KENNESAW STATE
UNIVERSITY

CS 7367
MACHINE VISION

ASSIGNMENT 1
IMAGE RESOLUTION

INSTRUCTOR

Dr. Sanghoon Lee

Your Name: Adam Kangiser
KSU ID: 000681701

1. ABSTRACT

This document explores digital image processing methodologies using MATLAB, with a primary focus on resizing and gray level quantization. Initially, techniques for down-sampling and up-sampling a given "rose.jpg" image to different resolutions were discussed. Traditional MATLAB functions were employed, followed by a custom implementation utilizing the nearest neighbor interpolation approach. Despite its simplicity, the latter method presented challenges, particularly in handling boundary conditions, which were iteratively resolved.

Subsequently, the discourse shifted to grayscale quantization of 8-bit images. From an original distinction of 256 gray levels, techniques were detailed to quantize the image to varying gray levels, including 128, 64, 32, 16, 8, 4, and 2. This process essentially reduced the number of discrete grayscale values in the image, leading to increased posterization. The results demonstrated the significance of gray levels in retaining image details, and how their reduction impacts visual clarity.

This report not only showcases MATLAB's prowess in image processing but also delves into the intricacies of custom algorithmic implementations and their respective challenges.

2. TEST RESULTS

Down-Sampled Images:

- Figure 1b: Image resized to 512x512 pixels using the nearest neighbor interpolation approach.

- Figure 1c: Image resized to 256x256 pixels using the nearest neighbor interpolation approach.

- Figure 1d: Image resized to 128x128 pixels using the nearest neighbor interpolation approach.

- Figure 2a: Image resized to 64x64 pixels using the nearest neighbor interpolation approach.

- Figure 2b: Image resized to 32x32 pixels using the nearest neighbor interpolation approach.

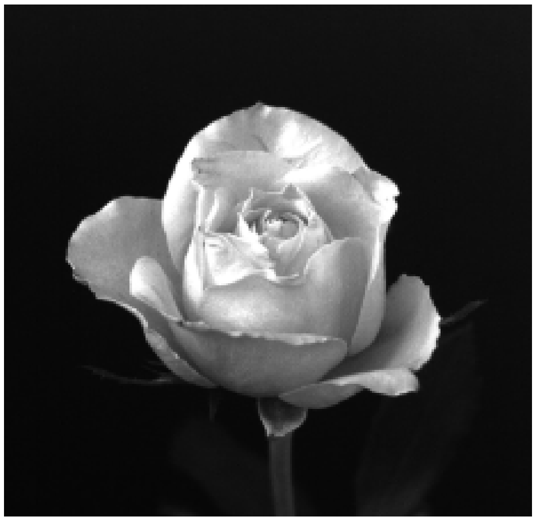
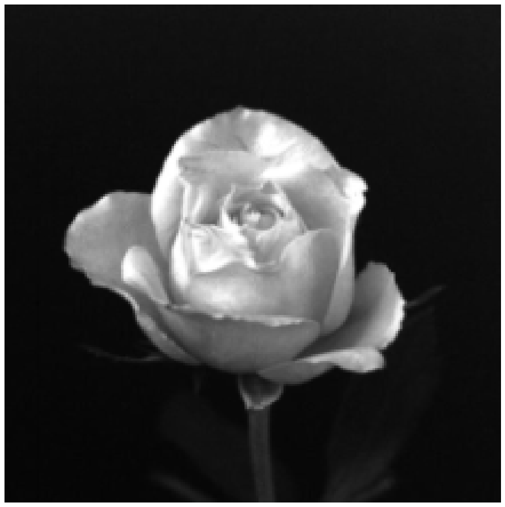
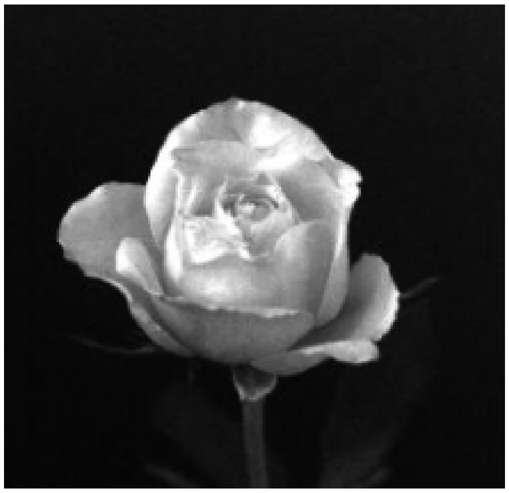
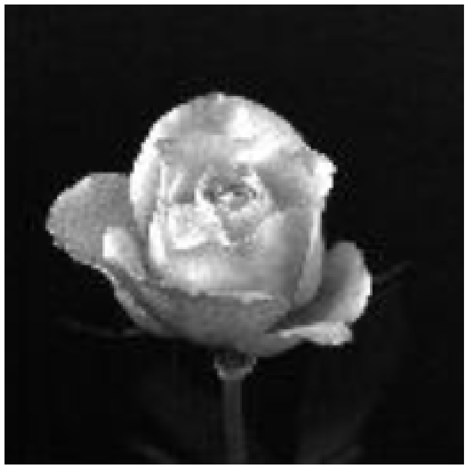
 (a)	 (b)
 (c)	 (d)

Figure 1: (a-b) Original grayscale image 1024x1024 (rose.jpg), downsampled (rose_512x512.jpg), (c-d) downsampled further (rose_256x256.jpg)(rose_128x128.jpg).

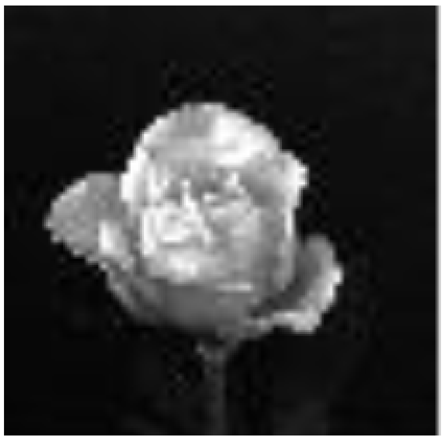
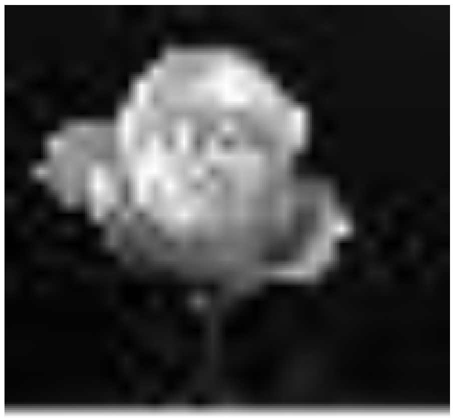
 **(a)**	 **(b)**

Figure 2: (a-b) Downscale image (rose_64x64.jpg), downsampled (rose_32x32.jpg)

Images that have been upsampled from resolution versions;

Figure 3b; This image has been increased in size from 32x32 pixels to 1024x1024 pixels using the neighbor interpolation method.

Figure 3c; This image has been enlarged from 64x64 pixels to 1024x1024 pixels using the neighbor interpolation approach.

Figure 3d; This image has been upscaled from its size of 128x128 pixels to 1024x1024 pixels using the neighbor interpolation technique.

Figure 4a; This image has undergone an upsampling process increasing its dimensions from 256x256 pixels to a size of 1024x1024 pixels through the use of neighbor interpolation.

Figure 4b; An image that was initially sized, at 512x512 pixels has been expanded to a dimension of 1024x1024 pixels utilizing the neighbor interpolation method.

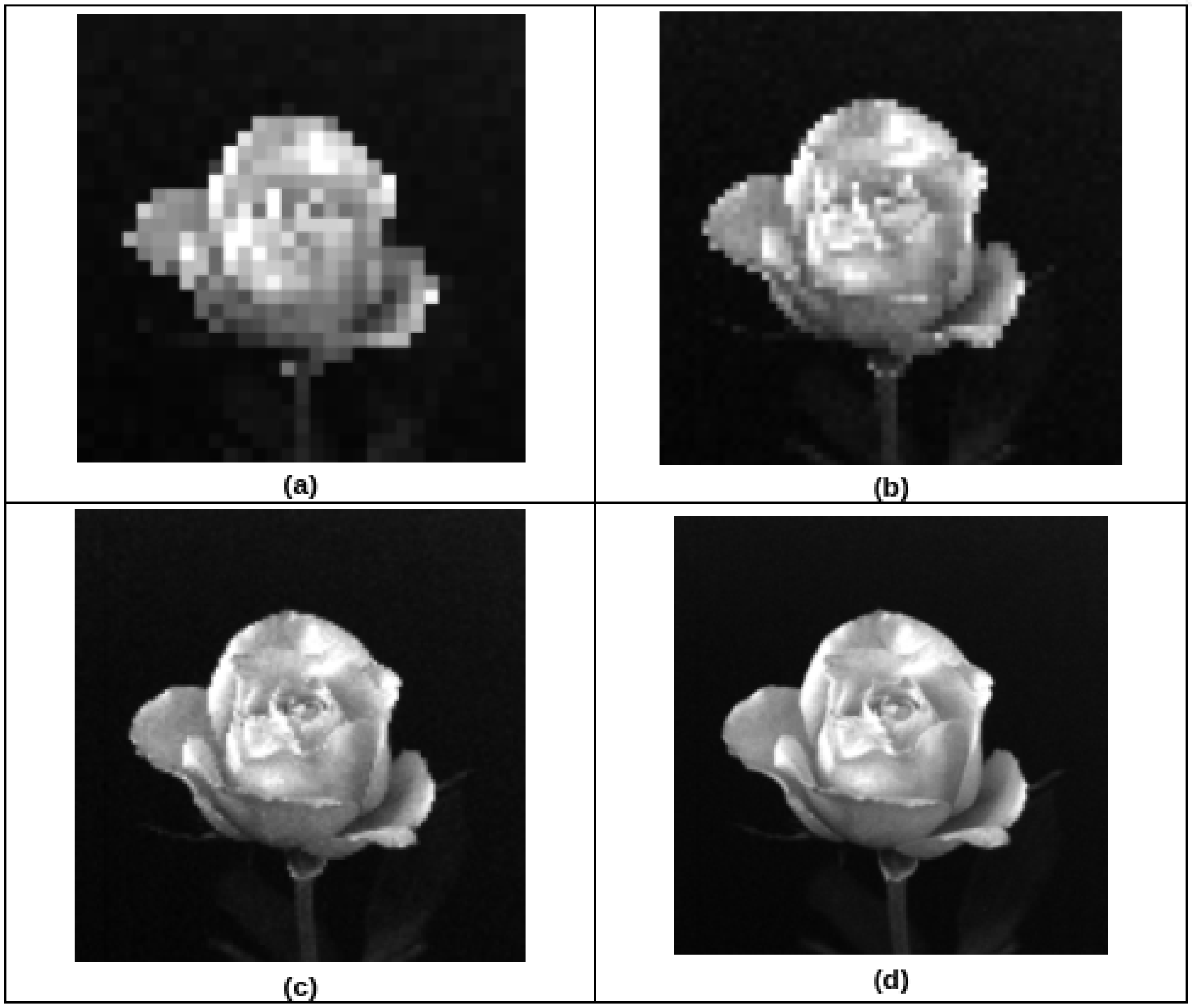


Figure 3: (a-b) Upscaled image 32x32 and 64x64, (c-d) upscaled 128x128 and 256x256

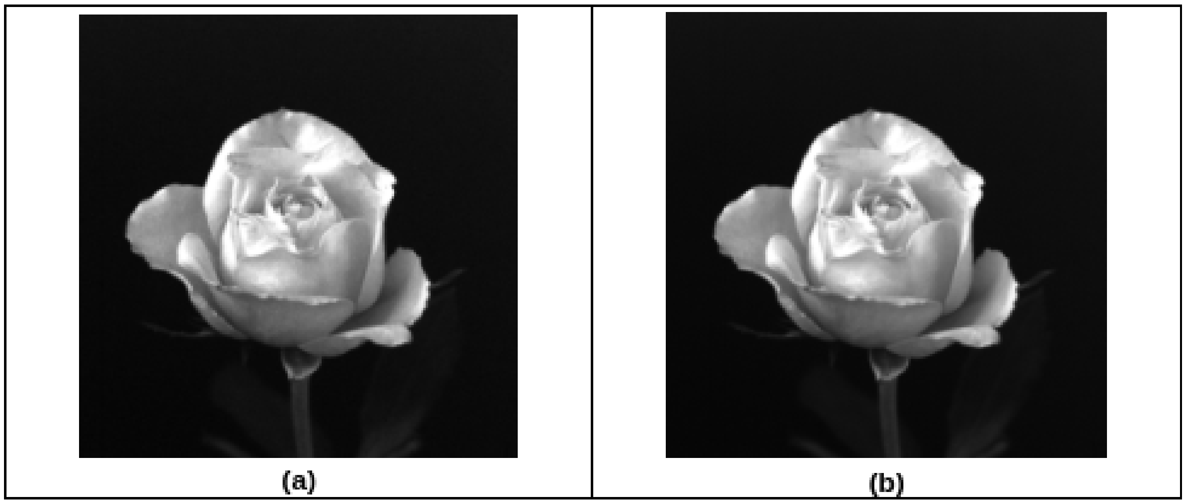


Figure 4: (a-b) upscaled image 512x512 and 1024x1024

Quantized Images;

In Figure 5b the image has been reduced to 128 levels. This means that the original grayscale intensities have been mapped to 128 values resulting in a decreased range.

Moving on to Figure 5c the image is quantized even further to only 64 gray levels. This additional mapping of intensities leads to posterization.

Figure 5d shows the image quantized to 32 gray levels.

In Figure 6a we have an image quantized down to 16 gray levels.

Taking it a step further Figure 6b presents an image with 8 levels.

The severity increases in Figure 6c as the image is now quantized to 4 levels. At this stage a significant amount of detail in the image becomes compromised.

Finally Figure 6d demonstrates quantization where the image is reduced to two gray levels. Essentially this converts the image into a representation.

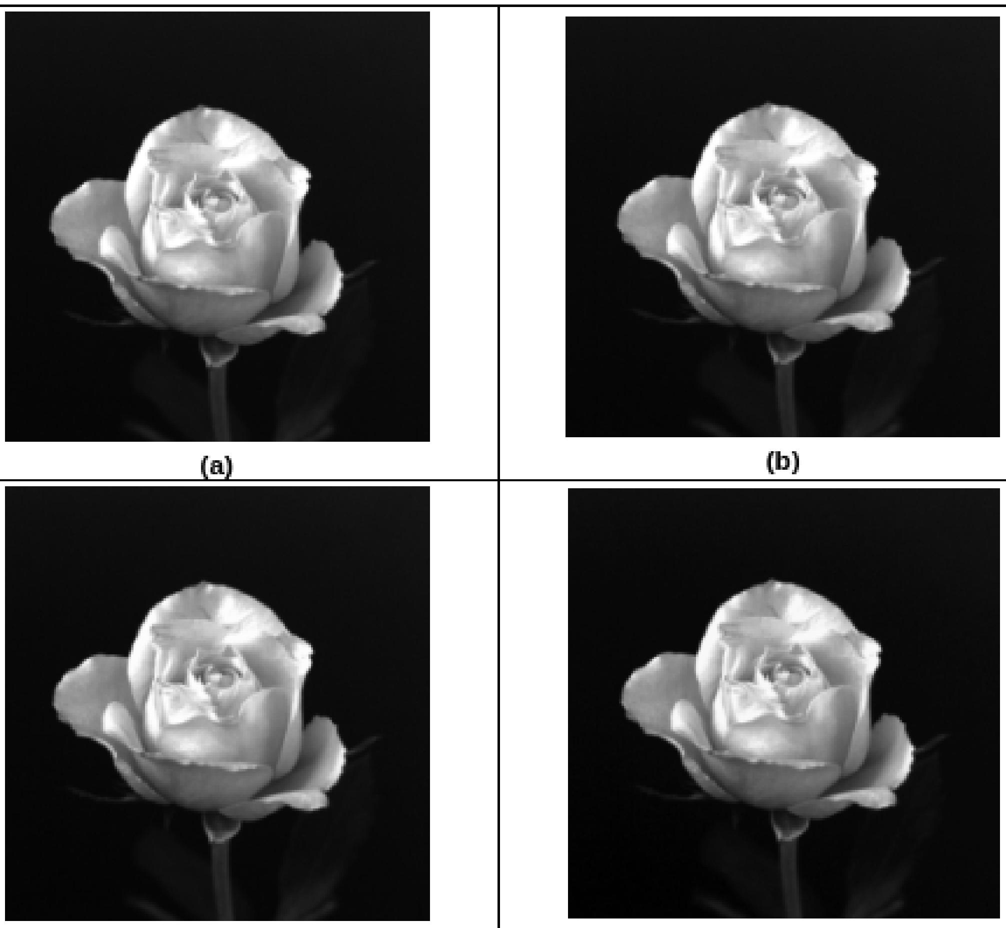


Figure 5: (a-b) Grey level 256 and 128, (c-d) grey level 64 and 32

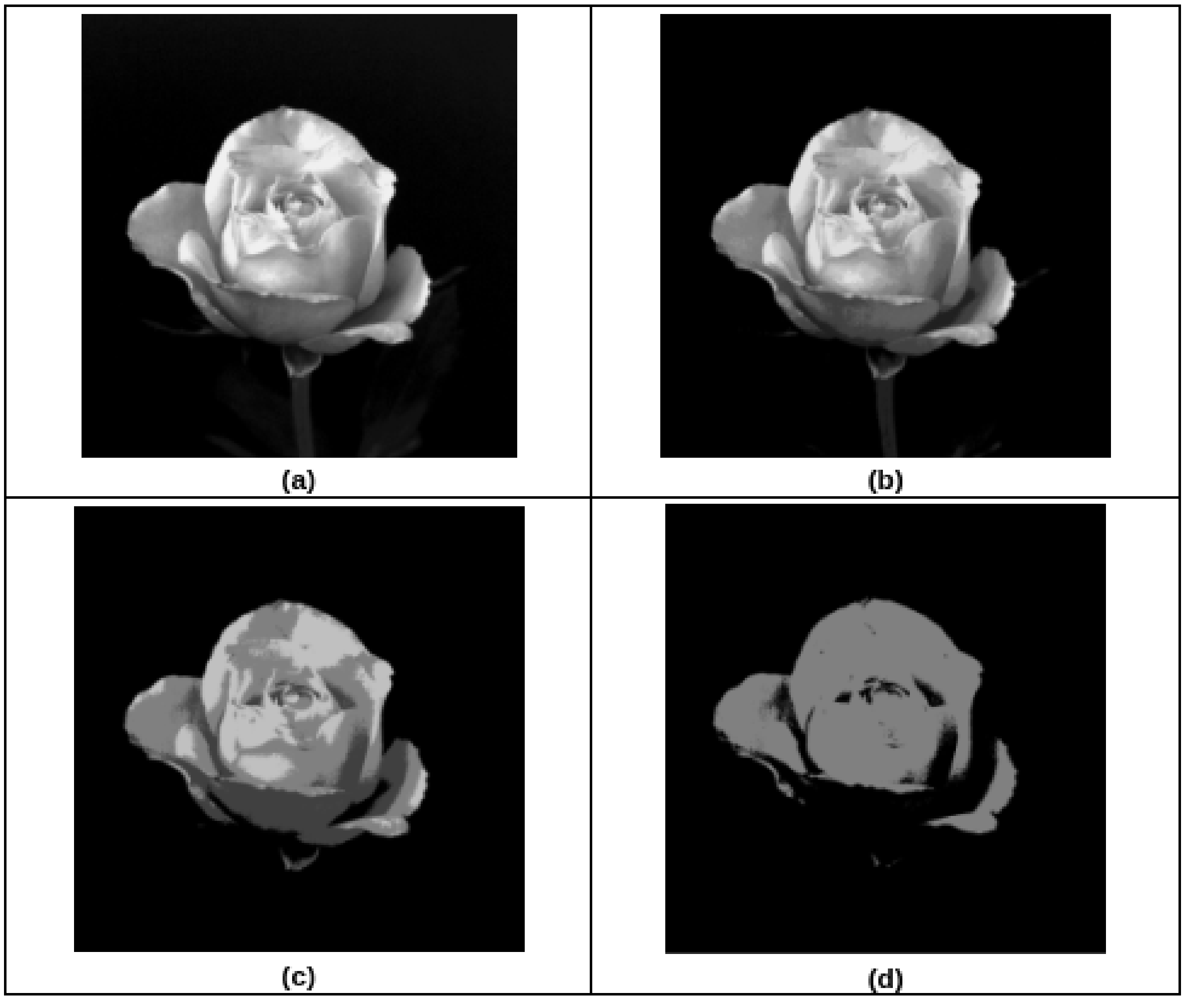


Figure 6: (a-b) Grey level 16 and 8, (c-d) gray level 4 and 2

The experiment of reducing and increasing image resolution showed how it affects the clarity of the image. As we anticipated reducing the resolution resulted in a loss of detail that couldn't be restored when we increased it again. However when we quantized the levels it became clear how crucial they are, in preserving image details. As we decreased the number of levels the visual quality of the image worsened, emphasizing the variations and smooth transitions that higher gray levels contribute to an image.

3. Discussion

Throughout this project I was amazed by the effects of adjusting image resolution and gray level quantization. As I worked with MATLAB to navigate through the sampling and quantization processes I became keenly aware of the trade offs involved in image size, clarity and computational efficiency.

When I down sampled the images it was no surprise that some details were lost. However witnessing this loss firsthand during sampling gave me an appreciation for the intricate nuances that high resolution images capture. Implementing a custom algorithm also reminded me how crucial it is to handle boundaries with care. It served as a reminder that even small oversights in programming can lead to errors.

Gray level quantization provided another eye opening experience. As I reduced the number of levels it became evident how much depth and detail are added by shades in an image. Watching the posterization with gray levels was like witnessing the richness held within 8 bit grayscale images.

Reflecting on my journey although MATLAB proved to be a tool engaging in hands-on exploration truly enhanced my understanding of these concepts. Given time I would have loved to delve into interpolation methods for image resizing such as bilinear or bicubic interpolation.

To gain an understanding of their strengths and weaknesses it would be valuable to compare these methods with the nearest neighbor approach I previously employed. Moreover I am particularly fascinated by the possibility of exploring color quantization or applying these techniques to a collection of images such as videos, in order to uncover any complexities and revelations they might unveil.

4. CODES

3.1 Code for Upscale and Downscale

```
% Name: Adam Kangiser
% KSU Number: 000681701
% Assignment 1
close all;
clear;
clc;
% Reading the image
img = imread('rose.jpg');
% Define the desired sizes for down-sampling
sizes = [512, 256, 128, 64, 32];
for s = sizes
% Down-sampling the image using custom function
resized_img = custom_resize(img, s, s);
% Saving the down-sampled image
filename = sprintf('rose_%dx%d.jpg', s, s);
imwrite(resized_img, filename);
% Up-sampling the down-sampled image back to 1024x1024 using custom function
upsampled_img = custom_resize(resized_img, 1024, 1024);
% Saving the up-sampled image
upsampled_filename = sprintf('rose_%dx%d_upsampled.jpg', s, s);
imwrite(upsampled_img, upsampled_filename);
end
```

****

3.2 Code for Gray Level

```
% Name: Adam Kangiser
% KSU Number: 000681701
% Assignment 1
close all;
clear;
clc;
% Read the grayscale image
img = imread('rose.jpg');
if size(img, 3) == 3
img = rgb2gray(img); % Convert to grayscale if it's a color image
end
% Define the desired gray levels
gray_levels = [128, 64, 32, 16, 8, 4, 2];
for level = gray_levels
% Quantize the image
quantized_img = uint8(floor(double(img) / 256 * level) * (256/level));
% Save the quantized image
filename = sprintf('rose_gray_%d_levels.jpg', level);
imwrite(quantized_img, filename);
end
```

****

3.3 Function for Custom Resize

```
% Name: Adam Kangiser
% KSU Number: 000681701
% Assignment 1
function resized = custom_resize(image, new_width, new_height)
[original_height, original_width, channels] = size(image);
width_scale = original_width / new_width;
height_scale = original_height / new_height;
resized = zeros(new_height, new_width, channels, class(image));
for i = 1:new_height
for j = 1:new_width
x = max(1, min(round(j * width_scale), original_width));
y = max(1, min(round(i * height_scale), original_height));
resized(i, j, :) = image(y, x, :);
end
end
end
```