安徽大学《数字图像处理（双语）》实验报告（2）

学号： WA2214014 专业： 人工智能 姓名及签字： 杨跃浙

实验日期： 24.12.04 实验成绩： 教师签字：

**【实验名称】: MATLAB入门及数字图像处理编程基础**

**【实验目的】：**

**1. 熟悉和掌握基于点处理的图像增强原理**

**2. 通过MATLAB编程实现图像求反的增强技术**

**3. 熟悉和掌握图像灰度等级分辨率的含义并通过MATLAB改变图像的灰度等级分辨率**

**【实验内容】**

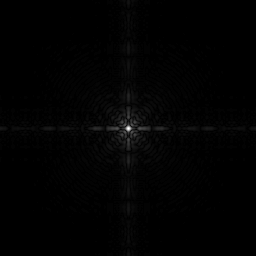
# PROJECT 03-01

## Image Enhancement Using Log Transformations

The focus of this project is to experiment with intensity transformations to enhance an image.

Enhance the image “Fig\_DFT\_no\_log.tif” by the log transformation of Eq.: *s* = *clog* (1 + *r*).

Change the only free parameter *c* until (according to your judgment) you have the best visual result for the transformation



Fig\_DFT\_no\_log.tif

# PROJECT 03-02

## Image Enhancement Using Power-law Transformations

Enhance the image “Fig\_fractured\_spine.tif” by a power-law transformation of the form

shown in Eq.:*s*  *cr*

. Change the two parameters, *c* and *r* until (according to your

judgment) you have the best visual result for the transformation.



Fig\_fractured\_spine.tif

# PROJECT 03-03

## Image Enhancement Using Thresholding

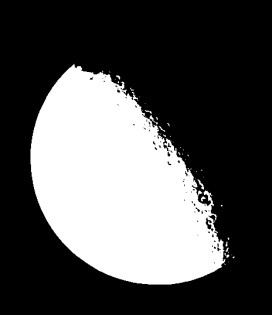
1. Write a MATLAB function **averageIntensity** which calculates the average intensity level of an image. From the command line use this function on the image “Fig\_blurry\_moon.tif”.

**(Hint**: To calculate the average intensity of the pixels in an image simply iterate through every pixel in the image, summing all of their values and finally divide this sum by the total number of pixels.)

1. Write a MATLAB function **thresholdImage** which thresholds an image based on a threshold level given as a parameter to the function. The function should take two parameters – the image to be thresholded and the threshold level. The result of the function should be a new thresholded image. This function would be called as follows:

## ThresholdedMoon = thresholdImage(Moon, ave);

Use this new function from the command line on the image to give images similar to the following:



Thresholding means that a new image is generated in which each pixel has intensity 1.0 if the corresponding pixel in the original image has a value above the threshold and 0 otherwise.

**【实验代码和结果】**

**Project03-01：**

**Code:**

**Project1.m:**

% Read the original image

image1 = imread("/Users/youngbean/Desktop/Experiments in Digital Image Processing/Class2/Data/Fig\_DFT\_no\_log.tif");

% Normalize the original image to the range [0, 1]

% Convert the image to double type and divide by 255 to map pixel values to [0, 1]

image1 = double(image1) / 255;

% Define the range for c (different scaling factors for log transformation)

c = [0.1, 0.5, 1, 1.5, 2, 3, 5, 10]; % List of c values to test

num\_c = length(c); % Get the total number of c values

% Create a new figure window for displaying results

figure;

% Display the original image in the first subplot

subplot(ceil((num\_c + 1) / 3), 3, 1); % The first subplot is reserved for the original image

imshow(image1); % Display the normalized original image

title('Original Image'); % Set the title for the original image

% Loop through all c values and display the log-transformed results

for i = 1:num\_c

% Apply the log transform with the current c value

% 9 \* image1 amplifies the normalized image before applying the log

% log10 is used for base-10 logarithm, and +1 ensures no log(0) occurs

image2 = c(i) \* log10(9 \* image1 + 1);

% Display the transformed image in the subplot

subplot(ceil((num\_c + 1) / 3), 3, i + 1); % Place the result in the appropriate subplot

imshow(image2); % Display the transformed image

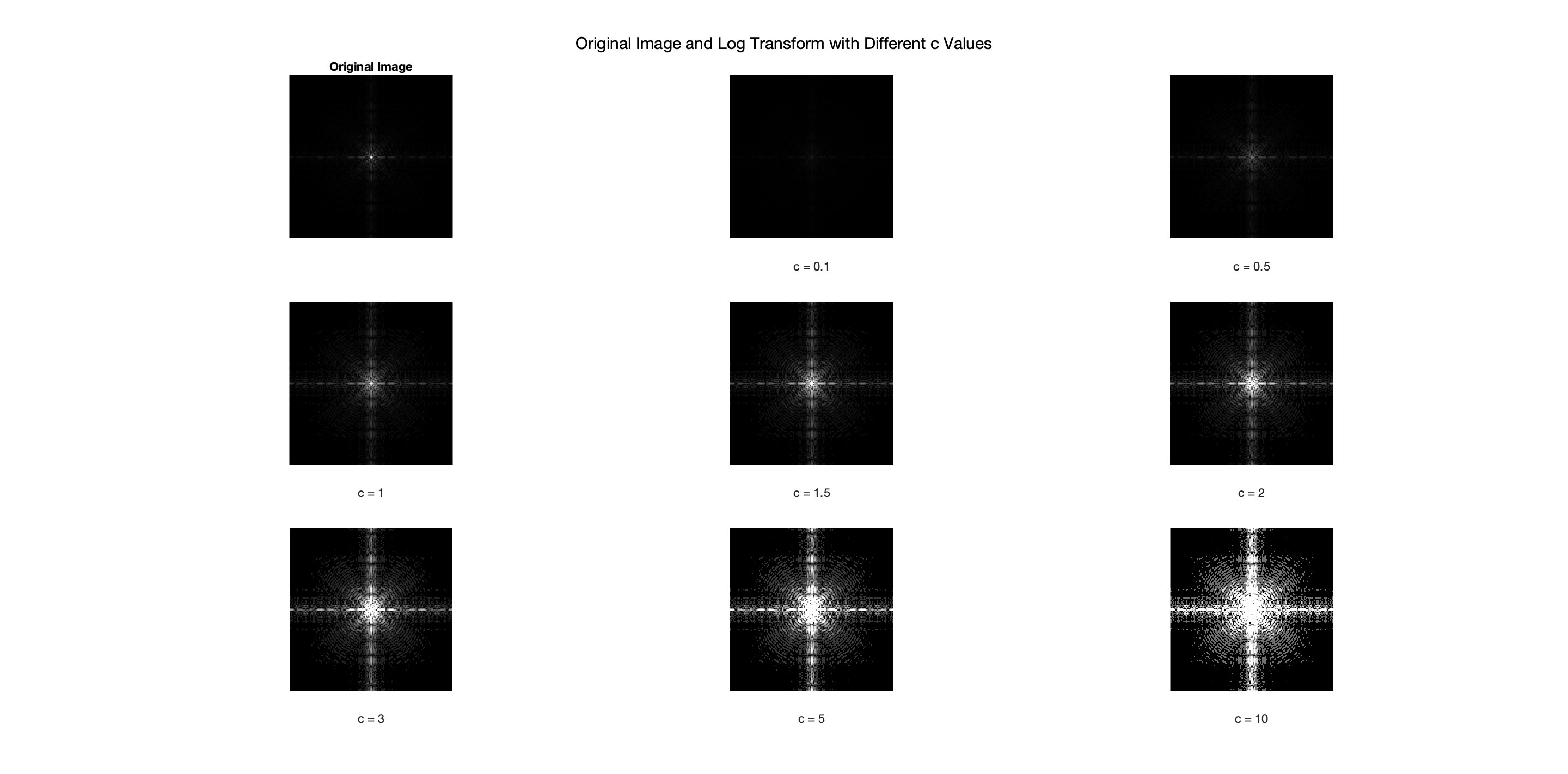
xlabel(['c = ' num2str(c(i))]); % Label the subplot with the current c value

end

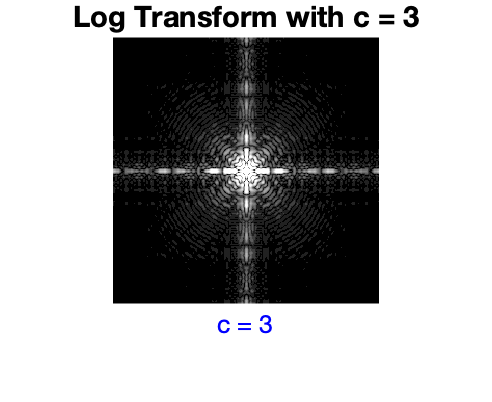
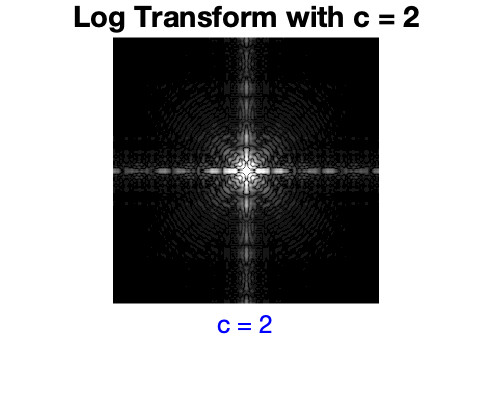
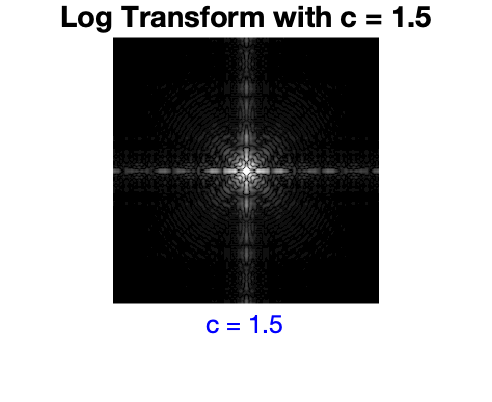
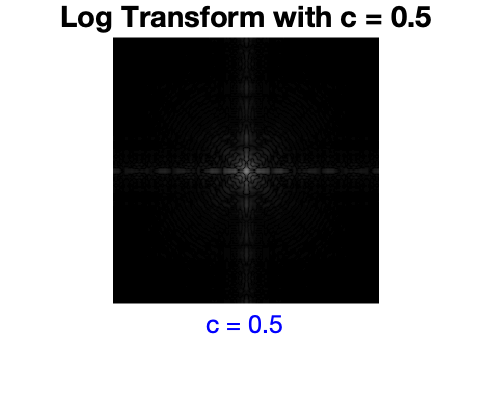
% Add a global title for the figure to describe the experiment

sgtitle('Original Image and Log Transform with Different c Values');

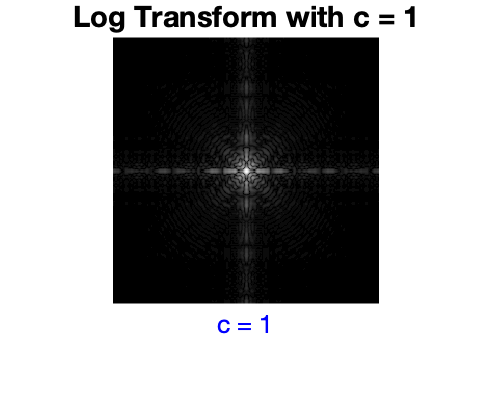
**Result:**



**放大的示例：**



**最佳结果：**



**Project03-02:**

**Code:**

**Project2.m:**

% Read the original image

image = imread('/Users/youngbean/Desktop/Experiments in Digital Image Processing/Class2/Data/Fig\_fractured\_spine.tif');

% Normalize the image to the range [0, 1]

image = double(image) / 255; % Convert to double and normalize

% Define the range for parameters c and gamma

c\_values = [0.5, 1, 2, 3]; % Scaling factors

gamma\_values = [0.2, 0.4, 0.6, 0.8, 1.0, 1.2,1.4,1.6,1.8,2.0]; % Gamma values

% Create a new figure for displaying results

figure;

% Display the original image in a larger subplot

subplot(2, 1, 1); % Use the top half of the figure for the original image

imshow(image);

title('Original Image', 'FontSize', 14); % Larger title for better emphasis

% Calculate the number of subplots needed for enhanced images

num\_c = length(c\_values);

num\_gamma = length(gamma\_values);

total\_images = num\_c \* num\_gamma;

% Create a tiled layout for enhanced images (adjusting rows/cols)

cols = num\_gamma; % Columns match the number of gamma values

rows = num\_c; % Rows match the number of c values

figure;

subplot(2, 1, 2); % Use the bottom half for the tiled layout

t = tiledlayout(rows, cols, 'TileSpacing', 'compact', 'Padding', 'compact');

% Add row and column labels for better comparison

for r = 1:rows

for g = 1:cols

% Compute the current c and gamma values

c = c\_values(r);

gamma = gamma\_values(g);

% Apply the power-law transformation

transformed\_image = c \* (image .^ gamma);

% Plot the transformed image in the corresponding tile

nexttile;

imshow(transformed\_image);

title(['c=', num2str(c), ', \gamma=', num2str(gamma)], 'FontSize', 10);

end

end

% Add global labels for rows and columns

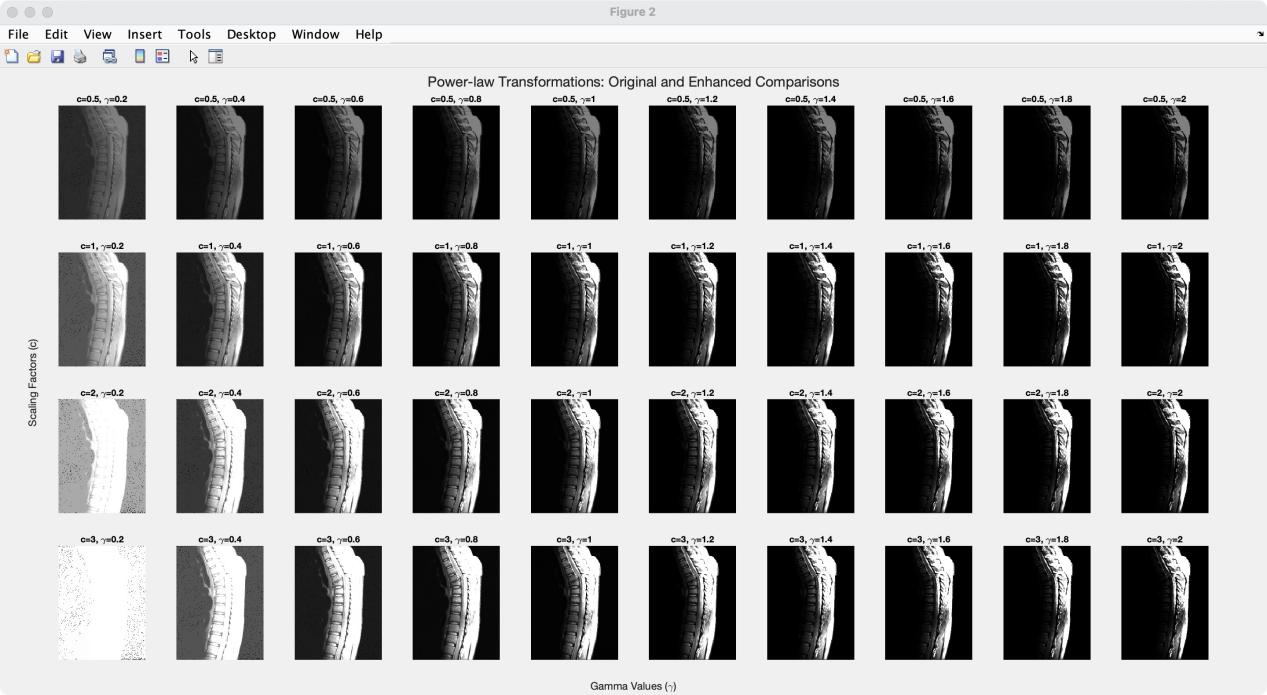
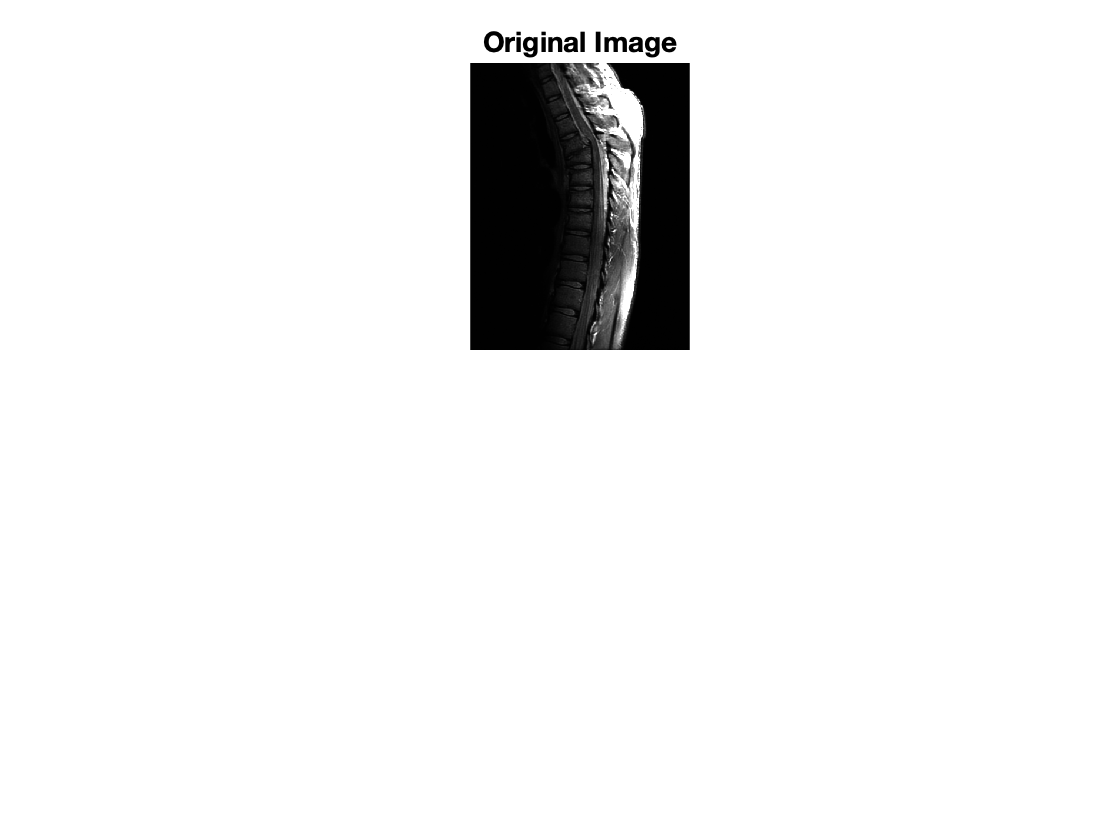
xlabel(t, 'Gamma Values (\gamma)', 'FontSize', 12);

ylabel(t, 'Scaling Factors (c)', 'FontSize', 12);

% Add a global title

sgtitle('Power-law Transformations: Original and Enhanced Comparisons', 'FontSize', 16);

**Result:**



**放大的示例：**



**最佳的结果：**  


**Project03-03:**

**Code:**

**averageIntensity.m:**

function avg = averageIntensity(image)

% Ensure the image is in double format for accurate calculation

image = double(image);

% Calculate the sum of all pixel intensities

totalIntensity = sum(image(:));

% Calculate the total number of pixels

numPixels = numel(image);

% Compute the average intensity

avg = totalIntensity / numPixels;

end

**thresholdImage.m:**

function thresholdedImage = thresholdImage(image, threshold)

% Ensure the image is in double format for accurate comparison

image = double(image);

% Generate the thresholded image

% Pixels greater than the threshold are set to 1, others are set to 0

thresholdedImage = image > threshold;

end

**Project3.m:**

% Read the input image

Moon = imread('/Users/youngbean/Desktop/Experiments in Digital Image Processing/Class2/Data/Fig\_blurry\_moon.tif');

% Calculate the average intensity using the averageIntensity function

ave = averageIntensity(Moon);

% Apply the thresholdImage function using the average intensity

ThresholdedMoon = thresholdImage(Moon, ave);

% Display the original and thresholded images

figure;

subplot(1, 2, 1);

imshow(Moon);

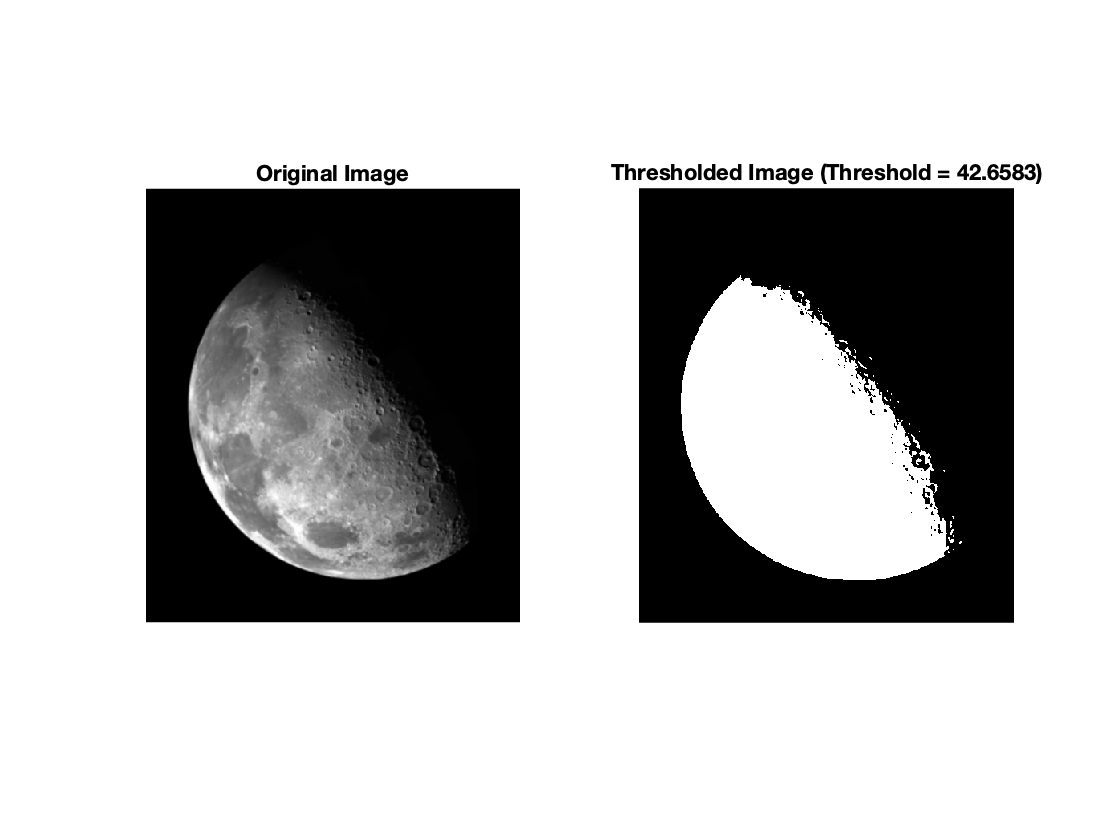
title('Original Image');

subplot(1, 2, 2);

imshow(ThresholdedMoon);

title(['Thresholded Image (Threshold = ', num2str(ave), ')']);

**Result:**

****

**【小结或讨论】**

本次实验以 MATLAB 为工具，完成了数字图像增强的多个项目，包括对数变换、幂律变换以及图像阈值化处理。通过实验，我们深入理解了相关理论并掌握了其实际应用。以下是实验中的具体总结和讨论。

## 图像增强原理及实现

### 1.对数变换

对数变换用于压缩动态范围大或增强暗部细节的图像，其公式为：

其中，表示归一化的像素值，是调整像素值幅度的参数，是对数的基底，是缩放因子。通过实验，我们对图像 Fig\_DFT\_no\_log.tif 应用 不同的值，结果表明对数变换能有效提升暗部区域的细节。 编程中遇到的问题与改进

最初编程中，使用了标准的对数公式,但结果显示对暗部细节的增强效果有限。在参考理论后，我们将公式调整为带基底和缩放因子的通用形式(如上公式),显著改善了动态范围的调整能力。

### 2. 幂律变换

幂律变换用于调节图像的亮度和对比度，其公式为：

其中， 是比例因子，是指数参数。实验中，通过调整 和对图像 Fig\_fractured\_spine.tif 进行增强，结果显示，较低的值能突出暗 部细节，而较高的 值则提升亮部区域的对比度。 编程中，为方便不同参数组合的结果比较，使用了MATLAB的 tiledlayout 功能，优化了显示布局。

### 3. 图像阈值化处理

实验中，首先使用 averageIntensity 函数计算图像 Fig\_blurry\_moon.tif 的平均强度，公式为：

然后利用平均强度作为阈值，将图像二值化，其公式为：

阈值化处理的结果直观显示了如何通过简单的规则突出目标区域，但编程中需特别注意数据类型的转换，以避免计算错误。

实验中通过理论和实践结合，加深了对数字图像增强技术的理解。调整对数变换公式后，增强效果显著改善，说明理论公式在实际应用中需根据场景调整。此外，在幂律变换和阈值化处理实验中，通过灵活调节参数，我们直观感受到不同参数对图像亮度、对比度和区域分割的影响。

实验仍存在以下问题与改进空间：

1. 增强效果的主观性：实验结果主要依赖肉眼评估，缺乏量化指标（如峰值信噪比或均方误差），未来可引入客观指标进行评价。

2. 效率问题：在处理较大图像时，运算速度相对较慢，可尝试使用 GPU 加速或优化 MATLAB 代码结构。

总体而言，本次实验不仅巩固了对图像增强理论的理解，还通过解决实际问题提升了编程能力，为后续复杂图像处理技术的学习奠定了坚实基础。