**Department of Computer Science and Engineering**

**National Institute of Technology, Hamirpur**

**Name: Manas Gupta**

**Roll Number: 21BCS010**

**Digital Image Processing CS-325**

**Solution Laboratory Assignment -1**

**Topic: Spatial Image Enhancement functions and interpretation of results**

Solution 1:

*from* PIL *import* Image

*import* numpy *as* np

*import* matplotlib.pyplot *as* plt

def read\_bmp(*file\_path*):

image = Image.open(*file\_path*)

*return* image

def intensity\_averaging(*image*):

*return* *image*.convert('L')

def intensity\_inversion(*image*):

*return* Image.fromarray(255 - np.array(*image*))

def sub\_sampling(*image*, *factor*):

width, height = *image*.size

new\_width, new\_height = width // *factor*, height // *factor*

*return* *image*.resize((new\_width, new\_height))

def main():

*# Replace 'input.bmp' with the path to your BMP file*

file\_path = 'MRIspineFracture.bmp'

*# Read the BMP file*

original\_image = read\_bmp(file\_path)

*# Perform intensity averaging*

averaged\_image = intensity\_averaging(original\_image)

*# Perform intensity inversion*

inverted\_image = intensity\_inversion(original\_image)

*# Perform sub-sampling with factor 2*

subsampled\_image = sub\_sampling(original\_image, *factor*=2)

*# Display the images*

plt.figure(*figsize*=(10, 5))

plt.subplot(2, 3, 1)

plt.imshow(original\_image)

plt.title('Original Image')

plt.subplot(2, 3, 2)

plt.imshow(averaged\_image, *cmap*='gray')

plt.title('Intensity Averaging')

plt.subplot(2, 3, 3)

plt.imshow(inverted\_image, *cmap*='gray')

plt.title('Intensity Inversion')

plt.subplot(2, 3, 4)

plt.imshow(original\_image)

plt.title('Original Image')

plt.subplot(2, 3, 5)

plt.imshow(subsampled\_image)

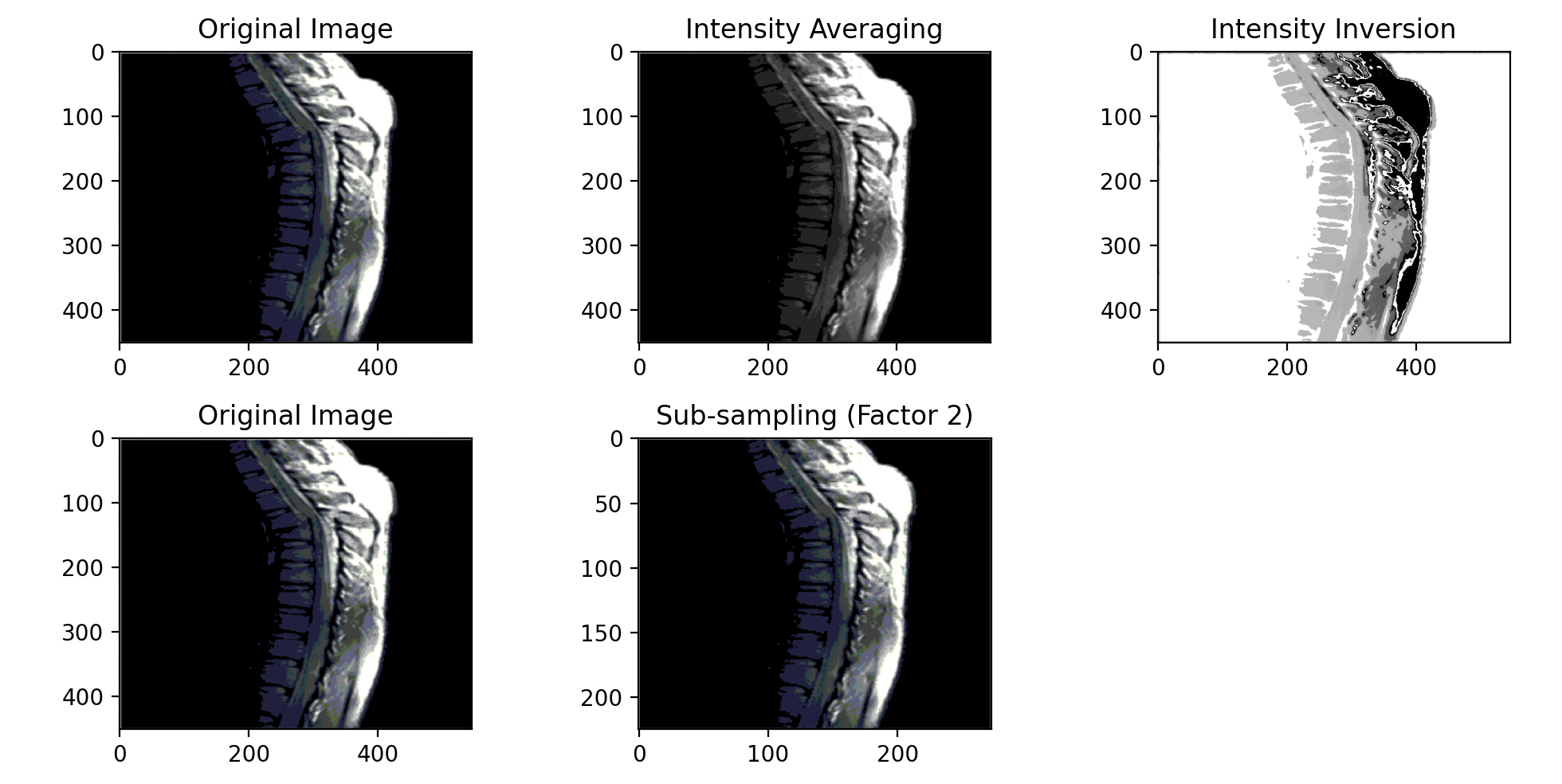
plt.title('Sub-sampling (Factor 2)')

plt.tight\_layout()

plt.show()

*if* \_\_name\_\_ == "\_\_main\_\_":

main()



Solution 2:

*import* matplotlib.pyplot *as* plt

*import* matplotlib.image *as* mpimg

*import* numpy *as* np

def contrast\_stretching(*image\_array*):

*# Minimum and maximum pixel values*

min\_val = np.min(*image\_array*)

max\_val = np.max(*image\_array*)

*# Apply contrast stretching formula*

stretched\_array = ((*image\_array* - min\_val) / (max\_val - min\_val)) \* 255.0

*# Clip values to ensure they are within the valid range*

stretched\_array = np.clip(stretched\_array, 0, 255)

*return* stretched\_array

def main():

*# Replace 'input.bmp' and 'input.pgm' with the paths to your BMP and PGM files*

bmp\_input\_path = 'boats.bmp'

pgm\_input\_path = 'lungs.pgm'

*# Read BMP file*

bmp\_image = mpimg.imread(bmp\_input\_path)

*# Apply contrast stretching to BMP*

enhanced\_bmp = contrast\_stretching(bmp\_image)

*# Save enhanced BMP using Matplotlib*

plt.imsave('enhanced\_bmp\_matplotlib.bmp', enhanced\_bmp.astype(np.uint8), *cmap*='gray')

*# Display original and enhanced BMP using Matplotlib*

plt.figure(*figsize*=(10, 5))

plt.subplot(1, 2, 1)

plt.imshow(bmp\_image, *cmap*='gray')

plt.title('Original BMP')

plt.subplot(1, 2, 2)

plt.imshow(enhanced\_bmp, *cmap*='gray')

plt.title('Enhanced BMP')

plt.show()

*# Read PGM file*

pgm\_image = np.loadtxt(pgm\_input\_path, *skiprows*=3)

*# Apply contrast stretching to PGM*

enhanced\_pgm = contrast\_stretching(pgm\_image)

*# Save enhanced PGM using Matplotlib*

plt.imsave('enhanced\_pgm\_matplotlib.pgm', enhanced\_pgm.astype(np.uint8), *cmap*='gray', *format*='pgm')

*# Display original and enhanced PGM using Matplotlib*

plt.figure(*figsize*=(10, 5))

plt.subplot(1, 2, 1)

plt.imshow(pgm\_image, *cmap*='gray')

plt.title('Original PGM')

plt.subplot(1, 2, 2)

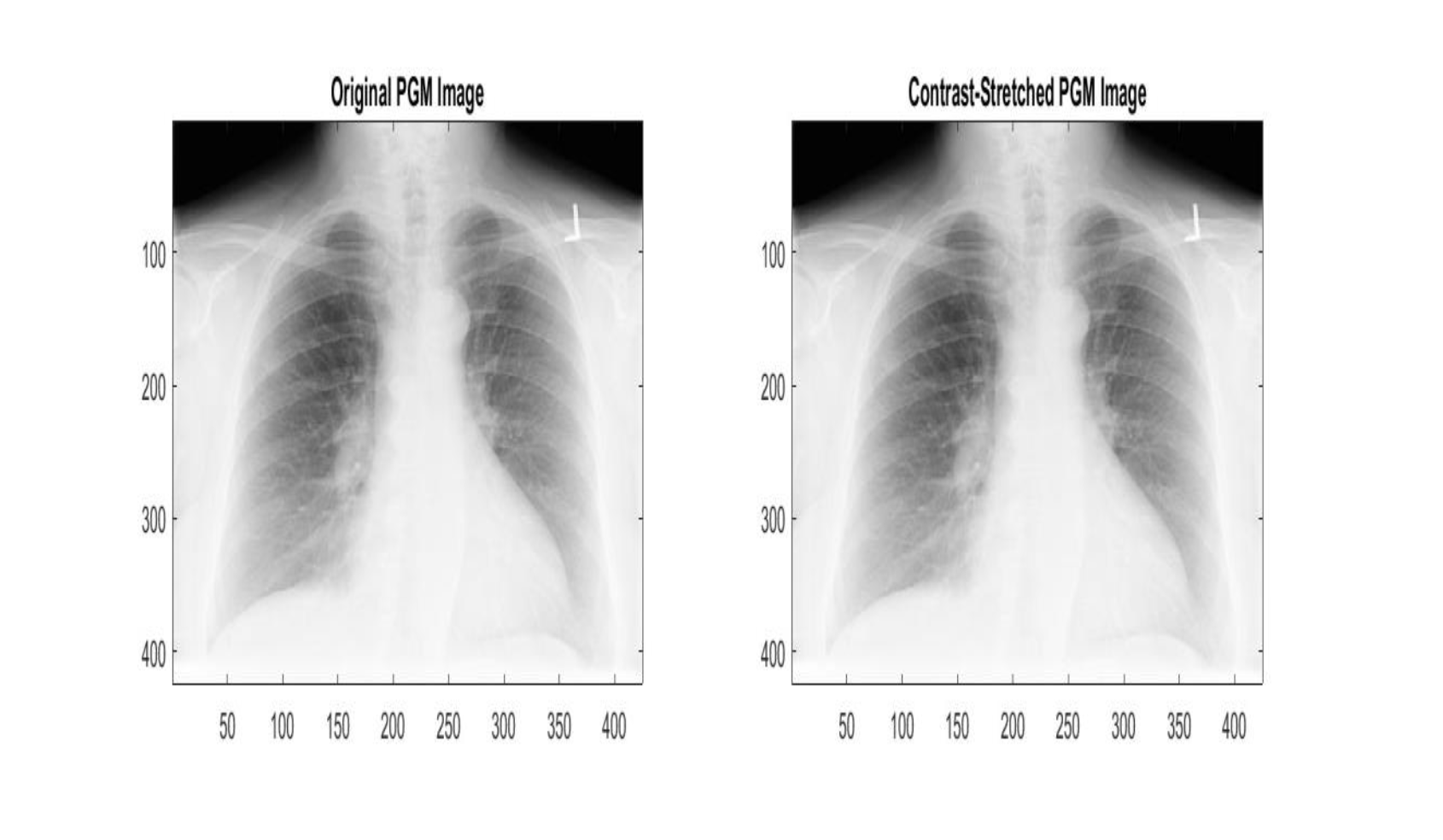
plt.imshow(enhanced\_pgm, *cmap*='gray')

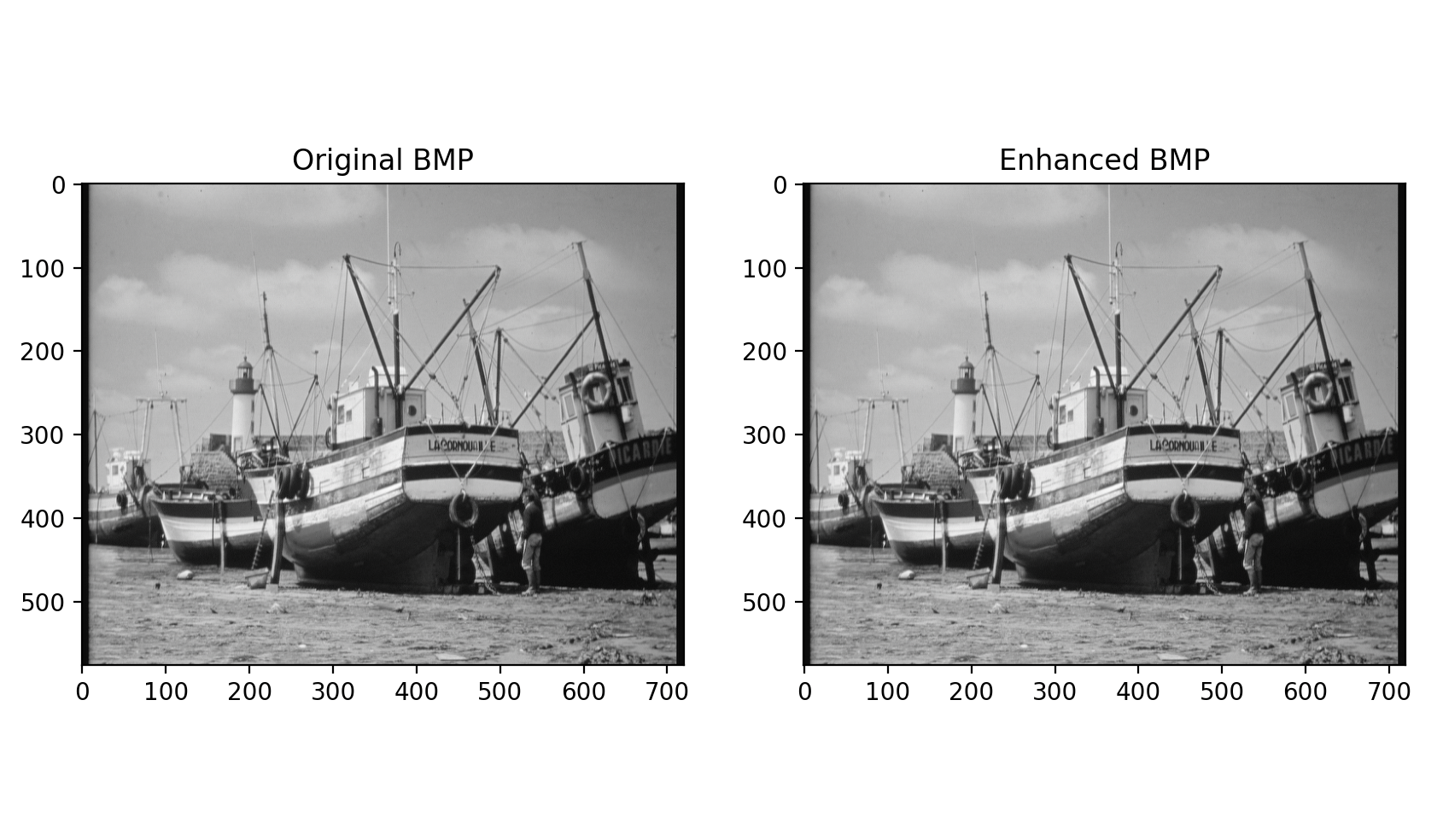
plt.title('Enhanced PGM')

plt.show()

*if* \_\_name\_\_ == "\_\_main\_\_":

main()





(b)

*from* PIL *import* Image

*import* numpy *as* np

*import* matplotlib.pyplot *as* plt

def log\_transform\_bmp(*image*):

*# Convert BMP image to numpy array*

img\_array = np.array(*image*)

*# Add a small constant to avoid taking the log of zero*

transformed\_array = np.log1p(img\_array)

*# Normalize to the range [0, 255]*

transformed\_array = (transformed\_array / np.max(transformed\_array)) \* 255.0

*# Convert back to uint8*

transformed\_array = transformed\_array.astype(np.uint8)

*# Create a new Image object from the transformed array*

transformed\_image = Image.fromarray(transformed\_array)

*return* transformed\_image

def main\_bmp():

*# Replace 'input.bmp' with the path to your BMP file*

bmp\_input\_path = 'boats.bmp'

*# Read BMP file*

bmp\_image = Image.open(bmp\_input\_path)

*# Apply log transform to BMP*

transformed\_bmp = log\_transform\_bmp(bmp\_image)

transformed\_bmp.save('transformed\_bmp\_log.bmp')

plt.figure(*figsize*=(10, 5))

plt.subplot(1, 2, 1)

plt.imshow(bmp\_image, *cmap*='gray')

plt.title('Original BMP')

plt.subplot(1, 2, 2)

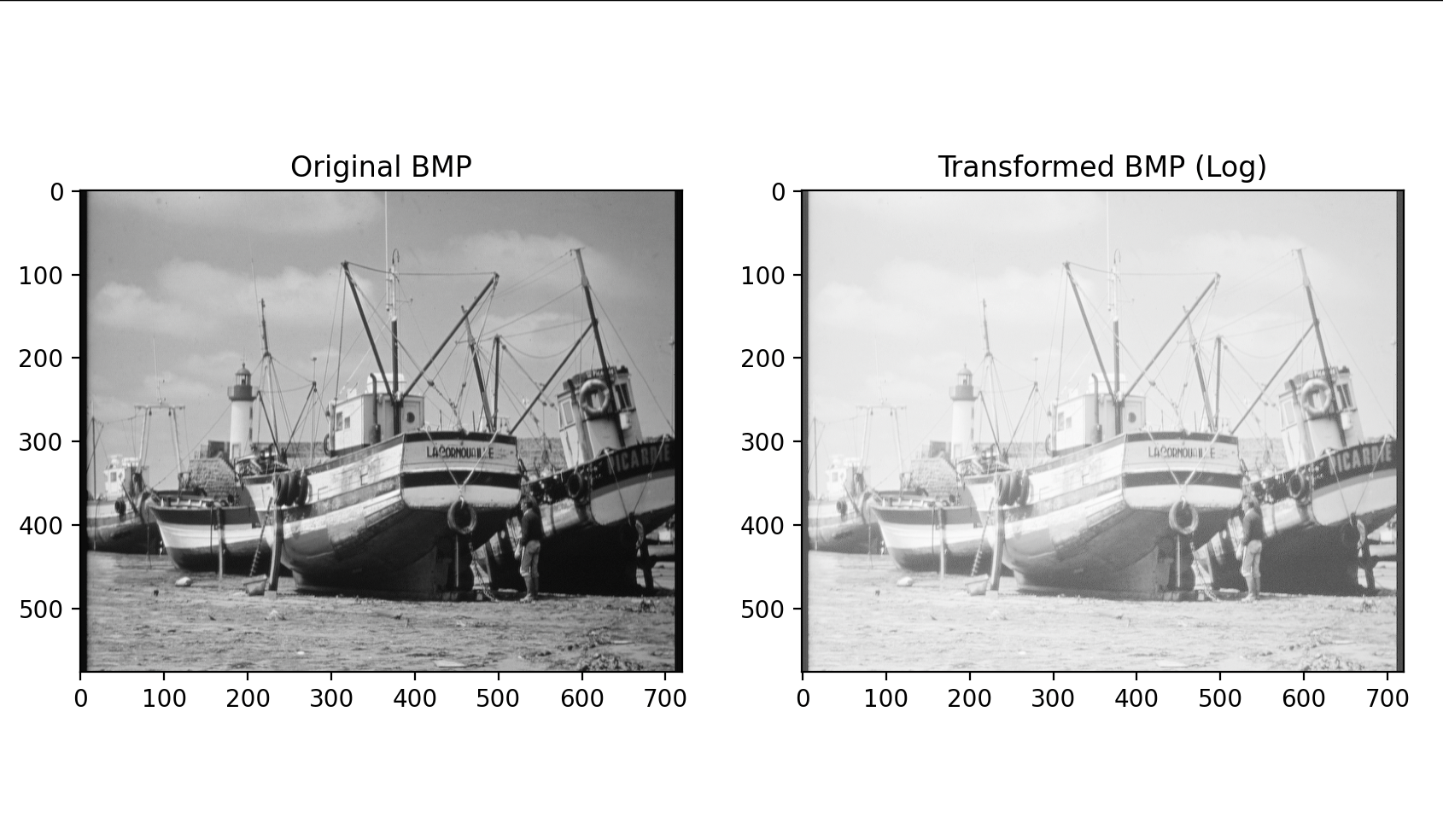
plt.imshow(transformed\_bmp, *cmap*='gray')

plt.title('Transformed BMP (Log)')

plt.show()

*if* \_\_name\_\_ == "\_\_main\_\_":

main\_bmp()



*import* matplotlib.pyplot *as* plt

*import* numpy *as* np

def log\_transform\_pgm(*image\_array*):

*# Add a small constant to avoid taking the log of zero*

transformed\_array = np.log1p(*image\_array*)

*# Normalize to the range [0, 255]*

transformed\_array = (transformed\_array / np.max(transformed\_array)) \* 255.0

*return* transformed\_array

def main\_pgm():

*# Replace 'input.pgm' with the path to your PGM file*

pgm\_input\_path = 'lungs.pgm'

*# Read PGM file*

pgm\_image = np.loadtxt(pgm\_input\_path, )

*# Apply log transform to PGM*

transformed\_pgm = log\_transform\_pgm(pgm\_image)

*# Save transformed PGM using Matplotlib*

plt.imsave('transformed\_pgm\_log.pgm', transformed\_pgm.astype(np.uint8), *cmap*='gray', *format*='pgm')

*# Display original and transformed PGM using Matplotlib*

plt.figure(*figsize*=(10, 5))

plt.subplot(1, 2, 1)

plt.imshow(pgm\_image, *cmap*='gray')

plt.title('Original PGM')

plt.subplot(1, 2, 2)

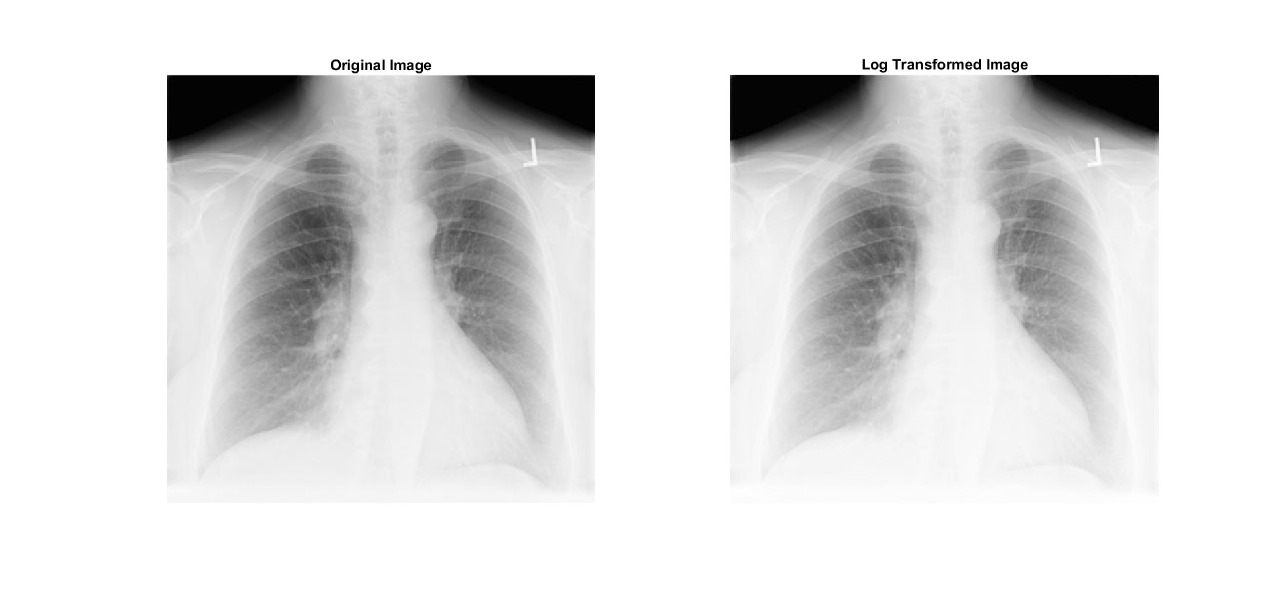
plt.imshow(transformed\_pgm, *cmap*='gray')

plt.title('Transformed PGM (Log)')

plt.show()

*if* \_\_name\_\_ == "\_\_main\_\_":

main\_pgm()



(c)

*from* PIL *import* Image

*import* numpy *as* np

*import* matplotlib.pyplot *as* plt

def power\_law\_transform\_bmp(*image*, *gamma*):

*# Convert BMP image to numpy array*

img\_array = np.array(*image*)

*# Normalize pixel values to the range [0, 1]*

normalized\_array = img\_array / 255.0

*# Apply power-law transform*

transformed\_array = np.power(normalized\_array, *gamma*)

*# Scale back to the range [0, 255]*

transformed\_array = (transformed\_array \* 255).astype(np.uint8)

*# Create a new Image object from the transformed array*

transformed\_image = Image.fromarray(transformed\_array)

*return* transformed\_image

def main\_bmp():

*# Replace 'input.bmp' with the path to your BMP file*

bmp\_input\_path = 'MRIspineFracture.bmp'

*# Read BMP file*

bmp\_image = Image.open('MRIspineFracture.bmp')

*# Set the gamma value (adjust as needed)*

gamma = 1.5

*# Apply power-law transform to BMP*

transformed\_bmp = power\_law\_transform\_bmp(bmp\_image, gamma)

transformed\_bmp.save(f'transformed\_bmp\_power\_law\_gamma\_{gamma}.bmp')

*# Display original and transformed BMP using Matplotlib*

plt.figure(*figsize*=(10, 5))

plt.subplot(1, 2, 1)

plt.imshow(bmp\_image, *cmap*='gray')

plt.title('Original BMP')

plt.subplot(1, 2, 2)

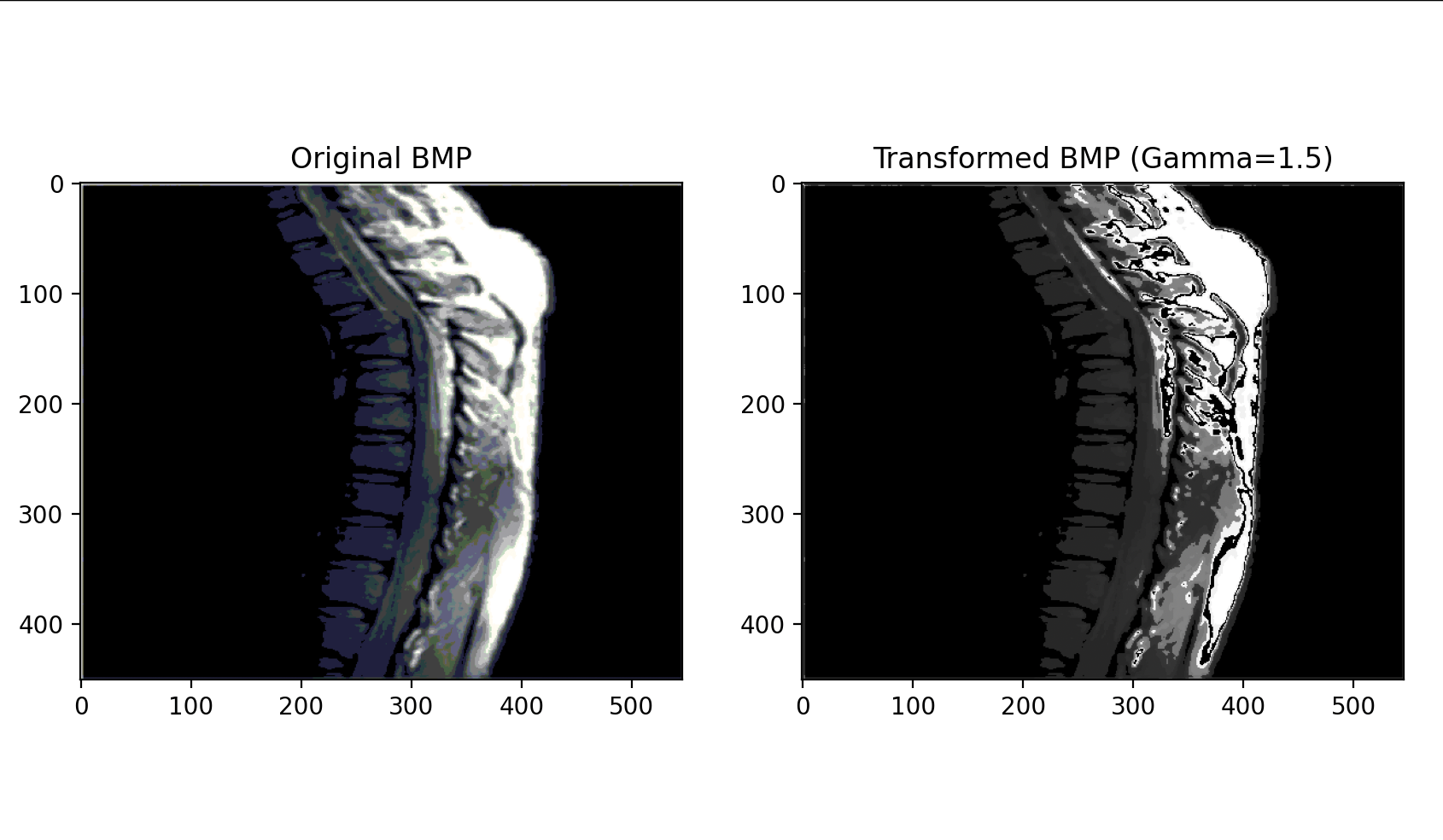
plt.imshow(transformed\_bmp, *cmap*='gray')

plt.title(Transformed BMP (Gamma={gamma})')

plt.show()

*if* \_\_name\_\_ == "\_\_main\_\_":

main\_bmp()



*import* matplotlib.pyplot *as* plt

*import* numpy *as* np

def power\_law\_transform\_pgm(*image\_array*, *gamma*):

*# Normalize pixel values to the range [0, 1]*

normalized\_array = *image\_array* / np.max(*image\_array*)

*# Apply power-law transform*

transformed\_array = np.power(normalized\_array, *gamma*)

*# Scale back to the range [0, 255]*

transformed\_array = (transformed\_array \* 255).astype(np.uint8)

*return* transformed\_array

def main\_pgm():

*# Replace 'input.pgm' with the path to your PGM file*

pgm\_input\_path = 'lungs.pgm'

*# Read PGM file*

pgm\_image = np.loadtxt('lungs.pgm', )

*# Set the gamma value (adjust as needed)*

gamma = 1.6

*# Apply power-law transform to PGM*

transformed\_pgm = power\_law\_transform\_pgm(pgm\_image, gamma)

*# Save transformed PGM using Matplotlib*

plt.imsave(f'transformed\_pgm\_power\_law\_gamma\_{gamma}.pgm', transformed\_pgm.astype(np.uint8), *cmap*='gray', *format*='pgm')

*# Display original and transformed PGM using Matplotlib*

plt.figure(*figsize*=(10, 5))

plt.subplot(1, 2, 1)

plt.imshow(pgm\_image, *cmap*='gray')

plt.title('Original PGM')

plt.subplot(1, 2, 2)

plt.imshow(transformed\_pgm, *cmap*='gray')

plt.title(f'Transformed PGM (Gamma={gamma})')

plt.show()

*if* \_\_name\_\_ == "\_\_main\_\_":

main\_pgm()

