KENNESAW STATE UNIVERSITY

CS 7367
MACHINE VISION

ASSIGNMENT 1

<u>**INSTRUCTOR**</u>

Dr. Sanghoon Lee

Your Name: YELLU SIRI
KSU ID: 001165833

1. ABSTRACT

In this project, I worked on exploring basic image processing techniques mainly focusing on image resolution changes and gray level reduction. The main goal was to manually implement these techniques using loops and conditional statements by avoiding built-in MATLAB functions, to understand how these processes work step by step.

The first part focused on resolution changes. I started with a 1024x1024 image and reduced its size to 512x512, 256x256, 128x128, 64x64, and 32x32 by averaging pixel blocks. After this, I took the smaller images and scaled them back up to 1024x1024 by repeating pixel values. This allowed me to observe how much detail was lost during downsampling and how upsampling brought back a pixelated version of the image.

In the second part, I reduced the number of gray levels in an 8-bit grayscale image (with 256 gray levels). By dividing the pixel values into fewer groups, I created images with 128, 64, 32, 16, 8, 4, and 2 gray levels. This process showed how reducing gray levels simplifies the image but introduces visible steps between shades.

Through this project, I have developed understanding of how changes in resolution and gray levels will affect the visual appearance of an image. Project helped me understand the basics of image processing and how these changes affect the way visual data is represented.

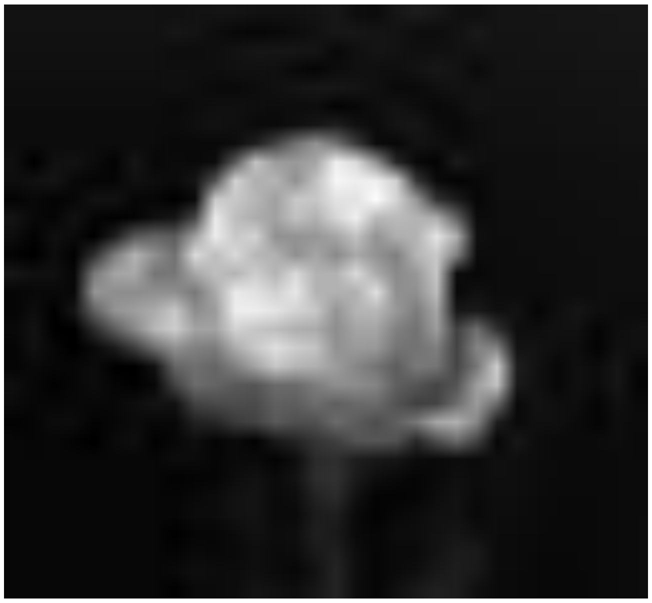
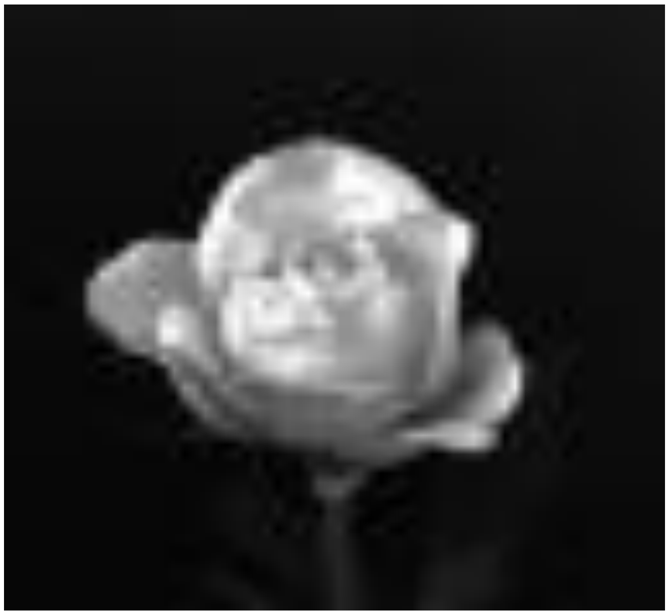
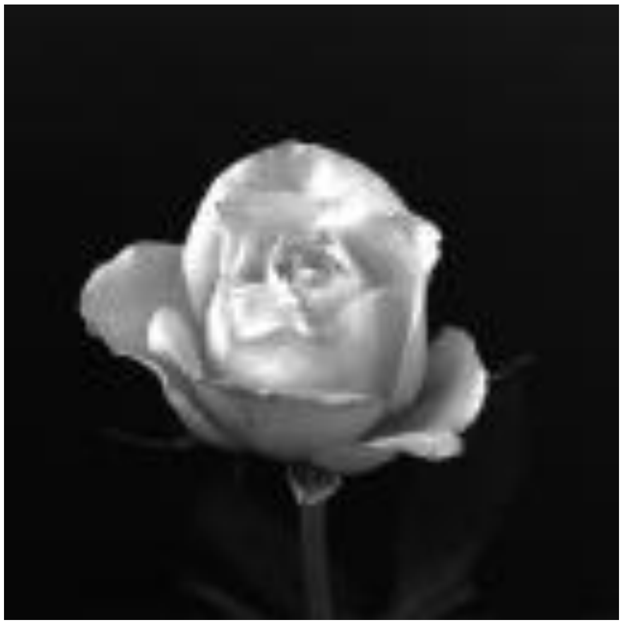
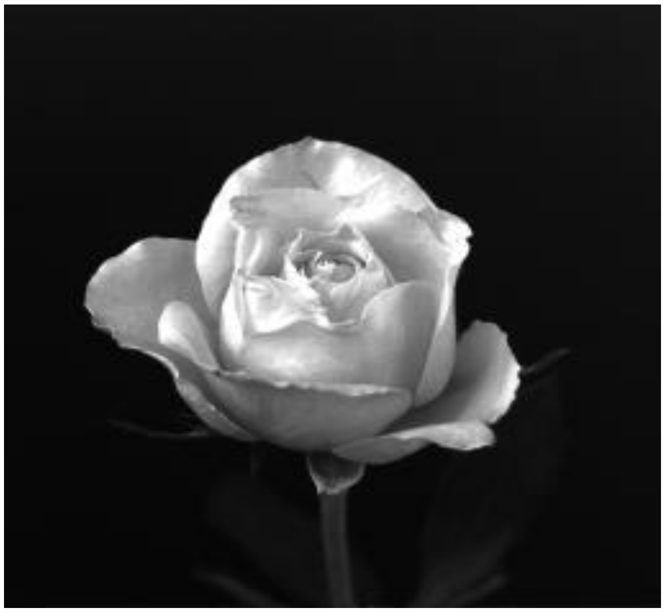
2. TEST RESULTS

2.1 Test Results for Downsampling and Upsampling

INPUT IMAGE:

rose.jpg

Down-Sampled Images:

 (a)	 (b)
 (c)	 (d)

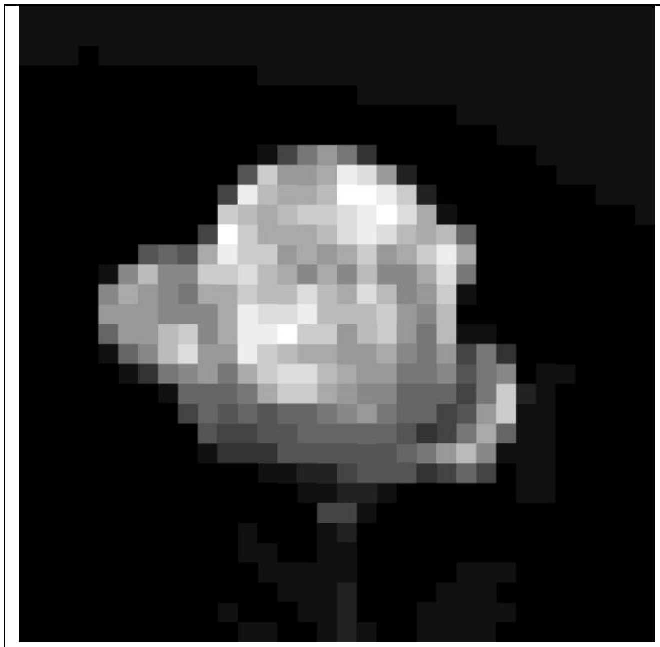
(e)

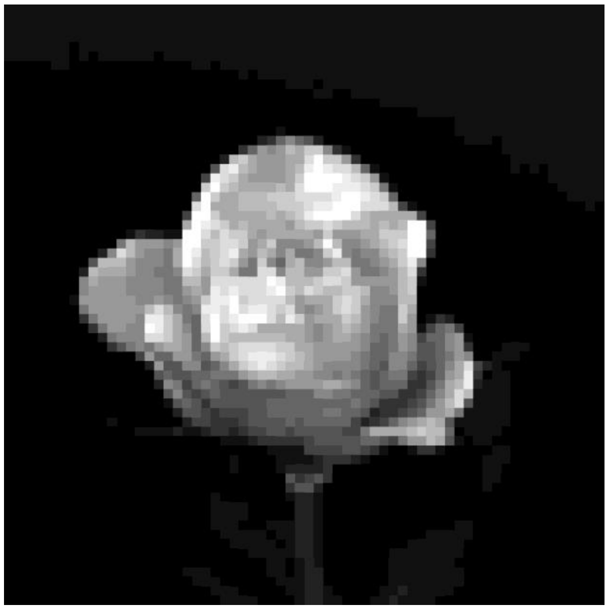
Figure/Table 1: (a-e): The original image rose.jpg, converted to grayscale and resized to 1024x1024 pixels, is **downsampled** to progressively smaller resolutions:

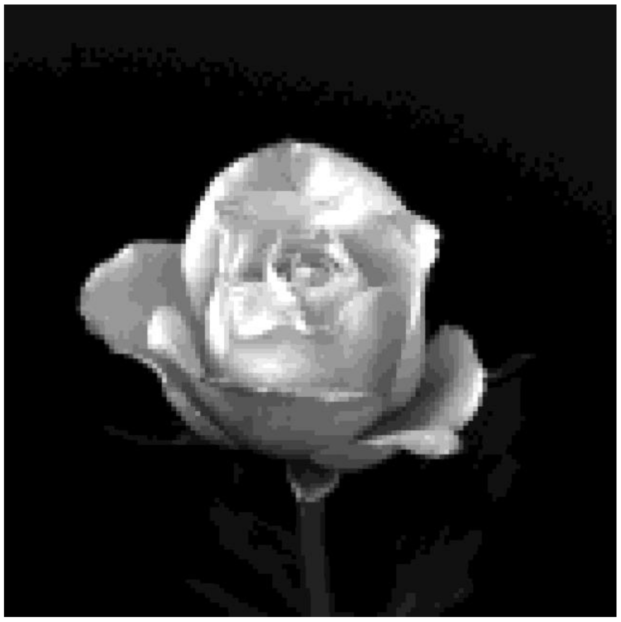
- **(a): 32x32 pixels.**
Parameters: Pixel skip rate = 32, Sampling factor = 32.
- **(b): 64x64 pixels.**
Parameters: Pixel skip rate = 16, Sampling factor = 16.
- **(c): 128x128 pixels.**
Parameters: Pixel skip rate = 8, Sampling factor = 8.
- **(d): 256x256 pixels.**
Parameters: Pixel skip rate = 4, Sampling factor = 4.
- **(e): 512x512 pixels.**
Parameters: Pixel skip rate = 2, Sampling factor = 2.

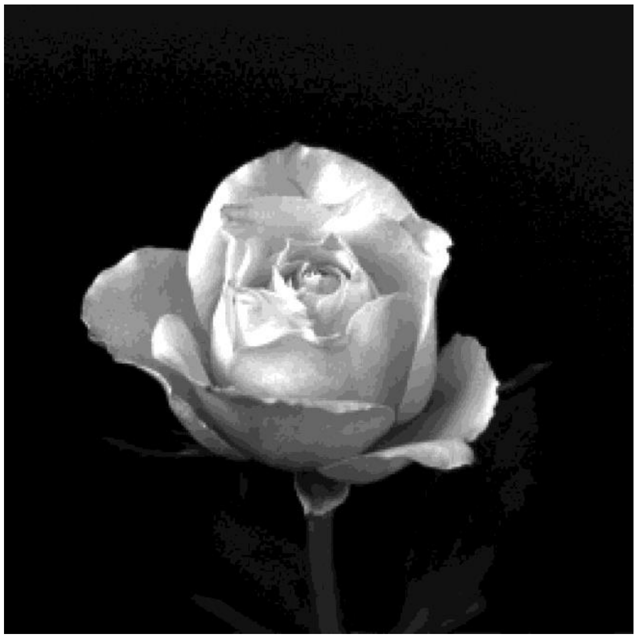
These images demonstrate the gradual loss of detail and sharpness as the resolution decreases, with fewer pixels available to represent the image.

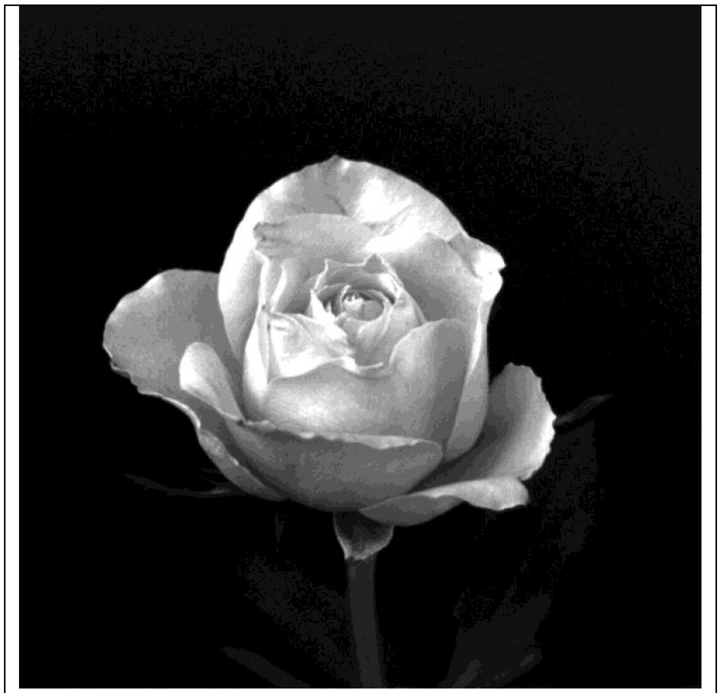
Up-Sampled Images:


(a)


(b)


(c)

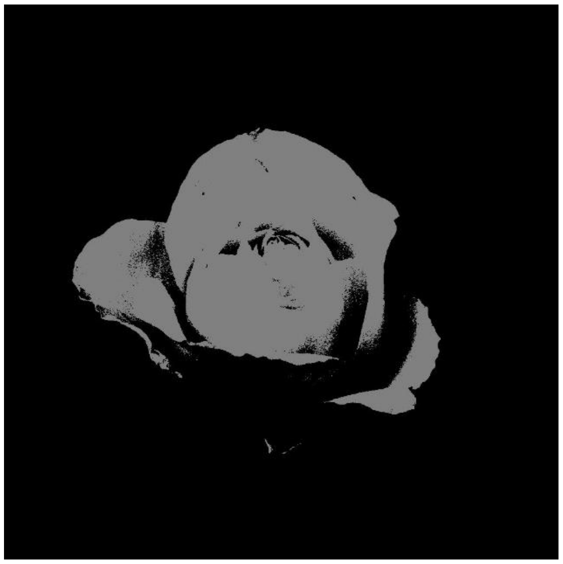
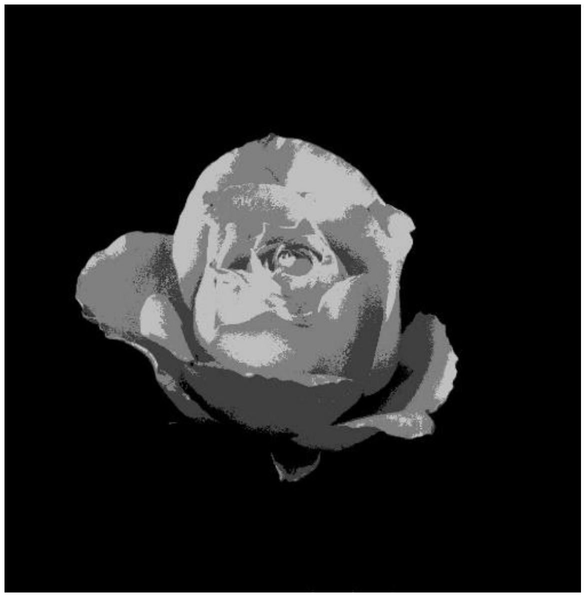
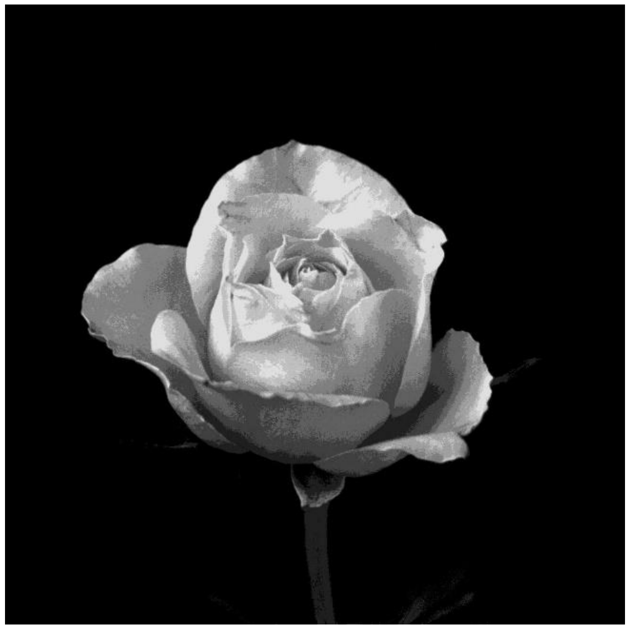
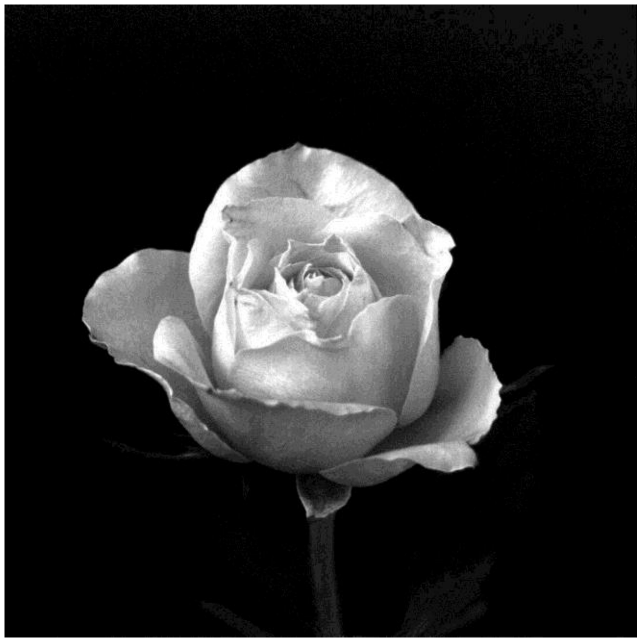

(d)

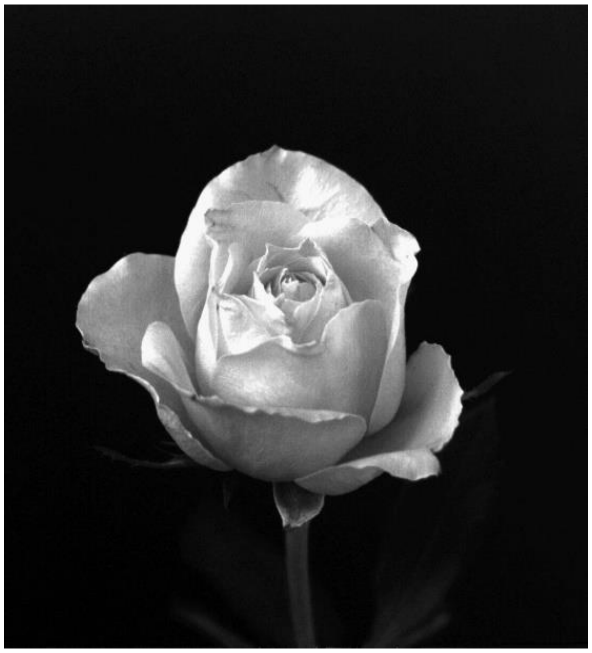
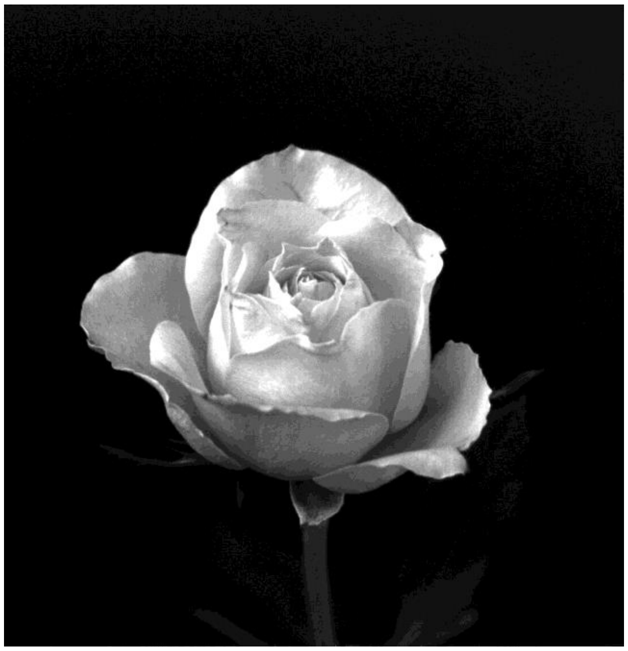

(e)

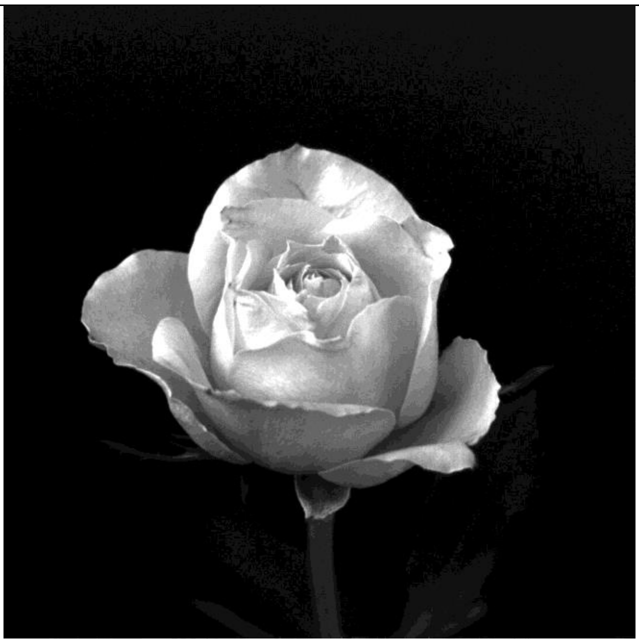
Figure/Table 2: (a-e) The **upsampled images**, where the downsampled images are restored back to 1024x1024 pixels using nearest-neighbor interpolation:

- **(a): Upsampled from 32x32 pixels.**
  Scaling Ratio: 1024/32=321024 / 32 = 321024/32=32.
  Parameters: Downsample size = 32x32, Upsampled back to 1024x1024.
- **(b): Upsampled from 64x64 pixels.**
  Scaling Ratio: 1024/64=161024 / 64 = 161024/64=16.
  Parameters: Downsample size = 64x64, Upsampled back to 1024x1024.
- **(c): Upsampled from 128x128 pixels.**
  Scaling Ratio: 1024/128=81024 / 128 = 81024/128=8.
  Parameters: Downsample size = 128x128, Upsampled back to 1024x1024.
- **(d): Upsampled from 256x256 pixels.**
  Scaling Ratio: 1024/256=41024 / 256 = 41024/256=4.
  Parameters: Downsample size = 256x256, Upsampled back to 1024x1024.
- **(e): Upsampled from 512x512 pixels.**
  Scaling Ratio: 1024/512=21024 / 512 = 21024/512=2.
  Parameters: Downsample size = 512x512, Upsampled back to 1024x1024.

Gray-Scale Quantized images:

 (a)	 (b)
 (c)	 (d)

 **(e)**	 **(f)**



(g)

Figure/Table 3: (a-g): Gray level reduced images of rose.jpg, showing progressively fewer gray levels, starting from 2 and increasing up to 128:

- (a): Reduced to 2 Gray Levels
  Parameters: Step size: 128, Quantization formula = floor(pixel_value/128)*128
- (b): Reduced to 4 Gray Levels
  Parameters: Step size: 64, Quantization formula = floor(pixel_value/64)*64
- (c): Reduced to 8 Gray Levels

  Parameters: Step size: 32, Quantization formula = floor(pixel_value/32)*32
- (d): Reduced to 16 Gray Levels
  Parameters: Step size: 16, Quantization formula = floor(pixel_value/16)*16
- (e): Reduced to 32 Gray Levels
  Parameters: Step size: 8, Quantization formula = floor(pixel_value/8)*8
- (f): Reduced to 64 Gray Levels
  Parameters: Step size: 4, Quantization formula = floor(pixel_value/4)*4
- (g): Reduced to 128 Gray Levels
  Parameters: Step size: 2, Quantization formula = floor(pixel_value/2)*2

3. DISCUSSION

Working on this project has helped me understand the basics of image processing more clearly and instead of using MATLAB's built-in functions I have written the code using loops and if statements, which allowed me to see how these techniques work step by step. Also, Resizing the images helped me understand better on how important resolution is because I have noticed details being lost during down sampling and how up sampling can make an image appear blurry.

Similarly, reducing the number of gray levels helped me understand better on how simplifying intensity values can drastically change the overall appearance of an image. These hands-on tasks made it easier to grasp how resolution and gray levels affect how an image looks.

One of the biggest challenges I have faced was implementing these techniques manually while ensuring accuracy. Writing the code for resizing and gray-level quantization helped me better understand how concepts like interpolation and intensity mapping work in practice.

In Future, I would want to explore other interpolation techniques, such as bilinear or bicubic interpolation, to see how they compare with nearest neighbor in terms of image quality, and it would be interesting to apply these methods to color images to understand how these changes in resolution and intensity can affect their appearance. Overall, this project has provided me with a strong foundation for understanding image processing and gave a better view for the impact of these techniques on visual data. It also enlightened the importance of balancing simplicity and detail in image manipulation tasks.

4. CODES

4.1 *resolution_changes.m*

```
% resolution_changes.m
% Name: YELLU SIRI
% KSU ID: 001165833

clear all;
close all;
clc;

% Read and preprocess image
img = imread('rose.jpg');
if size(img,3) == 3
    img = manual_rgb2gray(img);
end

% Ensure image is 1024x1024
if size(img,1) ~= 1024 || size(img,2) ~= 1024
    img = manual_resize(img, 1024);
end

% Resolution changes
sizes = [512 256 128 64 32];
figure('Name', 'Resolution Changes');

for i = 1:length(sizes)
    % Downsample
    down = downsample_image(img, sizes(i));
    subplot(2,5,i);
    imshow(down);
    title(['Down to ' num2str(sizes(i)) 'x' num2str(sizes(i))]);
    imwrite(down, sprintf('down_%dx%d.jpg', sizes(i), sizes(i)));

    % Upsample back to 1024x1024
    up = upsample_image(down, 1024);
    subplot(2,5,i+5);
    imshow(up);
    title(['Up from ' num2str(sizes(i))]);
    imwrite(up, sprintf('up_from_%d.jpg', sizes(i)));
end

function gray = manual_rgb2gray(rgb)
    r = double(rgb(:,:,1));
    g = double(rgb(:,:,2));
    b = double(rgb(:,:,3));
    gray = uint8(0.299 * r + 0.587 * g + 0.114 * b);
end

function resized = manual_resize(img, new_size)
    [rows, cols] = size(img);
    resized = zeros(new_size, new_size, 'uint8');

    row_ratio = rows / new_size;
    col_ratio = cols / new_size;
    for i = 1:new_size
        for j = 1:new_size
            row = min(rows, round(i * row_ratio));
            col = min(cols, round(j * col_ratio));
            if row == 0; row = 1; end
            if col == 0; col = 1; end
            resized(i,j) = img(row,col);
        end
    end
end

function down = downsample_image(img, target_size)
    [rows, cols] = size(img);
    down = zeros(target_size, target_size, 'uint8');

    window_rows = floor(rows/target_size);
    window_cols = floor(cols/target_size);

    for i = 1:target_size
        for j = 1:target_size
            row_start = (i-1)*window_rows + 1;
            row_end = min(rows, i*window_rows);
            col_start = (j-1)*window_cols + 1;
            col_end = min(cols, j*window_cols);

            window = img(row_start:row_end, col_start:col_end);
            down(i,j) = uint8(mean(window(:)));
        end
    end
end

function up = upsample_image(img, target_size)
    [rows, cols] = size(img);
    up = zeros(target_size, target_size, 'uint8');

    row_ratio = rows / target_size;
    col_ratio = cols / target_size;

    for i = 1:target_size
        for j = 1:target_size
            row = min(rows, max(1, ceil(i * row_ratio)));
            col = min(cols, max(1, ceil(j * col_ratio)));
            up(i,j) = img(row,col);
        end
    end
end
```

4.2 *gray_level_reduction.m*

```
% gray_level_reduction.m
% Name: YELLU SIRI
% KSU ID: 001165833
```

```
clear all;
close all;
clc;

% Read and preprocess image
img = imread('rose.jpg');
if size(img,3) == 3
    img = manual_rgb2gray(img);
end

% Ensure image is 1024x1024
if size(img,1) ~= 1024 || size(img,2) ~= 1024
    img = manual_resize(img, 1024);
end

% Gray level reduction
levels = [128 64 32 16 8 4 2];
figure('Name', 'Gray Level Reduction');

% Display original
subplot(2,4,1);
imshow(img);
title('Original (256 levels)');

% Reduce and display gray levels
for i = 1:length(levels)
    reduced = reduce_gray_levels(img, levels(i));
    subplot(2,4,i+1);
    imshow(reduced);
    title([num2str(levels(i)) ' levels']);
    imwrite(reduced, sprintf('gray_%d.jpg', levels(i)));
end

function gray = manual_rgb2gray(rgb)
    r = double(rgb(:,:,1));
    g = double(rgb(:,:,2));
    b = double(rgb(:,:,3));
    gray = uint8(0.299 * r + 0.587 * g + 0.114 * b);
end

function resized = manual_resize(img, new_size)
    [rows, cols] = size(img);
    resized = zeros(new_size, new_size, 'uint8');

    row_ratio = rows / new_size;
    col_ratio = cols / new_size;

    for i = 1:new_size
        for j = 1:new_size
            row = min(rows, round(i * row_ratio));
            col = min(cols, round(j * col_ratio));
            if row == 0; row = 1; end
            if col == 0; col = 1; end
            resized(i,j) = img(row,col);
        end
    end
end
```

```
function reduced = reduce_gray_levels(img, num_levels)
    max_val = 255;
    step = max_val / (num_levels - 1);
    reduced = uint8(round(double(img) / step) * step);
end
```