Image and Video Processing Lab

**Name: -** Yash Rajput
**Registration Number:** 211060042 **Date: -** 30.04.2024

Experiment 3
JPEG Compression

**Aim: -** The objective of this experiment is to evaluate the impact of JPEG compression on image file size and visual quality. The program should prompt the user to input parameters such as the binary or color image.

**Software Used: -** MATLAB

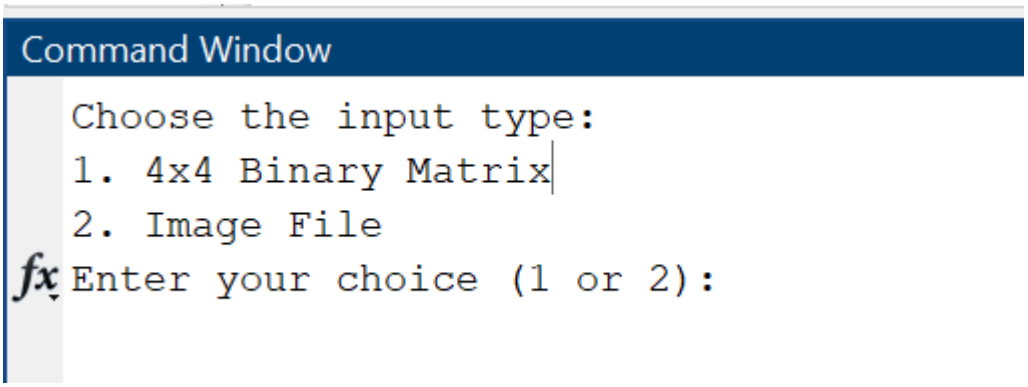
**Code:**

```
% IVP Lab – Experiment No: 3 – JPEG Compression
% Yash Rajput – TY EC - 211060042
clc;
try
    % Display menu to select input type
    fprintf('Choose the input type:\n');
    fprintf('1. 4x4 Binary Matrix\n');
    fprintf('2. Image File\n');
    choice = input('Enter your choice (1 or 2): ');

    if choice == 1
        % Option 1: Input a 4x4 binary matrix
        fprintf('Enter a 4x4 binary matrix (0s and 1s):\n');
        binary_matrix = input('');  % Prompt user to input the binary matrix

        % Validate input dimensions
        if ~isequal(size(binary_matrix), [4, 4])
            error('Invalid matrix dimensions. Please provide a 4x4 matrix.');
        end

        % Validate binary values (0s and 1s)
        if ~all(binary_matrix(:) == 0 | binary_matrix(:) == 1)
            error('Invalid matrix values. Please provide a matrix of 0s and 1s.');
        end

        % Convert binary matrix to uint8 image (scale to 0-255 range)
        input_image = uint8(binary_matrix) * 255;

    elseif choice == 2
        % Option 2: Select an image file
        [filename, pathname] = uigetfile({'*.png;*.jpg;*.jpeg', 'Image Files ( *.jpeg)'},
'Select an Image File');

        % Check if the user cancelled the file selection
        if isequal(filename, 0) || isequal(pathname, 0)
            disp('User cancelled the operation.');
            return;
        end
```

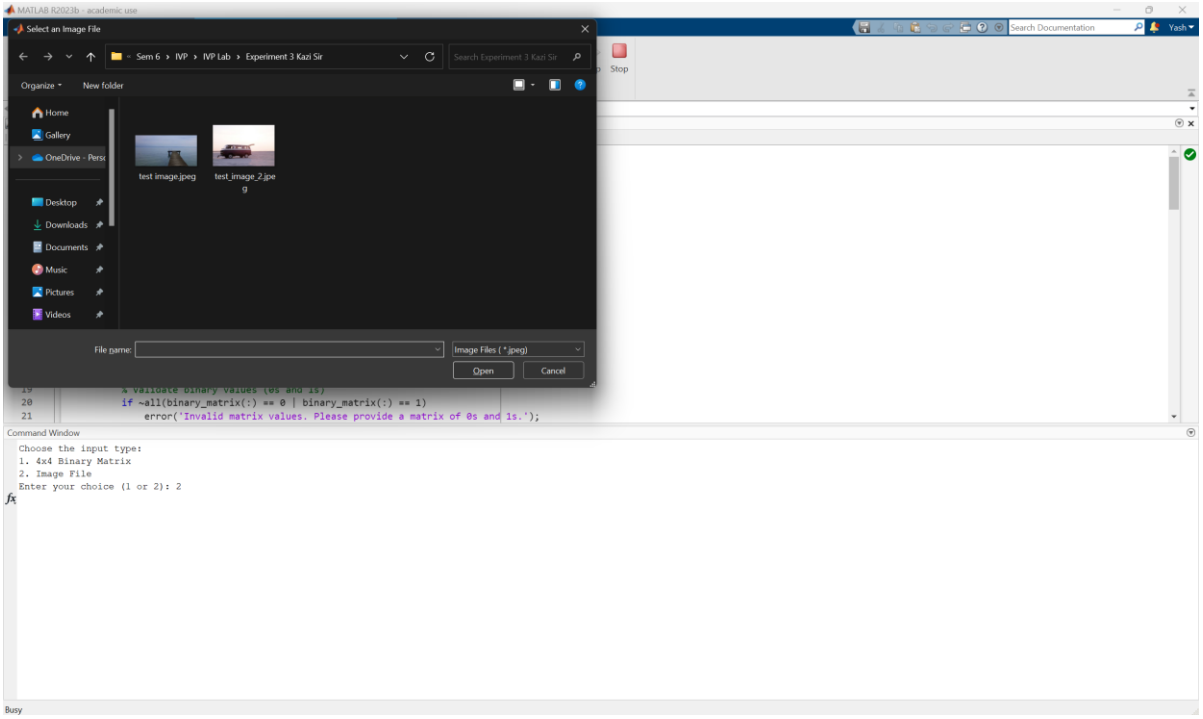
```
        % Construct the full path to the selected image file
        input_image_path = fullfile(pathname, filename);

        % Open the input image
        input_image = imread(input_image_path);

        % Check if the input image is grayscale (convert to RGB for display)
        if ndims(input_image) == 2 %#ok<ISMAT>
            input_image = cat(3, input_image, input_image, input_image);  % Convert to RGB
        end

    else
        error('Invalid choice. Please enter 1 or 2.');
    end

    % Display the input image
    figure('Position', [100, 100, 1000, 400]);

    % Display original image with information
    subplot(1, 2, 1);
    imshow(input_image);
    if choice == 1
        title('Input Binary Image');
        original_size = 4 * 4;  % Size of 4x4 binary matrix (in elements)
    else
        title('Input Image');
        original_info = dir(input_image_path);
        original_size = original_info.bytes;  % Size of original image file
    end
    axis off;

    % Resize the image
    [height, width, ~] = size(input_image);
    new_size = [height/2, width/2];
    resized_image = imresize(input_image, new_size);

    % Save the resized image
    if choice == 1
        output_image_path = 'compressed_binary_image.jpg';
    else
        [~, name, ext] = fileparts(input_image_path);
        output_image_path = fullfile(pathname, ['compressed_' name ext]);
    end
    imwrite(resized_image, output_image_path, 'Quality', 50);

    % Display the compressed image
    subplot(1, 2, 2);
    imshow(resized_image);
    title('Compressed Image');
    axis off;
    % Get compressed image size and calculate compression ratio
    compressed_info = dir(output_image_path);
    compressed_size = compressed_info.bytes;  % Size of compressed image file
    compression_ratio = (1 - compressed_size/original_size) * 100   ;

    fprintf('Original Size: %d bytes\n', original_size);
    fprintf('Compressed Size: %d bytes\n', compressed_size);
    fprintf('Compression Ratio: %.2f%%\n', (1 - compressed_size/original_size) * 100);
catch ME
    % Error Handling
    fprintf('Error: %s\n', ME.message);
end
```

**User Interface:**

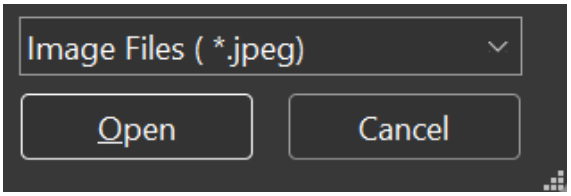
**Step 1:** The code begins with asking the user for the type of input




**Step 2:** Assuming that you select 2 where you are going to select image as input, making the code user friendly a file picker opens which can be used to select the image from any folder

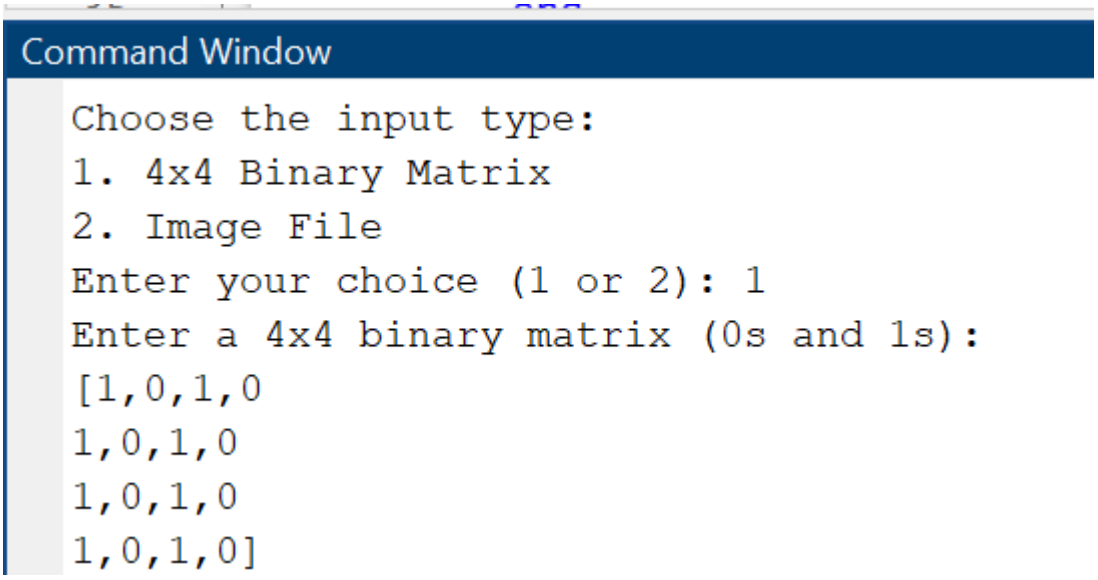



NOTE: For better error handling jpeg files can only be selected by regular expression set to jpeg

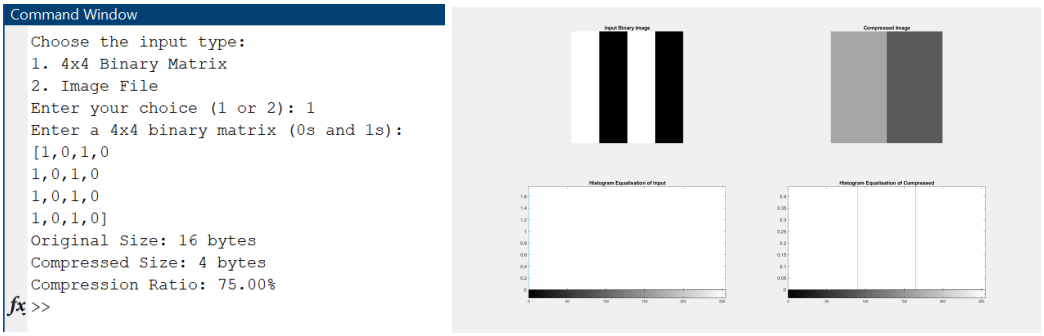



```
'Image Files ( *.jpeg)'},
```

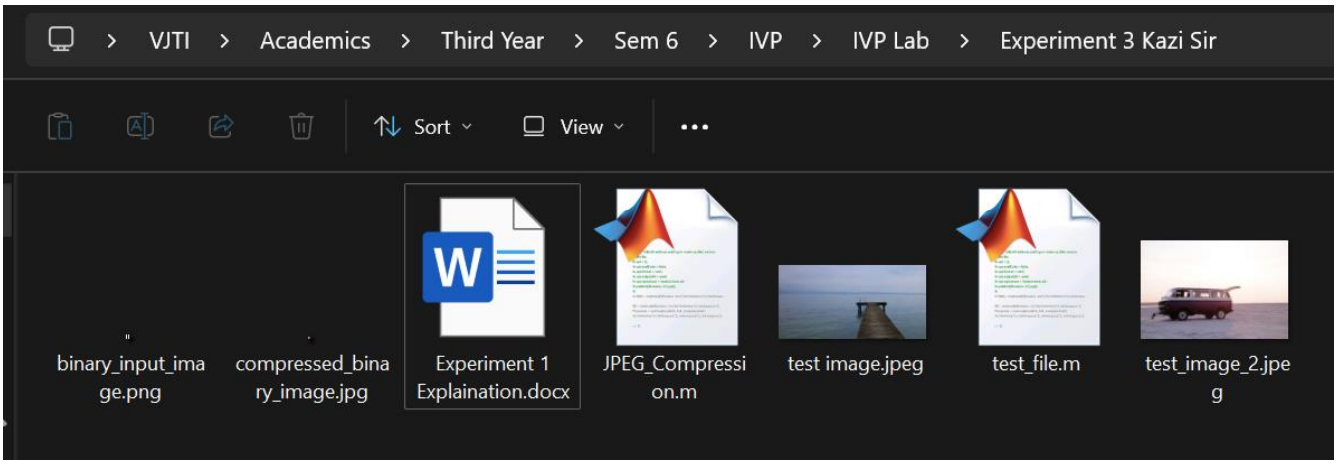
**Step 3:** If you select the 1st option you will be asked to enter the matrix in binary format




**Step 3:** After Entering your choice, you can see size of original image and compressed image along with the compression ratio in the command terminal and the Visual representation can be seen in the figure




**Step 4:** After the compression the if the user has selected binary input the compressed image and binary input image will be saved in your folder and in case of input image your compressed image will be saved in the same directory as your input image



## Error Handling:

### Case 1: User does not select an image and closes the file picker

```
Command Window
  Choose the input type:
  1. 4x4 Binary Matrix
  2. Image File
  Enter your choice (1 or 2): 2
  User canceled the operation.
fx >>
```

### Case 2: The Entered Matrix is not of 4x4 order

```
Command Window
  Choose the input type:
  1. 4x4 Binary Matrix
  2. Image File
  Enter your choice (1 or 2): 1
  Enter a 4x4 binary matrix (0s and 1s):
  [1,0,1
  1,0,1
  1,0,1
  1,0,1]
  Error: Invalid matrix dimensions. Please provide a 4x4 matrix.
```

### Case 3: The Entered Matrix contains non numeric number

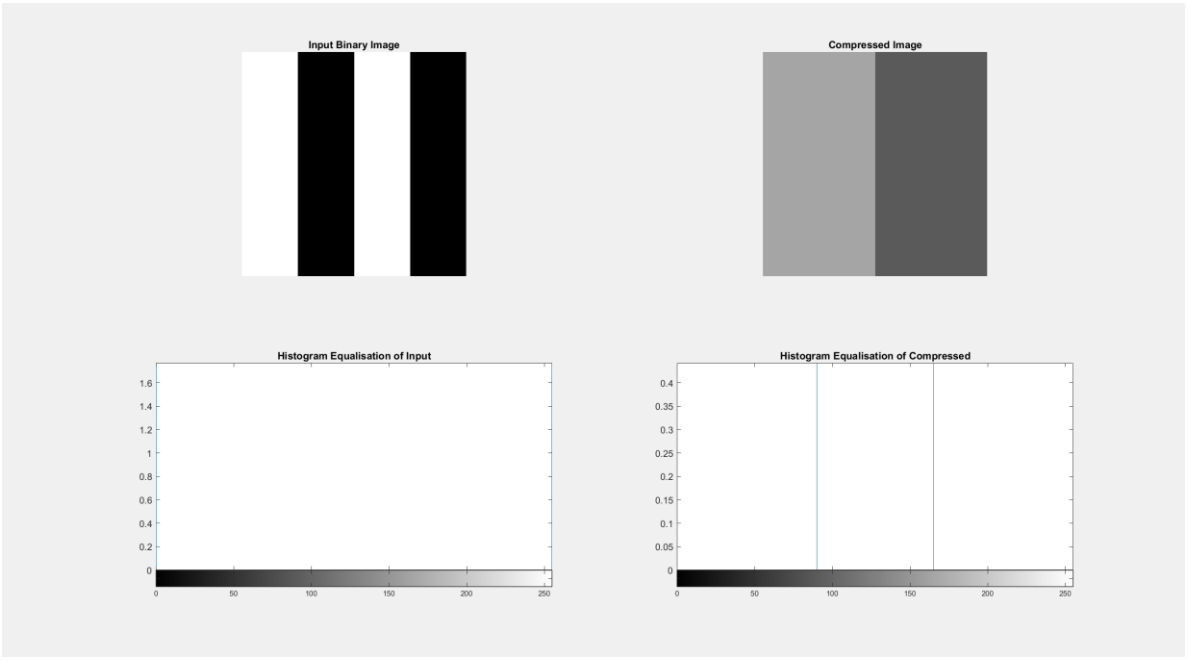
```
  Choose the input type:
  1. 4x4 Binary Matrix
  2. Image File
  Enter your choice (1 or 2): 1
  Enter a 4x4 binary matrix (0s and 1s):
  [1,0,1,a;
  1,0,1,0;
  1,0,1,0;
  1,0,1,0]
  Error: Non Numeric Character
fx >>
```

## Output Analysis

### Binary Test Case 1

```
Command Window
  Choose the input type:
  1. 4x4 Binary Matrix
  2. Image File
  Enter your choice (1 or 2): 1
  Enter a 4x4 binary matrix (0s and 1s):
  [1,0,1,0
  1,0,1,0
  1,0,1,0
  1,0,1,0]
  Original Size: 16 bytes
  Compressed Size: 4 bytes
  Compression Ratio: 75.00%
fx >>
```

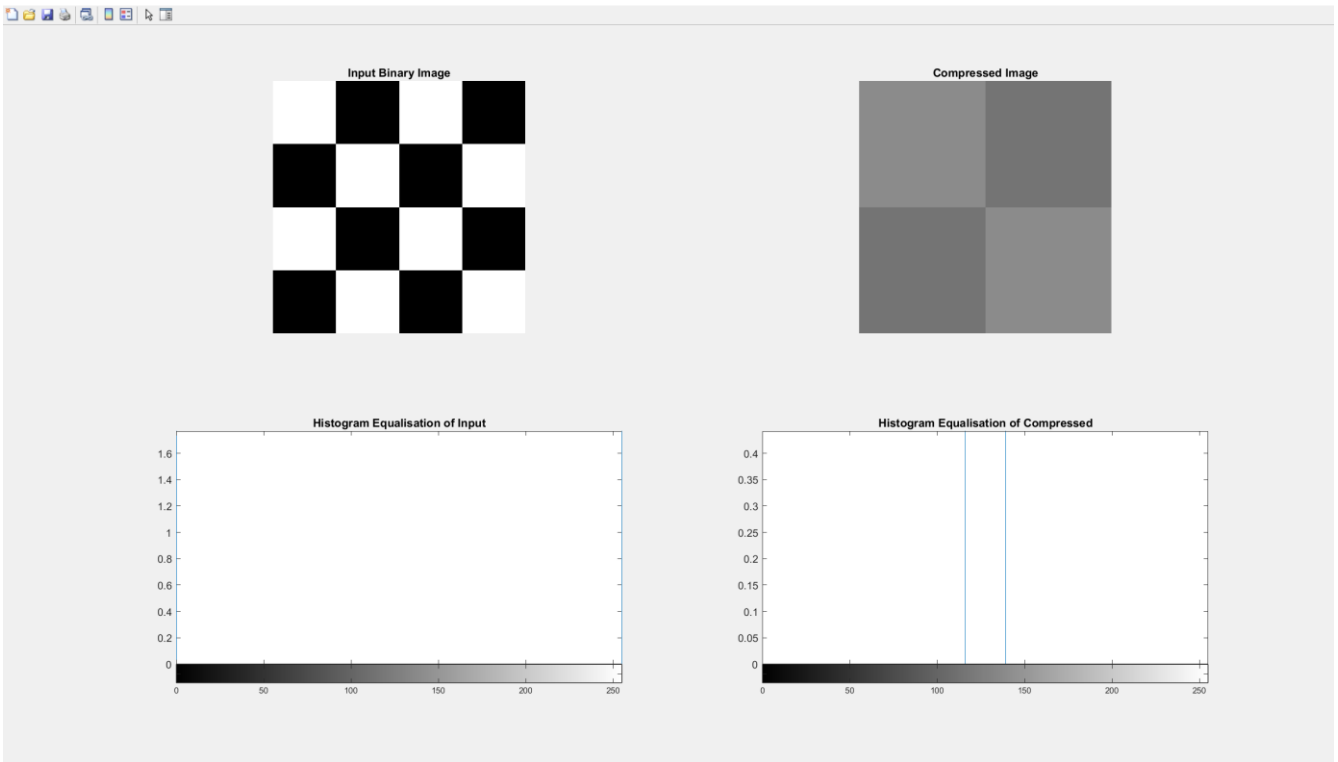
### Visual Representation:



## Binary Test Case 2

```
Command Window
Choose the input type:
1. 4x4 Binary Matrix
2. Image File
Enter your choice (1 or 2): 1
Enter a 4x4 binary matrix (0s and 1s):
[1,0,1,0,
0,1,0,1,
1,0,1,0,
0,1,0,1]
Original Size: 16 bytes
Compressed Size: 4 bytes
Compression Ratio: 75.00%
```

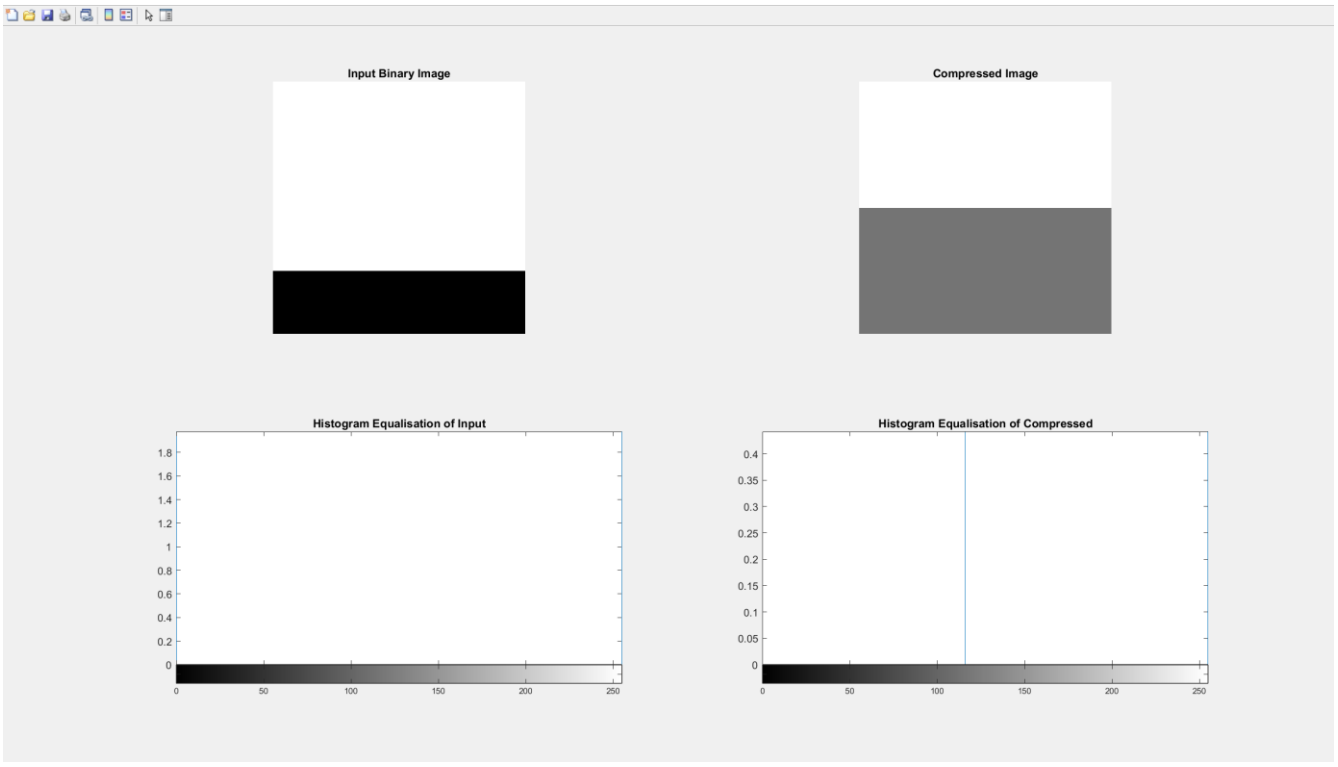
## Visual Representation:



## Binary Test Case 3

```
Command Window
  Choose the input type:
  1. 4x4 Binary Matrix
  2. Image File
  Enter your choice (1 or 2): 1
  Enter a 4x4 binary matrix (0s and 1s):
  [1,1,1,1
  1,1,1,1,
  1,1,1,1,
  0,0,0,0]
  Original Size: 16 bytes
  Compressed Size: 4 bytes
  Compression Ratio: 75.00%
fx >>
```

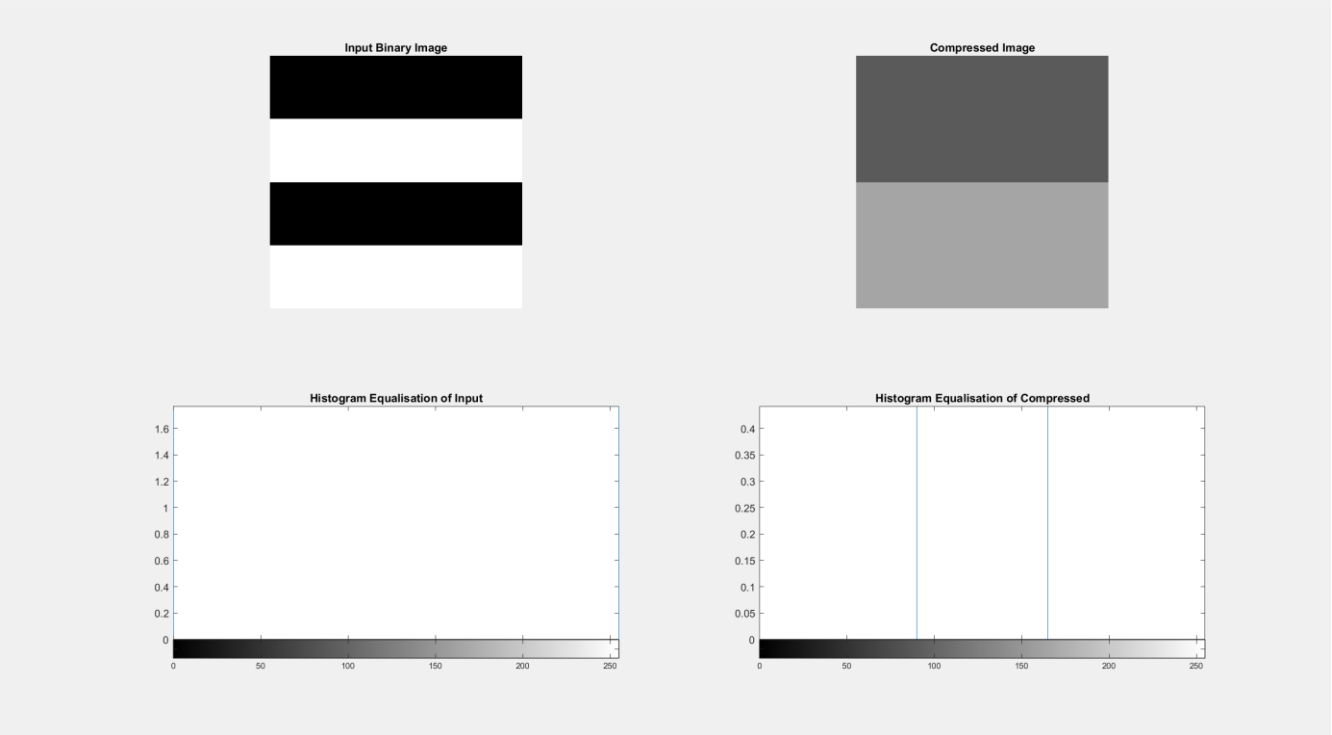
## Visual Representation:



## Binary Test Case 4

```
Command Window
  Choose the input type:
  1. 4x4 Binary Matrix
  2. Image File
  Enter your choice (1 or 2): 1
  Enter a 4x4 binary matrix (0s and 1s):
  [0,0,0,0,
  1,1,1,1,
  0,0,0,0,
  1,1,1,1]
  Original Size: 16 bytes
  Compressed Size: 4 bytes
  Compression Ratio: 75.00%
fx >>
```

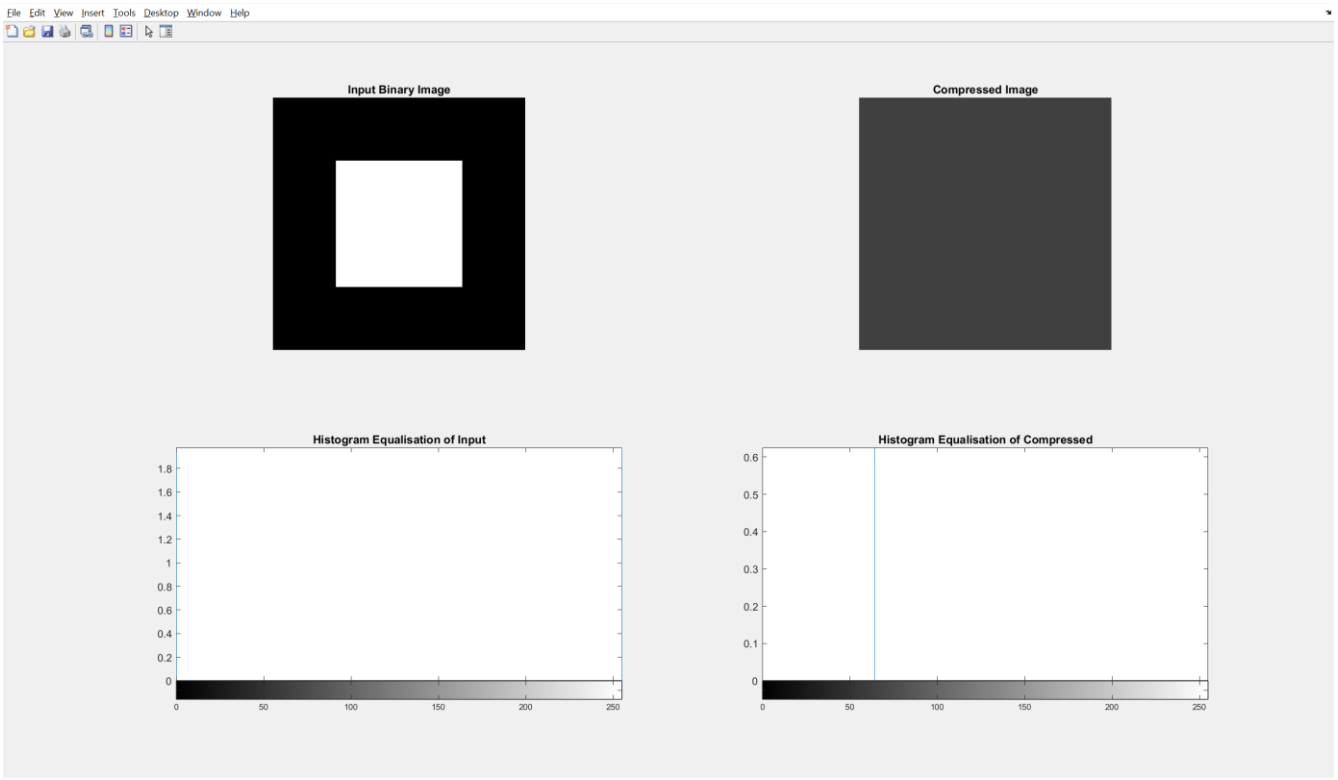
## Visual Representation:



## Binary Test Case 5

```
Command Window
  Choose the input type:
  1. 4x4 Binary Matrix
  2. Image File
  Enter your choice (1 or 2): 1
  Enter a 4x4 binary matrix (0s and 1s):
  [0,0,0,0,
  0,1,1,0,
  0,1,1,0
  0,0,0,0]
  Original Size: 16 bytes
  Compressed Size: 4 bytes
  Compression Ratio: 75.00%
fx >>
```

## Visual Representation:

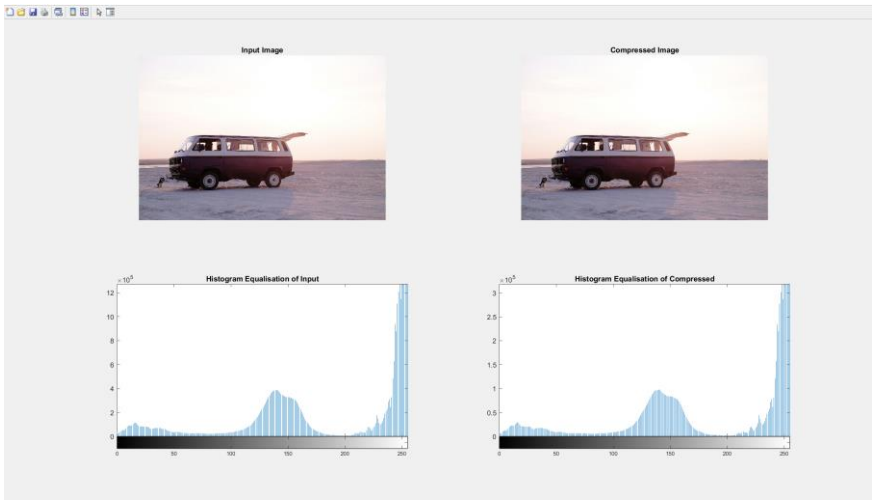


**NOTE**: The compression in the images is not quite visible which is why we can refer to the Size and histogram of the compressed image to see the compression, to see the visible compression zoomed in image of both has been provided

## Image Test Case 1

```
Command Window
  Choose the input type:
  1. 4x4 Binary Matrix
  2. Image File
  Enter your choice (1 or 2): 2
  Original Size: 1270878 bytes
  Compressed Size: 176457 bytes
  Compression Ratio: 86.12%
fx >>
```

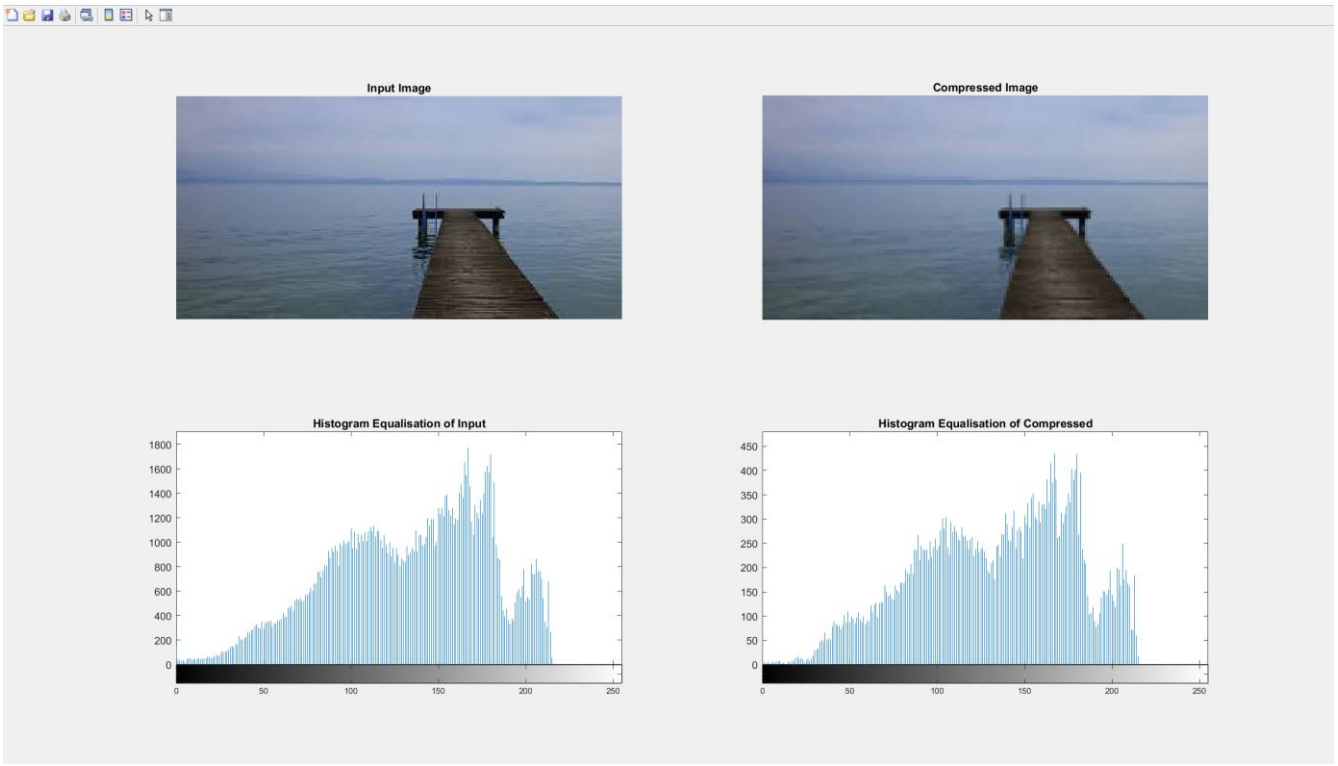
## Visual Representation:

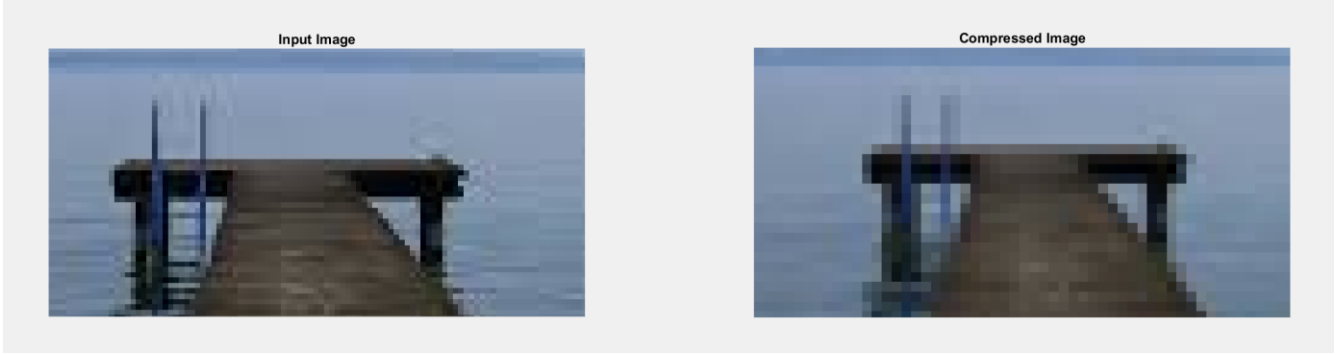



## Image Test Case 2

```
Command Window
  Choose the input type:
  1. 4x4 Binary Matrix
  2. Image File
  Enter your choice (1 or 2): 2
  Original Size: 3800 bytes
  Compressed Size: 1512 bytes
  Compression Ratio: 60.21%
fx >>
```

## Visual Representation:

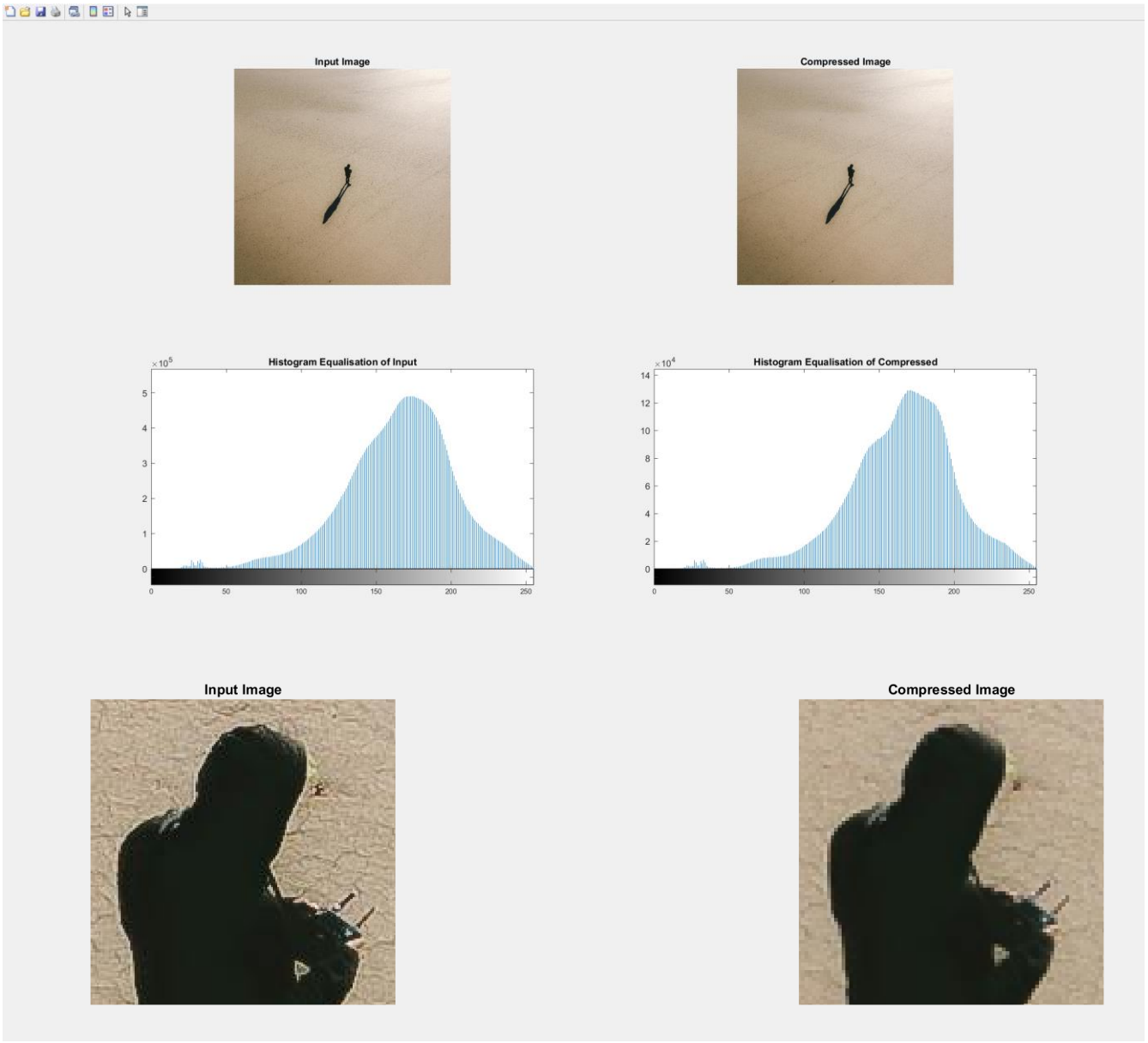


## Image Test Case 3

```
Command Window
  Choose the input type:
  1. 4x4 Binary Matrix
  2. Image File
  Enter your choice (1 or 2): 2
  Original Size: 2071980 bytes
  Compressed Size: 263782 bytes
  Compression Ratio: 87.27%
fx >>
```

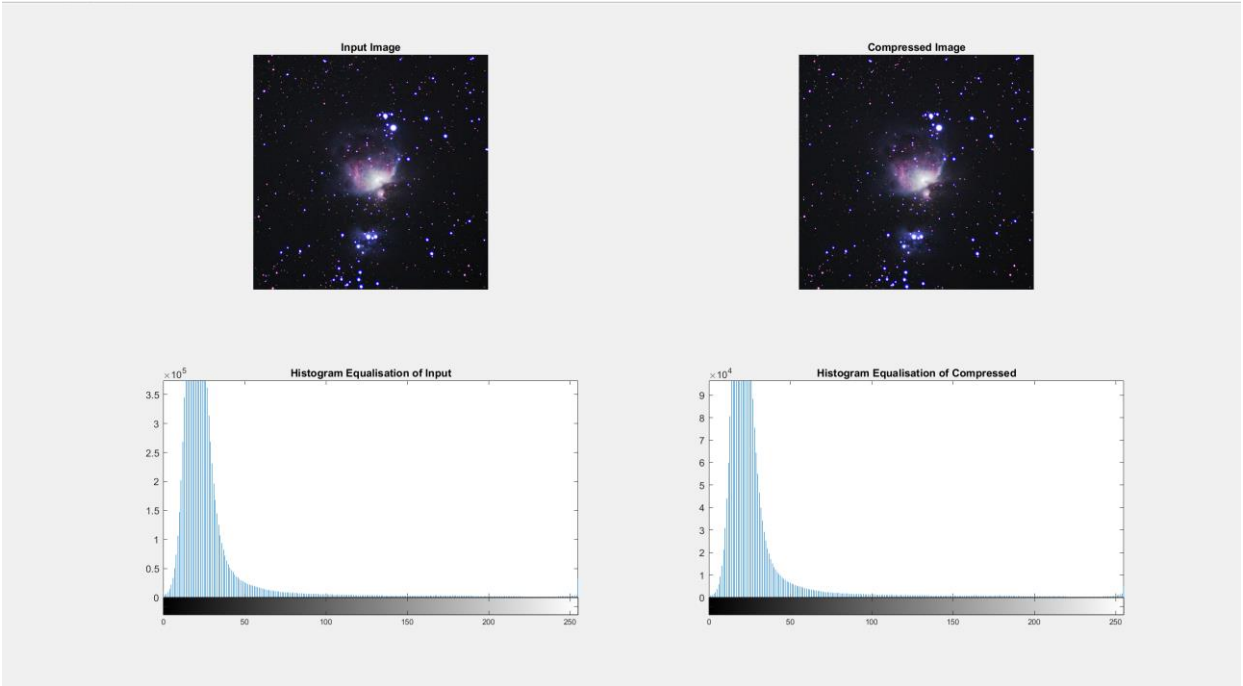
## Visual Representation:

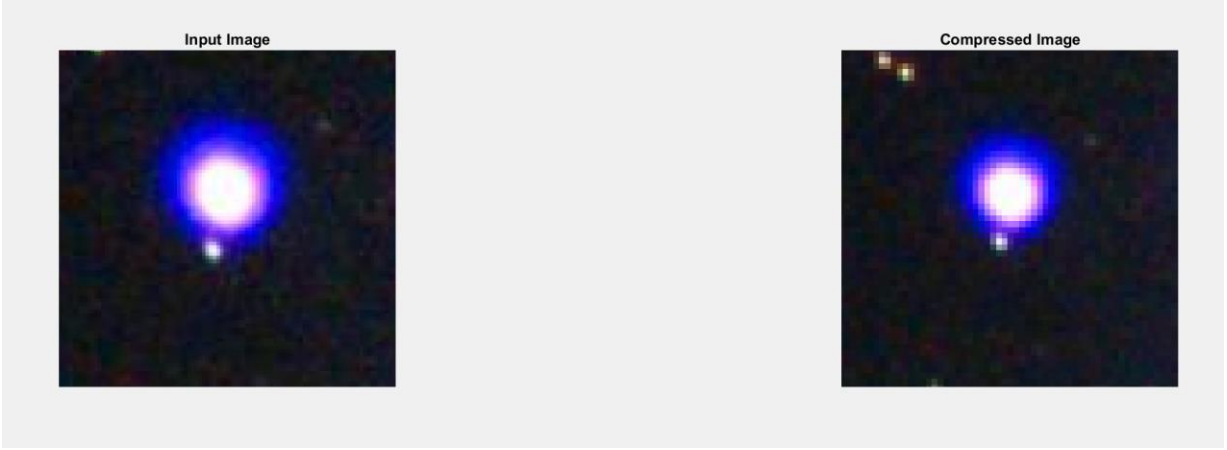


## Image Test Case 4

```
Command Window
  Choose the input type:
  1. 4x4 Binary Matrix
  2. Image File
  Enter your choice (1 or 2): 2
  Original Size: 429282 bytes
  Compressed Size: 59873 bytes
  Compression Ratio: 86.05%
fx >>
```

## Visual Representation:

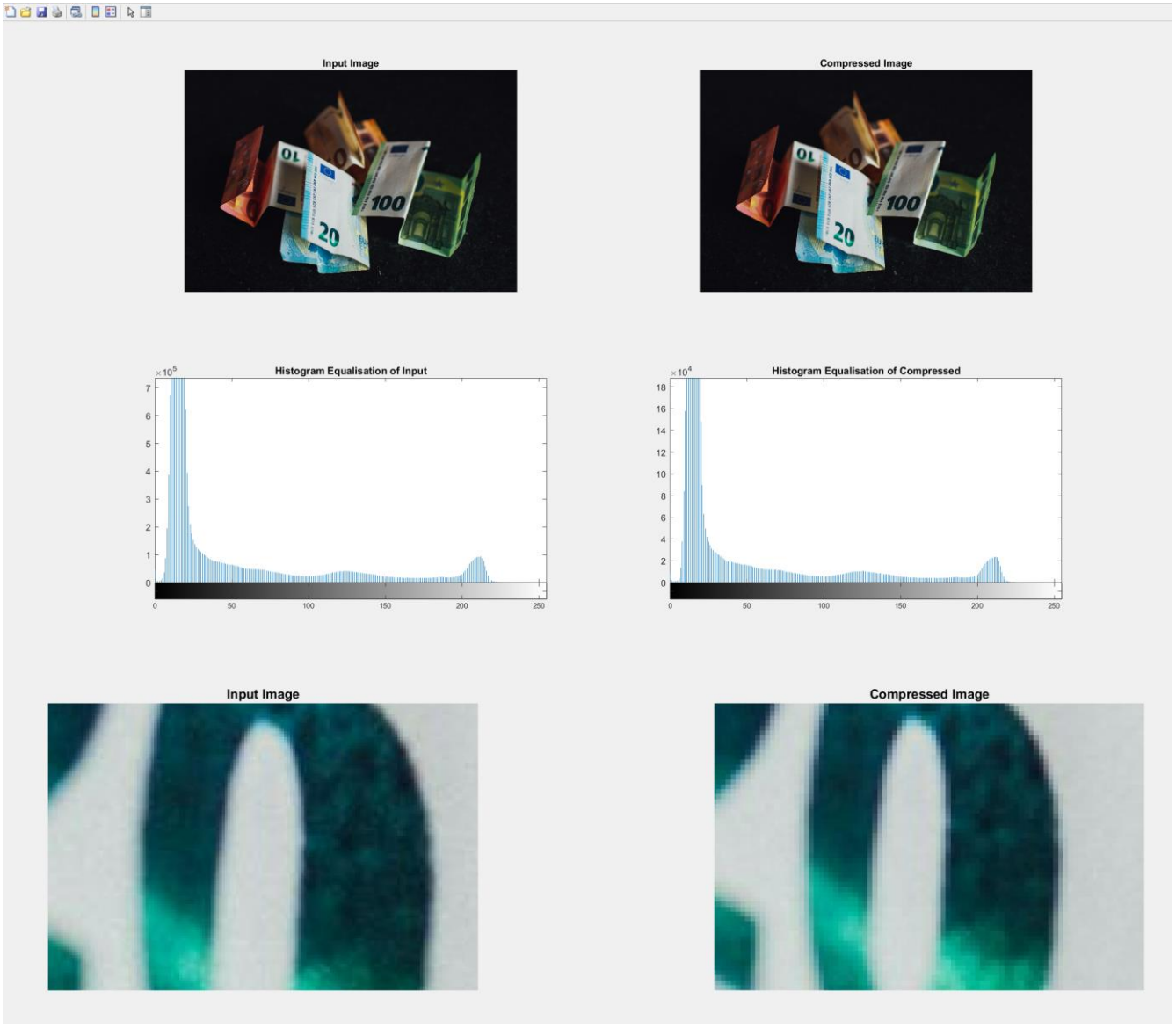


## Image Test Case 5

```
Command Window
  Choose the input type:
  1. 4x4 Binary Matrix
  2. Image File
  Enter your choice (1 or 2): 2
  Original Size: 501689 bytes
  Compressed Size: 87854 bytes
  Compression Ratio: 82.49%
fx >>
```

## Visual Representation:



**Conclusion:**

This code serves as a simplified demonstration of JPEG compression on small-scale images. It allows users to understand the impact of compression on image size and quality, particularly showcasing how resizing and compressing images can significantly reduce file size at the cost of some image fidelity. The compression ratios observed in the processed images range from 60.21% to 86.12%, highlighting the substantial reduction in file size achievable through compression techniques. The results underscore the trade-off between file size and image quality, providing valuable insights into practical image processing and compression strategies. Further development could enhance the code's capabilities to handle larger images and explore more advanced compression methods, aligning with specific educational objectives and practical applications.