

**Batch: A-3 Roll No.: 16010122104**

**Experiment No. 9**

**Grade: AA / AB / BB / BC / CC / CD /DD**

**Signature of the Staff In-charge with date**

**Title:** Image compression by lossless technique (Run Length Coding).

**Objective:** To understand image compression by lossless technique (Run Length Coding).

**Expected Outcome of Experiment:**

|  |  |
| --- | --- |
| **CO** | **Outcome** |
|  |  |
| **CO5** | Design and develop applications based on 1-D and 2-D digital signals. |
|  |
|  |
|  |  |

**Books/ Journals/ Websites referred:**

1. http://www.mathworks.com/support/
2. www.math.mtu.edu/~msgocken/intro/intro.html
3. www.mccormick.northwestern.edu/docs/efirst/matlab.pdf
4. A.Nagoor Kani “Digital Signal Processing”, 2nd Edition, TMH Education.

**Pre Lab/ Prior Concepts:**

Variable length code can be used to remove coding redundancy. One of the way to remove the inter pixel redundancy is run length coding. When inter pixel redundancy is removed by using run length coding the mapper transforms the input data in to usually non visual format. This operation is reversible and may or may not reduce directly the amount of data required to represent the image. Run length coding is the example of a mapping that directly results in data compression in the initial stage of overall source encoding process.

**ALGORITHM:**

Compression:

* Read the binary (monochrome) Image.
* Write the runs of the pixel in a text file.
* Use the text file created in a step 2 and writes each run by

using 8 bits in output file to create compressed image.

* Use the compress image.
* Expand the runs to create decompressed image.

**Implementation**

Code:

% RLE Compression and Decompression for Binary Images

clc; clear; close all;

%% Compression

input\_image = 'picture.bmp';

output\_bin = 'compressed.bin';

output\_txt = 'compressed.txt';

% Read and binarize image

img = imread(input\_image);

if size(img, 3) > 1

img = rgb2gray(img);

end

threshold = graythresh(img); % Automatic thresholding

img\_binary = imbinarize(img, threshold);

% Convert to 1D vector (column-wise)

img\_vector = img\_binary(:);

% Run-Length Encoding

values = [];

counts = [];

current\_val = img\_vector(1);

current\_count = 1;

for i = 2:length(img\_vector)

if img\_vector(i) == current\_val && current\_count < 255

current\_count = current\_count + 1;

else

values = [values; current\_val];

counts = [counts; current\_count];

current\_val = img\_vector(i);

current\_count = 1;

end

end

% Add last run

values = [values; current\_val];

counts = [counts; current\_count];

% Save to binary file (1 byte value, 1 byte count)

fid = fopen(output\_bin, 'wb');

for i = 1:length(values)

fwrite(fid, values(i), 'uint8');

fwrite(fid, counts(i), 'uint8');

end

fclose(fid);

% Save to text file for verification

fid = fopen(output\_txt, 'w');

fprintf(fid, '%d %d\n', [values'; counts']);

fclose(fid);

% Calculate compression metrics

original\_size = numel(img\_binary);

compressed\_size = length(values) \* 2; % 2 bytes per run

CR = original\_size / compressed\_size;

RD = 1 - (1/CR);

fprintf('Compression Ratio (CR): %.2f\n', CR);

fprintf('Redundancy (RD): %.2f\n', RD);

%% Decompression

output\_image = 'decompressed.bmp';

image\_size = size(img\_binary);

% Read binary compressed file

fid = fopen(output\_bin, 'rb');

data = fread(fid, [2, Inf], 'uint8');

fclose(fid);

% Reconstruct image vector

reconstructed = [];

for i = 1:size(data,2)

reconstructed = [reconstructed; repmat(data(1,i), data(2,i), 1)];

end

% Reshape and save

img\_reconstructed = reshape(reconstructed, image\_size);

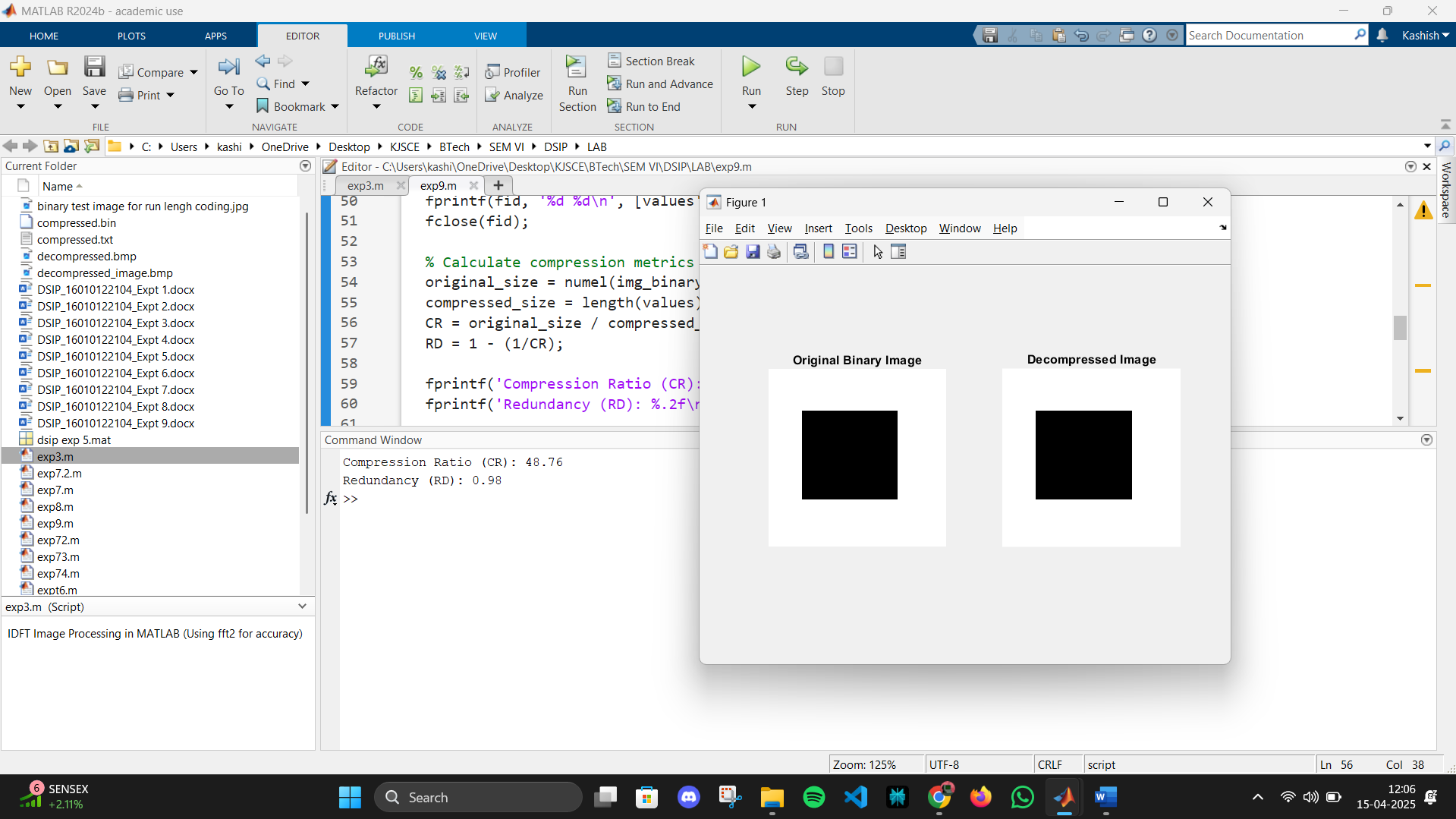
imwrite(img\_reconstructed, output\_image);

%% Verification

subplot(1,2,1), imshow(img\_binary), title('Original Binary Image')

subplot(1,2,2), imshow(img\_reconstructed), title('Decompressed Image')

Output:



**Conclusion:-**

By eliminating coding and inter pixel redundancy lossless compression is achieved. Here by applying run length coding interpixel redundancy is removed and relative data redundancy is calculated as:

****

where is the compression ratio.

**Post Lab Questions**

1. Compare Lossy and lossless compression.

**Ans:**

**Comparison of Lossy and Lossless Compression**

| **Feature** | **Lossy Compression** | **Lossless Compression** |
| --- | --- | --- |
| **Data Integrity** | **Permanently removes some data, reducing integrity. Cannot restore original file.** | **Retains all original data. Can restore file to exact original state.** |
| **Quality** | **Some quality loss occurs, especially after repeated compression or editing.** | **No quality loss; original quality is preserved after decompression.** |
| **File Size Reduction** | **Achieves much higher reduction in file size.** | **Reduces file size, but less than lossy methods.** |
| **Restoration** | **Irreversible; original data cannot be fully recovered.** | **Reversible; original data can be perfectly reconstructed.** |
| **Typical Algorithms** | **Discrete Cosine Transform (DCT), Wavelet Transform, Fractal Compression, etc.** | **Run-Length Encoding, Huffman Coding, Lempel-Ziv-Welch (LZW), Arithmetic.** |
| **Common Uses** | **Multimedia (images, audio, video) where some loss is acceptable (e.g., JPEG, MP3)** | **Text, program files, medical images, and critical data (e.g., PNG, ZIP).** |
| **Editing Impact** | **Quality degrades further with repeated edits and saves.** | **Quality remains unchanged after multiple edits and saves.** |
| **Best For** | **Streaming, web content, where small size and speed are priorities.** | **Archiving, legal, medical, or scientific data where accuracy is critical.** |