In [1]:
```
import pydicom
import numpy as np
import cv2
import os
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

from scipy import ndimage
from skimage import morphology
```

Load and Windowing Image

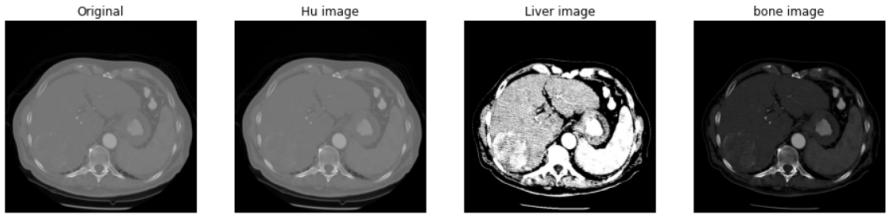
In [2]:
```
def transform_to_hu(medical_image, image):
    intercept = medical_image.RescaleIntercept
    slope = medical_image.RescaleSlope
    hu_image = image * slope + intercept

    return hu_image
```

In [3]:
```
def window_image(image, window_center, window_width):
    img_min = window_center - window_width // 2
    img_max = window_center + window_width // 2
    window_image = image.copy()
    window_image[window_image < img_min] = img_min
    window_image[window_image > img_max] = img_max

    return window_image
```

In [4]:
```
def load_and_plot_image(file_path, save=False):
    medical_image = pydicom.read_file(file_path)
    image = medical_image.pixel_array

    print(image.shape)

    hu_image = transform_to_hu(medical_image, image)
    liver_image = window_image(hu_image, 40, 80)
    bone_image = window_image(hu_image, 400, 1000)

    plt.figure(figsize=(20, 10))
    plt.style.use('grayscale')

    plt.subplot(151)
    plt.imshow(image)
    plt.title('Original')
    plt.axis('off')

    plt.subplot(152)
    plt.imshow(hu_image)
    plt.title('Hu image')
    plt.axis('off')

    plt.subplot(153)
    plt.imshow(liver_image)
    plt.title('Liver image')
    plt.axis('off')

    plt.subplot(154)
    plt.imshow(bone_image)
    plt.title('bone image')
    plt.axis('off')

    if save:
        mpimg.imsave(os.path.join(output_path, f'{file_path[:-4]}-original.png'), image)
        mpimg.imsave(os.path.join(output_path, f'{file_path[:-4]}-hu_image.png'), hu_image)
        mpimg.imsave(os.path.join(output_path, f'{file_path[:-4]}-liver_image.png'), liver_image)
        mpimg.imsave(os.path.join(output_path, f'{file_path[:-4]}-bone_image.png'), bone_image)
```

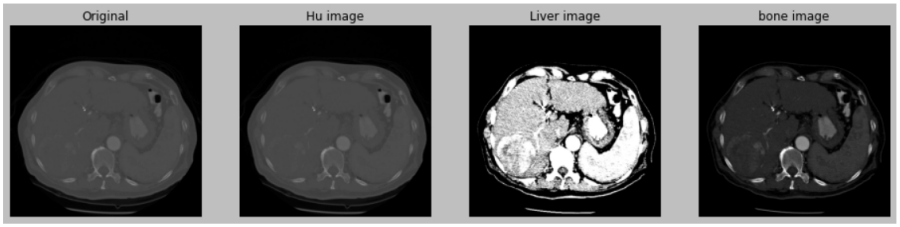
In [5]:
```
load_and_plot_image("Y:/Liver_dataset/liver-alldatasets/TCA DATA/1-35.dcm")
```

```
(512, 512)
```



In [6]:
```
load_and_plot_image("Y:/Liver_dataset/liver-alldatasets/TCA DATA/1-38.dcm")
```

```
(512, 512)
```

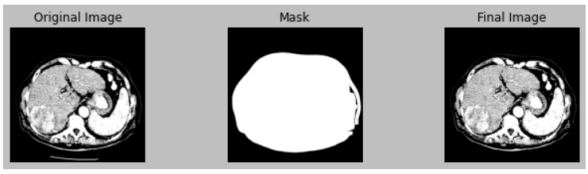


Improving Images

In [7]:
```
def remove_noise(file_path, display=False):
    medical_image = pydicom.read_file(file_path)
    image = medical_image.pixel_array

    hu_image = transform_to_hu(medical_image, image)
    liver_image = window_image(hu_image, 40, 80)

    # morphology.dilation creates a segmentation of the image
    # If one pixel is between the origin and the edge of a square of size
    # 5x5, the pixel belongs to the same class

    # We can instead use a circule using: morphology.disk(2)
    # In this case the pixel belongs to the same class if it's between the origin
    # and the radius

    segmentation = morphology.dilation(liver_image, np.ones((5, 5)))
    labels, label_nb = ndimage.label(segmentation)

    label_count = np.bincount(labels.ravel().astype(np.int))
    # The size of label_count is the number of classes/segmentations found

    # We don't use the first class since it's the background
    label_count[0] = 0

    # We create a mask with the class with more pixels
    # In this case should be the liver
    mask = labels == label_count.argmax()

    # Improve the liver mask
    mask = morphology.dilation(mask, np.ones((5, 5)))
    mask = ndimage.morphology.binary_fill_holes(mask)
    mask = morphology.dilation(mask, np.ones((3, 3)))

    # Since the the pixels in the mask are zero's and one's
    # We can multiple the original image to only keep the liver region
    masked_image = mask * liver_image

    if display:
        plt.figure(figsize=(15, 2.5))
        plt.subplot(141)
        plt.imshow(liver_image)
        plt.title('Original Image')
        plt.axis('off')

        plt.subplot(142)
        plt.imshow(mask)
        plt.title('Mask')
        plt.axis('off')

        plt.subplot(143)
        plt.imshow(masked_image)
        plt.title('Final Image')
        plt.axis('off')

    return masked_image
```

In [8]:
```
_ = remove_noise("Y:/Liver_dataset/liver-alldatasets/TCA DATA/1-35.dcm", display=True)
```

```
C:\Users\asus\AppData\Local\Temp/ipykernel_9764/753963014.py:19: DeprecationWarning: `np.int` is a deprecated alias for the bui
ltin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is safe. When replacing `
np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, ch
eck the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  label_count = np.bincount(labels.ravel().astype(np.int))
```



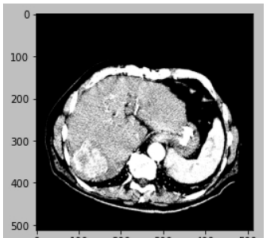
In [9]:
```
def crop_image(image, display=False):
    # Create a mask with the background pixels
    mask = image == 0

    # Find the liver area
    coords = np.array(np.nonzero(~mask))
    top_left = np.min(coords, axis=1)
    bottom_right = np.max(coords, axis=1)

    # Remove the background
    croped_image = image[top_left[0]:bottom_right[0],
                top_left[1]:bottom_right[1]]

    return croped_image
```

In [10]:
```
final_image = remove_noise("Y:/Liver_dataset/liver-alldatasets/TCA DATA/1-31.dcm")
```

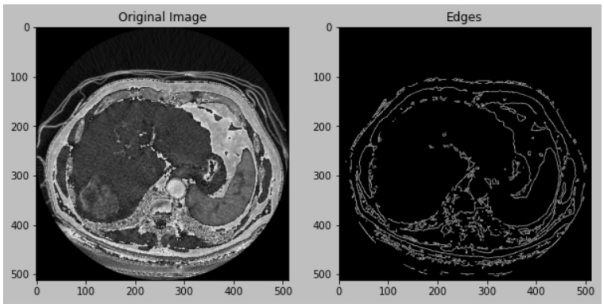
```
C:\Users\asus\AppData\Local\Temp/ipykernel_9764/753963014.py:19: DeprecationWarning: `np.int` is a deprecated alias for the bui
ltin `int`. To silence this warning, use `int` by itself. Doing this will not modify any behavior and is safe. When replacing `
np.int`, you may wish to use e.g. `np.int64` or `np.int32` to specify the precision. If you wish to review your current use, ch
eck the release note link for additional information.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations
  label_count = np.bincount(labels.ravel().astype(np.int))
```

In [11]:
```
plt.imshow(final_image)
```

Out[11]:
```
<matplotlib.image.AxesImage at 0x27df9e33580>
```

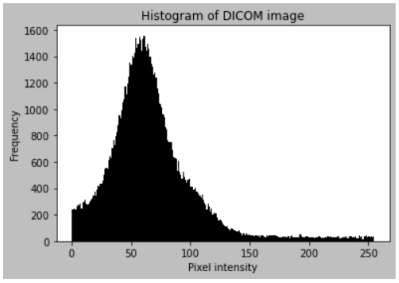


In [12]:
```
import pydicom
import cv2
import matplotlib.pyplot as plt

# read the DICOM image
dcm = pydicom.dcmread('Y:/Liver_dataset/liver-alldatasets/TCA DATA/1-31.dcm')

# get the pixel array
image = dcm.pixel_array.astype('uint8')

# apply Canny edge detection
edges = cv2.Canny(image, 1000, 500)

# display the original and edge images side by side
fig, ax = plt.subplots(1, 2, figsize=(10, 5))
ax[0].imshow(image, cmap='gray')
ax[0].set_title('Original Image')
ax[1].imshow(edges, cmap='gray')
ax[1].set_title('Edges')
plt.show()
```



In [13]:
```
#https://www.youtube.com/watch?v=cToG83MLkqw
```

In [14]:
```
import pydicom
import matplotlib.pyplot as plt

# Load the DICOM image
ds = pydicom.dcmread('Y:/Liver_dataset/liver-alldatasets/TCA DATA/1-31.dcm')

# Get the pixel data and flatten it
pixel_data = ds.pixel_array.flatten()

# Plot the histogram
plt.hist(pixel_data, bins=256, range=(0, 255))
plt.title("Histogram of DICOM image")
plt.xlabel("Pixel intensity")
plt.ylabel("Frequency")
plt.show()
```



In [15]:
```
import pydicom
import numpy as np

# Load the DICOM image
ds = pydicom.dcmread('Y:/Liver_dataset/liver-alldatasets/TCA DATA/1-31.dcm')

# Get the pixel data and convert to Hounsfield units
pixels = ds.pixel_array
slope = ds.RescaleSlope
intercept = ds.RescaleIntercept
hu_pixels = (pixels * slope) + intercept

# Calculate the mean value in Hounsfield units
mean_hu = np.mean(hu_pixels)

print('Mean value in Hounsfield units:', mean_hu)
```

```
Mean value in Hounsfield units: -428.73548126220703
```

```
import pydicom
import numpy as np
import matplotlib.pyplot as plt

# load the DICOM file
dcm_file = pydicom.dcmread('')

# extract the image data
image = dcm_file.pixel_array

# select the ROI using slicing

roi = image[155:200, 100:200]


#ax2.add_patch(plt.Rectangle((roi_x, roi_y), roi_width, roi_height, edgecolor='r', fill=False))

# display the ROI and the whole image side by side
```

```
fig, ax = plt.subplots(1, 2, figsize=(10, 5))
ax[0].imshow(image, cmap='gray')
ax[0].set_title('Whole Image')
ax[1].imshow(roi, cmap='gray')
ax[1].set_title('ROI')
plt.show()
```

```
import pydicom
import numpy as np
import matplotlib.pyplot as plt

# load the DICOM file
dcm_file = pydicom.dcmread('Y:/Liver_dataset/liver-alldatasets/TCA DATA/1-36.dcm')

# extract the image data
image = dcm_file.pixel_array

# select the ROI using slicing
roi = image[220:300, 100:200]

# display the whole image
fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(image, cmap='gray')
ax.set_title('Whole Image')

# draw a rectangle around the ROI in the whole image
rect = plt.Rectangle((100, 220), 100, 100, linewidth=1, edgecolor='r', facecolor='none')
ax.add_patch(rect)

# display the selected ROI
fig, ax = plt.subplots(figsize=(5, 5))
ax.imshow(roi, cmap='gray')
ax.set_title('ROI')

plt.show()


#Get the pixel data and convert to Hounsfield units
# Extract the ROI pixel values
roi_pixels = dcm_file.pixel_array[200:300, 100:200]

# Convert the ROI pixel values to Hounsfield units
slope1 = dcm_file.RescaleSlope
intercept1 = dcm_file.RescaleIntercept
roi_hu = roi_pixels * slope1 + intercept1

# Calculate the mean Hounsfield unit value of the ROI
roi_mean_hu = np.mean(roi_hu)

print("Mean Hounsfield unit value of ROI: ", roi_mean_hu)
#print('Mean value in Hounsfield units:', mean_hu)
```

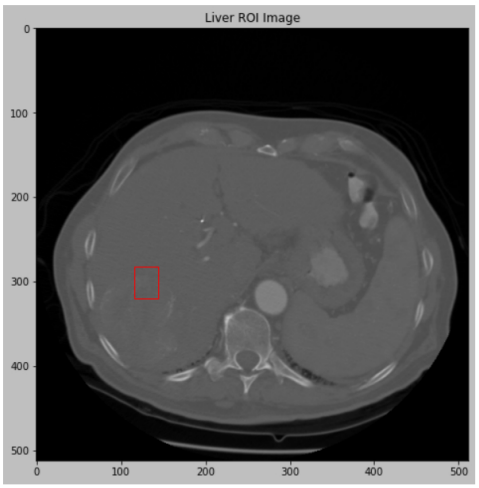
```
#http://www.radtechonduty.com/2017/03/ct-scan-of-liver-and-enhancing-phases.html
https://www.sciencedirect.com/topics/engineering/grayscale-
level#:~:text=In%20a%20CT%20image%2C%20each,beam%20attenuation%20to%20the%20tissue.
```

```
#'Y:/Liver_dataset/liver-alldatasets/TCA DATA/1-36.dcm'
import pydicom
import matplotlib.pyplot as plt
import numpy as np

# load the DICOM image
ds = pydicom.dcmread('Y:/Liver_dataset/liver-alldatasets/TCA DATA/1-36.dcm')

# extract the image array and metadata
image = ds.pixel_array
#metadata = ds.pixel_array

# define the ROI coordinates
x1, y1, x2, y2 = [180, 190, 160, 170]  # example coordinates

# extract the ROI from the image
roi = image[y1:y2, x1:x2]

# calculate the mean of the ROI in Hounsfield units
slope = ds.RescaleSlope
intercept = ds.RescaleIntercept
mean_hu = np.mean(roi) * slope + intercept

# display the image with the ROI
fig, ax = plt.subplots()
ax.imshow(image, cmap='gray')
ax.add_patch(plt.Rectangle((x1, y1), x2 - x1, y2 - y1, linewidth=1, edgecolor='r', facecolor='none'))
plt.show()

print('Mean Hounsfield Units in ROI: {:.2f}'.format(mean_hu))
```

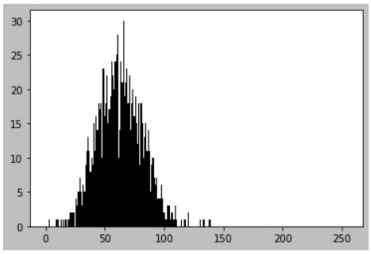
In [21]:
```
#FOR LIVER
import pydicom
import numpy as np
import matplotlib.pyplot as plt

# load the DICOM file
dcm_file = pydicom.dcmread('Y:/Liver_dataset/liver-alldatasets/TCA DATA/1-36.dcm')

# extract the image data
image = dcm_file.pixel_array

#define the ROI coordinates
x1, y1, x2, y2 = [116,282, 144,320]
#116,282
#100,200, 200,300 == correct
#180, 190, 160, 170]  # example coordinates
#200:300, 100:200
#y1:y2, x1:x2

# extract the ROI from the image
roi = image[y1:y2, x1:x2]

# display the whole image
fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(image, cmap='gray')
ax.set_title('Liver ROI Image')

# draw a rectangle around the ROI in the whole image
rect = plt.Rectangle((x1, y1), x2 - x1, y2 - y1, linewidth=1, edgecolor='r', facecolor='none')
```

```
#ax.add_patch(plt.Rectangle((x1, y1), x2 - x1, y2 - y1, linewidth=1, edgecolor='r', facecolor='none'))
ax.add_patch(rect)

#Get the pixel data and convert to Hounsfield units
# Extract the ROI pixel values
roi_pixels = dcm_file.pixel_array[y1:y2, x1:x2]
# Convert the ROI pixel values to Hounsfield units
slope1 = dcm_file.RescaleSlope
intercept1 = dcm_file.RescaleIntercept
roi_hu = roi_pixels * slope1 + intercept1

# Calculate the mean Hounsfield unit value of the ROI
roi_mean_hu = np.mean(roi_hu)

print("Mean Hounsfield unit value of ROI: ", roi_mean_hu)
#print('Mean value in Hounsfield units:', mean_hu)
```

```
Mean Hounsfield unit value of ROI:  74.77913533834587
```



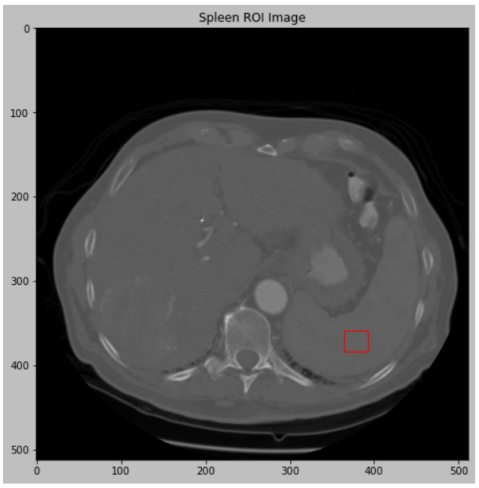
In [17]:
```
import pydicom
import matplotlib.pyplot as plt

# Load the DICOM image
ds = pydicom.dcmread('Y:/Liver_dataset/liver-alldatasets/TCA DATA/1-31.dcm')
x1, y1, x2, y2 = [116,282, 144,320]
# Get the pixel data and flatten it
roi_data = ds.pixel_array[y1:y2, x1:x2]

# Calculate the histogram
hist, bins = np.histogram(roi_data, bins=256, range=(0, 255))

# Plot the histogram
plt.bar(bins[:-1], hist, width=1)
plt.show()
```



In [20]:
```
#FOR SPLEEN
import pydicom
import numpy as np
import matplotlib.pyplot as plt

# load the DICOM file
dcm_file = pydicom.dcmread('Y:/Liver_dataset/liver-alldatasets/TCA DATA/1-36.dcm')
#Y:/Liver_dataset/TCIA DATA/manifest-1681666177860/HCC-TACE-Seg/HCC_004/11-28-1997-NA-CHEST ABD LIVER PROTOC-54067/4.000000-Recor
# extract the image data
image = dcm_file.pixel_array

#define the ROI coordinates
x1, y1, x2, y2 = [364,359, 392,383]
#116,282
#100,200, 200,300 == correct
#180, 190, 160, 170]  # example coordinates
#200:300, 100:200
#y1:y2, x1:x2

# extract the ROI from the image
roi = image[y1:y2, x1:x2]

# display the whole image
fig, ax = plt.subplots(figsize=(8, 8))
ax.imshow(image, cmap='gray')
#flag
#twilight
#gist_ncar
ax.set_title('Spleen ROI Image')

# draw a rectangle around the ROI in the whole image
rect = plt.Rectangle((x1, y1), x2 - x1, y2 - y1, linewidth=1, edgecolor='r', facecolor='none')
#ax.add_patch(plt.Rectangle((x1, y1), x2 - x1, y2 - y1, linewidth=1, edgecolor='r', facecolor='none'))
ax.add_patch(rect)

#Get the pixel data and convert to Hounsfield units
# Extract the ROI pixel values
roi_pixels = dcm_file.pixel_array[y1:y2, x1:x2]
# Convert the ROI pixel values to Hounsfield units
```

```
slope1 = dcm_file.RescaleSlope
intercept1 = dcm_file.RescaleIntercept
roi_hu = roi_pixels * slope1 + intercept1

# Calculate the mean Hounsfield unit value of the ROI
roi_mean_hu = np.mean(roi_hu)

print("Mean Hounsfield unit value of ROI: ", roi_mean_hu)
#print('Mean value in Hounsfield units:', mean_hu)
```

```
Mean Hounsfield unit value of ROI:  104.53869047619048
```



In [ ]:

In [ ]: