安徽大学《数字图像处理（双语）》实验报告（1）

学号：WA2214014 专业：人工智能 姓名及签字：杨跃浙

实验日期：24.11.26 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】：

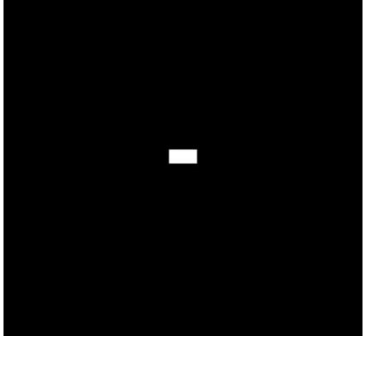
1. **熟悉和掌握 MATLAB 基本编程环境**
2. **熟悉和掌握基于 MATLAB 的数字图像处理编程基础**
3. **通过 MATLAB 编程实现创作图像以及图像的放大缩小**

【实验内容】

PROJECT 02-01

Image Creating

a) In Matlab workspace, generate an image with the size of 512 × 512 pixels, 8-bit grayscale, black background, the center having a 40 pixels width and 20 pixels height white rectangle. As shown in the following figure:



b) Save this image as a file “test.bmp”

c) Read out the image from the file test.bmp to the variable I

d) Display the image represented by the variable I in Matlab graphical interface

e) Convert the obtained image format into "*. tif", "*. jpg" format, check the data size of the volume of the documents with different formats.

f) Save or copy the image to the root directory of MATLAB program "work" folder for later experimental use.

PROJECT 02-02

Image negative

Write a computer program capable of producing the negative

images (logic operation NOT) on "Fig_blurry_moon.tif" as well as the images used in the previous sections.

PROJECT 02-03

Zooming and Shrinking Images by Pixel Replication

(a) Write a computer program capable of zooming and shrinking an image by pixel replication.

Assume that the desired zoom/shrink factors are integers.

(b) Use your program to shrink the image “Fig_rose.tif” by a factor of 16.

(c) Use your program to zoom the image in (b) back to the resolution of

the original. Explain the reasons for their differences.

PROJECT 02-04

Reducing the Number of Intensity Levels in an Image

(a) Write a computer program capable of reducing the number of intensity levels in an image from 256 to 2, in integer powers of 2. The desired number of intensity levels needs to be a variable input to your program.

(b) Perform the program on the following image “Fig_ctskull-256.tif” from the textbook

【实验代码和结果】

Project02-01:

Code:

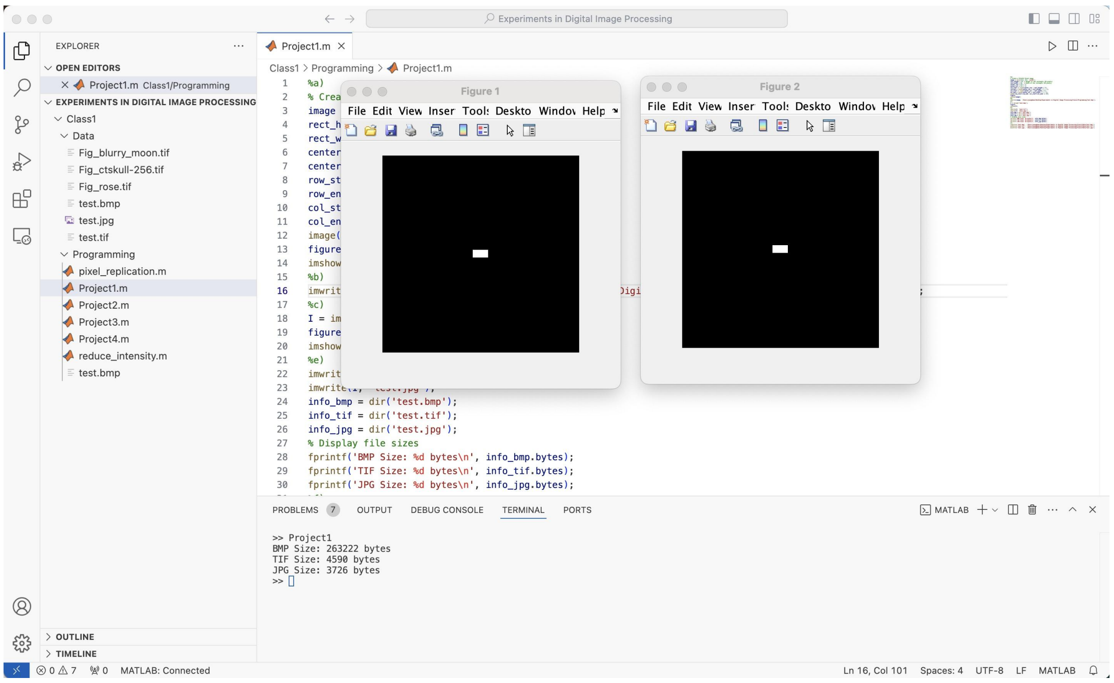
Project1.m:

```
%a)
% Create a 512x512 black image
image = zeros(512, 512, 'uint8');
rect_height = 20; % Height of the rectangle (20 pixels)
rect_width = 40; % Width of the rectangle (40 pixels)
center_row = 512 / 2;
center_col = 512 / 2;
row_start = round(center_row - rect_height / 2);
row_end = round(center_row + rect_height / 2 - 1);
col_start = round(center_col - rect_width / 2);
col_end = round(center_col + rect_width / 2 - 1);
image(row_start:row_end, col_start:col_end) = 255;
figure;
```

```
imshow(image);
%b)
imwrite(image, '/Users/youngbean/Desktop/Experiments in
Digital Image Processing/Class1/Programming/test.bmp');
%c)
I = imread('test.bmp');
figure;
imshow(I);
%e)
imwrite(I, 'test.tif');
imwrite(I, 'test.jpg');
info_bmp = dir('test.bmp');
info_tif = dir('test.tif');
info_jpg = dir('test.jpg');
% Display file sizes
fprintf('BMP Size: %d bytes\n', info_bmp.bytes);
fprintf('TIF Size: %d bytes\n', info_tif.bytes);
fprintf('JPG Size: %d bytes\n', info_jpg.bytes);
%f)
copyfile('test.bmp', '/Users/youngbean/Desktop/Experiments
in Digital Image Processing/Class1/Data/test.bmp');
movefile('test.tif', '/Users/youngbean/Desktop/Experiments
in Digital Image Processing/Class1/Data/test.tif');
movefile('test.jpg', '/Users/youngbean/Desktop/Experiments
in Digital Image Processing/Class1/Data/test.jpg');
```

Result:

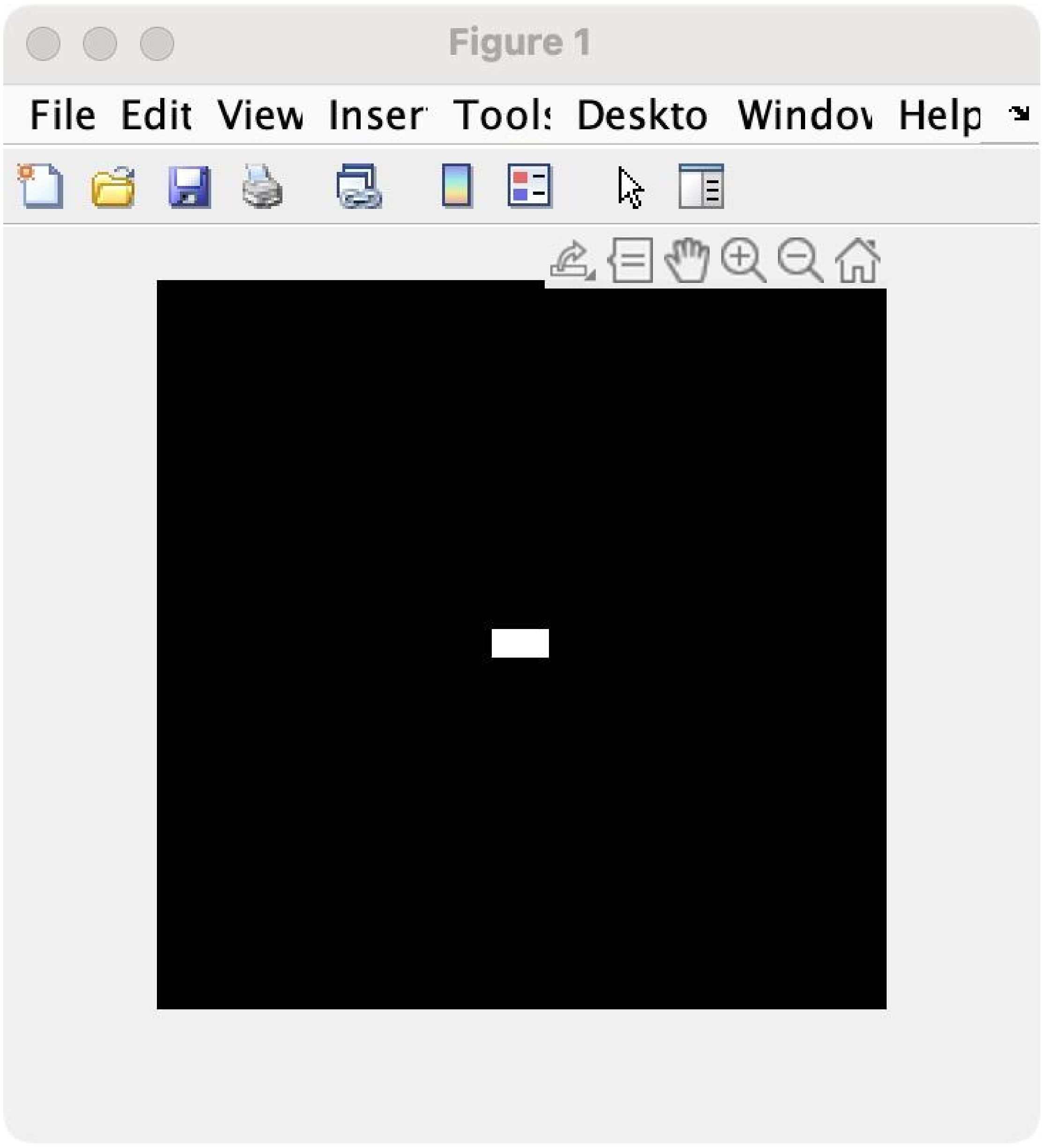
Digital Image Processing-Lab 1



```
%a)
% Crea
image
rect_h
rect_w
center
center
row_st
row_en
col_st
col_en
image(
figure
imshow
%b)
imwrit
%c)
I = im
figure
imshow
%e)
imwrit
imwrite(I, 'test.jpg');
info_bmp = dir('test.bmp');
info_tif = dir('test.tif');
info_jpg = dir('test.jpg');
% Display file sizes
fprintf('BMP Size: %d bytes\n', info_bmp.bytes);
fprintf('TIF Size: %d bytes\n', info_tif.bytes);
fprintf('JPG Size: %d bytes\n', info_jpg.bytes);
```

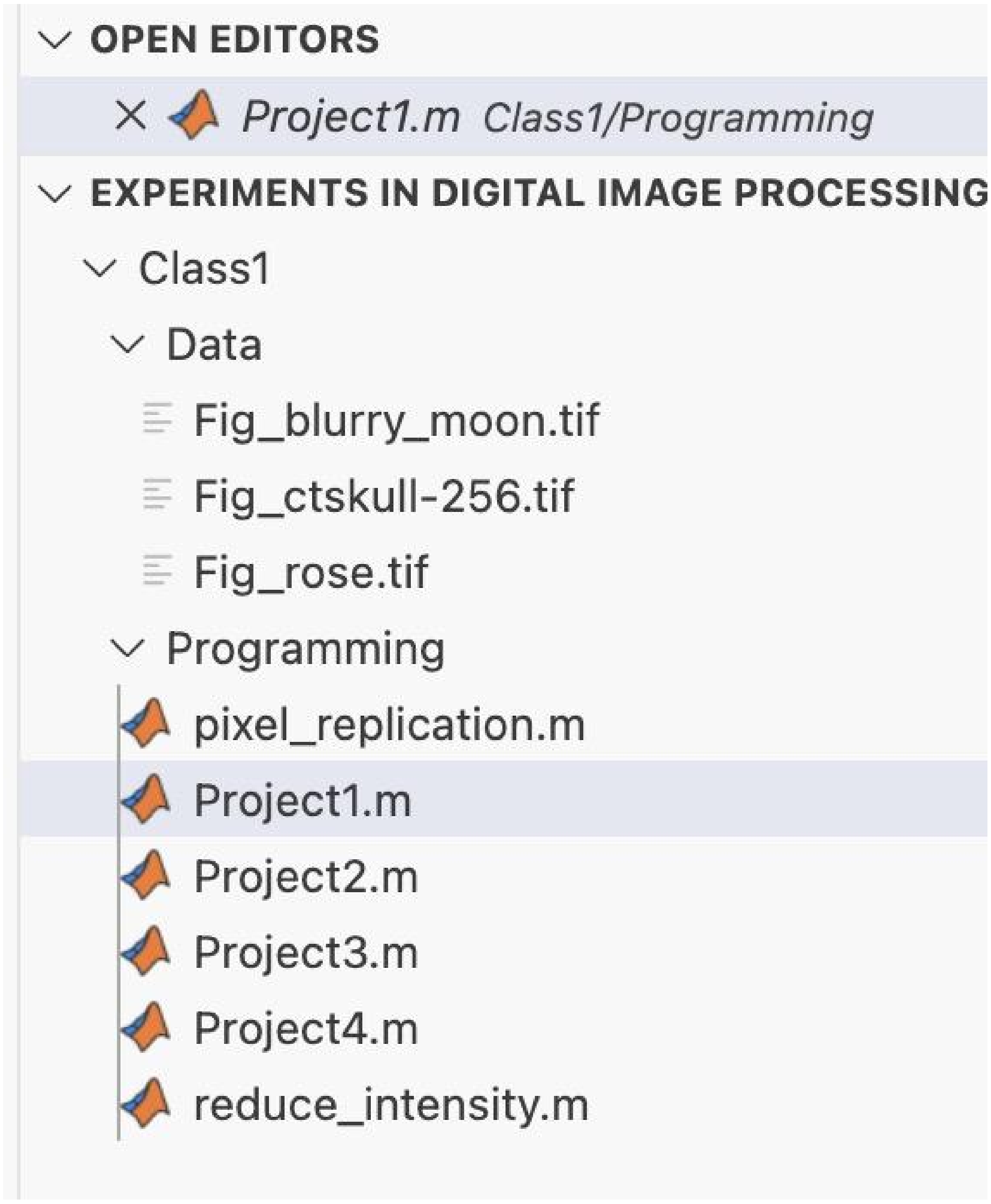
```
>> Project1
BMP Size: 263222 bytes
TIF Size: 4590 bytes
JPG Size: 3726 bytes
>>
```

Part (a):



Part (b):

Before:



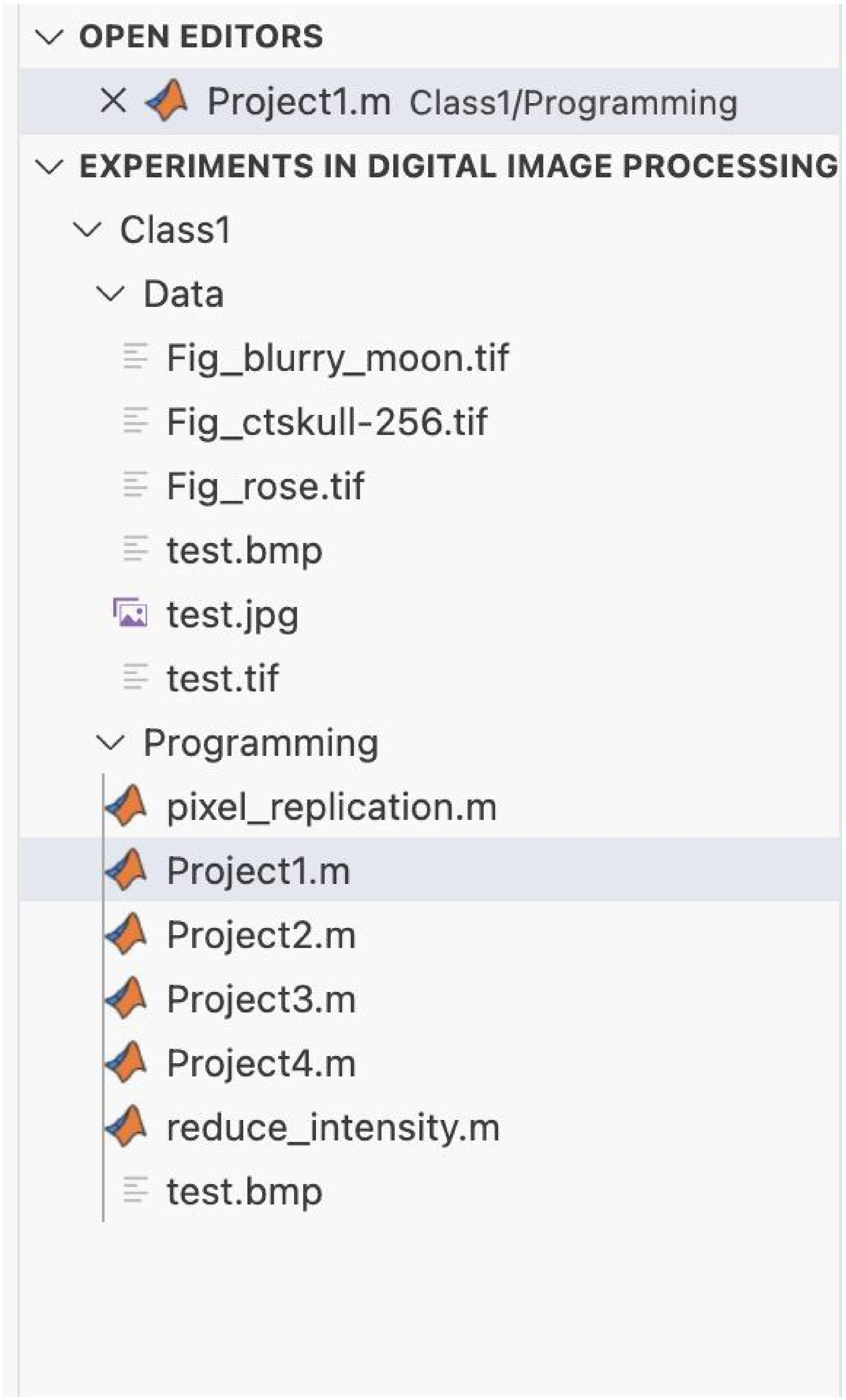
OPEN EDITORS

× *Project1.m Class1/Programming*

EXPERIMENTS IN DIGITAL IMAGE PROCESSING

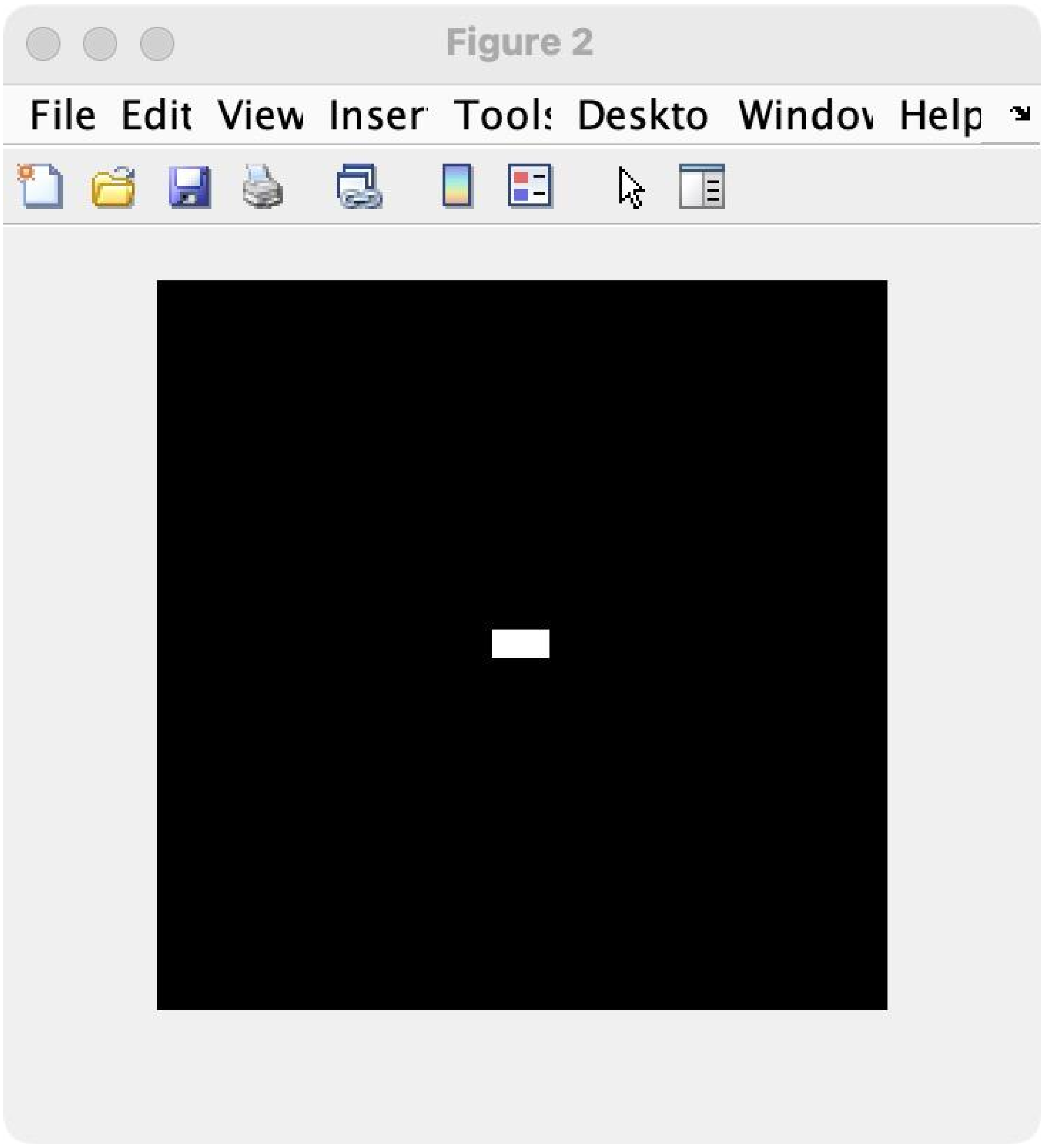
```
Class1
  Data
    Fig_blurry_moon.tif
    Fig_ctskull-256.tif
    Fig_rose.tif
  Programming
    pixel_replication.m
    Project1.m
    Project2.m
    Project3.m
    Project4.m
    reduce_intensity.m
```

After:因为我在 VScode 中运行 Matlab 并且是 MacOS 系统 因此比起遍历 work 环境，我直接简单的放在了当前目录下。运行到改步骤时 Data 未变。

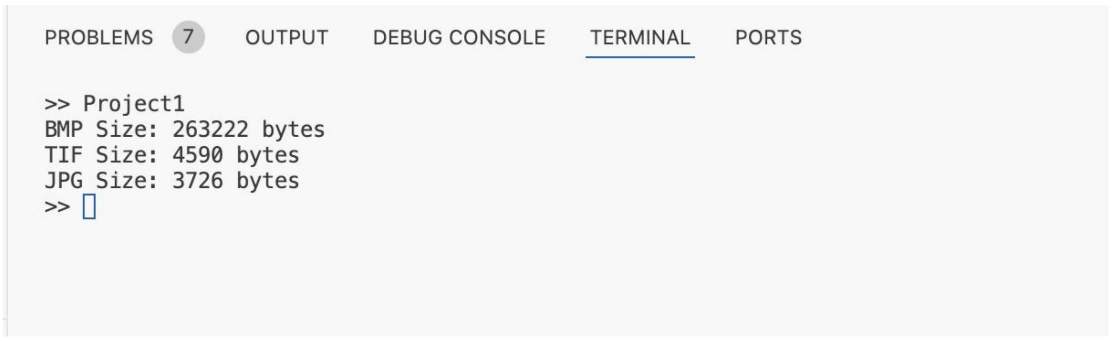


OPEN EDITORS
Project1.m Class1/Programming
EXPERIMENTS IN DIGITAL IMAGE PROCESSING
Class1
Data
Fig_blurry_moon.tif
Fig_ctskull-256.tif
Fig_rose.tif
test.bmp
test.jpg
test.tif
Programming
pixel_replication.m
Project1.m
Project2.m
Project3.m
Project4.m
reduce_intensity.m
test.bmp

Part (d):



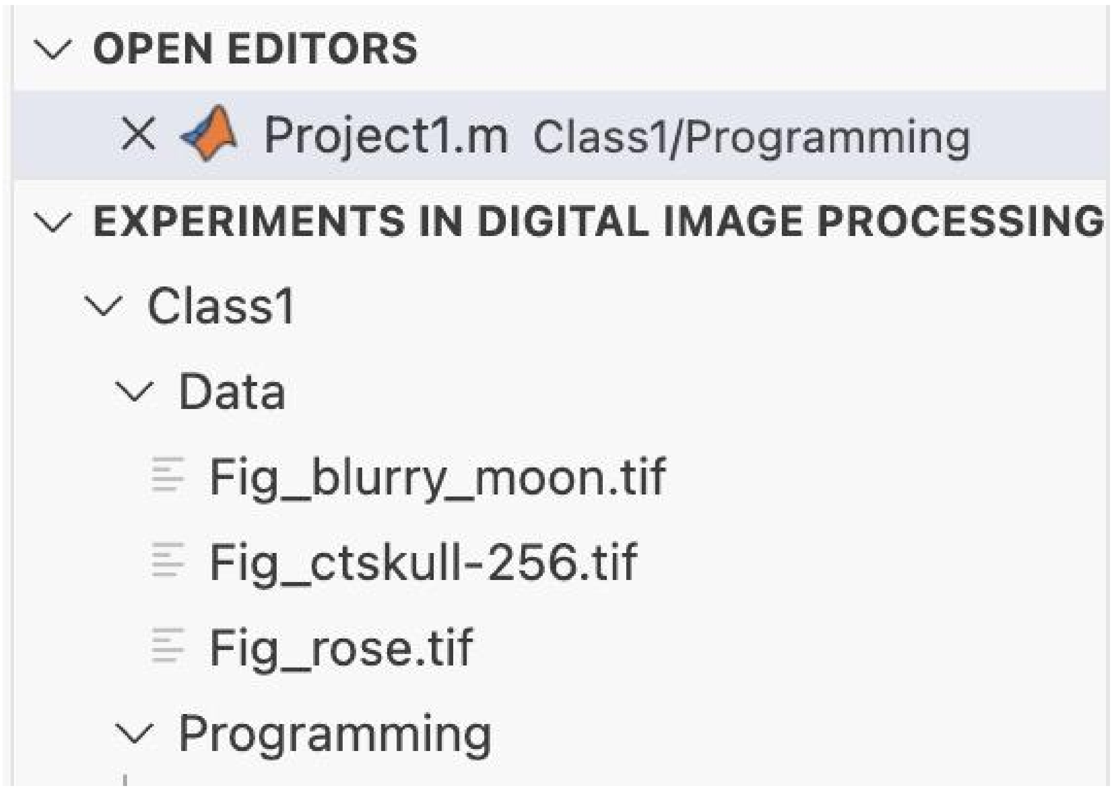
Part (e):



```
>> Project1
BMP Size: 263222 bytes
TIF Size: 4590 bytes
JPG Size: 3726 bytes
>> 
```

Part (f):

Before:



OPEN EDITORS

× Project1.m Class1/Programming

EXPERIMENTS IN DIGITAL IMAGE PROCESSING

Class1

Data

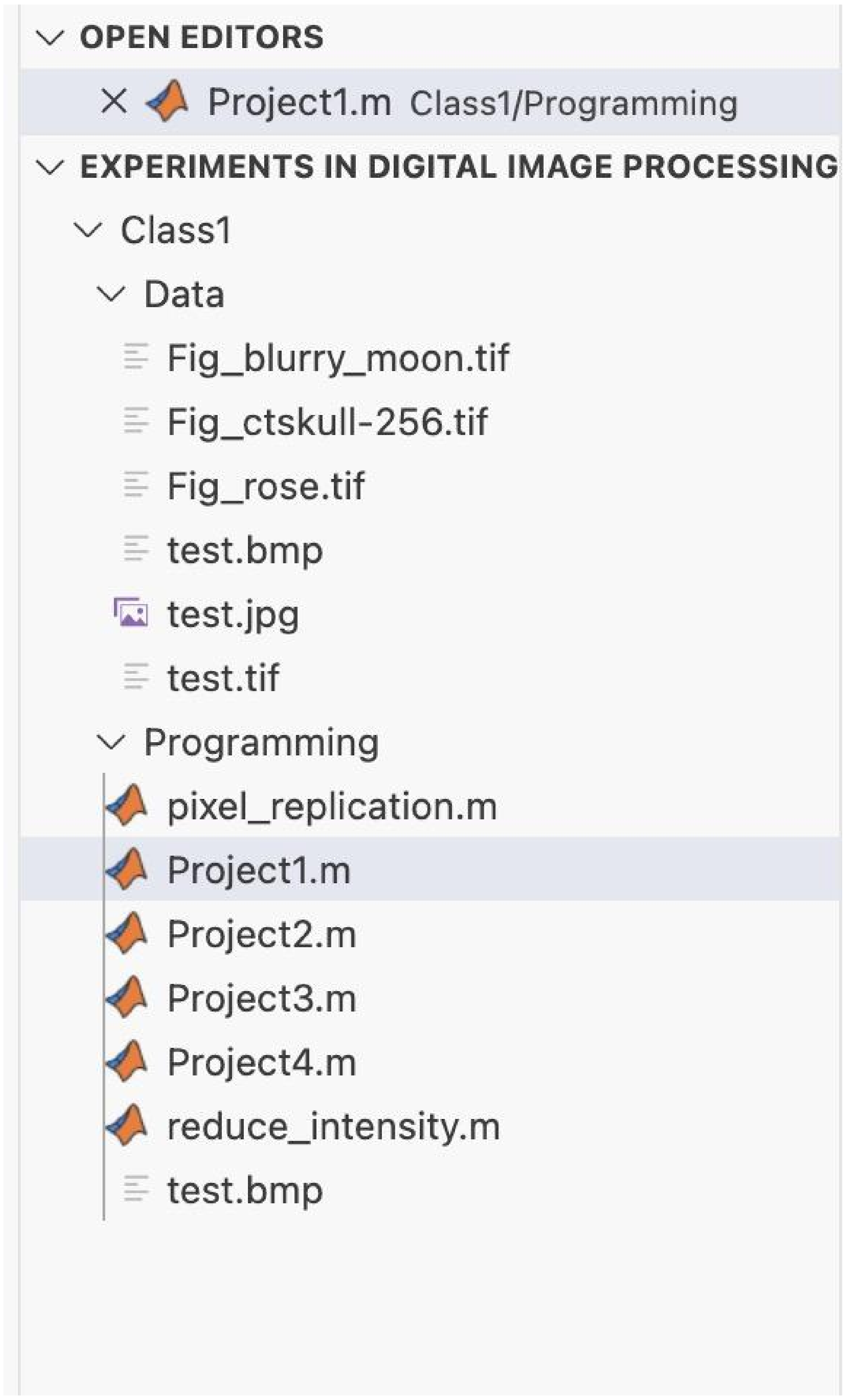
Fig_blurry_moon.tif

Fig_ctskull-256.tif

Fig_rose.tif

Programming

After:因为我在 VScode 中运行 Matlab 并且是 MacOS 系统 因此比起遍历 work 环境，我直接简单的放在了 Data（统一存储数据）目录下。bmp 是用的 copy 其他两个是 move。



```
OPEN EDITORS
    Project1.m  Class1/Programming
EXPERIMENTS IN DIGITAL IMAGE PROCESSING
  Class1
    Data
      Fig_blurry_moon.tif
      Fig_ctskull-256.tif
      Fig_rose.tif
      test.bmp
      test.jpg
      test.tif
    Programming
      pixel_replication.m
      Project1.m
      Project2.m
      Project3.m
      Project4.m
      reduce_intensity.m
      test.bmp
```

Project02-02:

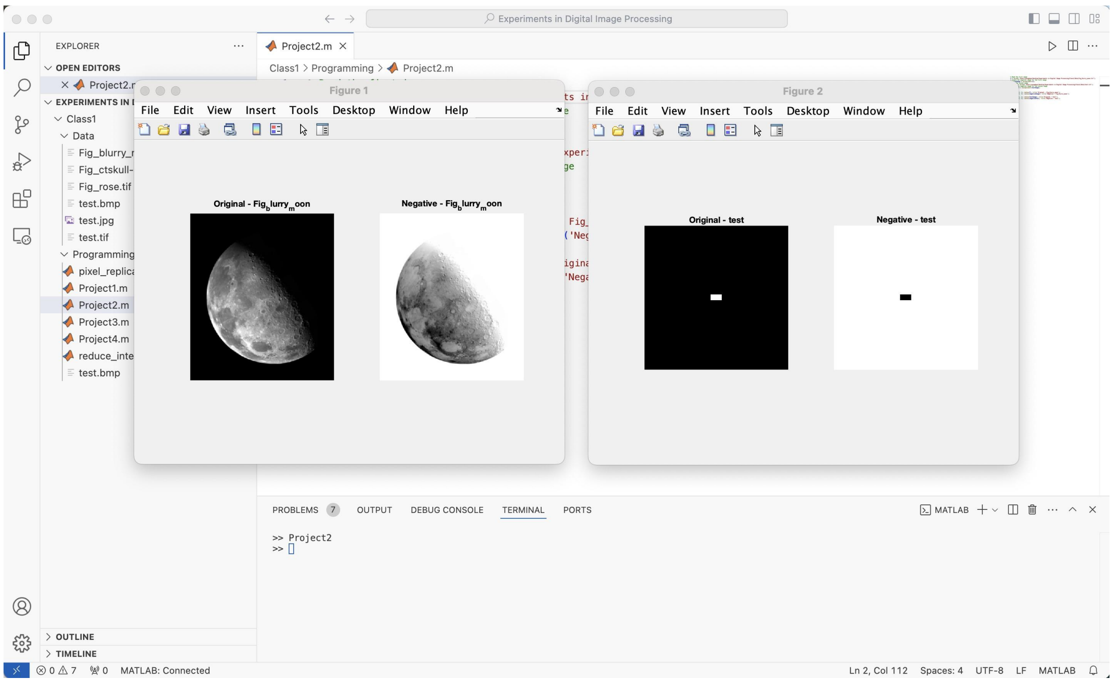
Code:

Project2.m:

```
% Read the first image
I = imread('/Users/youngbean/Desktop/Experiments in Digital
Image Processing/Class1/Data/Fig_blurry_moon.tif');
% Create the negative image for the first image
negativeImage = imcomplement(I);
% Read the second image
testImage = imread('/Users/youngbean/Desktop/Experiments in
Digital Image Processing/Class1/Data/test.tif');
% Create the negative image for the second image
negativeTest = imcomplement(testImage);

figure;
subplot(1, 2, 1); imshow(I); title('Original -
Fig_blurry_moon');
subplot(1, 2, 2); imshow(negativeImage); title('Negative -
Fig_blurry_moon');
figure;
subplot(1, 2, 1); imshow(testImage); title('Original -
test');
subplot(1, 2, 2); imshow(negativeTest); title('Negative -
test');
```

Result:

Digital Image Processing-Lab 1



```
>> Project2
>> 
```

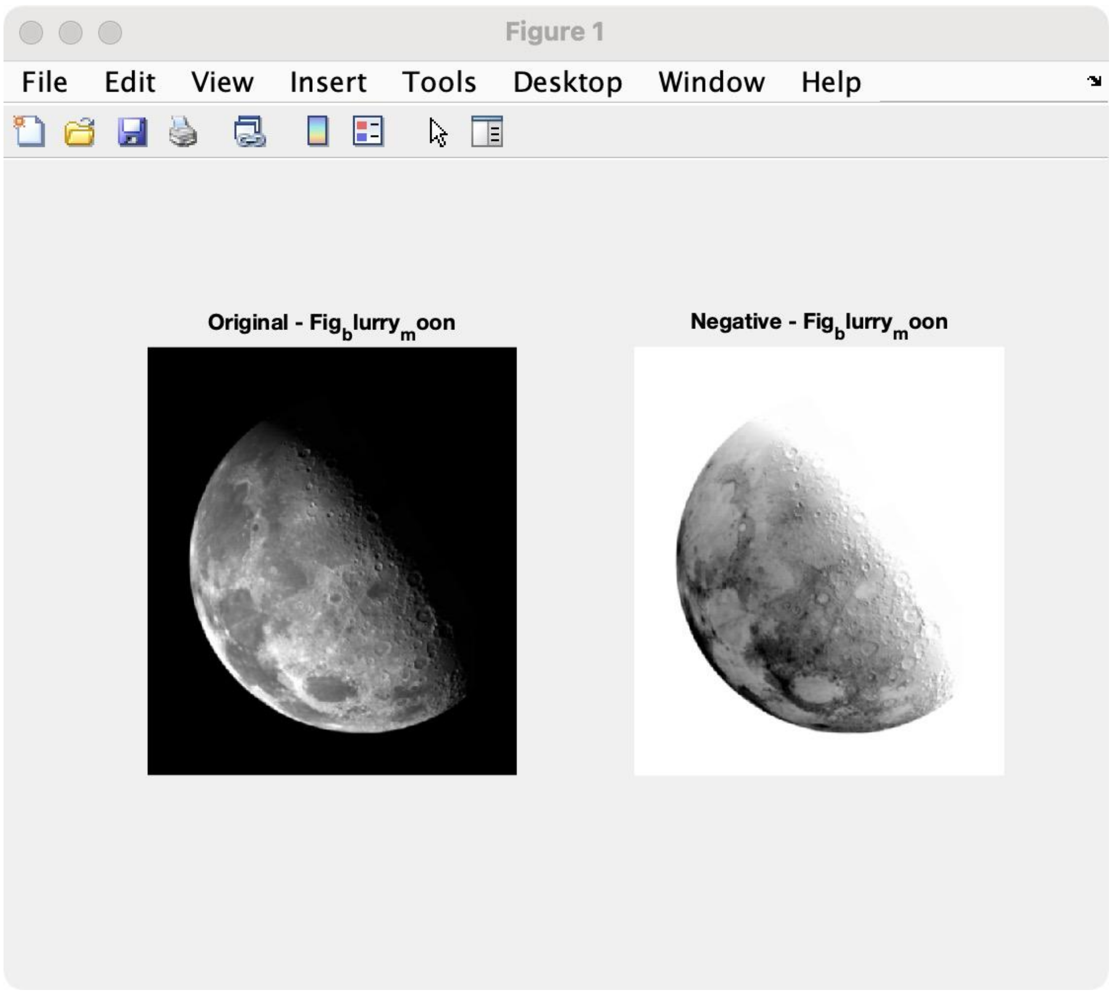
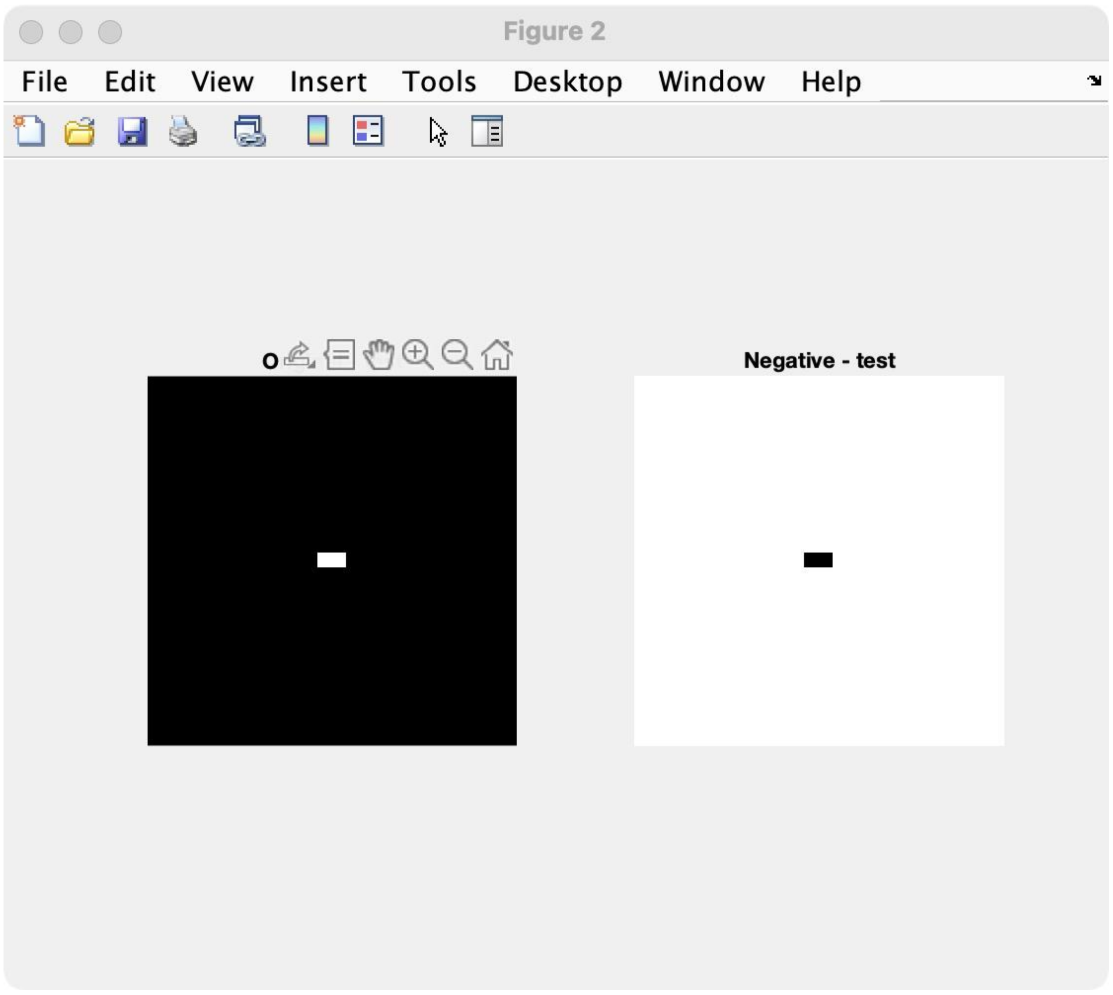


Figure 1
File Edit View Insert Tools Desktop Window Help
Original - Fig$_b$lurry$_m$oon
Negative - Fig$_b$lurry$_m$oon



Project02-03:

Code:

pixel_replication.m:

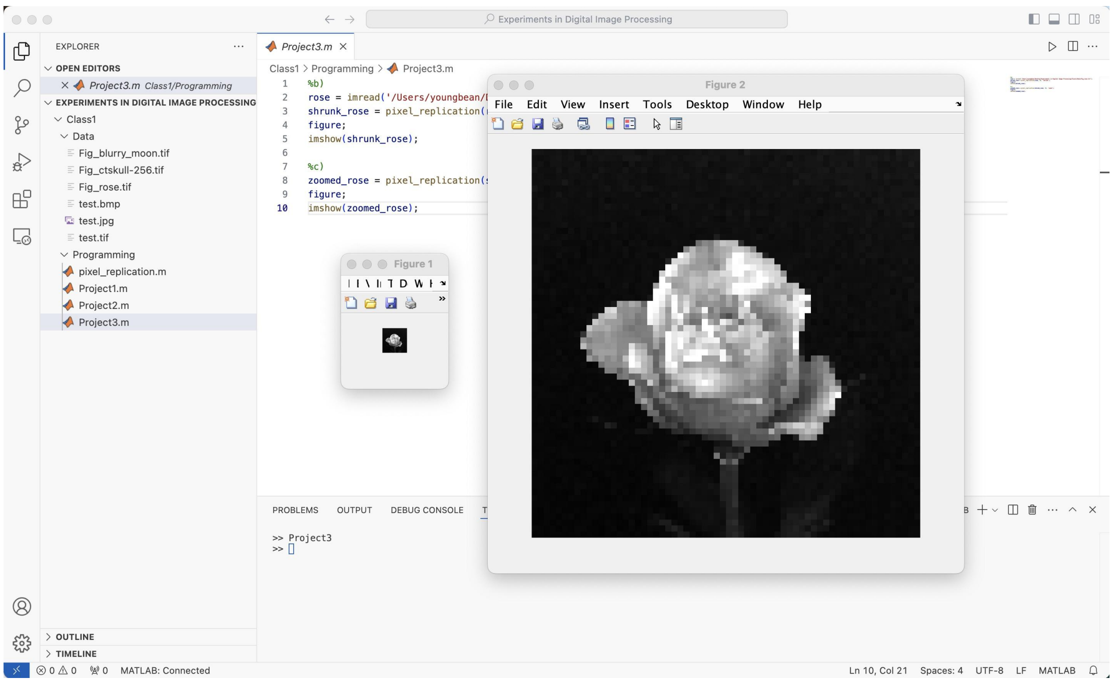
```
%a)
function resized_image = pixel_replication(image, factor,
mode)
if strcmp(mode, 'shrink')
resized_image = image(1:factor:end, 1:factor:end);
elseif strcmp(mode, 'zoom')
resized_image = repelem(image, factor, factor);
end
end
```

Project3.m:

```
%b)
```

```
rose = imread('/Users/youngbean/Desktop/Experiments in
Digital Image Processing/Class1/Data/Fig_rose.tif');
shrunk_rose = pixel_replication(rose, 16, 'shrink');
figure;
imshow(shrunk_rose);

%c)
zoomed_rose = pixel_replication(shrunk_rose, 16, 'zoom');
figure;
imshow(zoomed_rose);
```

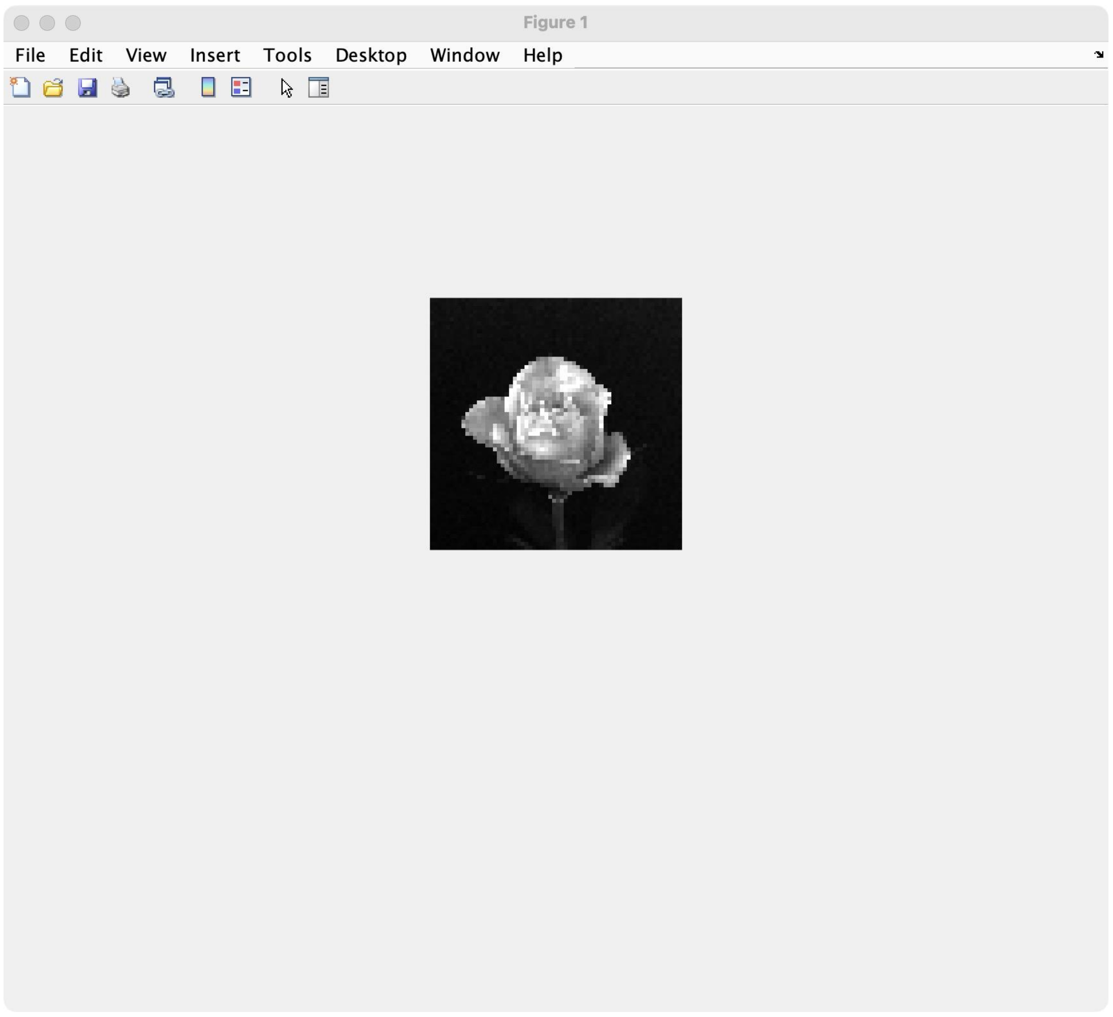
Result:


Experiments in Digital Image Processing
EXPLORER
OPEN EDITORS
Project3.m Class1/Programming
EXPERIMENTS IN DIGITAL IMAGE PROCESSING
Class1
Data
Fig_blurry_moon.tif
Fig_ctskull-256.tif
Fig_rose.tif
test.bmp
test.jpg
test.tif
Programming
pixel_replication.m
Project1.m
Project2.m
Project3.m
OUTLINE
TIMELINE
Project3.m
Class1 > Programming > Project3.m
Figure 1
Figure 2
File Edit View Insert Tools Desktop Window Help
PROBLEMS OUTPUT DEBUG CONSOLE
MATLAB: Connected
Ln 10, Col 21 Spaces: 4 UTF-8 LF MATLAB

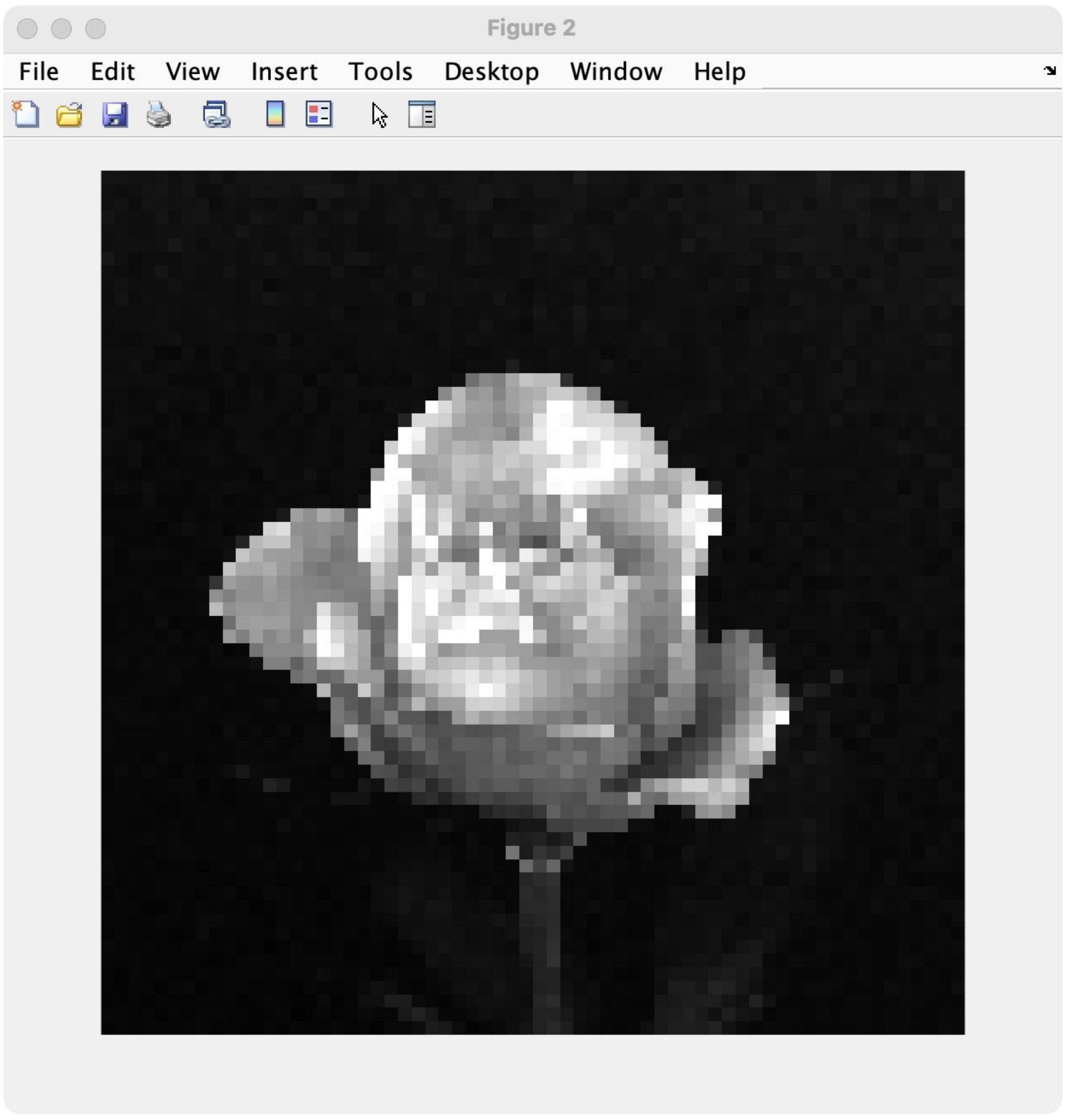
```
%b)
rose = imread('/Users/youngbean/I
shrunk_rose = pixel_replication(
figure;
imshow(shrunk_rose);

%c)
zoomed_rose = pixel_replication(s
figure;
imshow(zoomed_rose);
```

```
>> Project3
>>
```

Part (b):



Part (c):



Explanation:

When zooming the image shrunk in part (b) back to its original resolution, noticeable differences arise between the zoomed image and the original. These differences can be attributed to the nature of pixel replication and the irreversible loss of information during the shrinking process. When the image is shrunk by a factor of 16, the pixel replication method retains only one pixel from each 16×16 block of the original image, discarding all other details. As a result, finer textures, gradients, and edge information present in the original image are permanently lost. The shrunken image, therefore, represents a much simpler version of the original, with significant detail reduction.

In the subsequent zooming process, pixel replication simply duplicates each pixel of the shrunken image into a 16×16 block to match the original resolution. However, this method does not restore the lost details. Instead, the zoomed image becomes a magnified version of the shrunken image, and blocky artifacts are clearly visible. These artifacts arise because each pixel is uniformly repeated, resulting in abrupt transitions between blocks and the absence of the smooth gradients found in the original image. Features such as edges and textures appear jagged and pixelated, highlighting the limitations of pixel replication for high-quality resizing.

The irreversibility of the shrinking process further explains the differences between the zoomed image and the original. Since much of the original image's data is discarded during shrinking, it is impossible to recover this lost information through zooming. The zoomed image cannot recreate the original's fine details or natural transitions; it can only expand the limited information retained in the shrunken image. Consequently, the zoomed image serves as a coarse approximation of the original, lacking its complexity and visual fidelity. This highlights the need for more advanced interpolation techniques, such as bicubic or bilinear interpolation, in scenarios where preserving image quality is essential.

Project02-04:

Code:

reduce_intensity.m

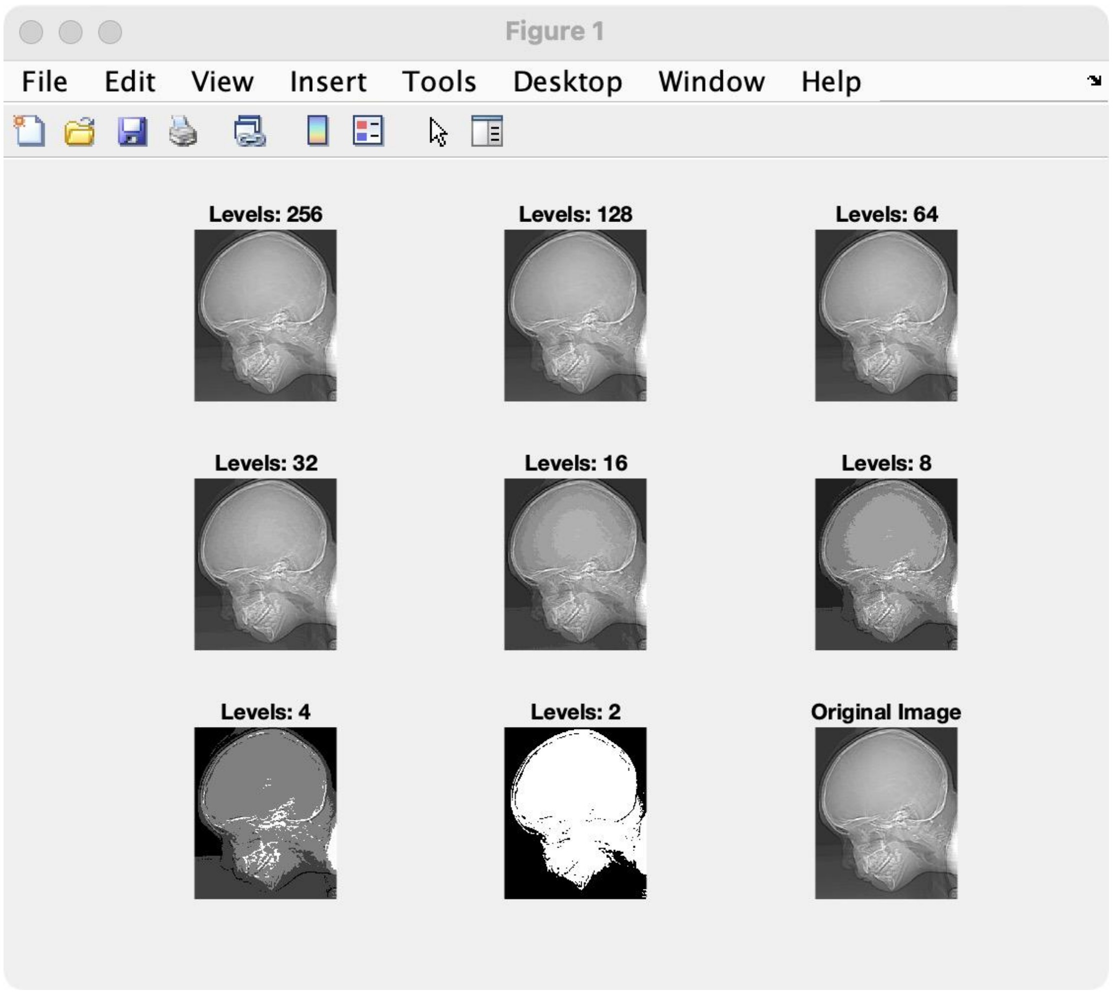
```
%a)
function reduced_image = reduce_intensity(image, levels)
image = uint8(image);
factor = 256 / levels;
reduced_image = uint8(floor(double(image) / factor) *
factor);
reduced_image(reduced_image == (256 - factor)) = 255;
end
```

Project4.m:

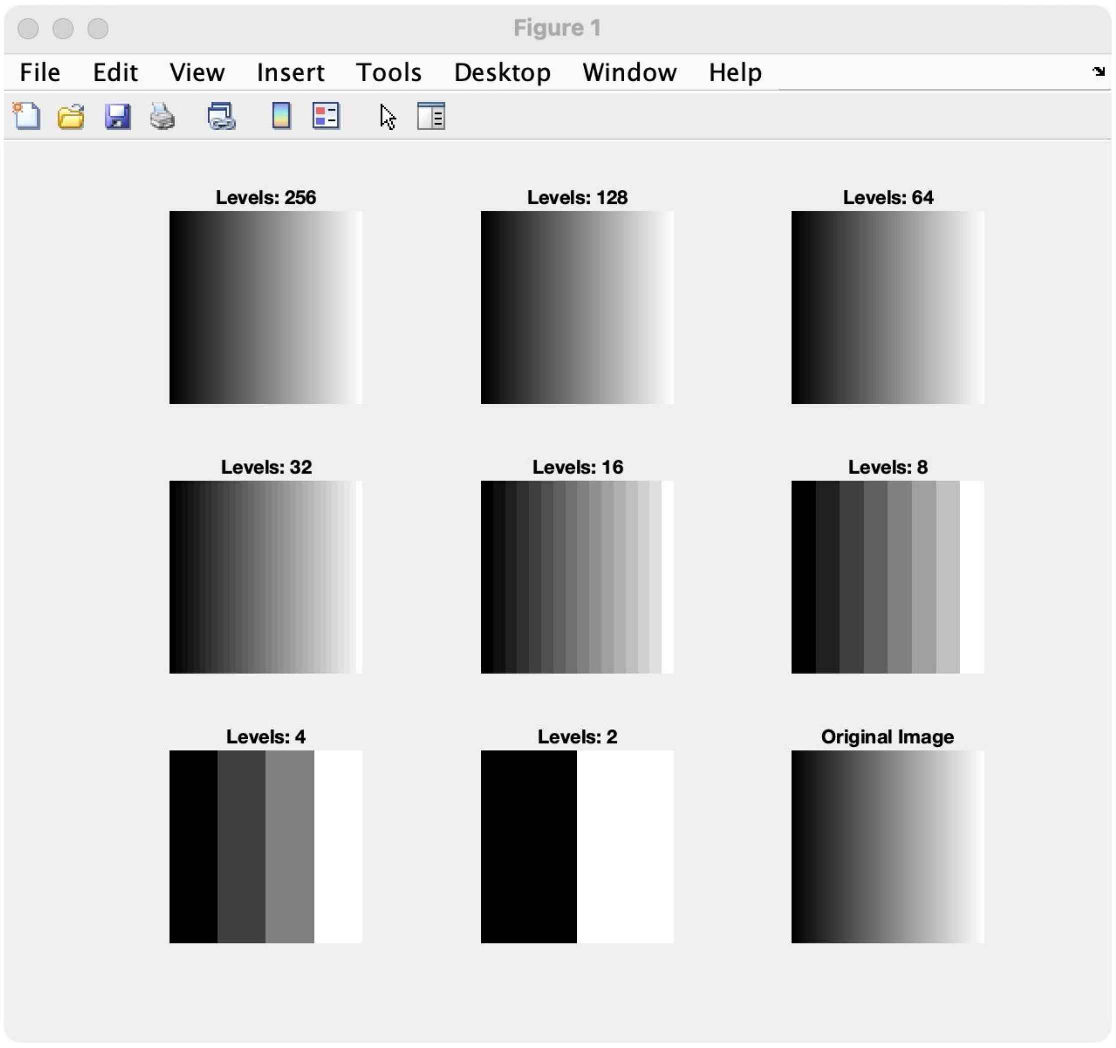
```
%b)
%ct_skull = uint8(repmat(0:255, [256, 1]));
ct_skull = imread('/Users/youngbean/Desktop/Experiments in
Digital Image
Processing/Class1/Data/Fig_ctskull-256.tif');
levels = 256;
num_reductions = log2(256);
figure;
for i = 0:num_reductions
% Calculate current levels (256, 128, ..., 2)
current_levels = 256 / (2^i);
```

```
reduced_image = reduce_intensity(ct_skull, current_levels);
subplot(3, 3, i + 1); % Arrange in a 3x3 grid
imshow(reduced_image);
title(['Levels: ', num2str(current_levels)]);
end
% Display the original image in the last subplot
subplot(3, 3, num_reductions + 1);
imshow(ct_skull);
title('Original Image');
% Reduce the intensity levels of the image to the
user-specified value
answer = inputdlg('Enter the desired number of intensity
levels (must be a power of 2 between 2 and 256):', ...
'Input Intensity Levels', [1 50]);
user_levels = str2double(answer{1});
user_reduced_image = reduce_intensity(ct_skull,
user_levels);
output_filename = ['reduced_image_levels_',
num2str(user_levels), '.png'];
imwrite(user_reduced_image, output_filename);
fprintf('Image reduced to %d levels saved as: %s\n',
user_levels, output_filename);
figure;
subplot(1, 2, 1);
imshow(ct_skull, [0 255]);
title('Original Image');
subplot(1, 2, 2);
imshow(user_reduced_image, [0 255]);
title(['Reduced to ', num2str(user_levels), ' Levels']);
```

Result:

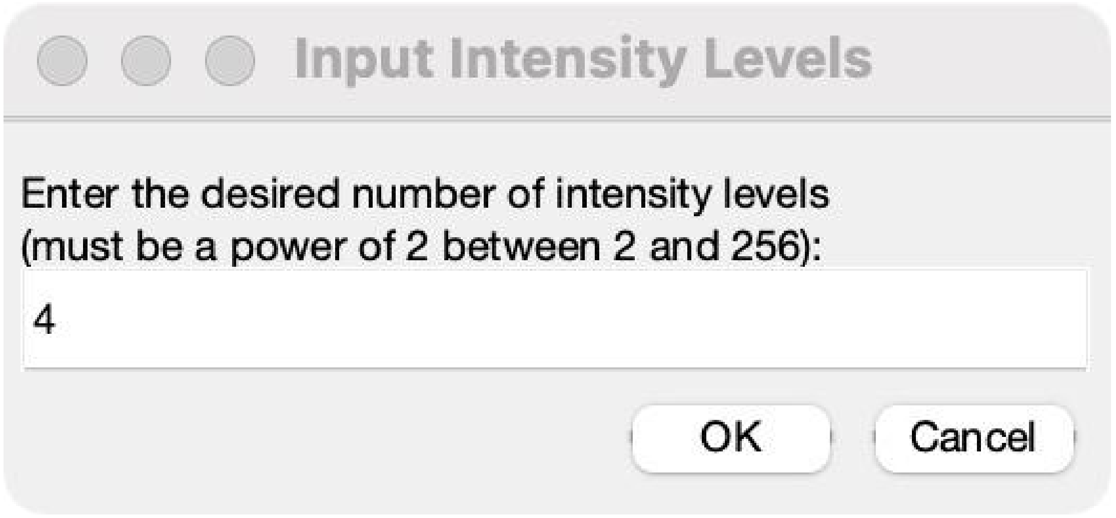


为了充分验证我是正确的，我还补充了另一个实验，直接生成一张 0-255 的图：



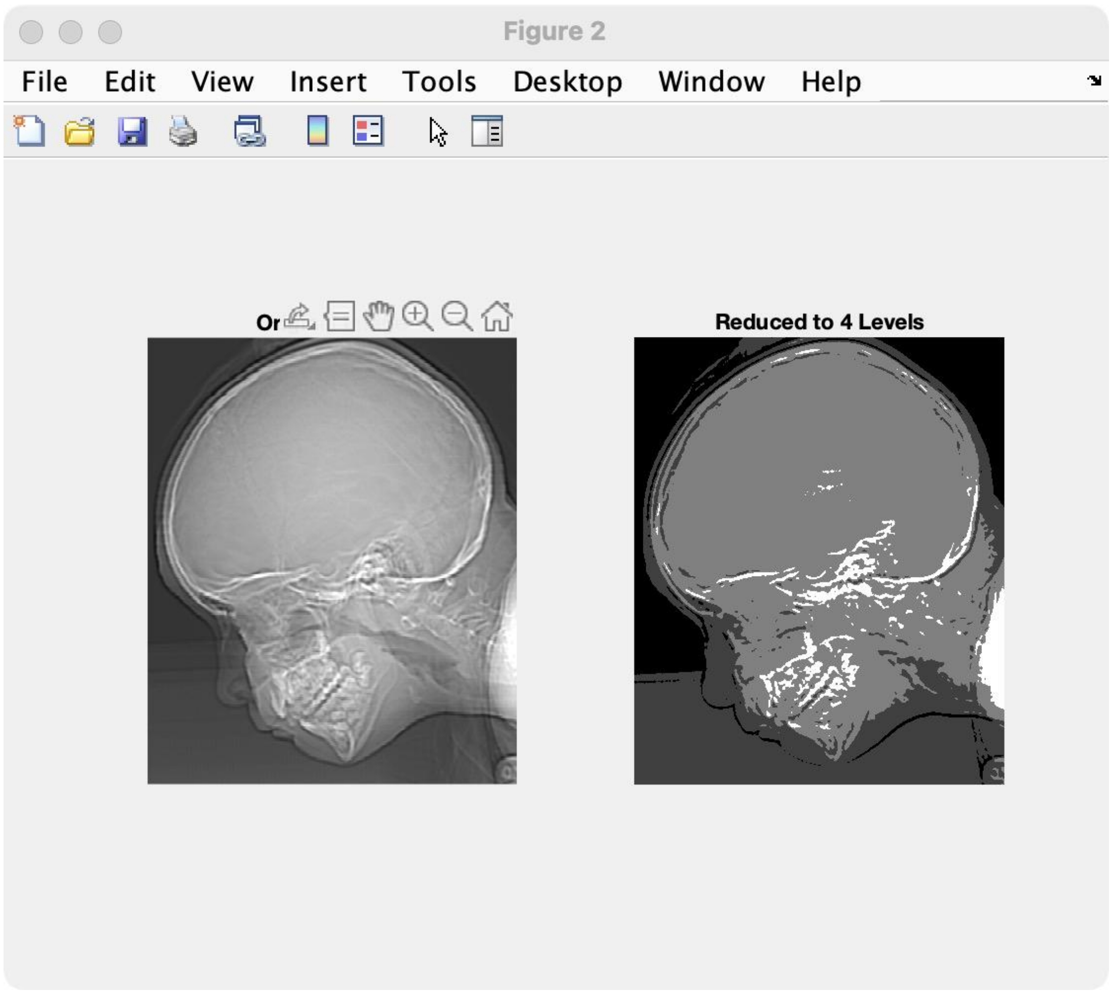
用户的动态输入由于 VScode 不支持在终端 input，所以这里采用了 inputdlg，产生一个输入对话框，让用户输入想得到的图像 Level。

Example 1:

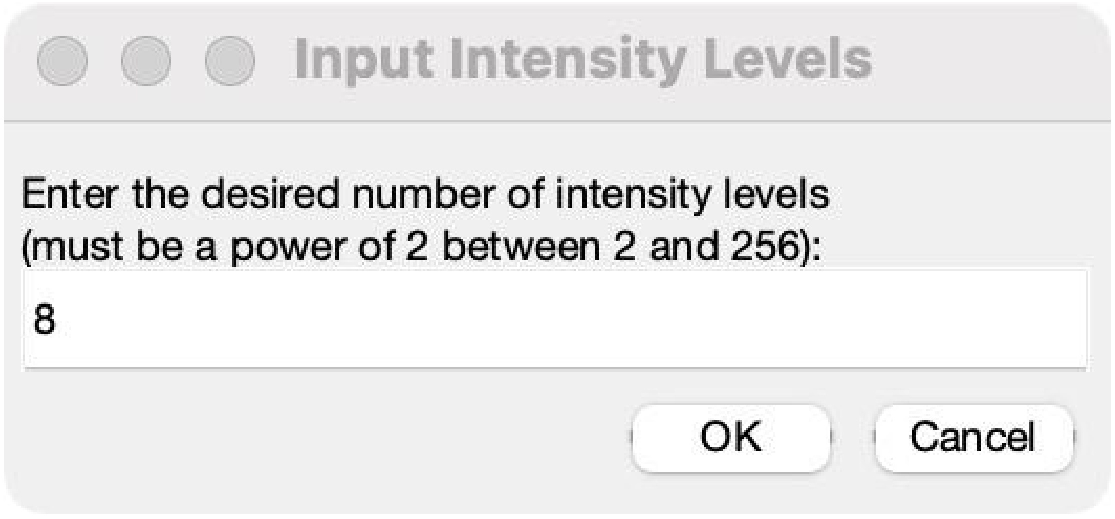


Enter the desired number of intensity levels
(must be a power of 2 between 2 and 256):

4

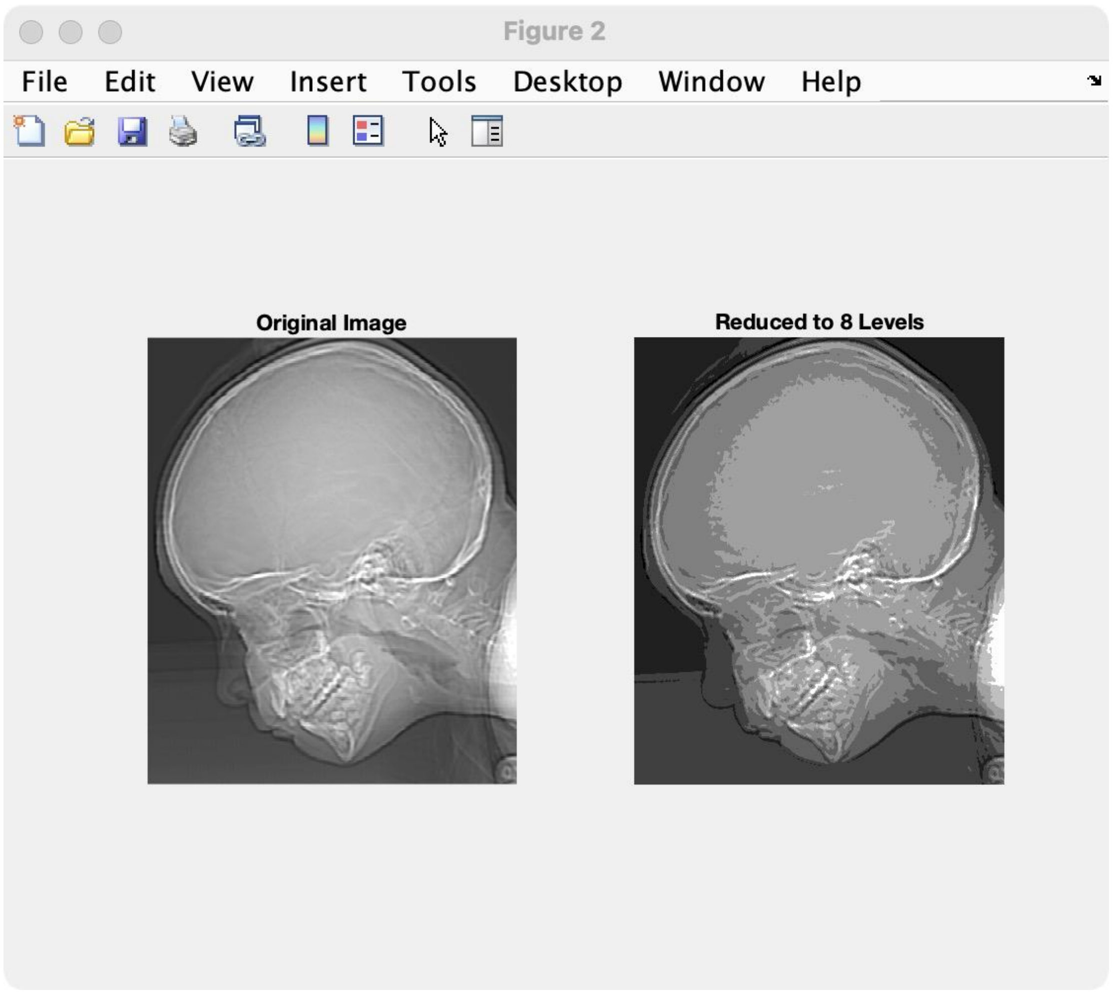


Example 2:

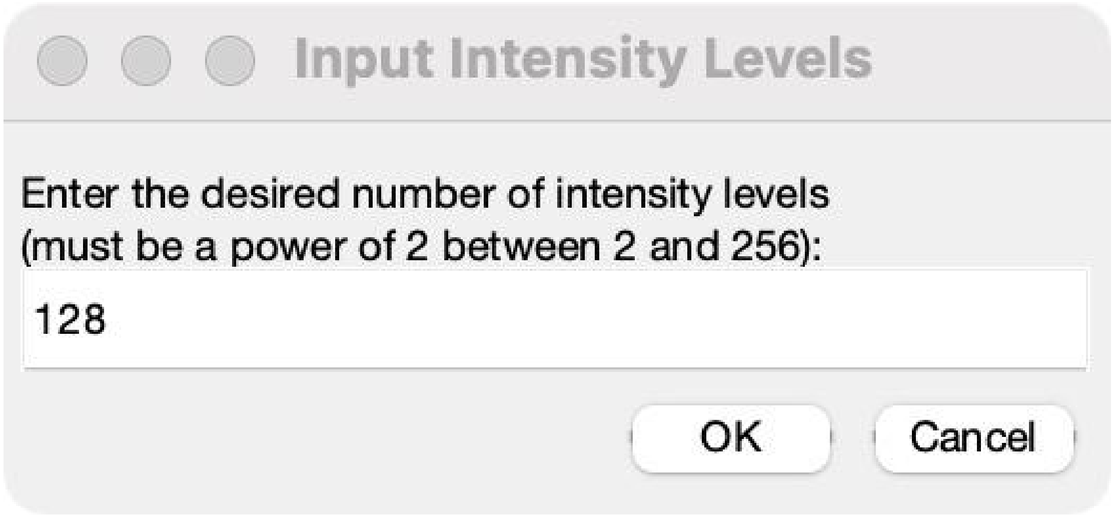


Enter the desired number of intensity levels
(must be a power of 2 between 2 and 256):

8

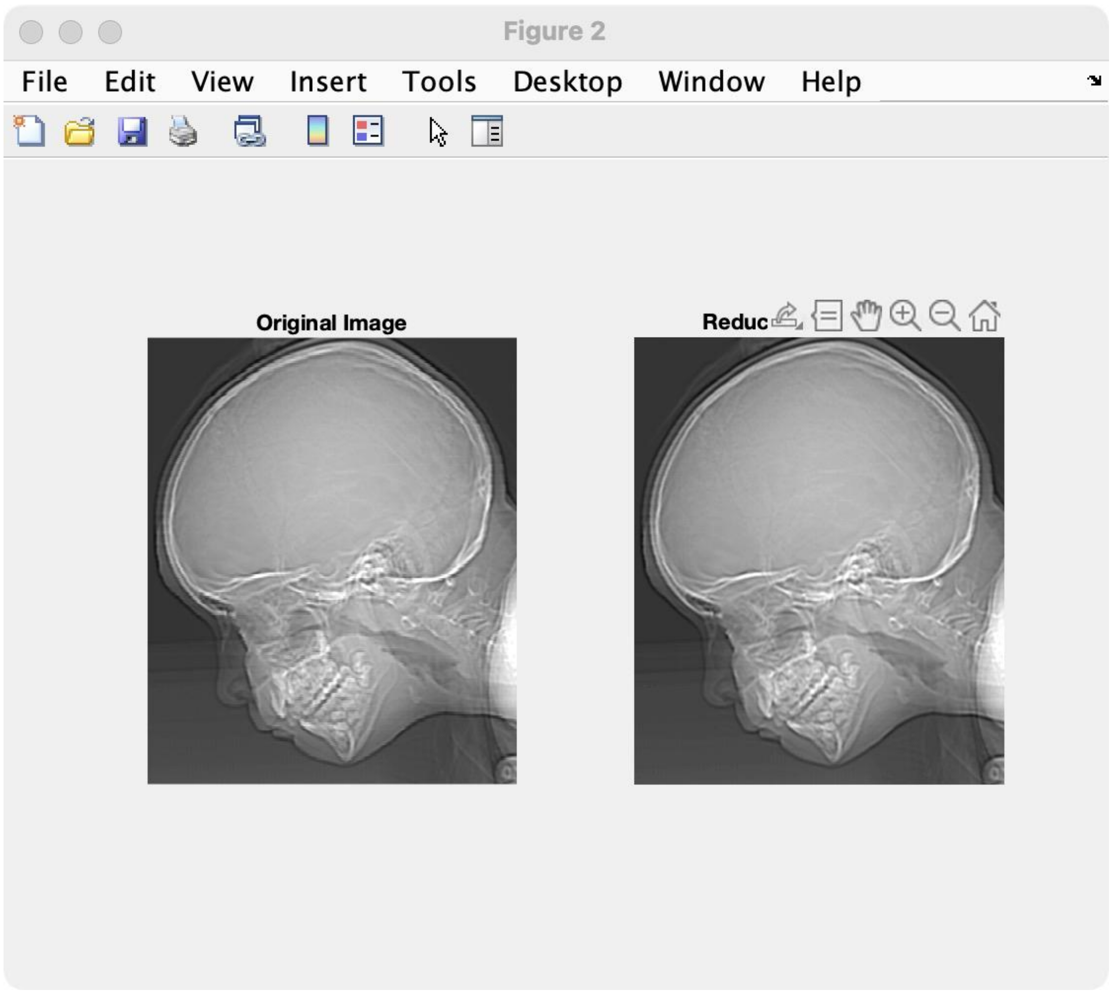


Example3:



Enter the desired number of intensity levels
(must be a power of 2 between 2 and 256):

128



PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
>> Project4
Image reduced to 4 levels saved as: reduced_image_levels_4.png
>> Project4
Image reduced to 8 levels saved as: reduced_image_levels_8.png
>> Project4
Image reduced to 128 levels saved as: reduced_image_levels_128.png
>> 
```

【小结或讨论】

本次实验通过 MATLAB 编程完成了图像生成、格式转换、负片处理、缩放操作以及灰度级缩减等任务，并成功达到了实验预期目标。在图像生成与格式转换环节，实验生成了一张中心包含白色矩形的黑色背景图像，并保存为 BMP、TIF 和 JPG 格式。通过比较发现，

不同图像格式的存储体积差异较大，其中 BMP 格式未压缩，体积较大，而 JPG 格式因压缩机制存储体积最小。

在图像负片处理方面，利用 MATLAB 的逻辑操作生成了图像的负片，成功实现亮度反转。这种负片处理技术能够增强图像对比度，尤其在医学影像分析等领域具有广泛应用价值。此外，实验还通过编写像素复制函数，实现了对图像的放大与缩小。实验表明，图像缩小会导致细节丢失，而放大由于像素重复填充，会出现明显的马赛克效应。这进一步说明，高精度应用场景中需要更高级的插值算法来提升缩放后的图像质量。

灰度级缩减实验展示了通过减少灰度级数量降低图像存储需求的方法。在将灰度级从 256 减少到 2 后，图像细节锐减，部分区域显示粗糙，但对简单的图像表现仍具有一定应用价值。这表明灰度级压缩是一种有效的图像存储优化方式，但需在细节保留和存储节省之间取得平衡。

实验充分展示了 MATLAB 在图像处理中的强大功能，如便捷的函数调用和高效的可视化操作，为数字图像处理提供了强有力的支持。同时，本次实验也凸显了图像压缩与细节保留之间的矛盾。例如，简单的像素复制缩放虽然实现了图像调整，但对于高质量图像处理还需采用更高级的算法。