安徽大学《数字图像处理（双语）》实验报告（3）

学号：WA2214014 专业：人工智能 姓名及签字：杨跃浙

实验日期：2024 年 12 月 10 日 实验成绩：________ 教师签字：________

【实验名称】： 基于直方图均衡的图像增强技术

【实验目的】：

1. 熟悉和掌握图像的直方图的意义

2. 通过 MATLAB 编程实现图像直方图的绘制并分析其和图像外观的关系

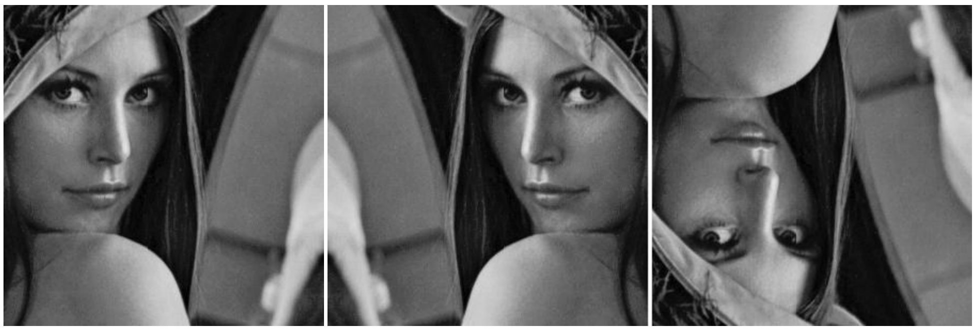
3. 通过 MATLAB 编程实现基于直方图均衡的图像增强

【实验内容】

PROJECT 03-04

Image Flipping

Write a MATLAB function **flipImage** which flips an image either vertically or horizontally. The function should take two parameters – the matrix storing the image data and a flag to indicate whether the image should be flipped vertically or horizontally. Use this function from the command line to flip the image woman.bmp both vertically and horizontally which should give the following results.



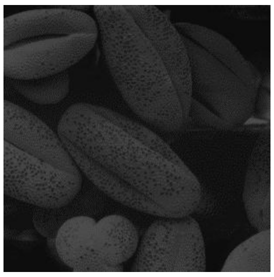
PROJECT 03-05

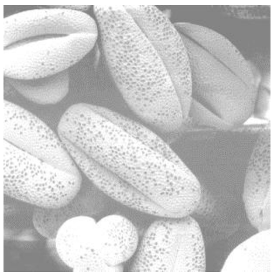
Image Histogram

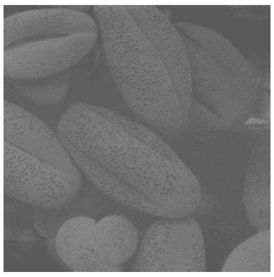
Write a MATLAB function, **generateHistogram**, which generates the histogram of an image. The function should take an image data array (with pixel values in the range 0 – 255) as its only parameter and return an array containing the histogram of the image. The histogram can be displayed using the built in MATLAB function hist. For example:

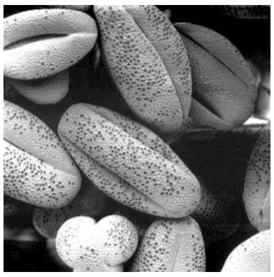
A = **generateHistogram**(Image); **bar**(A);

Use this new function to generate and display histograms for the following images (darkPollen.jpg, lightPollen.jpg, lowContrastPollen.jpg and pollen.jpg).


Dark Pollen


light Pollen


low Contrast Pollen


pollen

PROJECT 03-06

Histogram Equalization

a) Implement the histogram equalization using your own code instead of the function in MATLAB. Perform histogram equalization on the above 4 images.

b) Plot the histogram of the original images and the histogram-equalization enhanced images.

As a minimum, your report should include the original image, a plot of its histogram, the enhanced image, and a plot of its histogram. Use this information to explain why the resulting image was enhanced as it was.

【实验代码和结果】

Project03-04：

Code:

flipimage.m:

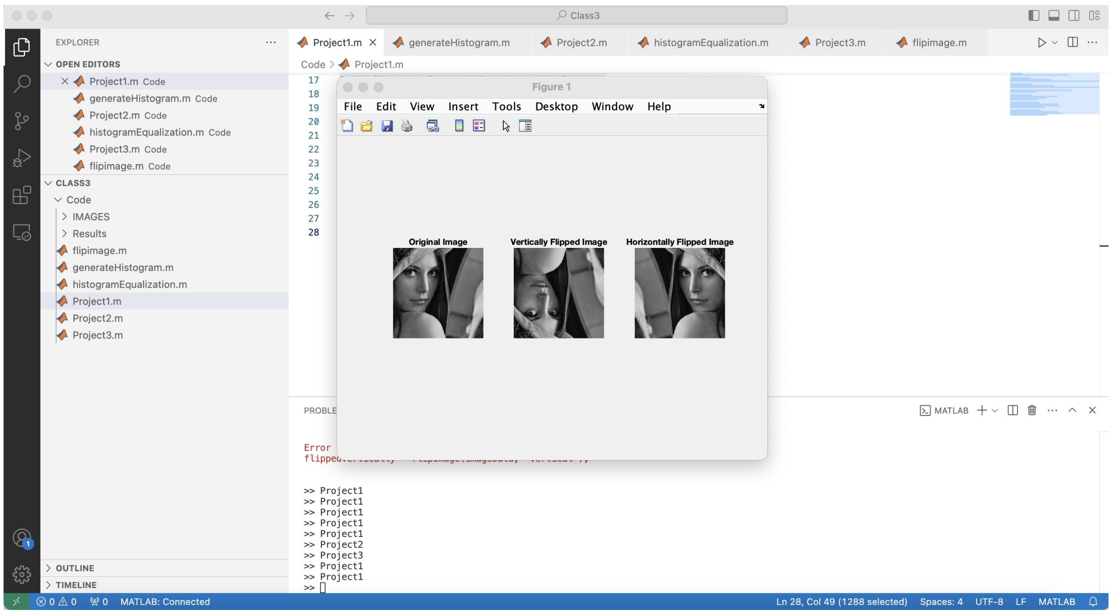
```
function flippedImage = flipimage(imageData, flipDirection)
% flipImage Flips an image vertically or horizontally.
% Parameters:
% imageData - A matrix storing the image data
% flipDirection - A string, either 'vertical' or 'horizontal'
% Returns:
% flippedImage - The flipped image matrix

if strcmpi(flipDirection, 'vertical')
flippedImage = flipud(imageData); % Flip the image vertically
elseif strcmpi(flipDirection, 'horizontal')
flippedImage = fliplr(imageData); % Flip the image
horizontally
else
error('Invalid flip direction. Use "vertical" or
"horizontal".');
end
end
```

Project1.m:

```
% Load the image
imageData =
imread('/Users/youngbean/Desktop/Class3/Code/IMAGES/woman.
tif');

% Flip the image vertically
flippedVertically = flipimage(imageData, 'vertical');
imwrite(flippedVertically,
'/Users/youngbean/Desktop/Class3/Code/Results/woman_flipp
ed_vertically.png'); % Save the vertically flipped image

% Flip the image horizontally
flippedHorizontally = flipimage(imageData, 'horizontal');
imwrite(flippedHorizontally,
'/Users/youngbean/Desktop/Class3/Code/Results/woman_flipp
ed_horizontally.png'); % Save the horizontally flipped image
```

```matlab
% Display original, vertically flipped, and horizontally
flipped images together
figure; % Open a new figure window

% Display the original image
subplot(1, 3, 1); % Create a subplot in a 1x3 grid, position
1
imshow(imageData); % Display the original image
title('Original Image'); % Add title

% Display the vertically flipped image
subplot(1, 3, 2); % Create a subplot in a 1x3 grid, position
2
imshow(flippedVertically); % Display the vertically flipped
image
title('Vertically Flipped Image'); % Add title

% Display the horizontally flipped image
subplot(1, 3, 3); % Create a subplot in a 1x3 grid, position
3
imshow(flippedHorizontally); % Display the horizontally
flipped image
title('Horizontally Flipped Image'); % Add title
```

Result:



```
Error
flipped

>> Project1
>> Project1
>> Project1
>> Project1
>> Project1
>> Project2
>> Project3
>> Project1
>> Project1
>> 
```

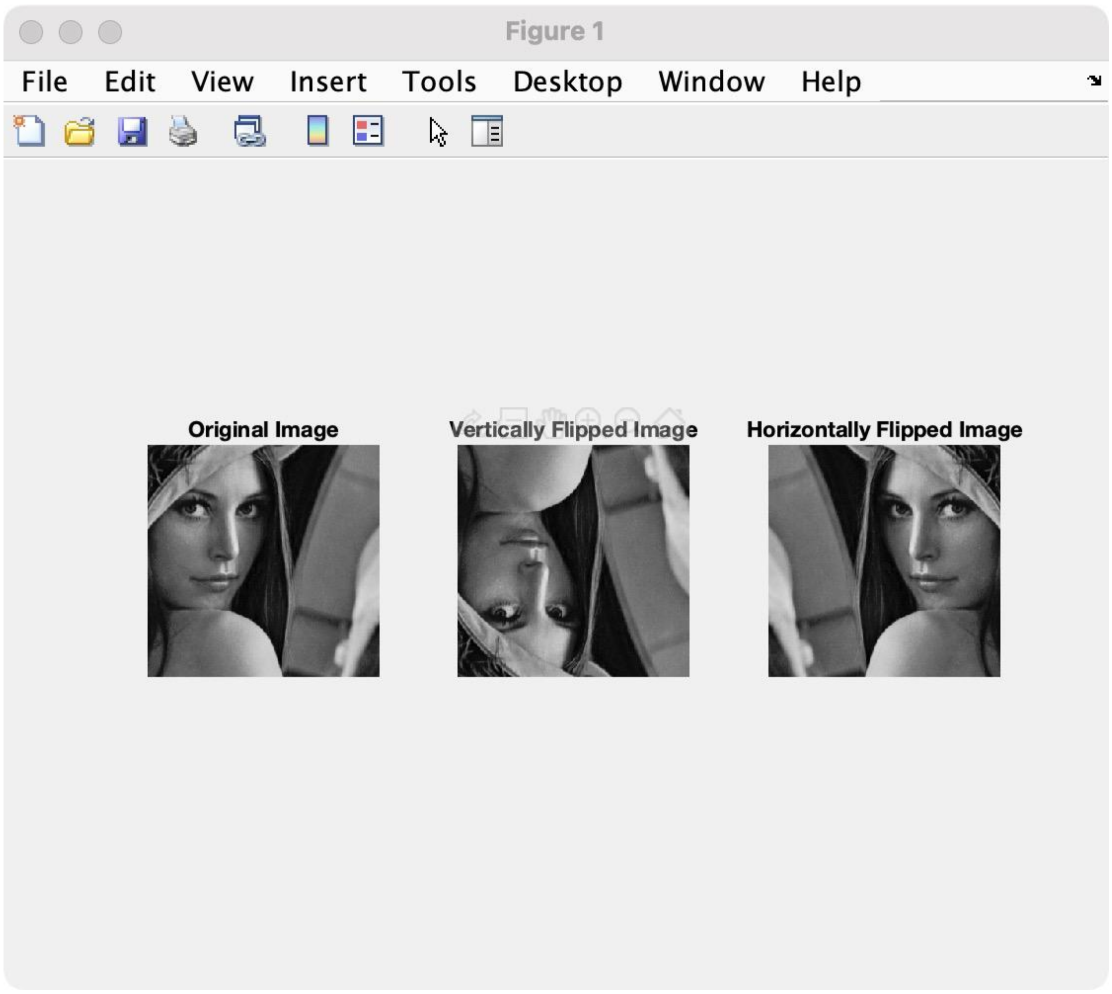


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help
Original Image
Vertically Flipped Image
Horizontally Flipped Image

Project03-05:

Code:

generateHistogram.m:

```
function histogramArray = generateHistogram(imageData)
% generateHistogram Generates the histogram of an image.
% Parameters:
% imageData - An image array with pixel values in the range
0-255.
% Returns:
% histogramArray - A 1x256 array containing the histogram
values.
% Convert image to grayscale if it is not already
if size(imageData, 3) == 3
imageData = rgb2gray(imageData); % Convert to grayscale
end

% Initialize histogram array
histogramArray = zeros(1, 256);

% Calculate histogram
for intensity = 0:255
histogramArray(intensity + 1) = sum(imageData(:) ==
intensity);
```

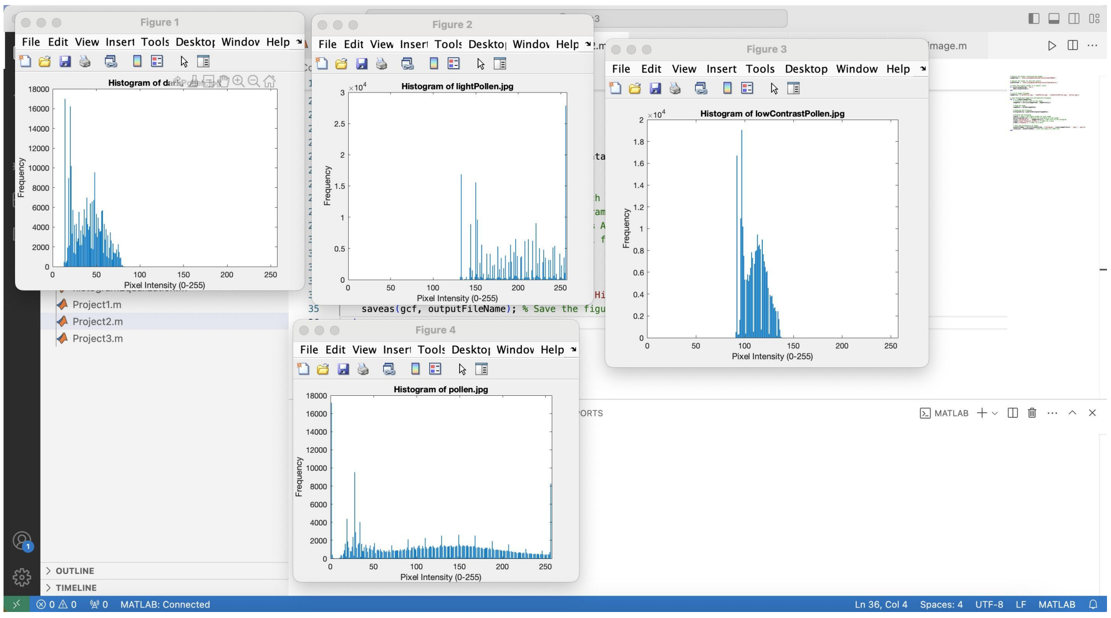
```
end
end
```

Project2.m:

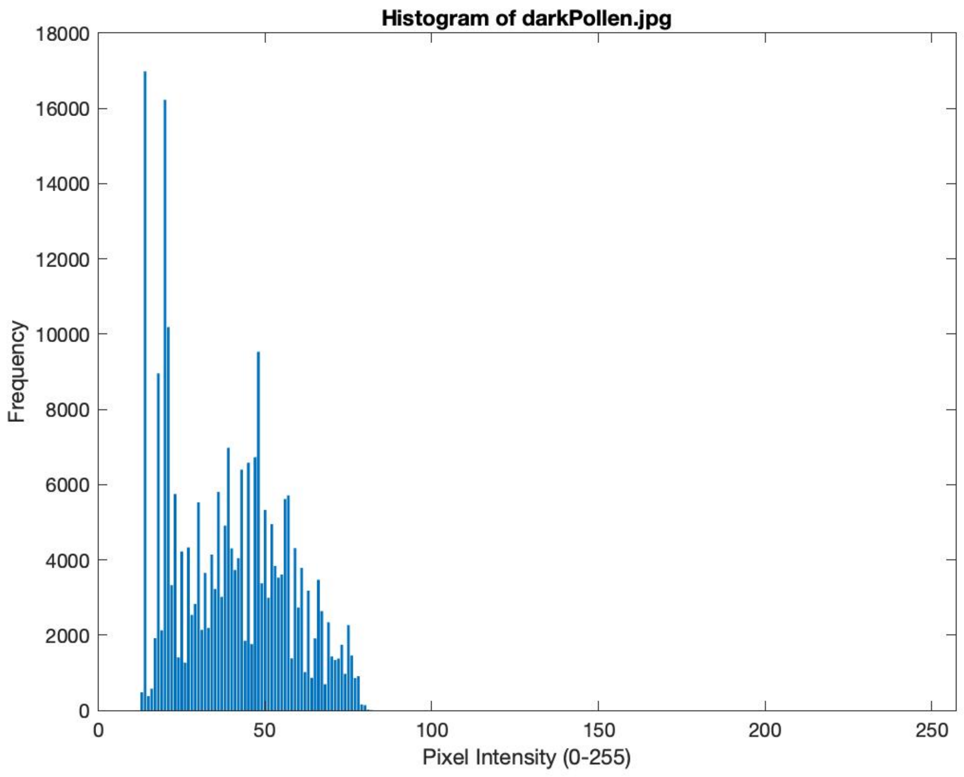
```
% Specify the folder containing the images
imageFolder =
'/Users/youngbean/Desktop/Class3/Code/IMAGES';

% Specify the folder to save the results
outputFolder =
'/Users/youngbean/Desktop/Class3/Code/Results';

% Create the results folder if it doesn't exist
if ~exist(outputFolder, 'dir')
mkdir(outputFolder);
end

% List of image filenames
imageFiles = {'darkPollen.jpg', 'lightPollen.jpg',
'lowContrastPollen.jpg', 'pollen.jpg'};

% Loop through each image file and generate histogram
for i = 1:length(imageFiles)
% Construct the full path of the image
imagePath = fullfile(imageFolder, imageFiles{i});
% Read the image
imageData = imread(imagePath);
% Generate the histogram
histogramArray = generateHistogram(imageData);
% Display the histogram
figure; % Open a new figure window for each image
bar(histogramArray); % Display the histogram as a bar graph
title(['Histogram of ', imageFiles{i}]); % Add title to the
histogram
xlabel('Pixel Intensity (0-255)'); % Label for x-axis
ylabel('Frequency'); % Label for y-axis
% Save the histogram as an image
outputFileName = fullfile(outputFolder, ['Histogram_',
erase(imageFiles{i}, '.jpg'), '.png']);
```

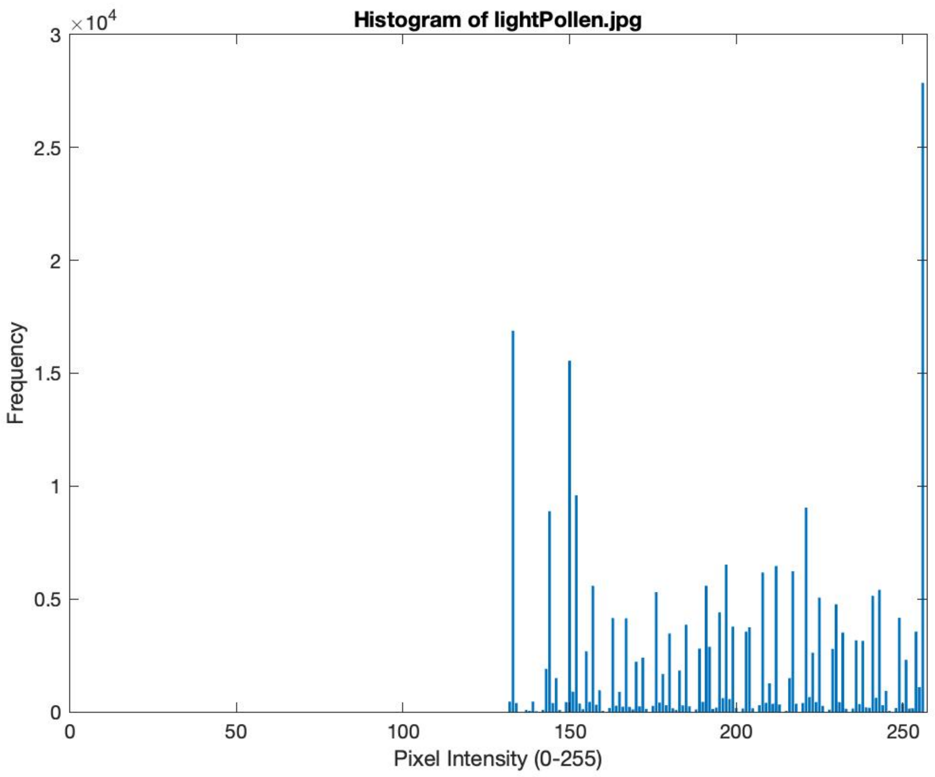
```
saveas(gcf, outputFileName); % Save the figure as a PNG file
end
```

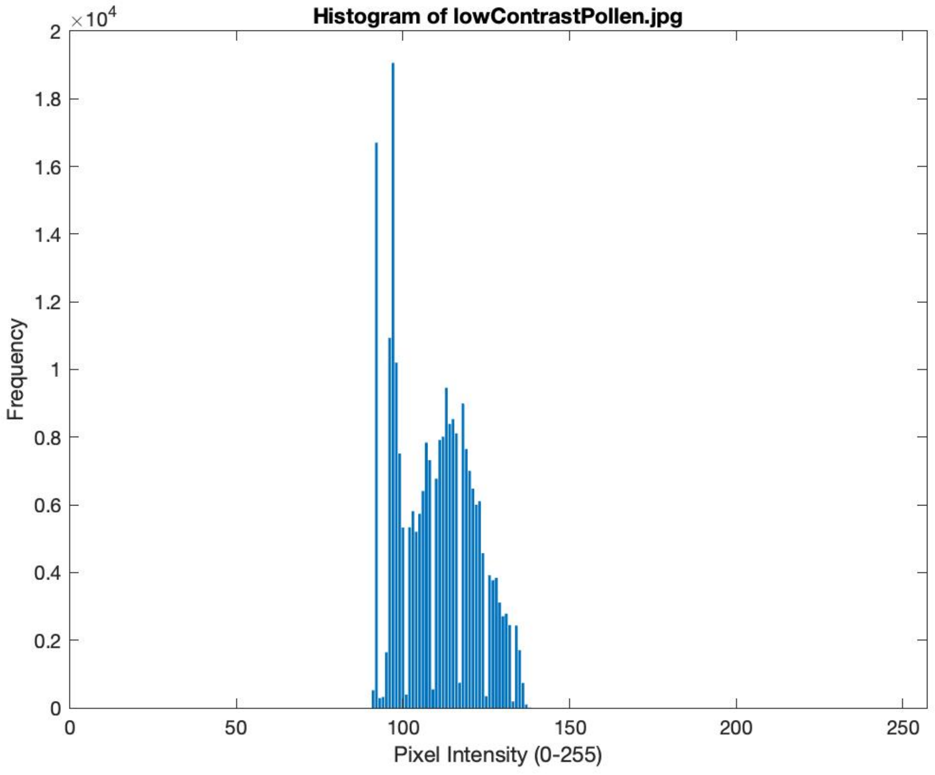
Result:

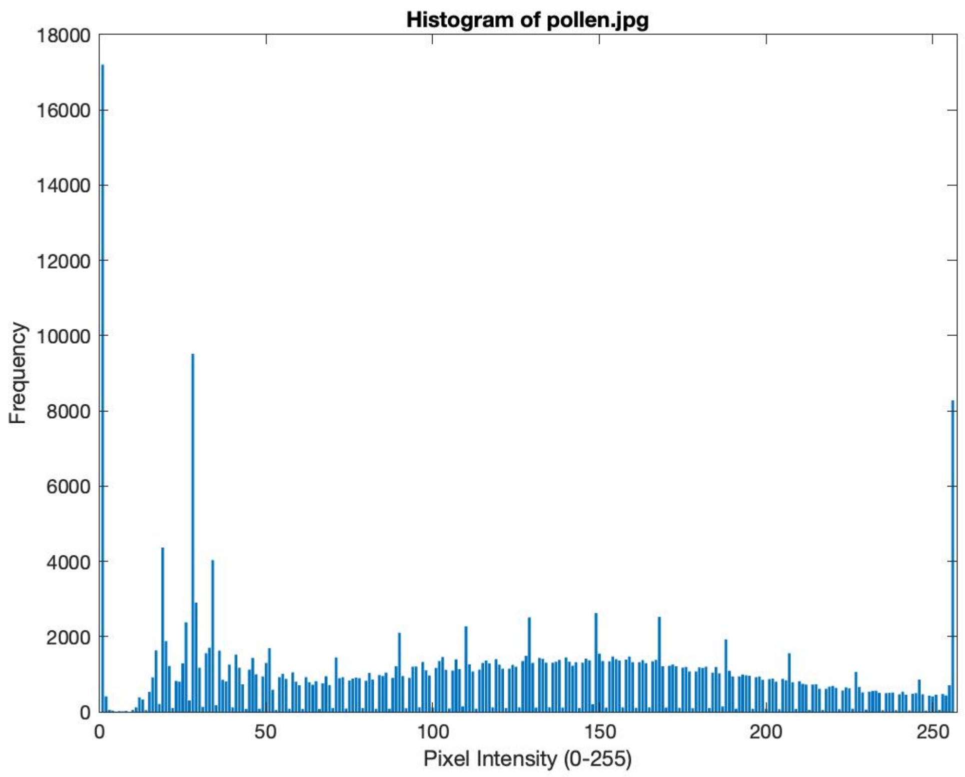


```
saveas(gcf, outputFileName); % Save the figu
```




Histogram of lightPollen.jpg
×10^4
3
2.5
2
1.5
1
0.5
0
Frequency
0
50
100
150
200
250
Pixel Intensity (0-255)


Histogram of lowContrastPollen.jpg
×10^4
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
0
Frequency
0
50
100
150
200
250
Pixel Intensity (0-255)



Project03-06:

Code:

histogramEqualization.m:

```
function enhancedImage = histogramEqualization(imageData)
% histogramEqualization Performs histogram equalization on
a grayscale image.
% Parameters:
% imageData - Grayscale image data (pixel values in 0-255
range).
% Returns:
% enhancedImage - Histogram-equalized grayscale image.
% Convert to grayscale if image is RGB
if size(imageData, 3) == 3
imageData = rgb2gray(imageData);
end
% Calculate the histogram
histArray = zeros(1, 256);
for intensity = 0:255
histArray(intensity + 1) = sum(imageData(:) == intensity);
```

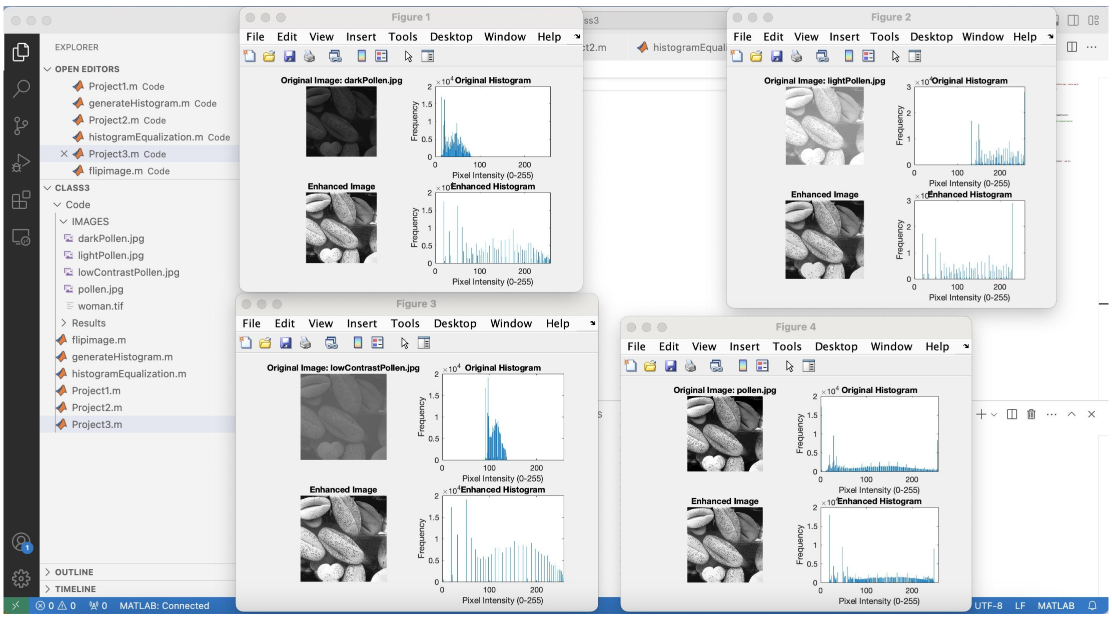
```
end
% Compute the cumulative distribution function (CDF)
cdf = cumsum(histArray) / numel(imageData);
% Map the original pixel values to new values based on the
CDF
equalized = uint8(cdf(imageData + 1) * 255);
% Return the equalized image
enhancedImage = equalized;
end
```

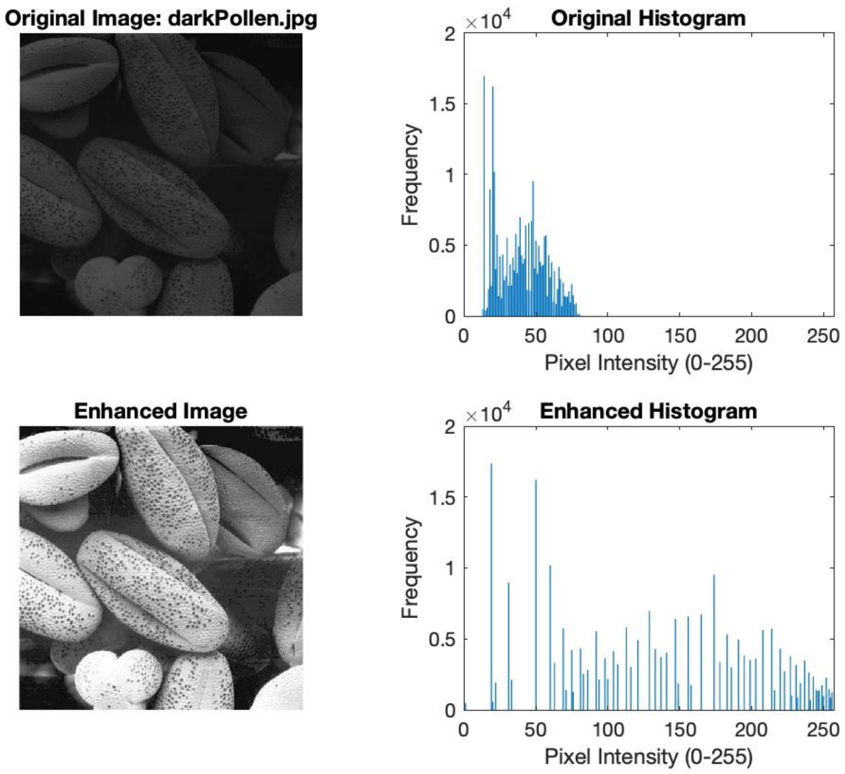
Project3.m:

```
% Specify the folder containing the images
imageFolder =
'/Users/youngbean/Desktop/Class3/Code/IMAGES';

% List of image filenames
imageFiles = {'darkPollen.jpg', 'lightPollen.jpg',
'lowContrastPollen.jpg', 'pollen.jpg'};

% Output folder for results
outputFolder =
'/Users/youngbean/Desktop/Class3/Code/Results';
if ~exist(outputFolder, 'dir')
mkdir(outputFolder);
end

% Loop through each image file
for i = 1:length(imageFiles)
% Construct the full path of the image
imagePath = fullfile(imageFolder, imageFiles{i});
% Read the image
imageData = imread(imagePath);
% Perform histogram equalization
enhancedImage = histogramEqualization(imageData);
% Save the enhanced image
enhancedImagePath = fullfile(outputFolder, ['enhanced_',
imageFiles{i}]);
imwrite(enhancedImage, enhancedImagePath);
% Compute histograms
```

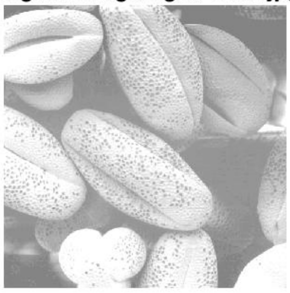
```
originalHist = generateHistogram(imageData); % Use previous
histogram function
enhancedHist = generateHistogram(enhancedImage);
% Plot the results
figure;
subplot(2, 2, 1);
imshow(imageData);
title(['Original Image: ', imageFiles{i}]);
subplot(2, 2, 2);
bar(originalHist);
title('Original Histogram');
xlabel('Pixel Intensity (0-255)');
ylabel('Frequency');
subplot(2, 2, 3);
imshow(enhancedImage);
title('Enhanced Image');
subplot(2, 2, 4);
bar(enhancedHist);
title('Enhanced Histogram');
xlabel('Pixel Intensity (0-255)');
ylabel('Frequency');
% Save the figure
saveas(gcf, fullfile(outputFolder, ['comparison_',
imageFiles{i}, '.png']));
end
```

Result:

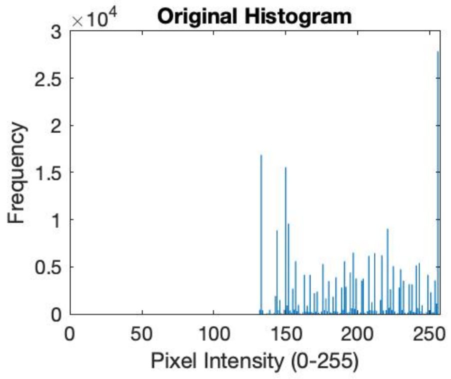

EXPLORER
OPEN EDITORS
Project1.m Code
generateHistogram.m Code
Project2.m Code
histogramEqualization.m Code
Project3.m Code
flipimage.m Code
CLASS3
Code
IMAGES
darkPollen.jpg
lightPollen.jpg
lowContrastPollen.jpg
pollen.jpg
woman.tif
Results
flipimage.m
generateHistogram.m
histogramEqualization.m
Project1.m
Project2.m
Project3.m
OUTLINE
TIMELINE
MATLAB: Connected
Figure 1
Original Image: darkPollen.jpg
Original Histogram
Enhanced Image
Enhanced Histogram
Figure 2
Original Image: lightPollen.jpg
Figure 3
Original Image: lowContrastPollen.jpg
Figure 4
Original Image: pollen.jpg
Frequency
Pixel Intensity (0-255)
UTF-8
LF
MATLAB


Original Image: darkPollen.jpg
Original Histogram
Enhanced Image
Enhanced Histogram
$\times 10^4$
Frequency
Pixel Intensity (0-255)
0
0.5
1
1.5
2
0
50
100
150
200
250

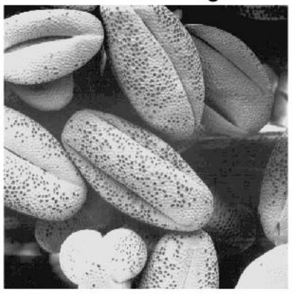
Original Image: lightPollen.jpg



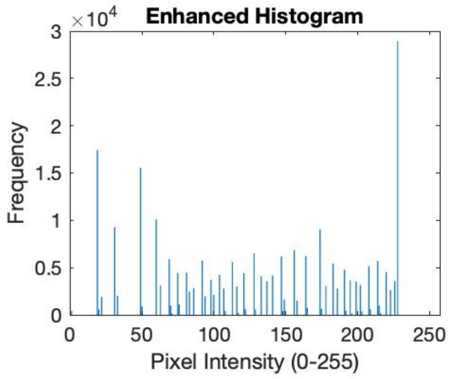
Original Histogram



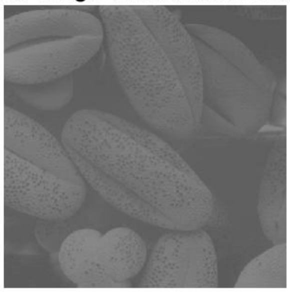
Enhanced Image



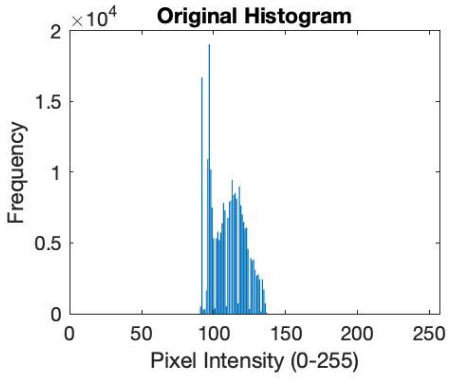
Enhanced Histogram



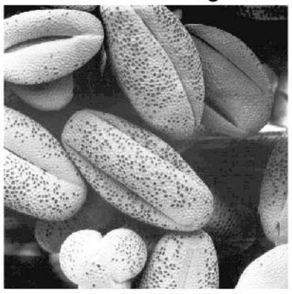
Original Image: lowContrastPollen.jpg



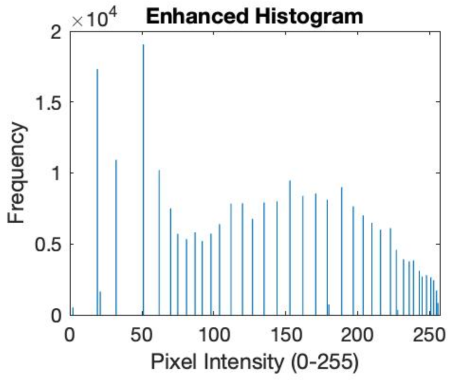
Original Histogram

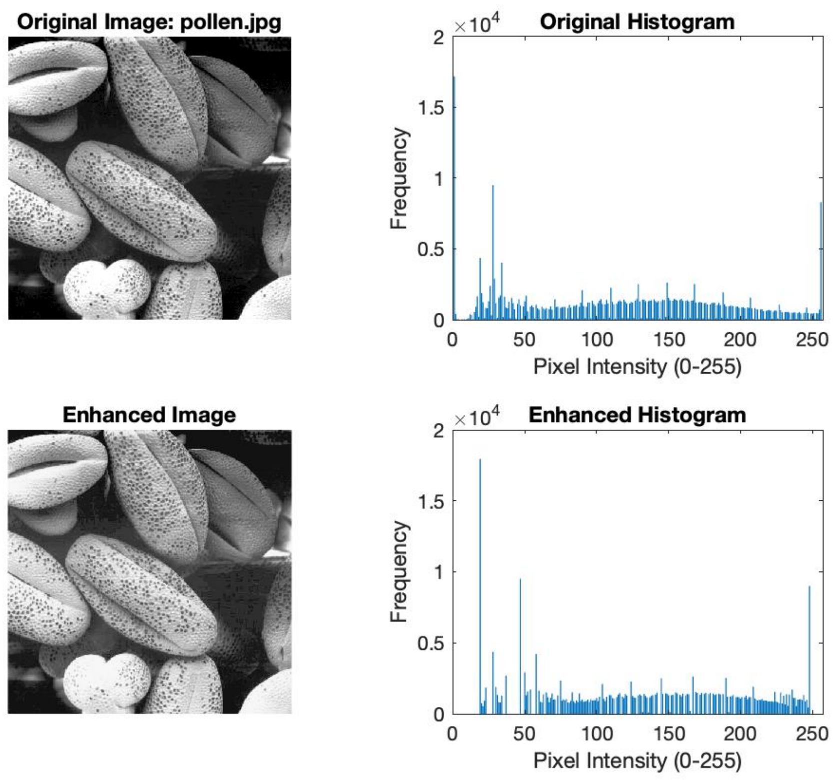


Enhanced Image

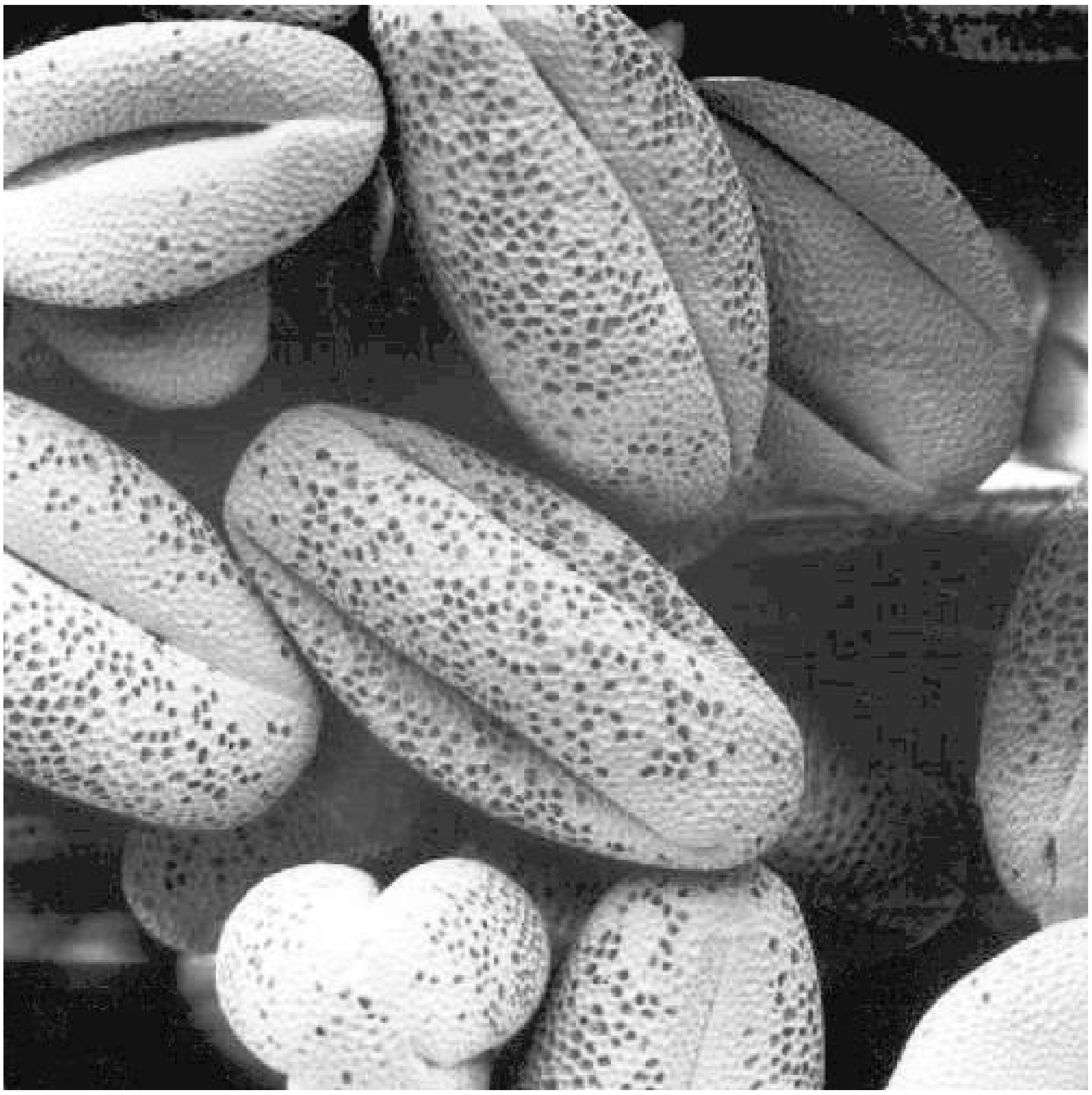


Enhanced Histogram

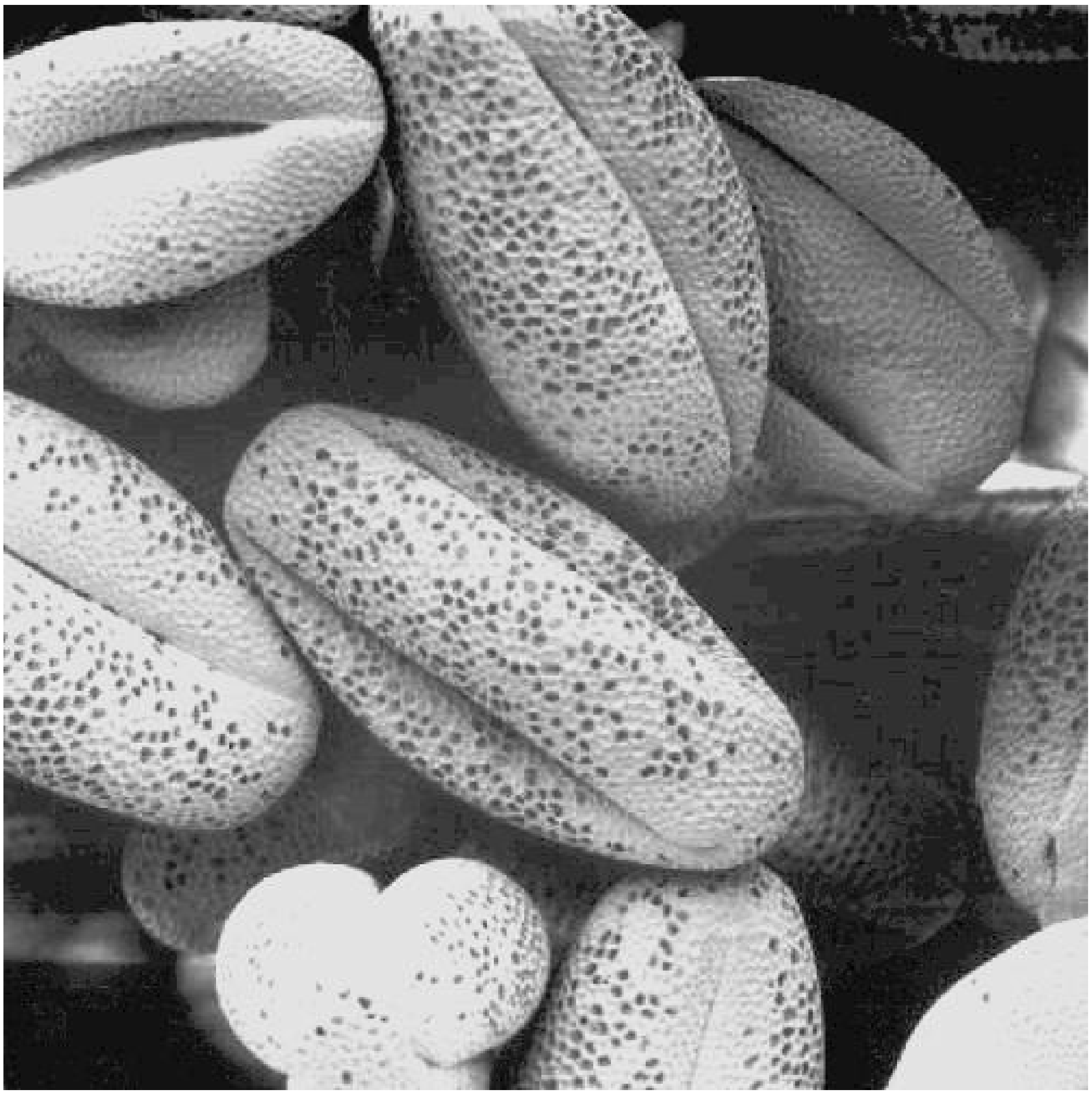


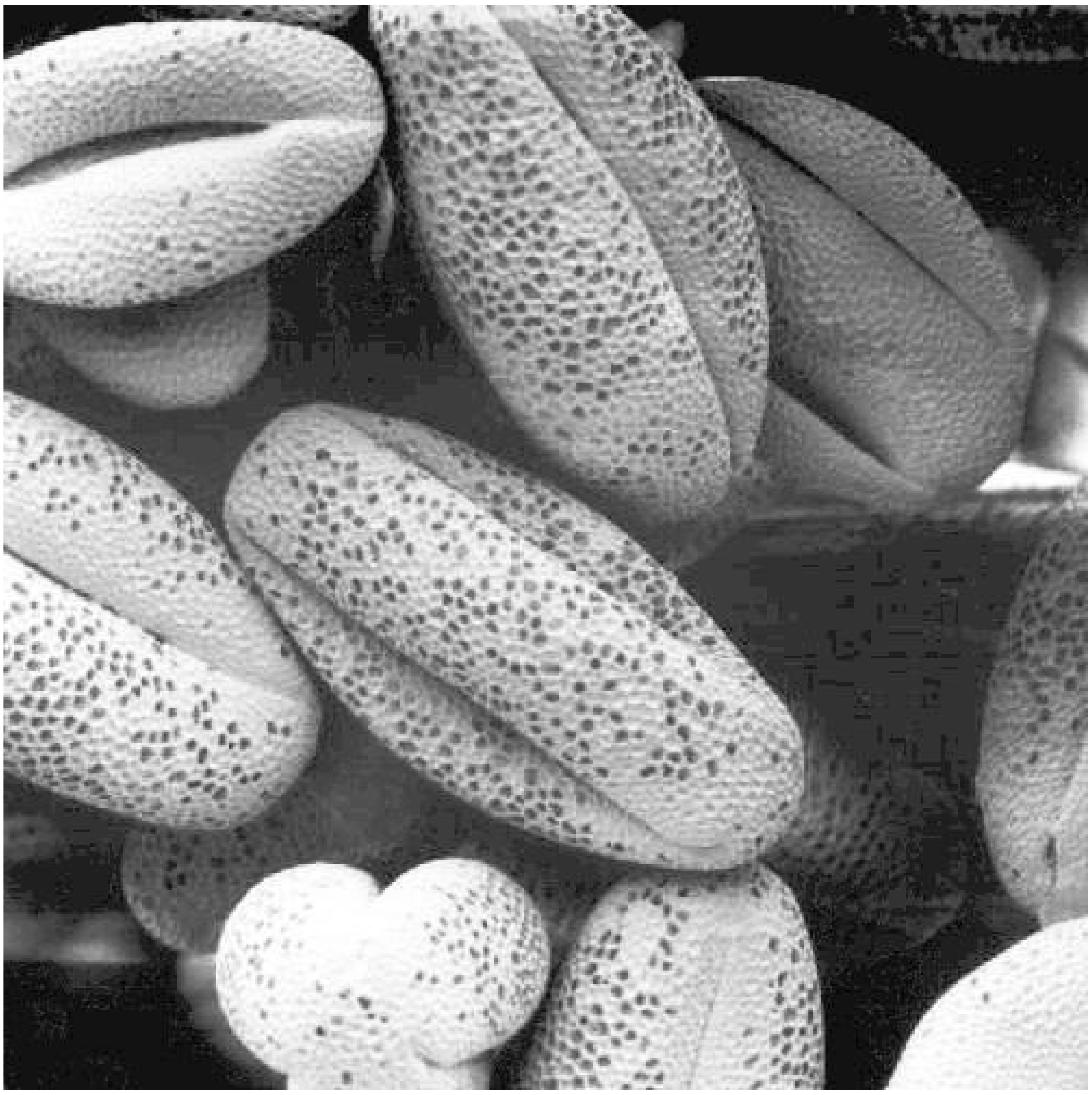


enhanced_darkPollen.jpg

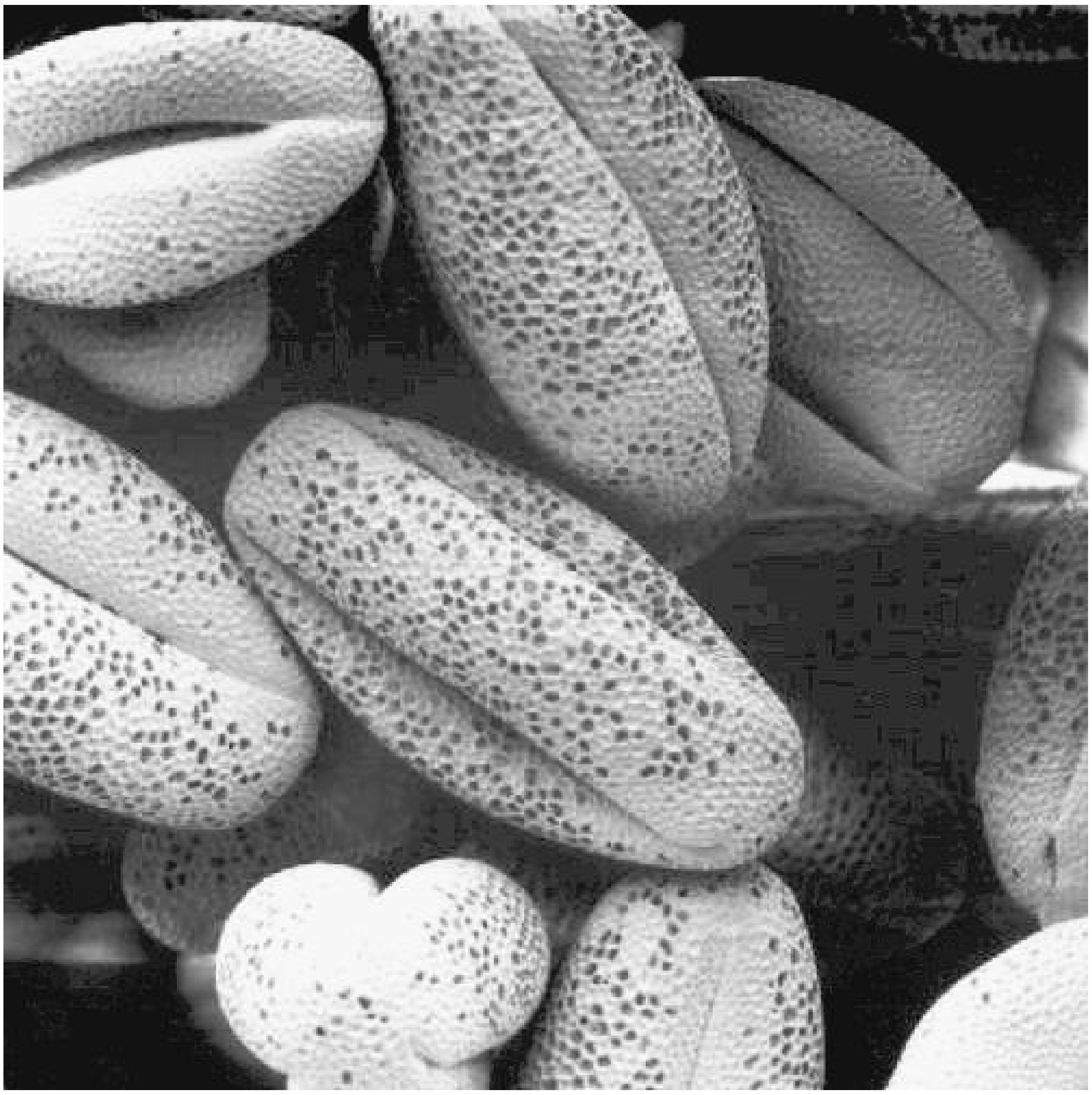


enhanced_lightPollen.jpg


enhanced_lowContrastPollen.jpg



enhanced_pollen.jpg



【小结或讨论】

在本次实验中，我深入探讨了图像直方图的特性以及其在图像增强中的应用，完成了图像翻转操作、直方图的生成与分析，以及基于直方图均衡化的图像增强技术。这些实验让我更加理解了图像处理的基本原理和 MATLAB 编程的实际应用。

首先，我通过编写代码生成了图像的直方图。直方图是反映图像灰度值分布的工具，对图像的亮度和对比度具有直观描

述。在实验中，我发现不同类型的图像直方图分布各有特点。例如，暗图像的直方图集中在灰度值的低范围，亮图像集中在灰度值的高范围，而低对比度图像的直方图范围窄。这些结果表明，直方图能够揭示图像的基本特性，为图像增强提供了方向。

为了生成直方图，我实现了以下计算公式:

$$H(i) = \sum_{x=1}^{M} \sum_{y=1}^{N} \delta\,(f(x,y) - i)$$

其中，$H(i)$ 表示灰度值为 i 的像素数，$f(x,y)$ 为图像在像素点 (x,y) 的灰度值，δ 为 Kronecker 函数，用于统计与灰度值匹配的像素数。通过这些公式，我实现了精确的像素统计，并绘制了直方图。

接下来，我在图像增强实验中实现了直方图均衡化。通过自定义的程序，我使用累计分布函数（CDF）调整了像素值分布，使图像的直方图更加均匀，视觉上对比度得到了显著提升。直方图均衡化的核心公式为:

$$T(r) = \operatorname{round}\left((L-1)\cdot\sum_{i=0}^{r}\frac{H(i)}{MN}\right)$$

其中，$T(r)$ 是灰度值 r 的映射结果，L 是灰度级总数，$H(i)$ 是灰度值 i 的频数，M 和 N 分别是图像的宽度和高度。

实验结果表明，直方图均衡化对于提升图像对比度具有重要作用。通过对低对比度图像进行均衡化，其直方图从集中分布变得更加均匀，图像细节变得更加清晰。同时，对于暗图

像和亮图像，均衡化使其灰度分布扩展到整个范围，图像亮度得到了改善。值得注意的是，对于已经具有较均匀直方图的图像，均衡化的效果相对较弱。

此外，我还实现了图像的翻转操作，通过垂直和水平翻转验证了图像数据的几何变换特性。通过这部分实验，我了解到如何使用简单的几何变换操作来实现图像校正或数据增强。

综合来看，本次实验让我深入理解了图像直方图的意义及其在图像增强中的作用，同时通过实际编程实践巩固了相关算法的应用能力。这为我后续研究更加复杂的图像处理技术奠定了坚实基础。