Assignment 1

ECE1512

Due: October 1, 2025

Data handed in: September 29, 2025

GitHub repository

Technical discussion. Traditional image processing is often an important preprocessing step, especially when preparing images for classical computer vision tasks such as classification or segmentation. These methods enhance and clean images, making subsequent analysis more effective. In this assignment, I focused on histogram analysis and equalization, intensity transformations, and specialized point functions.

Digital images consist of pixels located at spatial coordinates (x,y). Each pixel carries an intensity value: for an 8-bit grayscale image, values range from 0 (black) to 255 (white), while color images contain three channels (red, green, and blue), each ranging from 0 to 255. In this assignment, the focus is on grayscale images, specifically bimodal grayscale images, which are characterized by histograms with two dominant intensity peaks that often correspond to background and foreground regions.

The intensity of a pixel is the grayscale value at its location, and transformations on intensity can be applied either at the point level, where each pixel is processed independently, or at the neighborhood level, where surrounding pixels are also considered [1, p. 119]. In general, these operations can be described as a mapping function $s=T(r)$ where r is the input intensity and s is the transformed output intensity.

One widely used nonlinear transformation is the logarithmic transformation, defined as $s=c\cdot\log(1+r)$ where c is a positive optimizable constant [1, p. 124]. This transformation expands the range of dark regions with low intensity values and compresses bright regions with high intensity values. It is particularly useful for revealing hidden details in shadows or low-contrast parts of the image. Another important nonlinear transformation is the power-law or gamma transformation, defined as $s=c\cdot r^{\gamma}$, where c and γ are positive optimizable constants [125]. The effect of this transformation depends on the value of γ. When $\gamma<1$, bright regions are expanded while dark regions are compressed, and when $\gamma>1$, dark regions are expanded while bright regions are compressed.

For bimodal grayscale images, these transformations can help enhance the visibility of details that may otherwise be obscured in the original image, improving contrast and preparing the image for later processing steps such as segmentation.

Histogram of an image can describe the distribution of intensity values in an image which summarizes the probability and distribution of intensity values and explains how uniform or non-uniform the distribution is, how many peaks there are and how widespread the peaks are. Histogram shape of an image can also represent how the image appears [1, p. 133]. The darker grayscale images will have histogram concentrated on lower intensity values, brighter grayscale

images will have histogram concentrated on higher intensity values, uniform histogram gives the best contrast showing the most details [1, p. 134].

Formally, the histogram counts the number of pixels with a given intensity and expresses the proportion of that intensity relative to the total number of pixels. This can be written as

$$p(r_k) = \frac{h(r_k)}{MN} = \frac{n_k}{MN}$$

where MN is the total number of pixels (rows ×\times× columns), r_k is a specific intensity value, and n_k is the number of pixels with that intensity [1, p. 133]. It is also possible to bin intensities into ranges and represent the frequency of each bin. The normalized histogram is always a probability distribution, so the sum across all values (or bins) is equal to 1.

Histogram equalization is another type of intensity transformation that maps the original histogram of an image into a more uniform distribution. This increases the overall contrast and reveals more image details. For the transformation to be valid, two conditions must hold: (1) the mapping function s=T(r) where r is the input intensity and s is the output intensity, must be monotonic increasing to preserve order, and (2) the output intensity range must match the original image range. These conditions help avoid artifacts and ensure a proper transformation [1, p. 135].

The transformation function for histogram equalization is based on the cumulative distribution function (CDF) of the image's intensity histogram. It is given by

$$s = T(r) = (L\text{-}1)^* \int_0^r p_r(w)dw$$

where r is the original intensity, s is the transformed intensity, L−1 is the maximum intensity value (usually 255), $p_r(w)$ is the original probability distribution of intensities, and w is the dummy variable of integration [1, p. 136]. The integral represents the cumulative probability of intensities from 0 up to r. Multiplying by L−1 ensures that the output values are mapped back into the original intensity range.

The result of this transformation is that the output intensities follow a distribution that is ideally uniform, meaning that $p_s(s) \approx \frac{1}{L\text{-}1}$. In practice, with discrete images, the result is not perfectly uniform, but the contrast is significantly improved. For discrete intensity values, the integral is replaced by a summation over histogram bins, which serves as the discrete cumulative distribution function.

Discussion of results. As shown in Figure 1, row 1, the original image is bimodal, with two main intensity peaks around 0 and 255. The fractured spine is visible along the vertical middle of the

image, depicted in bright white to gray colors. Some parts of the spine are difficult to identify due to dark shadows or uneven lighting, as highlighted by the light blue circle in Figure 1, row 1.

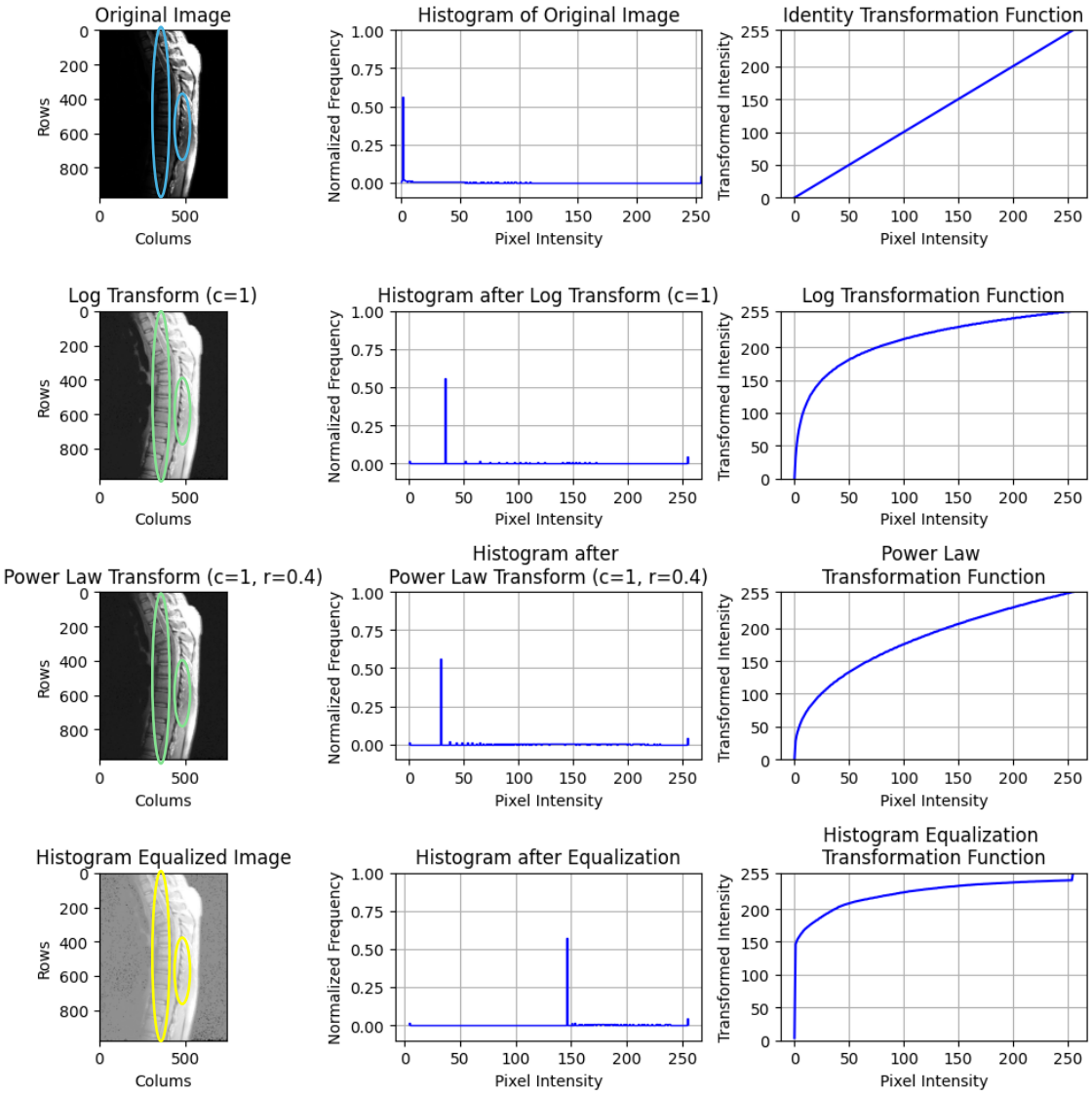

Figure 1. Visualization of image processing techniques. Each row corresponds to a different method: (1) original image with histogram and identity transformation, (2) log transformation, (3) power-law transformation, (4) histogram equalization.

The log transformation (Figure 2, row 2) brightened the important regions of the fractured spine. Results were similar for c ≥0.1 up to 5, indicating that log transformation can enhance image details with small c values, without extensive parameter tuning. Log transformation primarily extended low-intensity values to higher intensities, effectively compressing the dynamic range.

Power-law (gamma) transformation required more careful experimentation. Parameter $c \geq 0.1$ already provided near-optimal results, but gamma had a strong impact. Gamma values ≤ 0.4 brightened the image substantially. Gamma values > 0.4 and ≤ 0.6 enhanced the bright regions of the spine while preserving the very dark regions. However, gamma values > 0.8 and ≤ 1.8 reduced high-intensity values and preserved low intensities, resulting in overall darker images and loss of detail. Appendix Figure 3 shows these experiments. Based on these observations, gamma = 0.4 was chosen, as it enhanced spine details while preserving low-intensity contours for high contrast with the background.

Histogram equalization was challenging for this bimodal image. Due to the high proportion of minimum and maximum intensity values, histogram equalization failed to produce a uniform output histogram. Instead, it produced an overall brighter image, with output intensities concentrated at higher values, but failed to preserve contrast in the fractured bone regions.

In conclusion, the log transformation with $c \geq 0.1$ was the most effective technique for enhancing the fractured spine image, as it preserved contrast between foreground and background. However, it slightly reduced fine bone details due to brightening. For scenarios where preserving detailed anatomy is critical, a carefully optimized power-law transformation provides better contrast preservation and detail enhancement.

Citations.

[1] Gonzalez, R. C. and Woods, R. E. [2018]. Digital Image Processing, 4th ed., Pearson/Prentice Hall, NY.

Appendix.

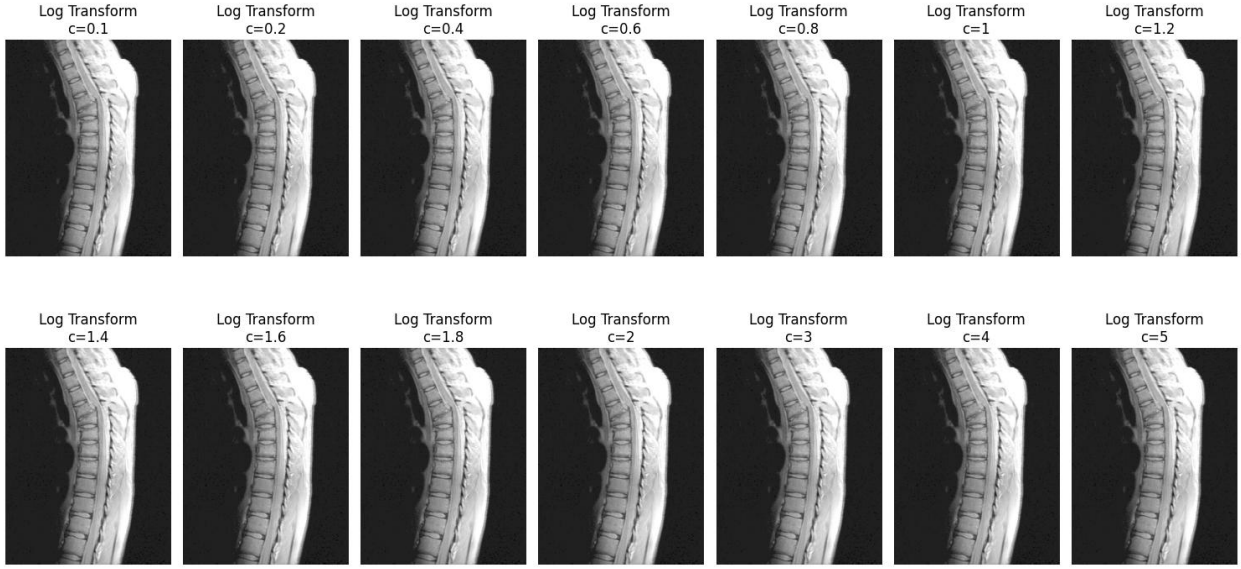

Figure 2. The results of log transformation for various c values.

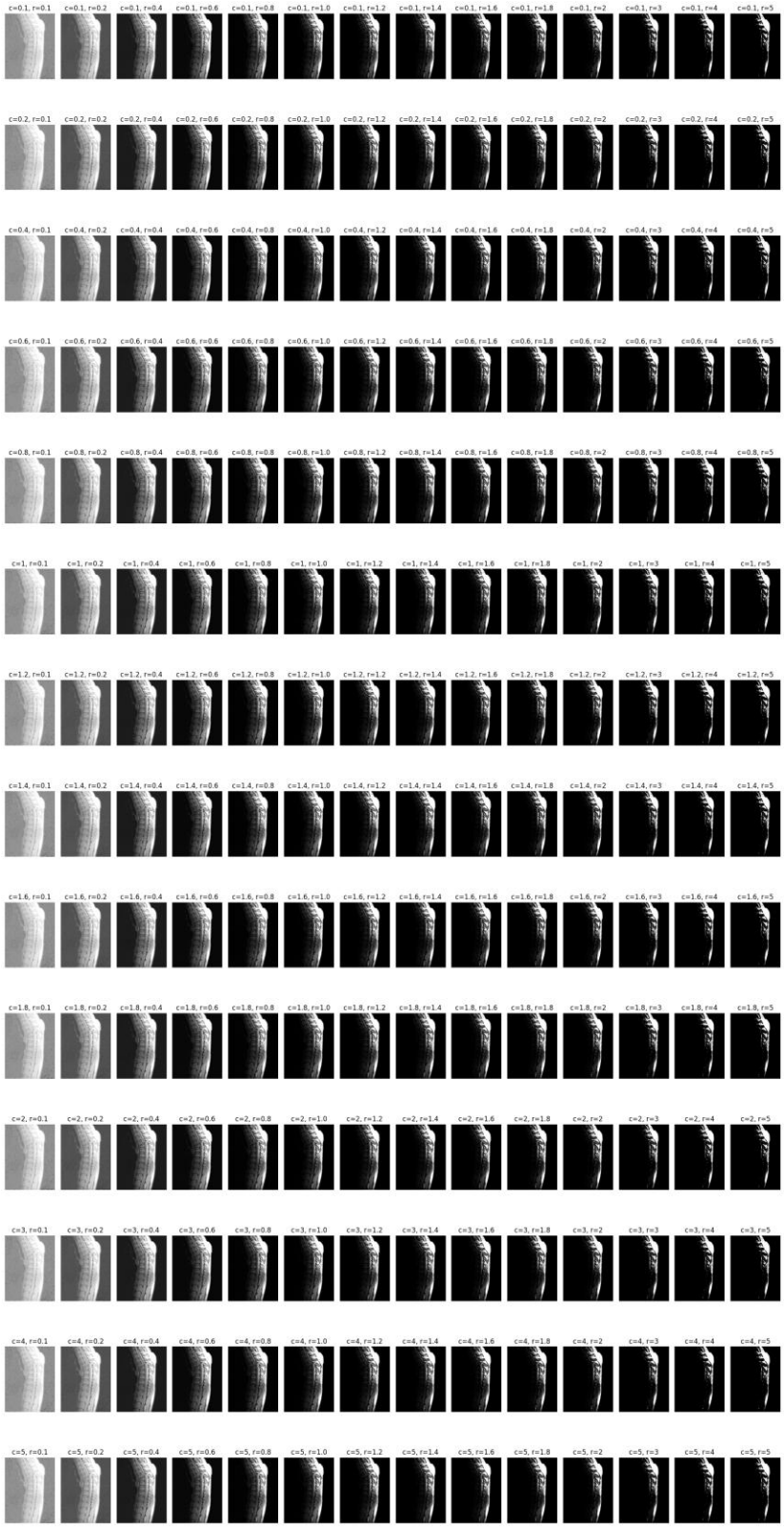

Figure 3. The power law transformation for various c and gamma values.

Libraries used:

Numpy

Matplotlib

Scikit-image

Experimental parameters:

c_values = [0.1, 0.2, 0.4, 0.6, 0.8, 1, 1.2, 1.4, 1.6, 1.8, 2, 3, 4, 5]

r_values = [0.1,0.2, 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6, 1.8, 2, 3,4,5]

Code for log transformation:

```
def log_transform(image, c=1.0):
    image = image.astype(np.uint8)
    image_log = c * np.log1p(image)
    image_log = image_log / np.max(image_log)  # Normalize to [0,1]
    image_log = img_as_ubyte(image_log)
    return image_log
```

Code for power law transformation:

```
def power_law_transform(image, c, r):
    image = image.astype(np.uint8)
    image_gamma = c * np.power(image, r)
    image_gamma = image_gamma / np.max(image_gamma)  # Normalize to [0,1]
    image_gamma = img_as_ubyte(image_gamma)
    return image_gamma
```

Code for histogram equalization:

```
image_eq = exposure.equalize_hist(original_image)
image_eq = img_as_ubyte(image_eq)
```

Code for producing figure in results:

```
figure, subplots = plt.subplots(4, 3, figsize=(10,10))


subplots[0, 0].imshow(original_image, cmap='gray', vmin=original_image.min(),
vmax=original_image.max())

subplots[0, 0].set_title('Original Image')

subplots[0,0].set_xlabel('Colums')

subplots[0,0].set_ylabel('Rows')


subplots[0, 1].hist(original_image.ravel(), bins=256, histtype='step',  density=True, color='black')

subplots[0, 1].set_title('Histogram of Original Image')

subplots[0,1].set_xlim([-5,255])

subplots[0,1].set_ylim([0,1])

subplots[0,1].set_xlabel('Pixel Intensity')

subplots[0,1].set_ylabel('Normalized Frequency')

subplots[0,1].grid(True)


subplots[0, 2].plot(np.arange(0,256), np.arange(0,256), color='blue')

subplots[0,2].set_title('Identity Transformation Function')

subplots[0,2].set_xlabel('Pixel Intensity')

subplots[0,2].set_ylabel('Transformed Intensity')

subplots[0,2].set_ylim([0,255])

# subplots[0,2].set_yticklabels([0,50,100,150,200,255])

subplots[0,2].set_yticks([0,50,100,150,200,255])

subplots[0,2].grid(True)
```

```
# log transformation with c=1
img_log = log_transform(original_image, c=1)
subplots[1, 0].imshow(img_log, cmap='gray', vmin=img_log.min(), vmax=img_log.max())
subplots[1, 0].set_title('Log Transform (c=1)')
subplots[1,0].set_xlabel('Colums')
subplots[1,0].set_ylabel('Rows')
subplots[1, 1].hist(img_log.ravel(), bins=256, histtype='step', density=True, color='black')
subplots[1, 1].set_title('Histogram after Log Transform (c=1)')
subplots[0,1].set_xlim([-5,255])
subplots[1,1].set_ylim([0,1])
subplots[1,1].set_xlabel('Pixel Intensity')
subplots[1,1].set_ylabel('Normalized Frequency')
subplots[1,1].grid(True)

intensities = np.arange(0, 256)
subplots[1,2].plot(intensities,
img_as_ubyte(np.log1p(intensities)/np.max(np.log1p(intensities))), color='blue')
subplots[1,2].set_title('Log Transformation Function')
subplots[1,2].set_xlabel('Pixel Intensity')
subplots[1,2].set_ylabel('Transformed Intensity')
subplots[1,2].set_ylim([0,255])
# subplots[1,2].set_yticklabels([0,50,100,150,200,255])
subplots[1,2].set_yticks([0,50,100,150,200,255])
subplots[1,2].grid(True)

# power law transformation with c=1, r=0.6
```

```
img_gamma = power_law_transform(original_image, c=1, r=0.4)

subplots[2, 0].imshow(img_gamma, cmap='gray', vmin=img_gamma.min(),
vmax=img_gamma.max())

subplots[2, 0].set_title('Power Law Transform (c=1, r=0.4)')

subplots[2,0].set_xlabel('Colums')

subplots[2,0].set_ylabel('Rows')

subplots[2, 1].hist(img_gamma.ravel(), bins=256, histtype='step', density=True, color='black')

subplots[2, 1].set_title('Histogram after \nPower Law Transform (c=1, r=0.4)')

subplots[0,1].set_xlim([-5,255])

subplots[2,1].set_ylim([0,1])

subplots[2,1].set_xlabel('Pixel Intensity')

subplots[2,1].set_ylabel('Normalized Frequency')

subplots[2,1].grid(True)


intensities = np.arange(0, 256)

subplots[2,2].plot(intensities, img_as_ubyte((intensities/255)**0.4), color='blue')

subplots[2,2].set_title('Power Law \nTransformation Function')

subplots[2,2].set_xlabel('Pixel Intensity')

subplots[2,2].set_ylabel('Transformed Intensity')

subplots[2,2].set_ylim([0,255])

# subplots[2,2].set_yticklabels([0,50,100,150,200,255])

subplots[2,2].set_yticks([0,50,100,150  ,200,255])

subplots[2,2].grid(True)


# histogram equalization

image_eq = exposure.equalize_hist(original_image)
```

```
image_eq = img_as_ubyte(image_eq)
subplots[3, 0].imshow(image_eq, cmap='gray', vmin=image_eq.min(), vmax=image_eq.max())
subplots[3, 0].set_title('Histogram Equalized Image')
subplots[3,0].set_xlabel('Colums')
subplots[3,0].set_ylabel('Rows')
subplots[3, 1].hist(image_eq.ravel(), bins=256, histtype='step', density=True, color='black')
subplots[3, 1].set_title('Histogram after Equalization')
subplots[0,1].set_xlim([-5,255])
subplots[3,1].set_ylim([0,1])
subplots[3,1].set_xlabel('Pixel Intensity')
subplots[3,1].set_ylabel('Normalized Frequency')
subplots[3,1].grid(True)

hist, bins = np.histogram(original_image.flatten(), bins=256, range=[0,256], density=True)
cdf = hist.cumsum()
cdf = cdf *255  # Normalize

subplots[3,2].plot(bins[:-1], cdf, color='blue')
subplots[3,2].set_title('Histogram Equalization \nTransformation Function')
subplots[3,2].set_xlabel('Pixel Intensity')
subplots[3,2].set_ylabel('Transformed Intensity')
subplots[3,2].set_ylim([0,255])
# subplots[3,2].set_yticklabels([0,50,100,150,200,255])
subplots[3,2].set_yticks([0,50,100,150,200,255])
subplots[3,2].grid(True)
```

```
plt.tight_layout()
```