Midterm for CSE 577 (Allen Tannenbaum)

Due by email Wednesday, March 22, 2023 by 12 noon.

All parameter values in these problems are suggested. You can and should try your own.

1. **(a) Take the image heart.jpg and find a threshold to pull out the two heart chambers. Please give me the threshold and results. See the power point medical.imaging.ppt.**
   (b) Take the image brain.tech.jpg and find a threshold to pull out the tumor. Please give me the threshold and results. See the power point medical.imaging.ppt.

2. **Look up the watershed segmentation method, explain it, and apply to the test images heart.jpg and brain.tech.jpg.**

3. **Apply the Laplacian of Gaussian (LoG) operator to find the edges in heart.jpg and brain.tech.jpg. Try several different values of the variance sigma. For example, sigma=1, 10, 100, 1000.**

4. **Apply histogram equalization to heart.jpg and brain.tech.jpg, and show results.**

5. **Let H be a Gaussian smoothing filter. Let F be the image heart.jpg. Consider the filter**

$$F + \alpha\,(F - F * H) = (1+\alpha)\,F - \alpha\,(F * H) = F * ([1+\alpha]e - H)$$

   What does this filter do? Take α=.5, and σ=10 (in the Gaussian), and apply to heart.jpg. Try a few other values of α and σ, and show the results. The symbol e in the above formula is the delta function.

Each problem is worth 20 points.

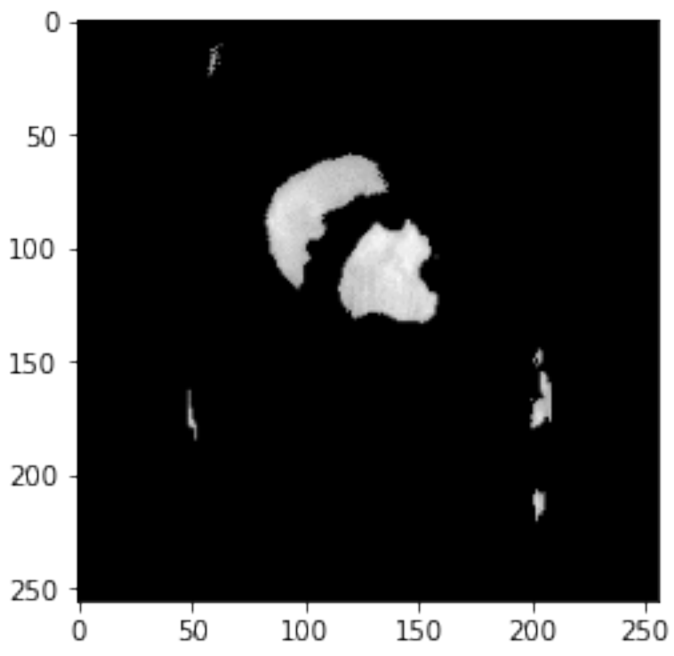
yizjia577mtq1

March 22, 2023

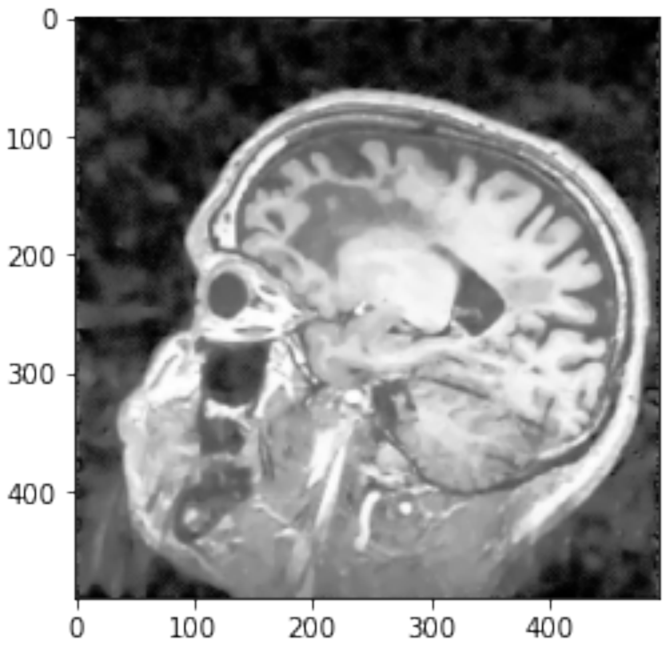
0.0.1 Problem 1: Pull out certain blocks

[32]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

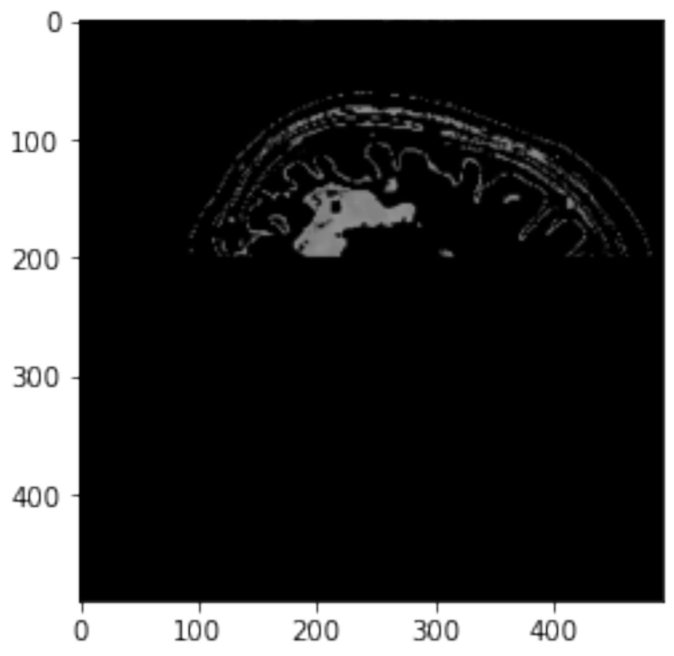
[33]:
```
# Extract the heart chambers from the CT scan image
heart_chambers = cv2.imread('heart.jpg')

# Threshold
heart_chambers[heart_chambers < 160] = 0

plt.imshow(heart_chambers)
plt.show()
```



[34]:
```
brain_tumor_o = cv2.imread("brain.tech.jpg")
brain_tumor = brain_tumor_o[60:550,190:683] # crop image to avoid pale frame
if brain_tumor is None:
    print("No image found!")
    sys.exit()
else:
    plt.imshow(brain_tumor)
    plt.show()
```



[37]:
```
# Threahold
brain_tumor[brain_tumor > 150] = 0
brain_tumor[brain_tumor < 120] = 0
brain_tumor[200:] = 0

plt.imshow(brain_tumor)
plt.show()
```


0
100
200
300
400
0
100
200
300
400

```
[ ]:
```

yizjia577mtq2

March 21, 2023

0.0.1 Problem 2: Watershed Segmentation Method

Watershed segmentation method is an image processing method that identifies the boundaries between objects by treating the image as a topographic map. It floods the image from local minima, defining the different regions based on the peaks reached by the water. The resulting boundaries are called watershed lines, and they define the boundaries between objects in the image. Watershed segmentation is useful for segmenting images with complex overlapping structures, but it can be sensitive to noise and may require post-processing techniques.

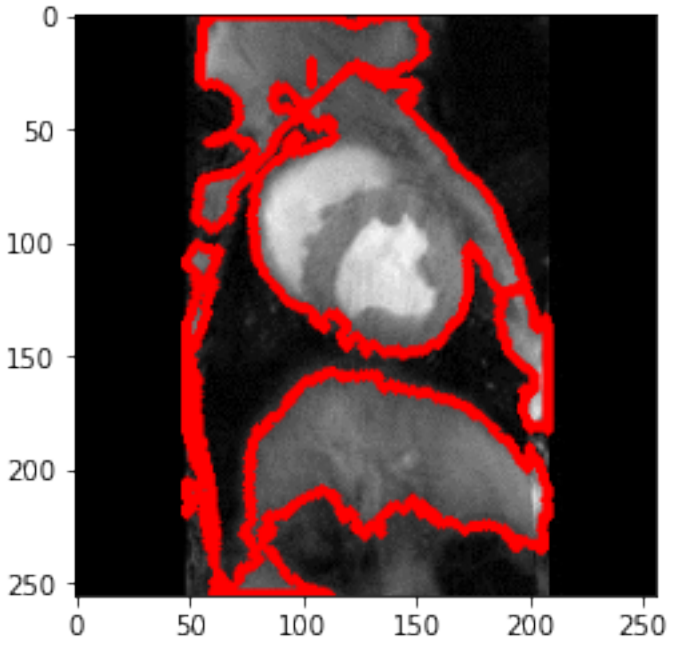
[3]:
```python
import cv2
import numpy as np
from skimage.feature import peak_local_max
from skimage.segmentation import watershed
from scipy import ndimage
import sys
import matplotlib.pyplot as plt
```

[4]:
```python
image_filename = "heart.jpg"
image = cv2.imread(image_filename)

# Check if the image is read successfully
if image is None:
    print("No image found")
    sys.exit()

# Load in image, convert to gray scale, and Otsu's threshold
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
thresh = cv2.threshold(gray, 128, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]

# Compute Euclidean distance from every binary pixel
# to the nearest zero pixel then find peaks
distance_map = ndimage.distance_transform_edt(thresh)
local_max = peak_local_max(distance_map, indices=False, min_distance=20,␣
 ↪labels=thresh)

# Perform connected component analysis then apply Watershed
markers = ndimage.label(local_max, structure=np.ones((3, 3)))[0]
labels = watershed(-distance_map, markers, mask=thresh)
```

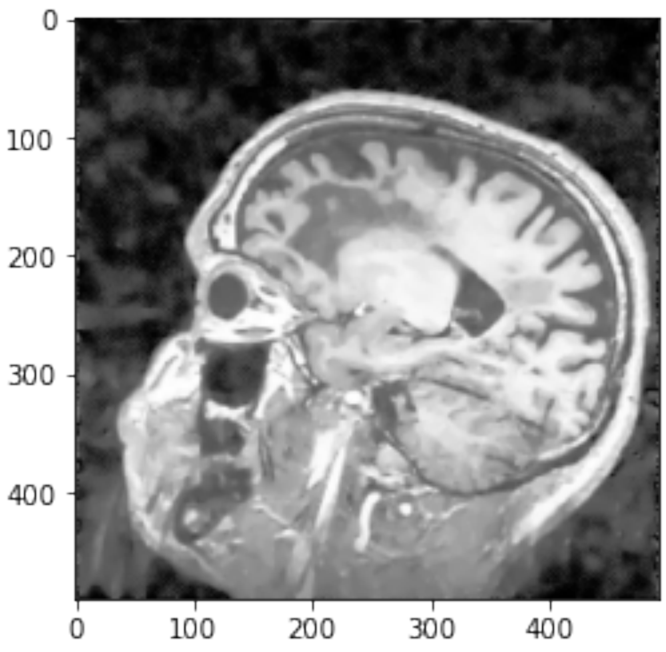
```
# Iterate through unique labels
total_area = 0
for label in np.unique(labels):
    if label == 0:
        continue

    # Create a mask
    mask = np.zeros(gray.shape, dtype="uint8")
    mask[labels == label] = 255

    # Find contours and determine contour area
    cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL, cv2.
 ↪CHAIN_APPROX_SIMPLE)
    cnts = cnts[0] if len(cnts) == 2 else cnts[1]
    c = max(cnts, key=cv2.contourArea)
    area = cv2.contourArea(c)
    total_area += area
    cv2.drawContours(image, [c], -1, (0,0,255), 4)

print("the total area is: ",total_area)

# Convert the image from BGR to RGB
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Show the processed image using matplotlib.pyplot
plt.imshow(image_rgb)
plt.show()

# Save the result image to the local folder using matplotlib.pyplot
result_image_filename = "heart_q2.jpg"
plt.imsave(result_image_filename, image_rgb)
```

```
the total area is:  21218.0
```

```
/var/folders/vn/30nktvjj7_gdjzzfypwf_5880000gp/T/ipykernel_3991/2150470762.py:14
: FutureWarning: indices argument is deprecated and will be removed in version
0.20. To avoid this warning, please do not use the indices argument. Please see
peak_local_max documentation for more details.
  local_max = peak_local_max(distance_map, indices=False, min_distance=20,
labels=thresh)
```



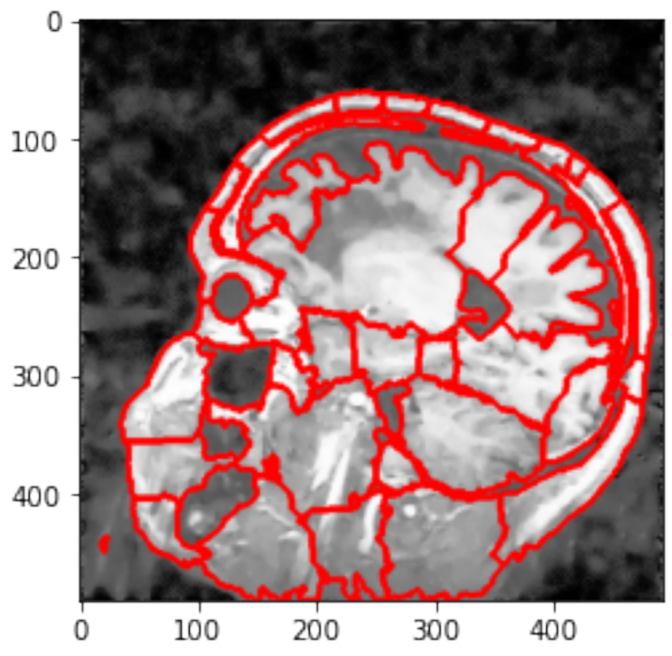
```
[42]: image_o = cv2.imread("brain.tech.jpg")
      image = image_o[60:550,190:683] # crop image to avoid pale frame
      if image is None:
          print("No image found!")
          sys.exit()
      else:
          plt.imshow(image)
          plt.show()
```



[43]:
```
# Load in image, convert to gray scale, and Otsu's threshold
gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
thresh = cv2.threshold(gray, 128, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)[1]

# Compute Euclidean distance from every binary pixel
# to the nearest zero pixel then find peaks
distance_map = ndimage.distance_transform_edt(thresh)
local_max = peak_local_max(distance_map, indices=False, min_distance=20,␣
 ↪labels=thresh)

# Perform connected component analysis then apply Watershed
markers = ndimage.label(local_max, structure=np.ones((3, 3)))[0]
labels = watershed(-distance_map, markers, mask=thresh)

# Iterate through unique labels
total_area = 0
for label in np.unique(labels):
    if label == 0:
        continue

    # Create a mask
    mask = np.zeros(gray.shape, dtype="uint8")
    mask[labels == label] = 255

    # Find contours and determine contour area
    cnts = cv2.findContours(mask.copy(), cv2.RETR_EXTERNAL, cv2.
 ↪CHAIN_APPROX_SIMPLE)
    cnts = cnts[0] if len(cnts) == 2 else cnts[1]
    c = max(cnts, key=cv2.contourArea)
    area = cv2.contourArea(c)
    total_area += area
    cv2.drawContours(image, [c], -1, (0,0,255), 4)

print("the total area is: ",total_area)

# Convert the image from BGR to RGB
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

# Show the processed image using matplotlib.pyplot
plt.imshow(image_rgb)
plt.show()

# Save the result image to the local folder using matplotlib.pyplot
result_image_filename = "brain.tech_q2.jpg"
plt.imsave(result_image_filename, image_rgb)
```

```
the total area is:  119189.0
```

```
/var/folders/vn/30nktvjj7_gdjzzfypwf_5880000gp/T/ipykernel_3991/3562561628.py:8:
FutureWarning: indices argument is deprecated and will be removed in version
0.20. To avoid this warning, please do not use the indices argument. Please see
peak_local_max documentation for more details.
  local_max = peak_local_max(distance_map, indices=False, min_distance=20,
labels=thresh)
```



[ ]:
```
```

yizjia577mtq3

March 21, 2023

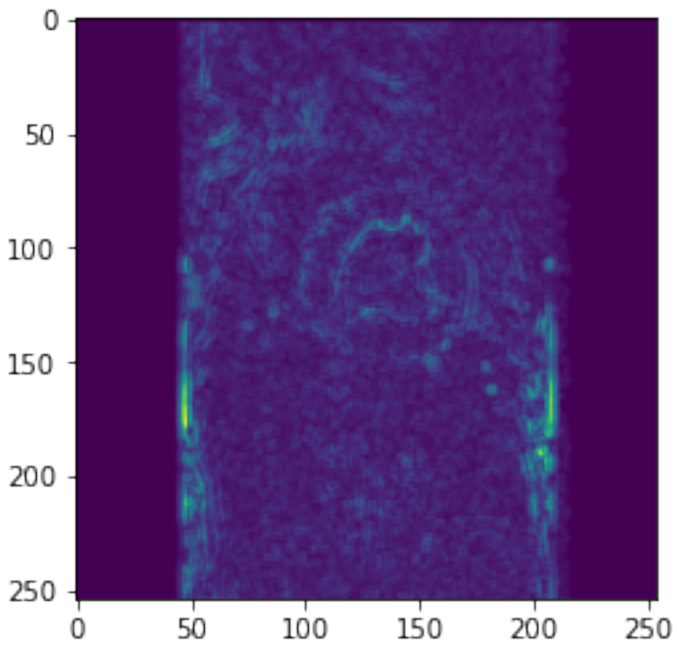
0.0.1 Problem 3: Laplacian of Gaussian

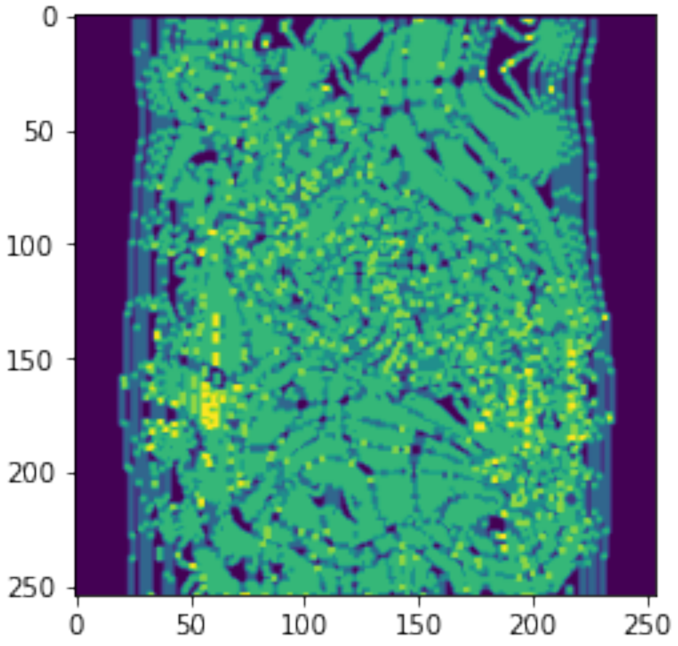
[6]:
```python
import numpy as np
import scipy.misc
import cv2
import matplotlib.pyplot as plt
```

[7]:
```python
def laplace_of_gaussian(gray_img, sigma, kappa=0.75, pad=False):
    """
    Applies Laplacian of Gaussians to grayscale image.
    :param gray_img: image to apply LoG to
    :param sigma:    Gauss sigma of Gaussian applied to image, <= 0. for none
    :param kappa:    difference threshold as factor to mean of image values, <=␣
 ↪0 for none
    :param pad:      flag to pad output w/ zero border, keeping input image size
    """
    assert len(gray_img.shape) == 2
    img = cv2.GaussianBlur(gray_img, (0, 0), sigma) if 0. < sigma else gray_img
    img = cv2.Laplacian(img, cv2.CV_64F)
    rows, cols = img.shape[:2]

    # min/max of 3x3-neighbourhoods
    min_map = np.minimum.reduce(list(img[r:rows-2+r, c:cols-2+c]
                                     for r in range(3) for c in range(3)))
    max_map = np.maximum.reduce(list(img[r:rows-2+r, c:cols-2+c]
                                     for r in range(3) for c in range(3)))

    # bool matrix for image value positiv (w/out border pixels)
    pos_img = 0 < img[1:rows-1, 1:cols-1]

    # bool matrix for min < 0 and 0 < image pixel
    neg_min = min_map < 0
    neg_min[1 - pos_img] = 0

    # bool matrix for 0 < max and image pixel < 0
    pos_max = 0 < max_map
    pos_max[pos_img] = 0
    # sign change at pixel
    zero_cross = neg_min + pos_max

    # values: max - min, scaled to 0--255; set to 0 for no sign change
    value_scale = 255. / max(1., img.max() - img.min())
    values = value_scale * (max_map - min_map)
    values[1 - zero_cross] = 0.

    # optional thresholding
    if 0. <= kappa:
        thresh = float(np.absolute(img).mean()) * kappa
        values[values < thresh] = 0.
    log_img = values.astype(np.uint8)
    if pad:
        log_img = np.pad(log_img, pad_width=1, mode='constant',␣
↪constant_values=0)
    return log_img
```

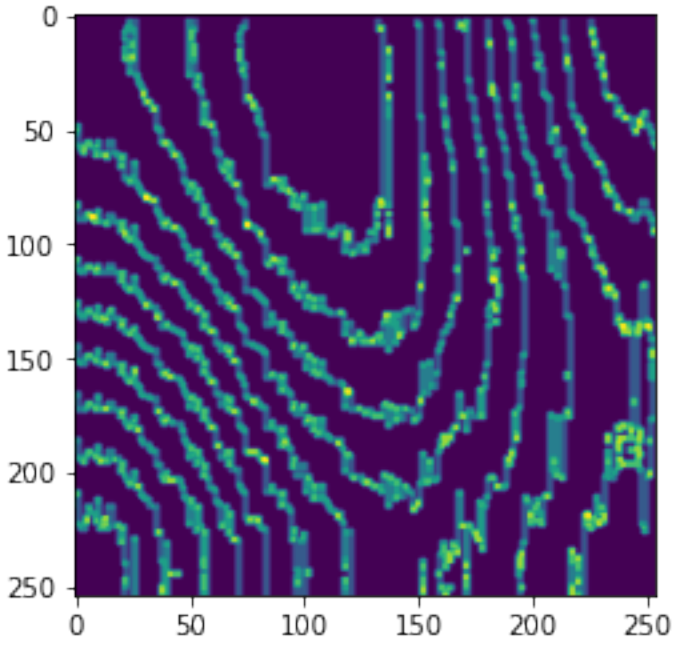
[8]:

```
img = cv2.imread("heart.jpg")
img = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

# apply LoG
sgm = [1,10,100,1000] # apply different sigma value
for i in sgm:
    log = laplace_of_gaussian(img, sigma = i)
    plt.imshow(log)
    plt.show()
```

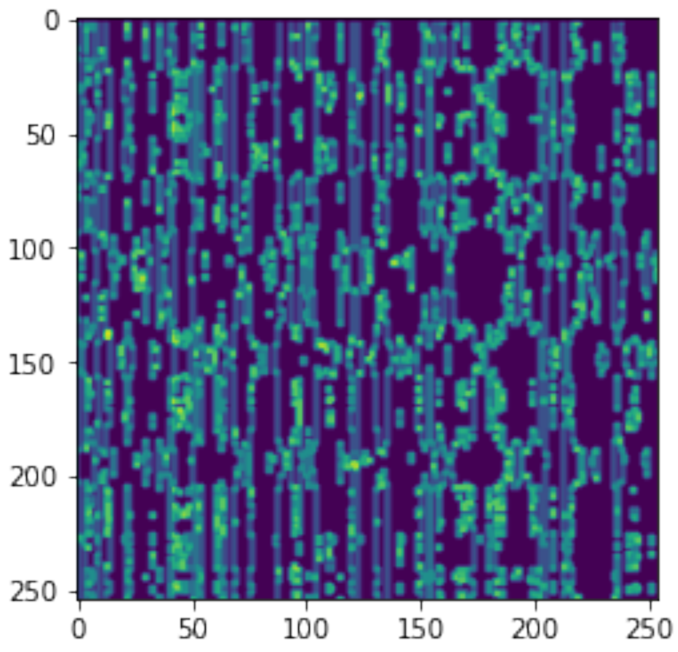




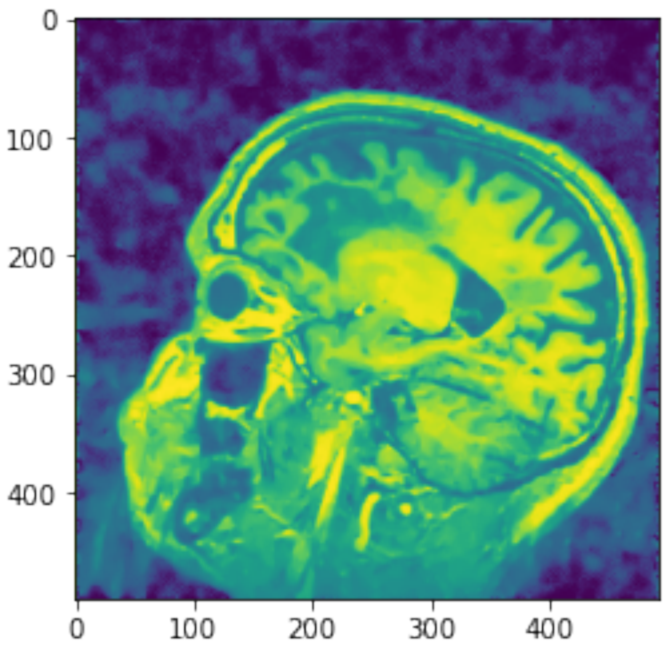
0
50
100
150
200
250
0
50
100
150
200
250

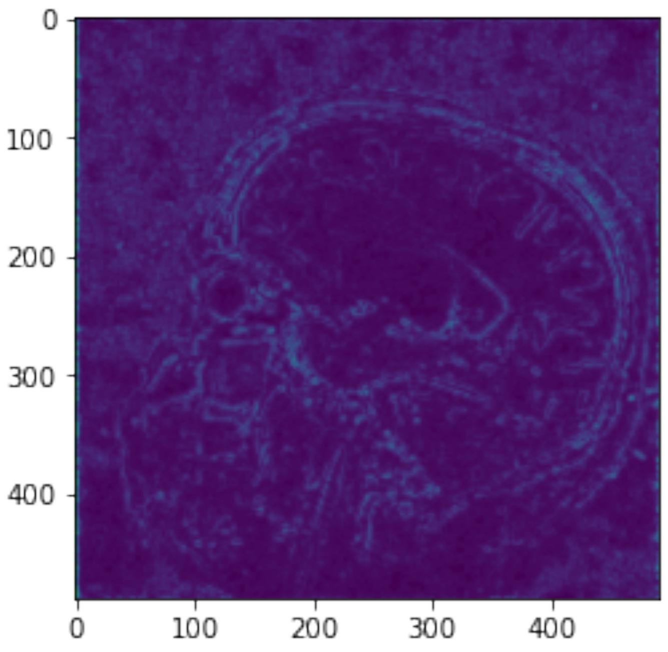


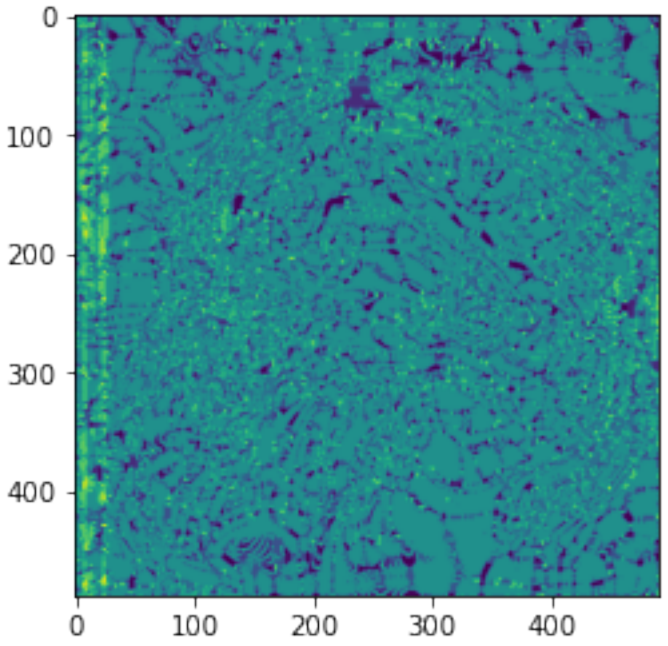
0
50
100
150
200
250
0
50
100
150
200
250

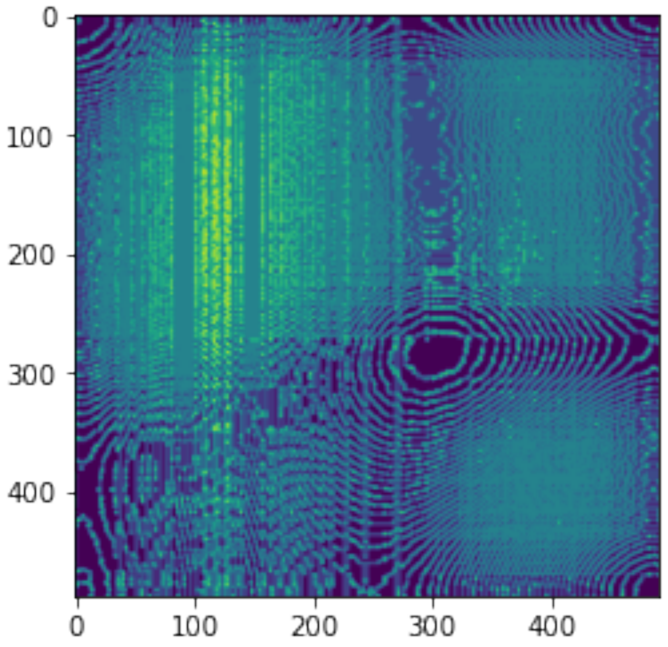


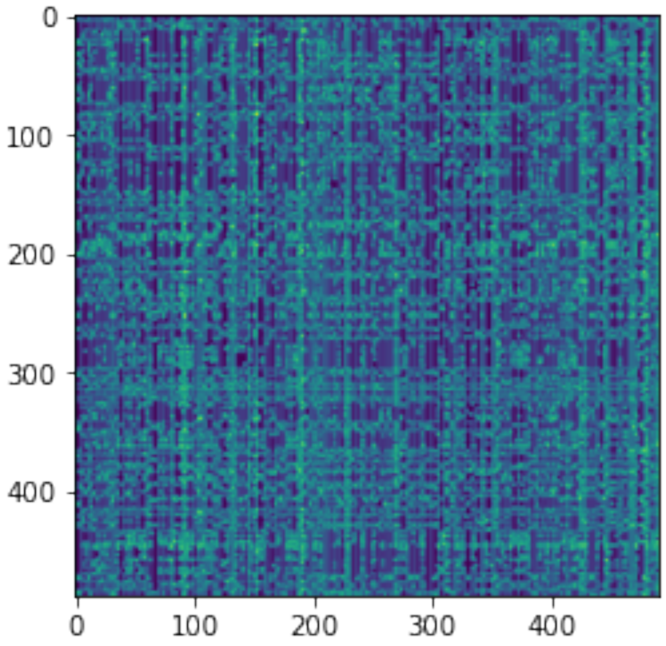
[9]:
```python
image_o = cv2.imread("brain.tech.jpg")
image = image_o[60:550,190:683] # crop image to avoid pale frame
img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
if img is None:
    print("no image!")
    sys.exit()
else:
    plt.imshow(img)
    plt.show()

# apply LoG
sgm = [1,10,100,1000] # apply different sigma value
for i in sgm:
    log = laplace_of_gaussian(img, sigma = i)
    plt.imshow(log)
    plt.show()
```


0
100
200
300
400
0
100
200
300
400


0
100
200
300
400
0
100
200
300
400


0
100
200
300
400
0
100
200
300
400


0
100
200
300
400
0
100
200
300
400


0
100
200
300
400
0
100
200
300
400

```
[ ]:
```

yizjia577mtq4

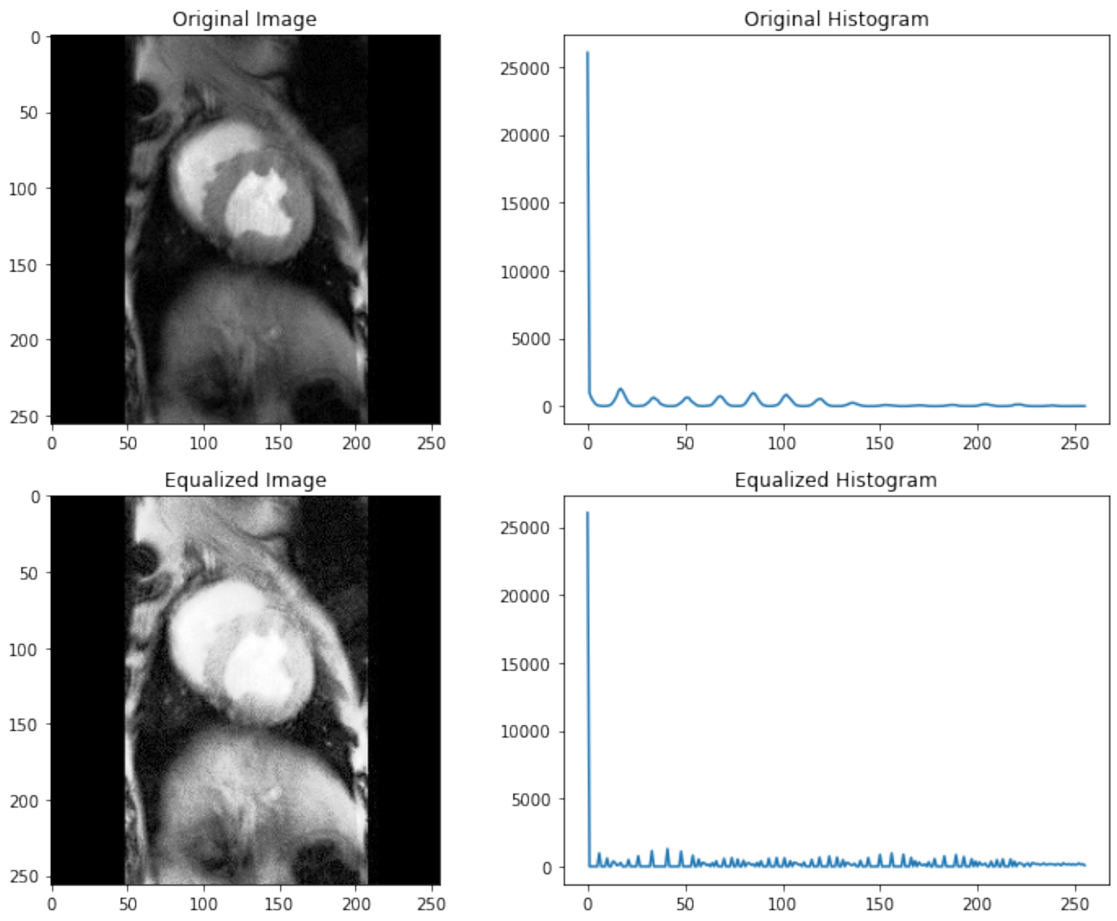
March 21, 2023

0.0.1 Problem 4: Histogram Equalization

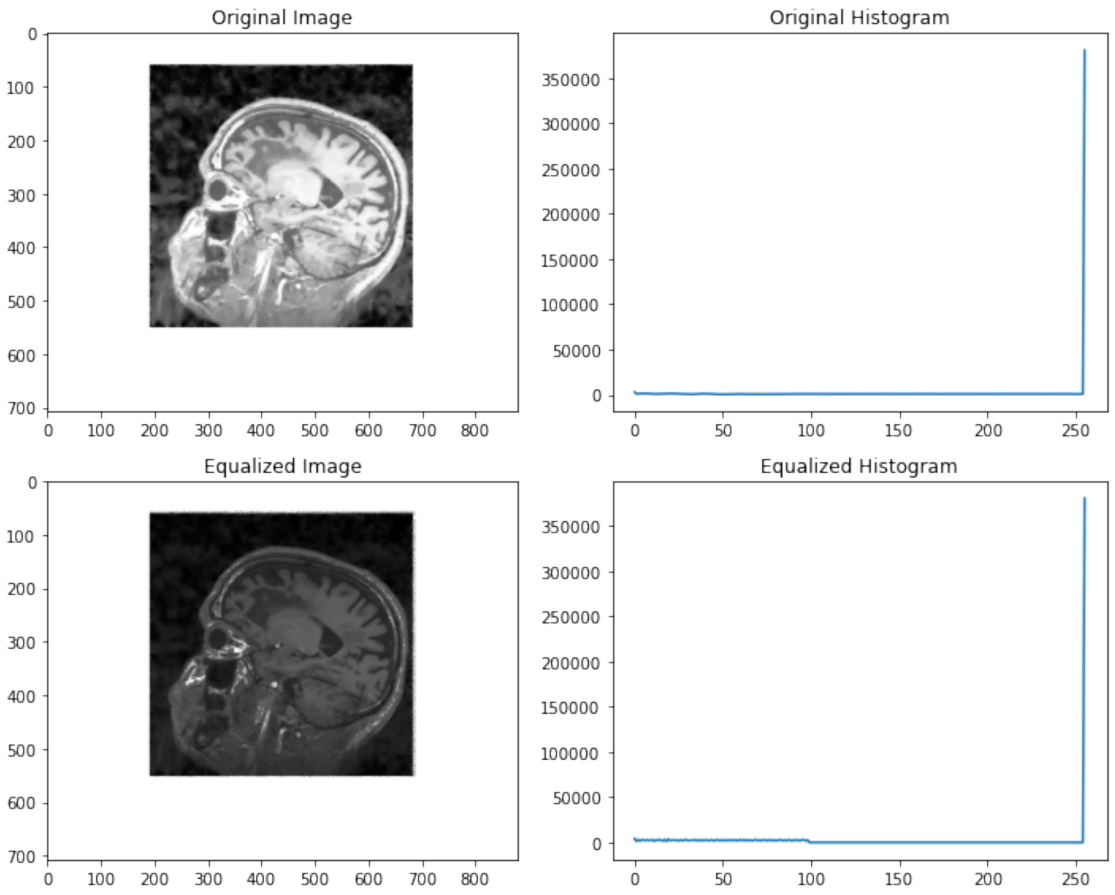
[12]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

[13]:
```
def histogram_equalization(image):
    # Load the image
    img = cv2.imread(image, cv2.IMREAD_GRAYSCALE)

    # Check if the image was loaded correctly
    if img is None:
        print("No image found!")

    # Compute the histogram of the original image
    hist, bins = np.histogram(img.flatten(), 256, [0, 256])

    # Apply histogram equalization
    img_eq = cv2.equalizeHist(img)

    # Compute the histogram of the equalized image
    hist_eq, bins_eq = np.histogram(img_eq.flatten(), 256, [0, 256])

    # Display the images and histograms
    fig, axes = plt.subplots(2, 2, figsize=(10, 8))

    axes[0, 0].imshow(img, cmap='gray')
    axes[0, 0].set_title('Original Image')

    axes[0, 1].plot(hist)
    axes[0, 1].set_title('Original Histogram')

    axes[1, 0].imshow(img_eq, cmap='gray')
    axes[1, 0].set_title('Equalized Image')

    axes[1, 1].plot(hist_eq)
    axes[1, 1].set_title('Equalized Histogram')
    plt.tight_layout()
    plt.show()
```

[14]:
```
histogram_equalization('heart.jpg')
```



[15]:
```
histogram_equalization('brain.tech.jpg')
```


Original Image
Original Histogram
Equalized Image
Equalized Histogram
0
100
200
300
400
500
600
700
800
50
150
250
50000
100000
150000
200000
250000
300000
350000

[ ]:

yizjia577mtq5

March 21, 2023

0.0.1 Problem 5: Gaussian Smoothing Filter and Relatives

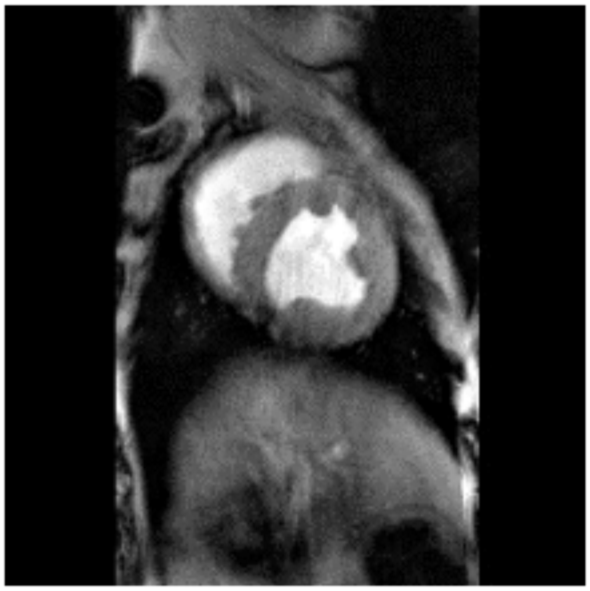
The filter allows me to control the degree of sharpening or smoothing applied to the image by adjusting the parameter alpha. If alpha is positive, the filter will amplify the high-frequency content, resulting in a sharper image. The greater the alpha value, the more pronounced the sharpening effect will be. If alpha is negative, the filter will reduce the high-frequency content, leading to a more smoothed or blurred image.

[10]:
```python
import cv2
import numpy as np
import matplotlib.pyplot as plt
```

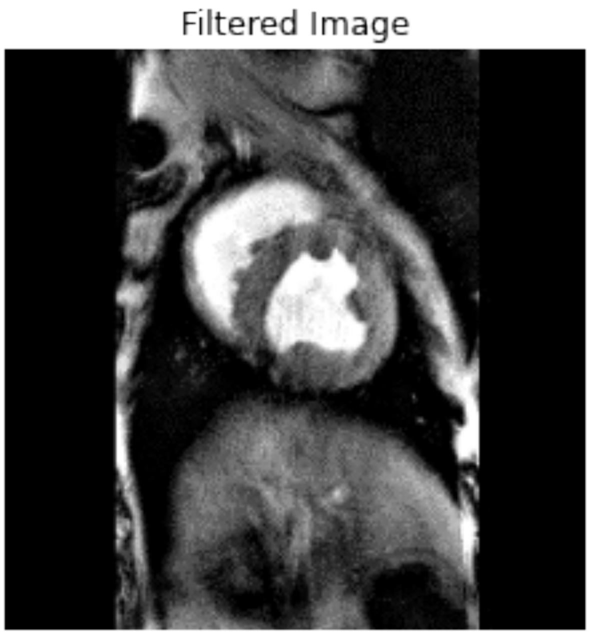
[11]:
```python
def apply_filter(image_path, alpha, sigma):
    # Read the image
    img = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)

    # Create the Gaussian filter
    ksize = int(6 * sigma) // 2 * 2 + 1  # Size of the kernel should be an odd␣
↪number
    gaussian_filter = cv2.getGaussianKernel(ksize, sigma)
    gaussian_filter = gaussian_filter * gaussian_filter.T

    # Convolve the image with the Gaussian filter
    img_smoothed = cv2.filter2D(img, -1, gaussian_filter)

    # Apply the filter formula
    img_filtered = (1 + alpha) * img - alpha * img_smoothed

    # Normalize the output image
    img_filtered = np.clip(img_filtered, 0, 255).astype(np.uint8)

    return img_filtered

# Set the parameters
image_path = 'heart.jpg'
alpha = 0.5
sigma = 10 # parameters in problem
```

```python
# Apply the filter
img_filtered = apply_filter(image_path, alpha, sigma)

# Save the filtered image
cv2.imwrite('heart_q5_1.jpg', img_filtered)

# Display the filtered image using matplotlib.pyplot
plt.imshow(img_filtered, cmap='gray')
plt.title('Filtered Image')
plt.axis('off')
plt.show()
```

Filtered Image

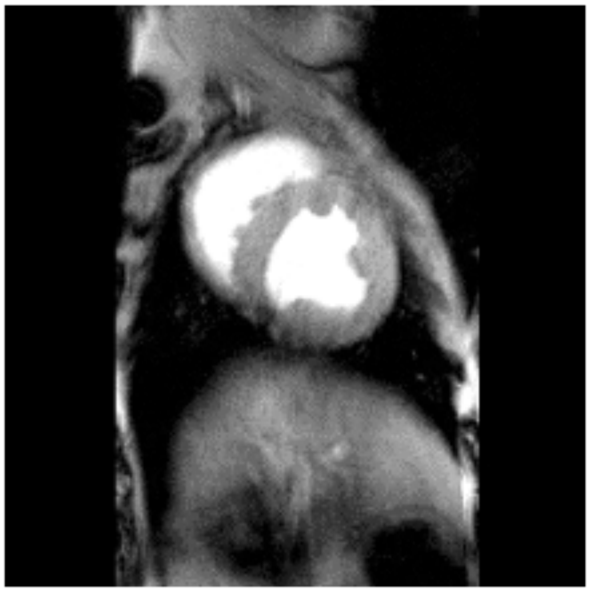


[12]:
```python
# Set the parameters
image_path = 'heart.jpg'
alpha = 1.2
sigma = 10 # alpha 0.5 -> 1.2, sigma stays still

# Apply the filter
img_filtered = apply_filter(image_path, alpha, sigma)

# Save the filtered image
cv2.imwrite('heart_q5_2.jpg', img_filtered)

# Display the filtered image using matplotlib.pyplot
plt.imshow(img_filtered, cmap='gray')
plt.title('Filtered Image')
plt.axis('off')
plt.show()
```



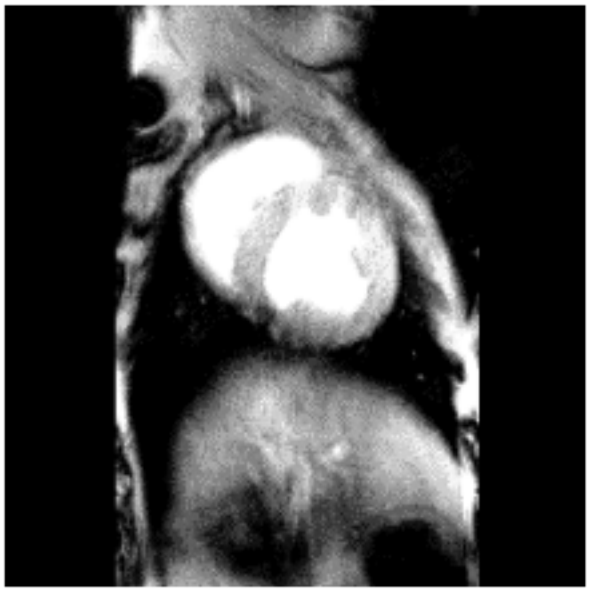
[13]:
```
# Set the parameters
image_path = 'heart.jpg'
alpha = 0.5
sigma = 1000 # sigma 10 -> 1000, alpha stays still

# Apply the filter
img_filtered = apply_filter(image_path, alpha, sigma)

# Save the filtered image
cv2.imwrite('heart_q5_3.jpg', img_filtered)

# Display the filtered image using matplotlib.pyplot
plt.imshow(img_filtered, cmap='gray')
plt.title('Filtered Image')
plt.axis('off')
plt.show()
```

Filtered Image



```
[14]: # Set the parameters
image_path = 'heart.jpg'
alpha = 1.2
sigma = 1000 # alpha 0.5 -> 1.2, sigma 10 -> 1000

# Apply the filter
img_filtered = apply_filter(image_path, alpha, sigma)

# Save the filtered image
cv2.imwrite('heart_q5_4.jpg', img_filtered)

# Display the filtered image using matplotlib.pyplot
plt.imshow(img_filtered, cmap='gray')
plt.title('Filtered Image')
plt.axis('off')
plt.show()
```

Filtered Image



```
[ ]:
```