FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERIG
Department of Computer Engineering

Experiment 3 - Bio Medial Image Pre processing and Segmentation

1. Course Details:

Academic Year	**2023 - 24**	**Estimated Time**	**Experiment No. 1 – 02 Hours**
Course & Semester	**B.E. (COMP) – Sem. VII**	**Subject Name**	**Data Science for Health and Social Care Lab**
Experiment Type	**Software Performance**	**Subject Code**	**HDSSBL701**

Name of Student	Atharva Pawar	**Roll No.**	9427
Date of Performance.:		**Date of Submission.:**	
CO Mapping	**HDSSBL701.3 - Demonstrate statistical data analysis, Image preprocessing, Segmentation and visualization techniques on healthcare data.**		

Aim: To perform Bio Medial Image Pre processing and Segmentation.
Objective: To explore, visualize and analyze Medical Image dataset to gain insights and understand the characteristics of data.

Tools :
Jupyter Notebook/ MATLAB

Libraries:
OpenCV, Numpy, Pandas, Tensor Flow

Dataset:
MRI, X-ray or CT scan datasets containing labeled images of various medical conditions (e.g., fractures, tumors, pneumonia, etc.). Datasets can be obtained from publicly available sources

Theory:

Image preprocessing is a vital step when working with image data.Medical Image preprocessing is used to improve the quality of medical images, making it easier to detect diseases or abnormalities. Some powerful image preprocessing techniques include noise reduction, contrast enhancement, image resizing, color correction, segmentation, feature extraction, etc. It is an essential step in image analysis that helps enhance the data in images and reduce clutter.

Techniques for Image Preprocessing

The choice of techniques depends on the nature of the image and the application. Here are a few techniques to improve image quality and suitability:

Noise Reduction: Noise in an image can be caused by various factors such as low light, sensor noise, and compression artifacts. Noise reduction techniques aim to remove noise from the image while preserving its essential features. Some common noise reduction techniques include Gaussian smoothing, median filtering, and wavelet denoising.

Contrast Enhancement: Contrast enhancement techniques aim to increase the contrast of an image, making it easier to distinguish between different image features. These techniques can be helpful in applications such as medical imaging and surveillance. Some standard contrast enhancement techniques include histogram equalization, adaptive histogram equalization, and contrast stretching.

Image Resizing: Image resizing techniques are used to adjust the size of an image. Resizing can be done to make an image smaller or larger or to change its aspect ratio. Some typical image resizing techniques include nearest neighbor interpolation, bilinear interpolation, and bicubic interpolation.

Color Correction: Color correction techniques are used to adjust the color balance of an image. Color correction is important in applications such as photography, where the color accuracy of an image is critical. Some common color correction techniques include gray world assumption, white balance, and color transfer.

Segmentation: Segmentation techniques are used to divide an image into regions based on its content. Segmentation can be helpful in applications such as medical imaging, where specific structures or organs must be isolated from the image. Some standard segmentation techniques include thresholding, edge detection, and region growing.

Feature Extraction: Feature extraction techniques are used to identify and extract relevant features from an image. These features can be used in object recognition and image classification applications. Some standard feature extraction techniques include edge detection, corner detection, and texture analysis.

Procedure:

1. Download any Sample image
2. Convert sample image to gray scale image
3. Apply Thresholding to see the changes (can use any thresholding method such as Binary thresholding or OTSU thresholding)
4. Apply median and gaussian filters for Noise reduction
5. Plot all images and their Histogram
6. Detection of edge

Results:

Advanced Learner Activity:

Study and perform Semantic Segmentation & Instance Segmentation

jupyter DSHC_EXP_3 Last Checkpoint: a few seconds ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Run Code

DSHC - EXP - 3

Demonstrate statistical data analysis, Image preprocessing, Segmentation and visualization techniques on healthcare data.bold text

In [12]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

In [13]:
```
# Step 1: Read the image
image_path = 'ID_0077_AGE_0074_CONTRAST_0_CT.png'
# image_path = 'ID_0078_AGE_0066_CONTRAST_0_CT.png'
# image_path = 'ID_0079_AGE_0071_CONTRAST_0_CT.png'
image = cv2.imread(image_path, cv2.IMREAD_COLOR)
```

In [15]:
```
# Step 2: Convert to grayscale
if image is None:
    print("Error: Image not loaded.")
else:
    gray_image = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    print("Success: Image loaded.")
```

```
Success: Image loaded.
```

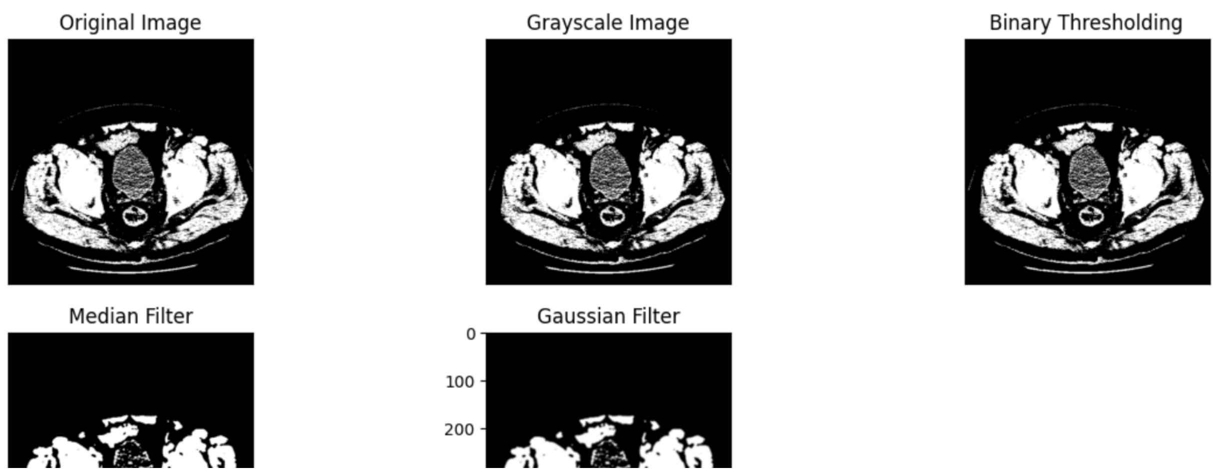
In [16]:
```
# Step 3: Apply Thresholding (Binary thresholding)
ret, binary_image = cv2.threshold(gray_image, 127, 255, cv2.THRESH_BINARY)
```

In [17]:
```
# Step 4: Apply Median Filter for Noise Reduction
median_filtered_image = cv2.medianBlur(binary_image, 5)
```

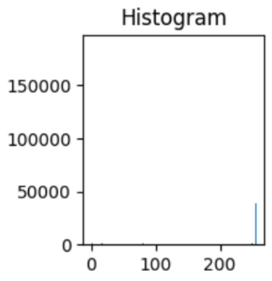
In [18]:
```
# Step 4: Apply Gaussian Filter for Noise Reduction
gaussian_filtered_image = cv2.GaussianBlur(median_filtered_image, (5, 5), 0)
```

In [19]:
```
# Step 5: Plot the Images and their Histograms
plt.figure(figsize=(15, 6))
plt.subplot(2, 3, 1), plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB), cmap='gray')
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(2, 3, 2), plt.imshow(gray_image, cmap='gray')
plt.title('Grayscale Image'), plt.xticks([]), plt.yticks([])
plt.subplot(2, 3, 3), plt.imshow(binary_image, cmap='gray')
plt.title('Binary Thresholding'), plt.xticks([]), plt.yticks([])
plt.subplot(2, 3, 4), plt.imshow(median_filtered_image, cmap='gray')
plt.title('Median Filter'), plt.xticks([]), plt.yticks([])
plt.subplot(2, 3, 5), plt.imshow(gaussian_filtered_image, cmap='gray')
plt.title('Gaussian Filter'), plt.xticks([]), plt.yticks()
```

Out[19]:
```
(Text(0.5, 1.0, 'Gaussian Filter'),
 ([], []),
 (array([-100.,    0.,  100.,  200.,  300.,  400.,  500.,  600.]),
  [Text(0, -100.0, '-100'),
   Text(0, 0.0, '0'),
   Text(0, 100.0, '100'),
   Text(0, 200.0, '200'),
   Text(0, 300.0, '300'),
   Text(0, 400.0, '400'),
   Text(0, 500.0, '500'),
   Text(0, 600.0, '600')]))
```



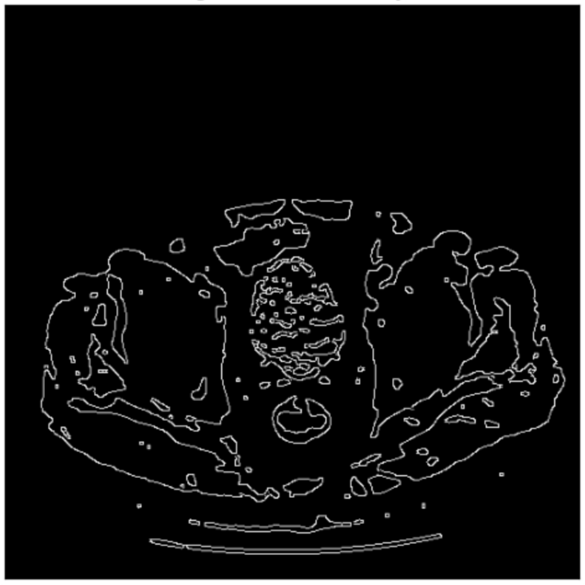
In [20]:
```python
# Plot histograms
plt.subplot(2, 3, 6), plt.hist(gaussian_filtered_image.ravel(), 256, [0, 256])
plt.title('Histogram')
plt.show()
```



In [21]:
```python
# Step 6: Detection of Edges (Canny Edge Detection)
edges = cv2.Canny(gaussian_filtered_image, 100, 200)
```

In [22]:
```python
# Display the edges
plt.figure(figsize=(6, 6))
plt.imshow(edges, cmap='gray')
plt.title('Edge Detection (Canny)')
plt.xticks([]), plt.yticks([])
plt.show()
```

Edge Detection (Canny)



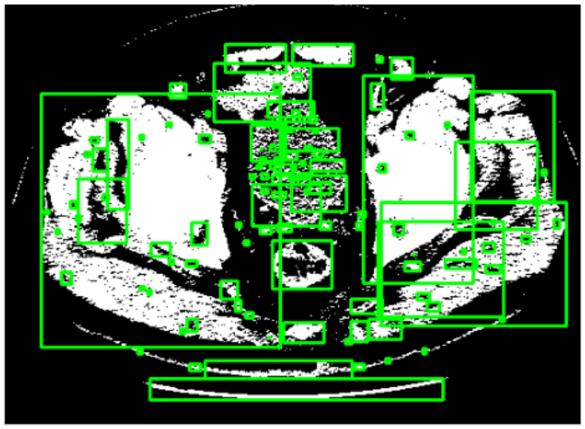
In [23]:
```python
# Find contours in the edge image
contours, _ = cv2.findContours(edges, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)

# Create a copy of the original image to draw bounding boxes on
image_with_boxes = image.copy()

# Iterate through the contours and draw bounding boxes
for contour in contours:
    x, y, w, h = cv2.boundingRect(contour)
    cv2.rectangle(image_with_boxes, (x, y), (x + w, y + h), (0, 255, 0), 2)  # Green rectangle

# Display the image with bounding boxes
plt.figure(figsize=(6, 6))
plt.imshow(cv2.cvtColor(image_with_boxes, cv2.COLOR_BGR2RGB))
plt.title('Image with Bounding Boxes')
plt.xticks([]), plt.yticks([])
plt.show()
```

Image with Bounding Boxes



```
In [ ]:
```