but 1

markers = markers+1

# Now, mark the region of unknown with zero

markers[unknown==255] = 0

img2 = cv2.imread(s,1)

img2 = cv2.medianBlur(img2,5)

markers = cv2.watershed(img2,markers)

img2[markers == -1] = [255,0,0]

return img2

if image\_no=='\image1.jpg':

img3 = Laplacian(img,0.239)

elif image\_no=='\image2.jpg':

img3 = GaborFilter(img)

img3 = Histeq(img3)

elif image\_no=='\image4.jpg':

img3 = GaborFilter(img)

img3 = Watershed(img)

plt.imshow(img3,'gray')

plt.title('Marked')

plt.xticks([]),plt.yticks([])