Preprocessing With Dicom

October 24, 2019

0.1 Applying Different Preprocessing on Dicom Images

[1]:
```
!conda install -c conda-forge pydicom=0.9.9 --yes
```

```
Collecting package metadata (current_repodata.json): done
Solving environment: failed with initial frozen solve. Retrying with flexible
solve.
Solving environment: failed with repodata from current_repodata.json, will retry
with next repodata source.
Collecting package metadata (repodata.json): failed

CondaHTTPError: HTTP 000 CONNECTION FAILED for url
<https://repo.anaconda.com/pkgs/main/noarch/repodata.json>
Elapsed: -

An HTTP error occurred when trying to retrieve this URL.
HTTP errors are often intermittent, and a simple retry will get you on your way.

If your current network has https://www.anaconda.com blocked, please file
a support request with your network engineering team.

ConnectionError(ReadTimeoutError("HTTPSConnectionPool(host='repo.anaconda.com',
port=443): Read timed out."))
```

[13]:
```
# this following line tells Jupyter to display images here in the browser,
# rather than in separate window.

%matplotlib inline

# import pydicom library

import pydicom as dicom

# import matplotlib and numpy
```

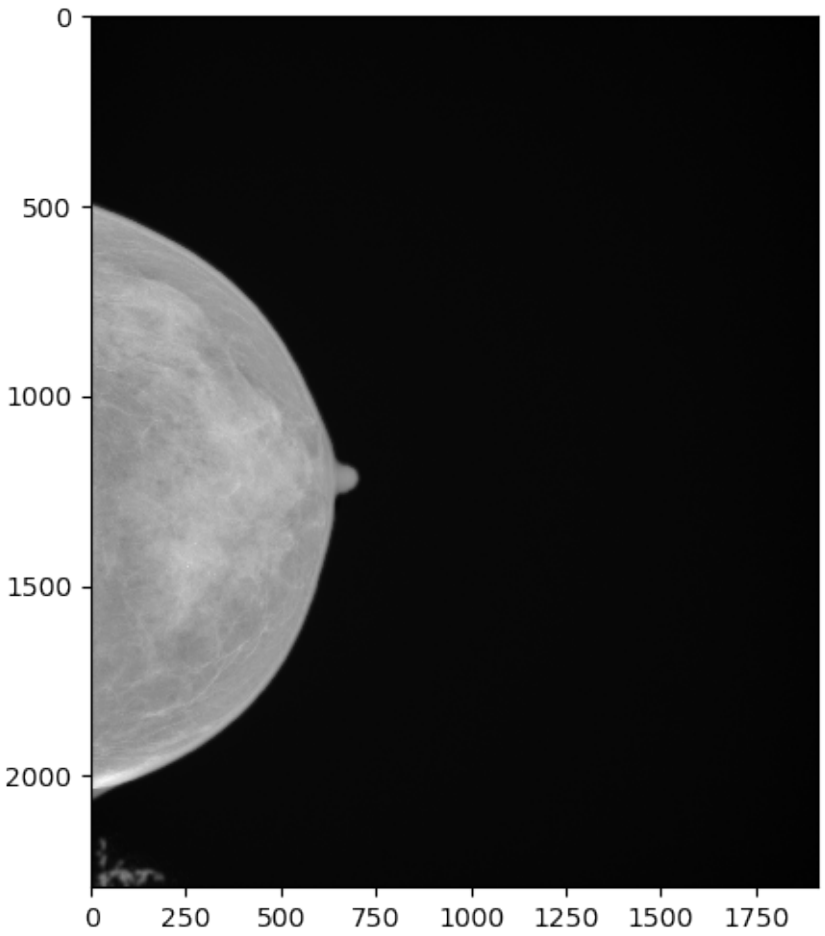
```
import matplotlib.pyplot as plt
import matplotlib.image as mpimage
import numpy as np
```

[14]:
```
# load some handy functions from the scikit-image library

from skimage import exposure
import skimage.morphology as morp
from skimage.filters import rank

# import operating system and glob libraries

import os, glob

# import some useful date functions

from datetime import datetime
```

[15]:
```
ds1 = dicom.read_file("/Users/nikhil/Downloads/562342730/SNUBH-MDB- Ci/0010.
↪dcm")  # read the DICOM image into memory
```

[16]:
```
# tell matplotlib to display our images as a 6 x 6 inch image, with resolution␣
↪of 100 dpi
plt.figure(figsize = (6,6), dpi=100)

# tell matplotlib to display our image, using a gray-scale lookup table.
plt.imshow(ds1.pixel_array, cmap=plt.cm.gray)
```

[16]:
```
<matplotlib.image.AxesImage at 0x1c1f0cbe50>
```



[17]:
```
ds1
```

[17]:
```
(0008, 0005) Specific Character Set              CS: 'ISO_IR 100'
(0008, 0008) Image Type                          CS: ['ORIGINAL', 'PRIMARY', '']
(0008, 0016) SOP Class UID                       UI: Digital Mammography X-Ray
Image Storage - For Presentation
(0008, 0018) SOP Instance UID                    UI:
9999.179205396445249921462244902608406690744
(0008, 0020) Study Date                          DA: '20010128'
(0008, 0023) Content Date                        DA: '20010128'
(0008, 0030) Study Time                          TM: ''
(0008, 0033) Content Time                        TM: ''
(0008, 0050) Accession Number                    SH: '1122236167559237'
(0008, 0060) Modality                            CS: 'MG'
(0008, 0064) Conversion Type                     CS: 'WSD'
(0008, 0068) Presentation Intent Type            CS: 'FOR PRESENTATION'
(0008, 0070) Manufacturer                        LO: ''
```

```
(0008, 0090) Referring Physician's Name          PN: ' '
(0008, 1030) Study Description                   LO: 'Mammography Routine'
(0008, 103e) Series Description                  LO: 'Mammography Routine'
(0008, 2218)  Anatomic Region Sequence   1 item(s) ----
   (0008, 0100) Code Value                          SH: 'T-04000'
   (0008, 0102) Coding Scheme Designator            SH: 'SNM3'
   (0008, 0104) Code Meaning                        LO: 'BREAST'
   ---------
(0010, 0010) Patient's Name                      PN: '2770951330'
(0010, 0020) Patient ID                          LO: '2770951330'
(0010, 0040) Patient's Sex                       CS: ''
(0012, 0062) Patient Identity Removed            CS: 'YES'
(0018, 0015) Body Part Examined                  CS: 'BREAST'
(0018, 0060) KVP                                 DS: "28"
(0018, 1110) Distance Source to Detector         DS: "660"
(0018, 1111) Distance Source to Patient          DS: "660"
(0018, 1114) Estimated Radiographic Magnificatio DS: "1"
(0018, 1147) Field of View Shape                 CS: 'RECTANGLE'
(0018, 1149) Field of View Dimension(s)          IS: ['229', '191']
(0018, 1150) Exposure Time                       IS: "421"
(0018, 1151) X-Ray Tube Current                  IS: "95"
(0018, 1152) Exposure                            IS: "36"
(0018, 1153) Exposure in uAs                     IS: "36300"
(0018, 1160) Filter Type                         SH: 'STRIP'
(0018, 1164) Imager Pixel Spacing                DS: ['0.094090909',
'0.094090909']
(0018, 1166) Grid                                CS: ['RECIPROCATING',
'FOCUSED']
(0018, 1190) Focal Spot(s)                       DS: "0.3"
(0018, 1191) Anode Target Material               CS: 'MOLYBDENUM'
(0018, 11a0) Body Part Thickness                 DS: "27"
(0018, 11a2) Compression Force                   DS: "90"
(0018, 1401) Acquisition Device Processing Code  LO: 'GEMS_FFDM_PV'
(0018, 1405) Relative X-Ray Exposure             IS: "3756"
(0018, 1508) Positioner Type                     CS: 'MAMMOGRAPHIC'
(0018, 1510) Positioner Primary Angle            DS: "0"
(0018, 1530) Detector Primary Angle              DS: "0"
(0018, 1700) Collimator Shape                    CS: 'RECTANGULAR'
(0018, 1702) Collimator Left Vertical Edge       IS: "0"
(0018, 1704) Collimator Right Vertical Edge      IS: "1913"
(0018, 1706) Collimator Upper Horizontal Edge    IS: "0"
(0018, 1708) Collimator Lower Horizontal Edge    IS: "2293"
(0018, 5101) View Position                       CS: 'CC'
(0018, 6000) Sensitivity                         DS: "0.01153066"
(0018, 7000) Detector Conditions Nominal Flag    CS: 'YES'
(0018, 7001) Detector Temperature                DS: "30.9"
(0018, 7004) Detector Type                       CS: 'SCINTILLATOR'
```

```
(0018, 7005) Detector Configuration              CS: 'AREA'
(0018, 701a) Detector Binning                    DS: ['1', '1']
(0018, 7020) Detector Element Physical Size      DS: ['0.1', '0.1']
(0018, 7022) Detector Element Spacing            DS: ['0.1', '0.1']
(0018, 7024) Detector Active Shape               CS: 'RECTANGLE'
(0018, 7026) Detector Active Dimension(s)        DS: ['192', '230.4']
(0018, 7030) Field of View Origin                DS: ['5', '1']
(0018, 7032) Field of View Rotation              DS: "0"
(0018, 7034) Field of View Horizontal Flip       CS: 'NO'
(0018, 7050) Filter Material                     CS: 'MOLYBDENUM'
(0018, 7060) Exposure Control Mode               CS: 'AUTOMATIC'
(0018, 7062) Exposure Control Mode Description   LT: 'AOP standard  RECTANGLE
1032 mm 250 mm 180 mm 240 mm EXP DOSE 67267 nGy PRE-EXP DOSE 2904 nGy PRE-EXP
THICK 34 mm PRE-EXP COMPO 75 % PRE-EXP KV 24 PRE-EXP TRACK Mo PRE-EXP FILTER Mo
PADDLE not detected FLATFIELD no'
(0018, 7064) Exposure Status                     CS: 'NORMAL'
(0020, 000d) Study Instance UID                  UI:
9999.104392271891125128751002392660835453701
(0020, 000e) Series Instance UID                 UI:
9999.122053378221540525088275740982289260 88
(0020, 0010) Study ID                            SH: ''
(0020, 0011) Series Number                       IS: "1178"
(0020, 0013) Instance Number                     IS: "2"
(0020, 0020) Patient Orientation                 CS: ['A', 'R']
(0020, 0062) Image Laterality                    CS: 'L'
(0028, 0002) Samples per Pixel                   US: 1
(0028, 0004) Photometric Interpretation          CS: 'MONOCHROME2'
(0028, 0010) Rows                                US: 2294
(0028, 0011) Columns                             US: 1914
(0028, 0100) Bits Allocated                      US: 16
(0028, 0101) Bits Stored                         US: 12
(0028, 0102) High Bit                            US: 11
(0028, 0103) Pixel Representation                US: 0
(0028, 0300) Quality Control Image               CS: 'NO'
(0028, 0301) Burned In Annotation                CS: 'NO'
(0028, 1040) Pixel Intensity Relationship        CS: 'LOG'
(0028, 1041) Pixel Intensity Relationship Sign   SS: -1
(0028, 1050) Window Center                       DS: "2593"
(0028, 1051) Window Width                        DS: "1184"
(0028, 1052) Rescale Intercept                   DS: "0"
(0028, 1053) Rescale Slope                       DS: "1"
(0028, 1054) Rescale Type                        LO: 'US'
(0028, 1055) Window Center & Width Explanation   LO: ['NORMAL', 'HARDER',
'SOFTER']
(0028, 1056) VOI LUT Function                    CS: 'SIGMOID'
(0028, 2110) Lossy Image Compression             CS: '00'
(0040, 0302) Entrance Dose                       US: 0
```

```
(0040, 0306) Distance Source to Entrance         DS: "633"
(0040, 0310) Comments on Radiation Dose          ST: '71 %'
(0040, 0316) Organ Dose                          DS: "0.00936"
(0040, 0318) Organ Exposed                       CS: 'BREAST'
(0045, 0010) Private Creator                     LO: 'GEMS_SENO_02'
(0054, 0220)  View Code Sequence   1 item(s) ----
   (0008, 0100) Code Value                          SH: 'R-10242'
   (0008, 0102) Coding Scheme Designator            SH: 'SNM3'
   (0008, 0104) Code Meaning                        LO: 'cranio-caudal'
   (0054, 0222)  View Modifier Code Sequence   0 item(s) ----
   ---------
(2050, 0020) Presentation LUT Shape              CS: 'IDENTITY'
(7fe0, 0010) Pixel Data                          OW: Array of 8781432 elements
```

[18]:
```
ds1.dir("patient")
```

[18]:
```
['DistanceSourceToPatient',
 'PatientID',
 'PatientIdentityRemoved',
 'PatientName',
 'PatientOrientation',
 'PatientSex']
```

[19]:
```
ds1.PatientName # show patient name
```

[19]:
```
'2770951330'
```

[20]:
```
ds1.PatientSex # show patient gender
```

[20]:
```
''
```

[21]:
```
ds1.PatientIdentityRemoved # show date of study acquisition
```

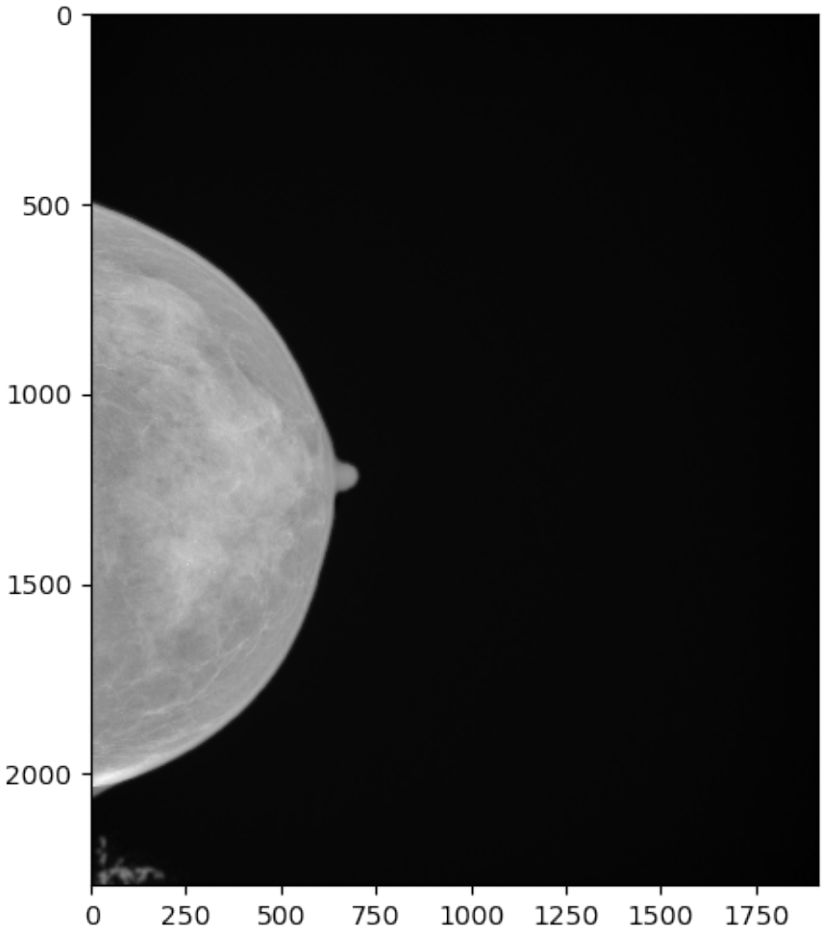
[21]:
```
'YES'
```

[22]:
```
plt.imsave('0010.jpg', ds1.pixel_array, cmap=plt.cm.gray)
plt.imsave('0010.png', ds1.pixel_array, cmap=plt.cm.gray)
#plt.imsave('0010.tif', ds1.pixel_array, cmap=plt.cm.gray)
```

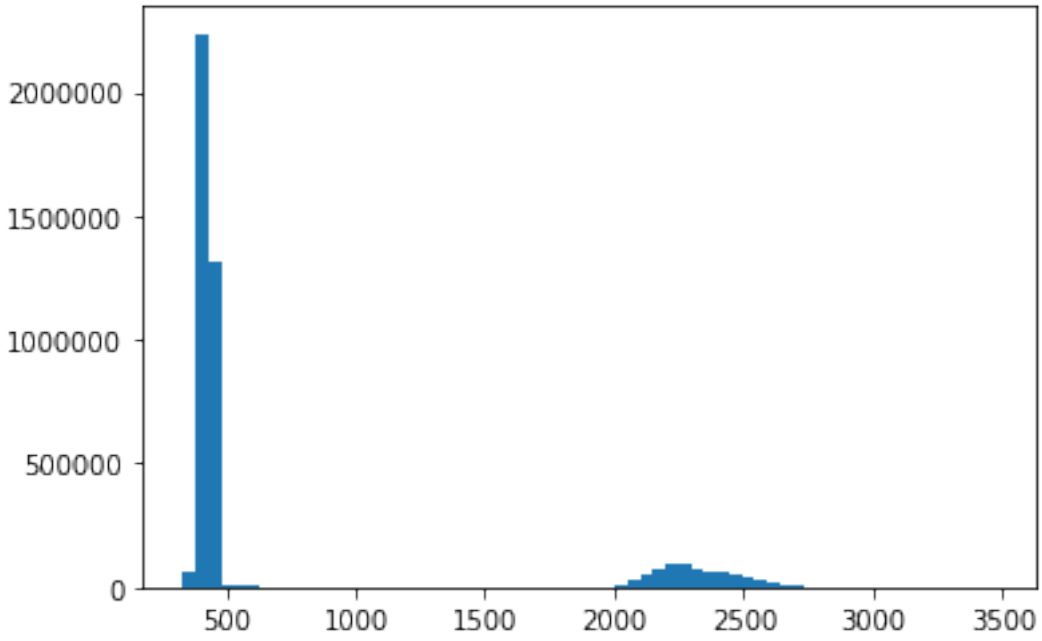
[23]:
```
ds1 = dicom.read_file("/Users/nikhil/Downloads/562342730/SNUBH-MDB- Ci/0010.
 ↪dcm")  # read the knee MR image into memory

# tell matplotlib to display our images as 6 x 6 inch image, with resolution of␣
 ↪100 dpi
plt.figure(figsize = (6,6), dpi=100)

# tell matplotlib to display our image, using a gray-scale lookup table.
```

```
plt.imshow(ds1.pixel_array, cmap=plt.cm.gray)
```

[23]: <matplotlib.image.AxesImage at 0x1c1f9f6710>



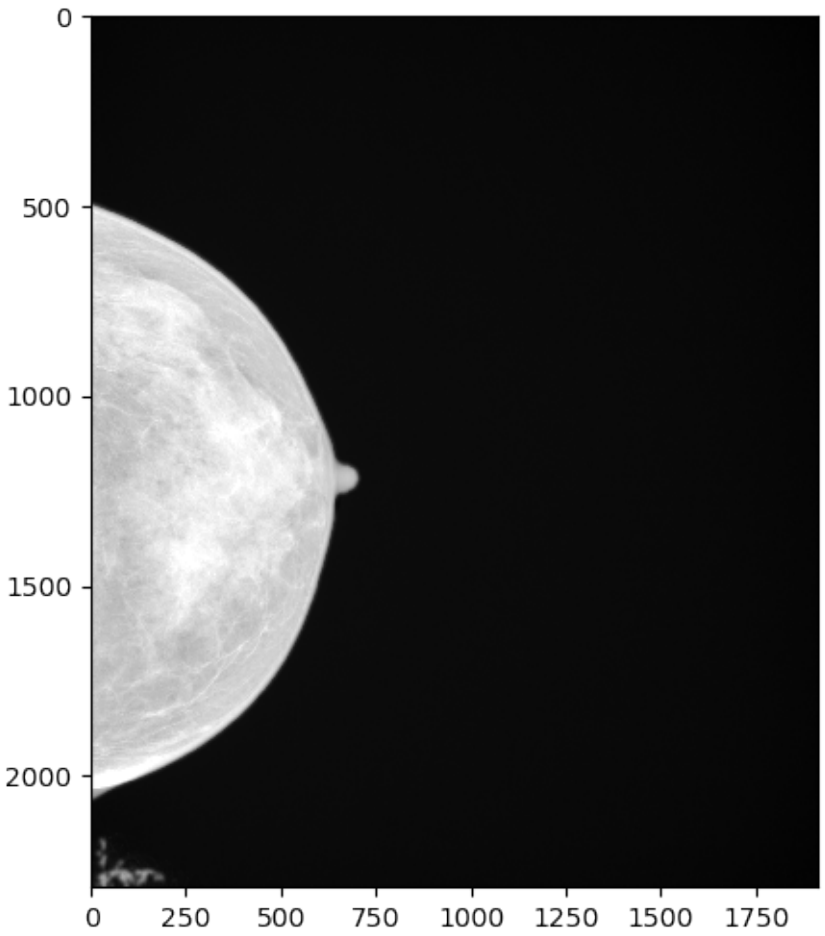
[24]:
```
plt.hist(ds1.pixel_array.flatten(), bins=64) # calculate a histogram of our␣
 ↪image
plt.show() # display that histogram

print ("pixel array = ", ds1.pixel_array.shape)
print("minimum value = ", np.amin(ds1.pixel_array)) # find minimum pixel value␣
 ↪in the image array
print("maximum value = ", np.amax(ds1.pixel_array)) # find maximum pixel value␣
 ↪in the image array
```



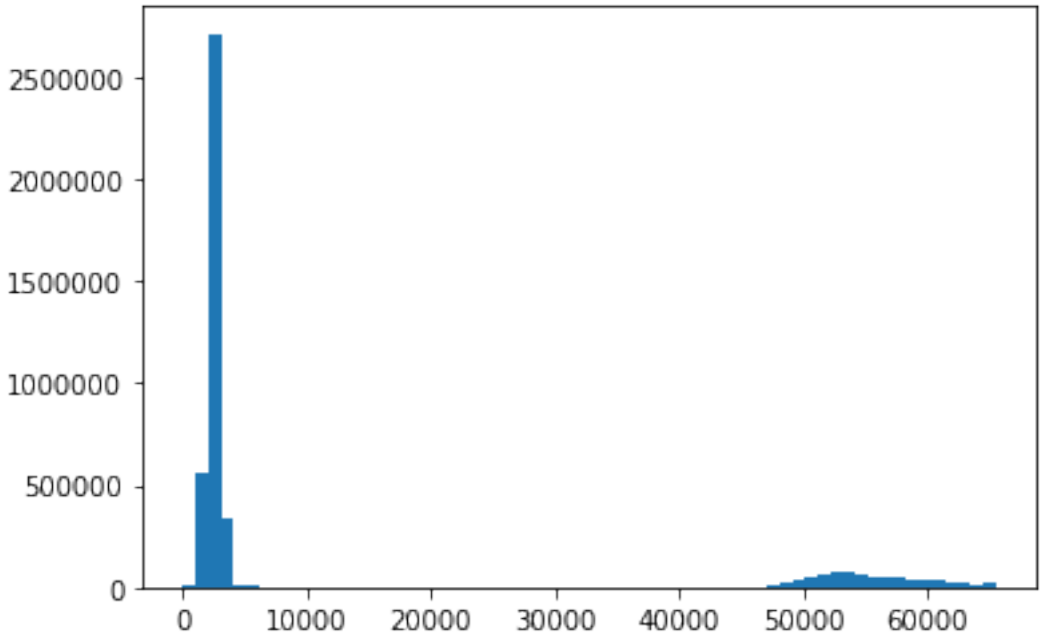

```
pixel array =  (2294, 1914)
minimum value =  334
maximum value =  3473
```

0.1.1 Contrast Streching

[25]:
```
# Contrast stretching try 1
plt.figure(figsize = (6,6), dpi=100)

p_lo, p_hi = np.percentile(ds1.pixel_array, (0, 99.75))

img_rescale_1 = exposure.rescale_intensity(ds1.pixel_array, in_range=(p_lo,␣
↪p_hi))

figure = plt.imshow(img_rescale_1, cmap=plt.cm.gray)
```

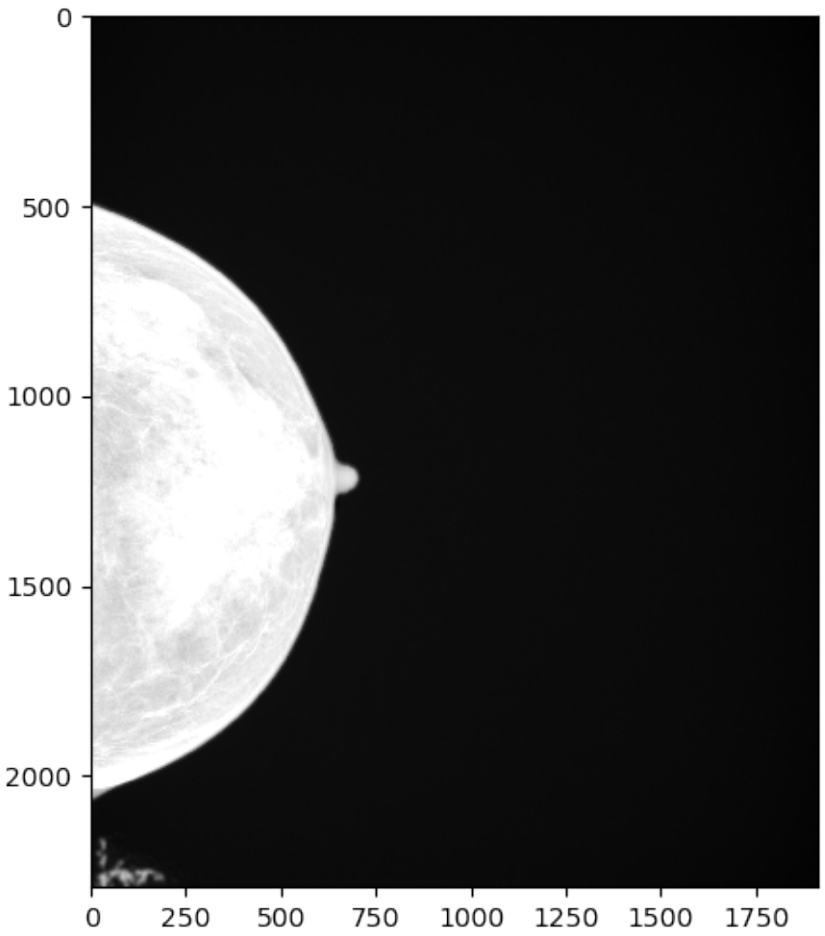


[26]:
```
plt.hist(img_rescale_1.flatten(), bins=64)
plt.show()
print("p_lo = ", p_lo)
print("p_hi = ",p_hi)
```

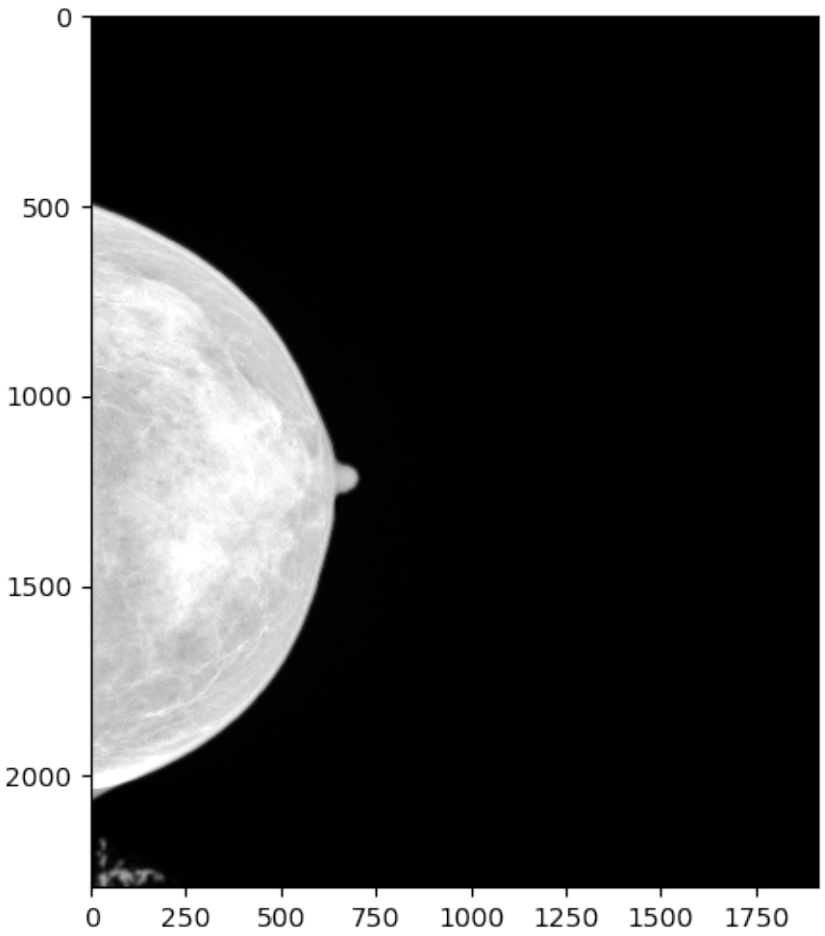



```
p_lo =  334.0
p_hi =  2678.0
```

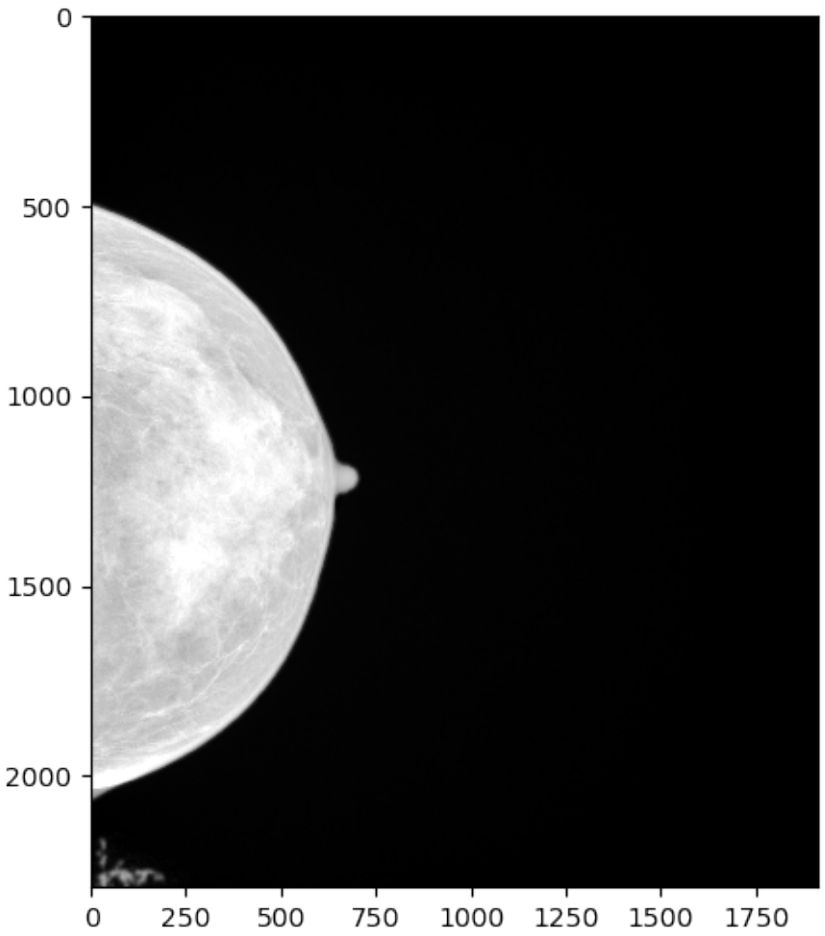
[27]:
```
# Contrast stretching try 2
plt.figure(figsize = (6,6), dpi=100)

p_lo, p_hi = np.percentile(ds1.pixel_array, (0, 95))
img_rescale_2 = exposure.rescale_intensity(ds1.pixel_array, in_range=(p_lo,␣
 ↪p_hi))

figure = plt.imshow(img_rescale_2, cmap=plt.cm.gray)
```



[28]:
```
# Contrast stretching try 3
plt.figure(figsize = (6,6), dpi=100)

p_lo, p_hi = np.percentile(ds1.pixel_array, (40, 99.5))
img_rescale_3 = exposure.rescale_intensity(ds1.pixel_array, in_range=(p_lo,␣
 ↪p_hi))

figure = plt.imshow(img_rescale_3, cmap=plt.cm.gray)
```



[29]:
```
# Contrast stretching try 4
plt.figure(figsize = (6,6), dpi=100)

p_lo, p_hi = np.percentile(ds1.pixel_array, (20, 99.5))
img_rescale_4 = exposure.rescale_intensity(ds1.pixel_array, in_range=(p_lo,␣
 ↪p_hi))

figure = plt.imshow(img_rescale_4, cmap=plt.cm.gray)

# save a copy of this image as a .PNG file
plt.imsave('knee_MR_contrast_stretching_try_4.png', img_rescale_4, cmap=plt.cm.
 ↪gray)
```



[30]:
```
from ipywidgets import interactive, interact, widgets, Layout, Button, Box
from IPython.display import display
```

[31]:
```
# turn off annoying pink warnings
import warnings
warnings.filterwarnings('ignore')
```

[32]:
```
ds2 = ds1
```

0.1.2 Using Contrast Interactive Method

[37]:
```
def contrast_stretch(image_name, percentile_lo, percentile_hi):

    image_name_global = image_name
    p_lo, p_hi = np.percentile(eval(image_name), (percentile_lo, percentile_hi))
```

```
    img_rescale = exposure.rescale_intensity(eval(image_name), in_range=(p_lo,␣
 ↪p_hi))

    # save optimized image array to global variable so other functions can use␣
 ↪it
    #img_rescale_interactive, image_name_global1
    img_rescale_interactive = img_rescale

    plt.figure(figsize = (6,6), dpi=100)
    plt.imshow(img_rescale, cmap=plt.cm.gray)

    plt.show()
```

[38]:
```
w = interactive(contrast_stretch, image_name="ds1.pixel_array",␣
 ↪percentile_lo=(1,100,.5), percentile_hi=(1,100,.5))

display(w)
```

```
interactive(children=(Text(value='ds1.pixel_array', description='image_name'), FloatSlider(valu
```

[39]:
```
# create three buttons
button_TIFF = widgets.Button(description = "Save .TIFF version")
button_JPG = widgets.Button(description = "Save .JPG version")
button_PNG = widgets.Button(description = "Save .PNG version")

# package them up in a list
items = [
    button_TIFF,
    button_JPG,
    button_PNG
]

# set up a layout for our box
box_layout = Layout(display = 'flex',
                    flex_flow = 'row',
                    align_items ='stretch'
                    )

# create a box containing our buttons
box = Box(children = items, layout = box_layout)


# these button actions will save the tweaked image in the desired format

def TIFF_button_clicked(b):
```

```
    plt.imsave(image_name_global+".tif", img_rescale_interactive, cmap=plt.cm.
 ↪gray)

def JPG_button_clicked(b):
    plt.imsave(image_name_global+".jpg", img_rescale_interactive, cmap=plt.cm.
 ↪gray)

def PNG_button_clicked(b):
    plt.imsave(image_name_global+".png", img_rescale_interactive, cmap=plt.cm.
 ↪gray)

# call the appropriate function when each button is clicked

button_TIFF.on_click(TIFF_button_clicked)
button_JPG.on_click(JPG_button_clicked)
button_PNG.on_click(PNG_button_clicked)
```

0.1.3 With save buttons

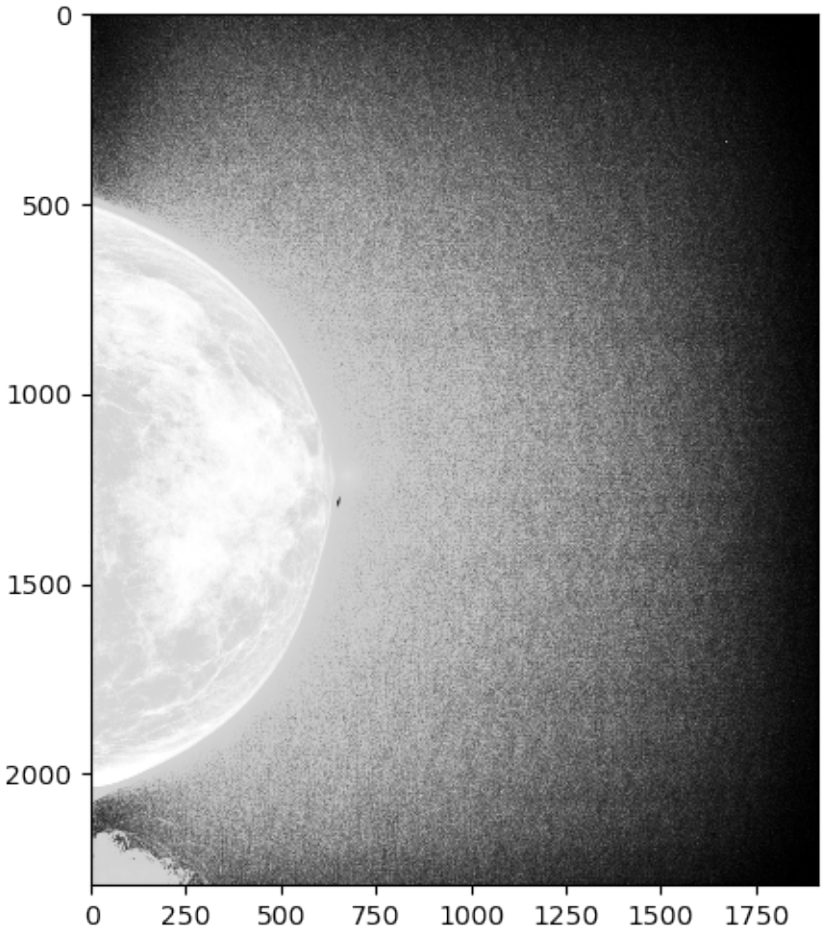
[41]:
```
w = interactive(contrast_stretch, image_name="ds1.pixel_array",␣
 ↪percentile_lo=(1,100,.5), percentile_hi=(1,100,.5))

display(w)

box
```

```
interactive(children=(Text(value='ds1.pixel_array', description='image_name'), FloatSlider(valu
```

```
Box(children=(Button(description='Save .TIFF version', style=ButtonStyle()), Button(descriptio
```

0.1.4 Global Equalization

[42]:
```
# global equalization

plt.figure(figsize = (6,6), dpi=100)

img_global_eq = exposure.equalize_hist(ds1.pixel_array)
figure = plt.imshow(img_global_eq, cmap=plt.cm.gray)
```



[43]:
```
w = interactive(contrast_stretch, image_name="img_global_eq",␣
↪percentile_lo=(1,100,.5), percentile_hi=(1,100,.5))

display(w)

box
```

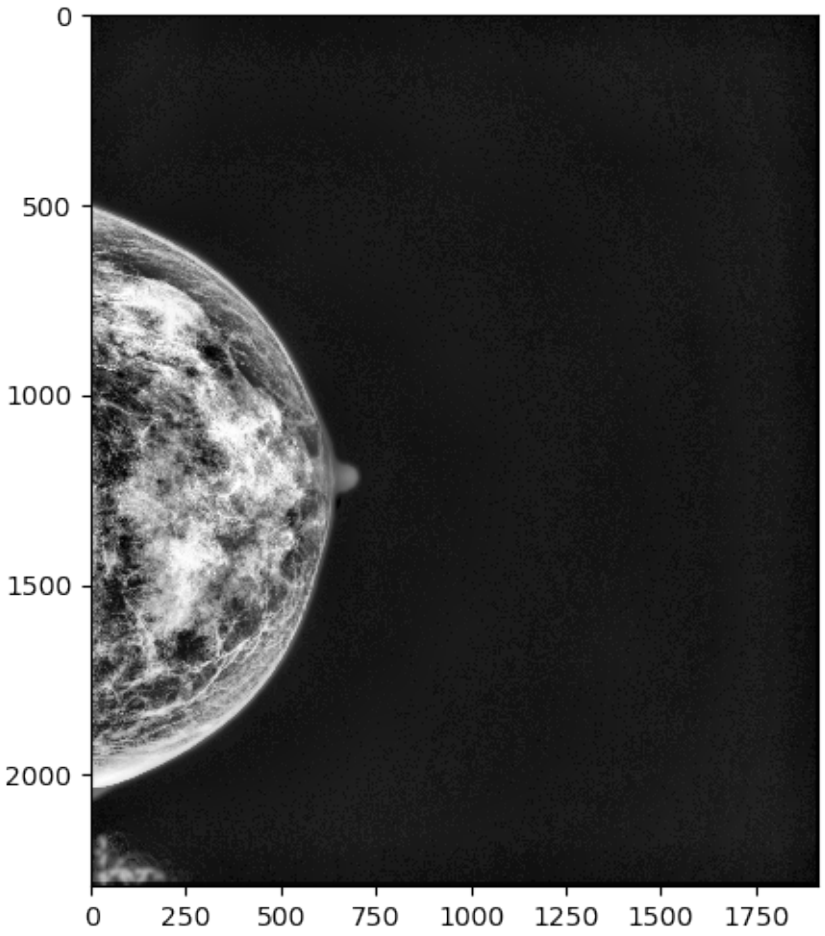
```
interactive(children=(Text(value='img_global_eq', description='image_name'), FloatSlider(value=
```

```
Box(children=(Button(description='Save .TIFF version', style=ButtonStyle()), Button(description
```

[44]:
```
import warnings
warnings.filterwarnings('ignore')
```

0.1.5 Adaptive Equalization

[46]:
```
plt.figure(figsize = (6,6), dpi=100)

img_adaptive = exposure.equalize_adapthist(ds1.pixel_array, clip_limit=0.040)
figure = plt.imshow(img_adaptive, cmap=plt.cm.gray)
```



[49]:
```
def adaptive_histogram_equalization(clip):
    plt.figure(figsize = (6,6), dpi=100)

    img_adaptive = exposure.equalize_adapthist(ds1.pixel_array, clip_limit=clip)
    plt.imshow(img_adaptive, cmap=plt.cm.gray)

    plt.show()
```

[50]:
```
w = interactive(adaptive_histogram_equalization, clip=(.005,.2,.005))
```

```
display(w)
```

```
interactive(children=(FloatSlider(value=0.1, description='clip', max=0.2, min=0.005, step=0.005
```

[55]:
```
def adaptive_histogram_equalization_plus_stretch(image_name, clip,␣
↪percentile_lo, percentile_hi):
    plt.figure(figsize = (6,6), dpi=100)

    img_adaptive = exposure.equalize_adapthist(eval(image_name),␣
↪clip_limit=clip)
#    plt.imshow(img_adaptive, cmap=plt.cm.gray)

    image_name_global = image_name+"adaptive_stretch"
    p_lo, p_hi = np.percentile(img_adaptive, (percentile_lo, percentile_hi))

    img_rescale = exposure.rescale_intensity(img_adaptive, in_range=(p_lo,␣
↪p_hi))

    # save optimized image array to global variable so other functions can use␣
↪it
    #global img_rescale_interactive, image_name_global
    img_rescale_interactive = img_rescale

    plt.figure(figsize = (6,6), dpi=100)
    plt.imshow(img_rescale, cmap=plt.cm.gray)

    plt.show()
```

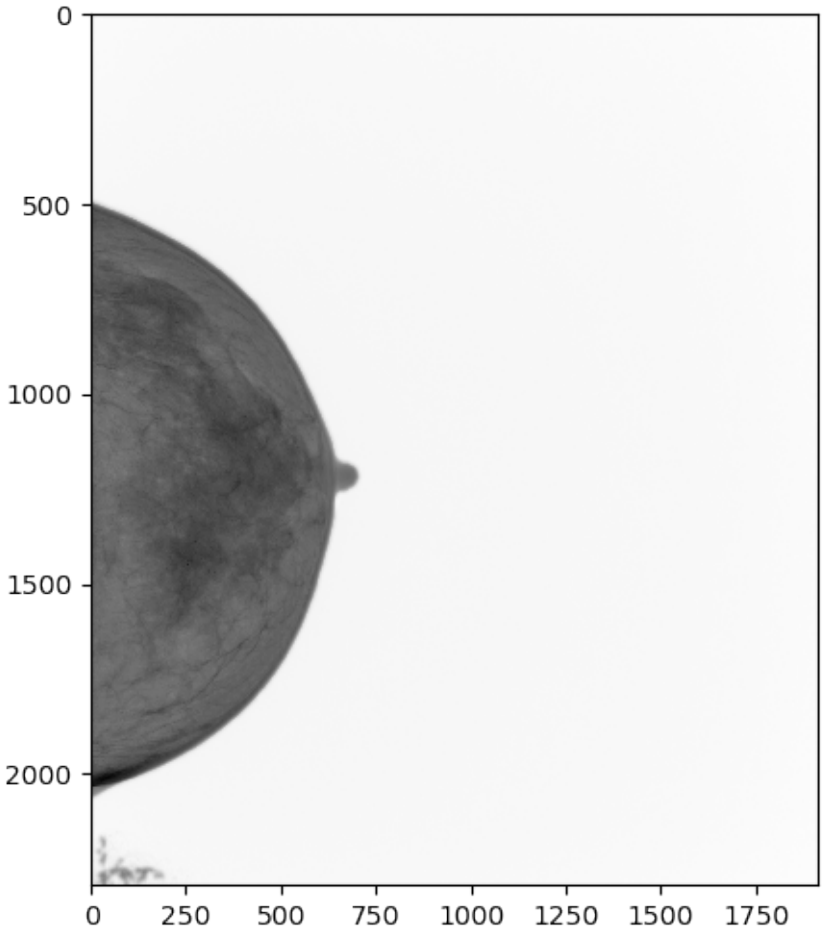
0.1.6 Adaptive Equalization with Contrast Streching

[56]:
```
w = interactive(adaptive_histogram_equalization_plus_stretch, image_name="ds1.
↪pixel_array", clip=(.005,.2,.005), percentile_lo=(1,100,.5),␣
↪percentile_hi=(1,100,.5))

display(w)

box
```

```
interactive(children=(Text(value='ds1.pixel_array', description='image_name'), FloatSlider(valu
```

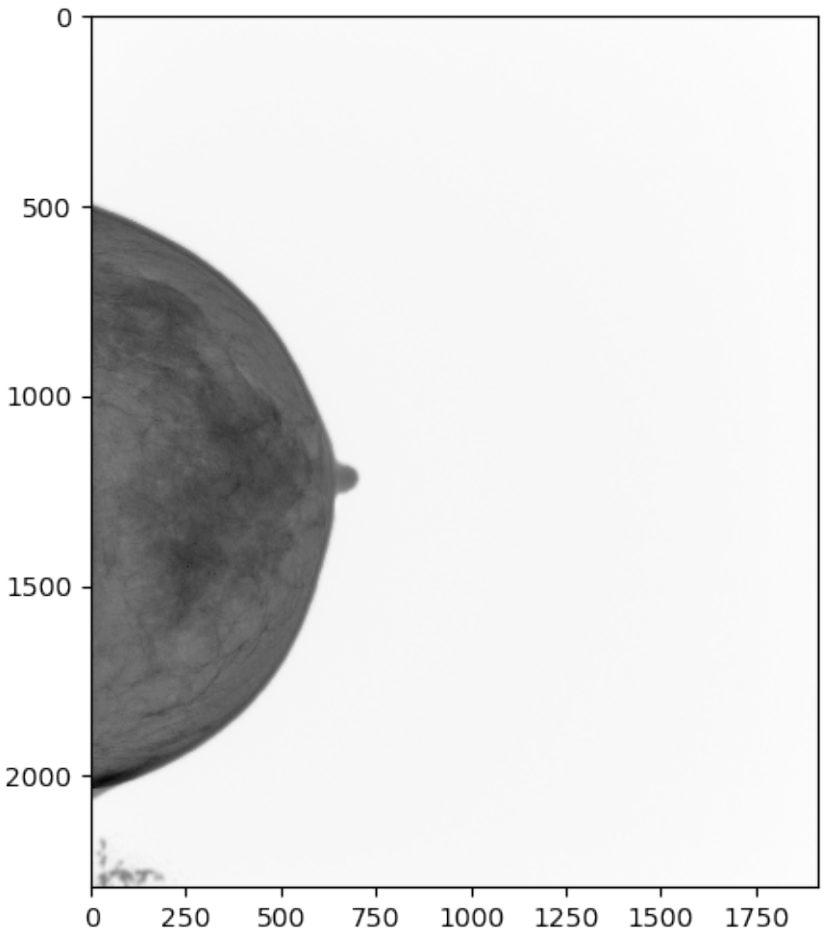
```
Box(children=(Button(description='Save .TIFF version', style=ButtonStyle()), Button(description
```

[58]:
```
plt.figure(figsize = (6,6), dpi=100)

figure = plt.imshow(ds1.pixel_array, cmap=plt.cm.gist_yarg)
```



[59]:
```
pixel_range = np.amin(ds1.pixel_array) + np.amax(ds1.pixel_array)

array_inverted = pixel_range - ds1.pixel_array

plt.figure(figsize = (6,6), dpi=100)

plt.imshow(array_inverted, cmap=plt.cm.gray)
```

[59]:
```
<matplotlib.image.AxesImage at 0x1c20883990>
```

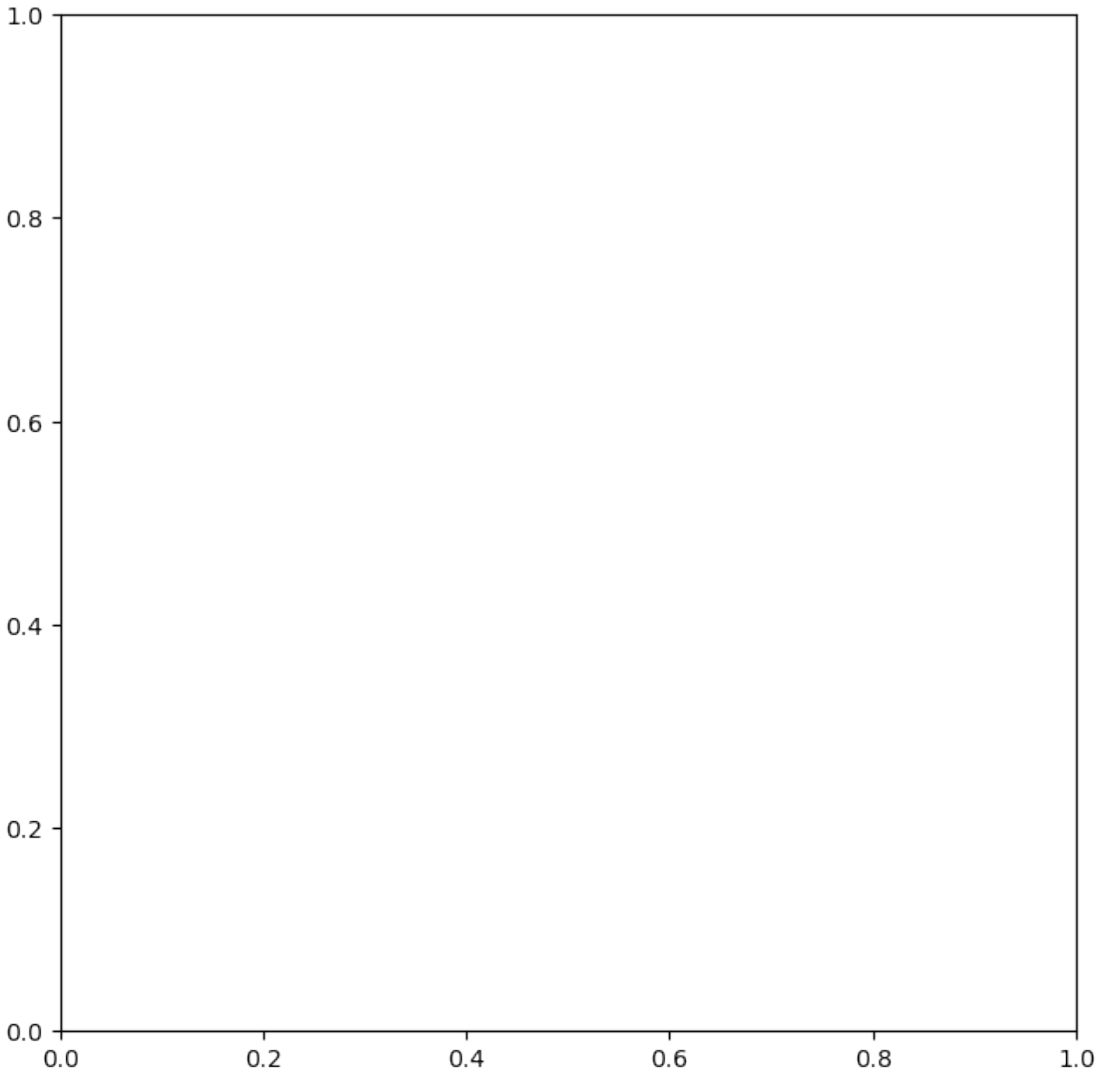


[60]:

```
      ␣
↪---------------------------------------------------------------------------

      TypeError                                 Traceback (most recent call␣
↪last)

      <ipython-input-60-f8a3afd6697a> in <module>
        3 plt.figure(figsize = (7.5,10), dpi=100) # embiggen the figure a bit
        4
  ----> 5 figure = plt.imshow(ds6.pixel_array[0])
        6 plt.show() # display the image
```

```
      ~/anaconda3/lib/python3.7/site-packages/matplotlib/pyplot.py in␣
↪imshow(X, cmap, norm, aspect, interpolation, alpha, vmin, vmax, origin,␣
↪extent, shape, filternorm, filterrad, imlim, resample, url, data, **kwargs)
     2681         filternorm=filternorm, filterrad=filterrad, imlim=imlim,
     2682         resample=resample, url=url, **({"data": data} if data is not
  -> 2683         None else {}), **kwargs)
     2684     sci(__ret)
     2685     return __ret


      ~/anaconda3/lib/python3.7/site-packages/matplotlib/__init__.py in␣
↪inner(ax, data, *args, **kwargs)
     1599     def inner(ax, *args, data=None, **kwargs):
     1600         if data is None:
  -> 1601             return func(ax, *map(sanitize_sequence, args), **kwargs)
     1602
     1603         bound = new_sig.bind(ax, *args, **kwargs)


      ~/anaconda3/lib/python3.7/site-packages/matplotlib/cbook/deprecation.py␣
↪in wrapper(*args, **kwargs)
      367                 f"%(removal)s.  If any parameter follows {name!r},␣
↪they "
      368                 f"should be pass as keyword, not positionally.")
  --> 369         return func(*args, **kwargs)
      370
      371     return wrapper


      ~/anaconda3/lib/python3.7/site-packages/matplotlib/cbook/deprecation.py␣
↪in wrapper(*args, **kwargs)
      367                 f"%(removal)s.  If any parameter follows {name!r},␣
↪they "
      368                 f"should be pass as keyword, not positionally.")
  --> 369         return func(*args, **kwargs)
      370
      371     return wrapper


      ~/anaconda3/lib/python3.7/site-packages/matplotlib/axes/_axes.py in␣
↪imshow(self, X, cmap, norm, aspect, interpolation, alpha, vmin, vmax, origin,␣
↪extent, shape, filternorm, filterrad, imlim, resample, url, **kwargs)
     5669                                 resample=resample, **kwargs)
     5670
  -> 5671         im.set_data(X)
     5672         im.set_alpha(alpha)
     5673         if im.get_clip_path() is None:
```

```
       ~/anaconda3/lib/python3.7/site-packages/matplotlib/image.py in␣
↪set_data(self, A)
       688                 or self._A.ndim == 3 and self._A.shape[-1] in [3,␣
↪4]):
       689             raise TypeError("Invalid shape {} for image data"
   --> 690                             .format(self._A.shape))
       691
       692         if self._A.ndim == 3:


       TypeError: Invalid shape (1914,) for image data
```



[ ]:

[ ]: