KENNESAW STATE
UNIVERSITY

CS 4732
MACHINE VISION

ASSIGNMENT 1

IMAGE RESOLUTION

INSTRUCTOR

Dr. Sanghoon Lee

Your Name: Max Haviv
KSU ID: 001029496

1. ABSTRACT

In this project I learn to down and up sample images using just for loops. I then also learn about how to adjust intensity levels in gray scale images using just for loops as well. In this assignment I use python, and matplotlib to visualize, read and visualize images. I also use numpy to create empty image arrays.

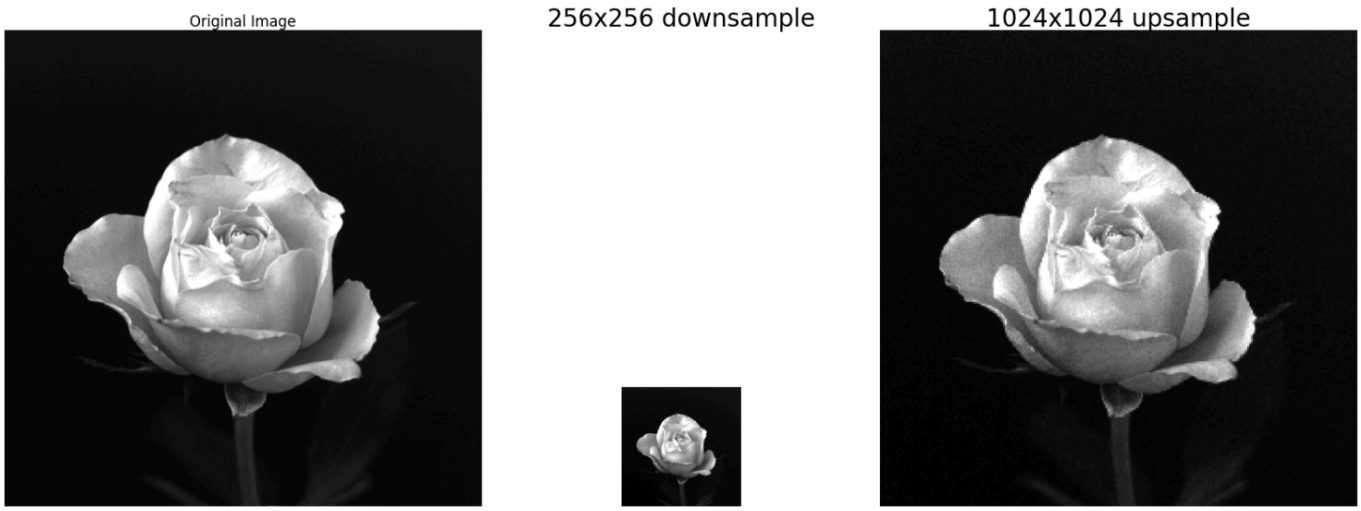
2. TEST RESULTS

2.1 Test Results for Image Downsampling and UpSampling

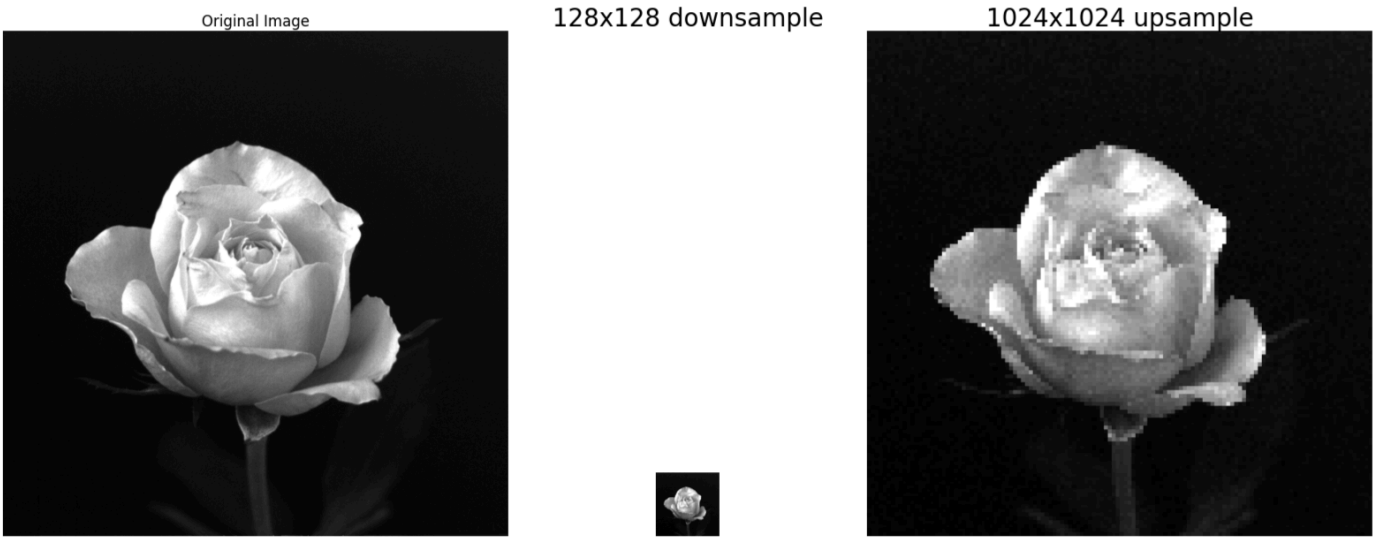
All images were put through my own down sampling function that would take the original image
and create a new image that was scaled down by the factor of the original image dimensions divided by the output image dimensions.

Then those same images were run through my own up sampling function that would create a new image by looping through the down sampled image and spreading the values of each pixel through the new image by the factor calculated earlier.

Fig (1) Original Image down sampled to 512x512 then back up to 1024x1024



Fig(2) Original Image down sampled to 256x256 then back up to 1024x1024



Fig(3) Original Image down sampled to 128x128 then back up to 1024x1024

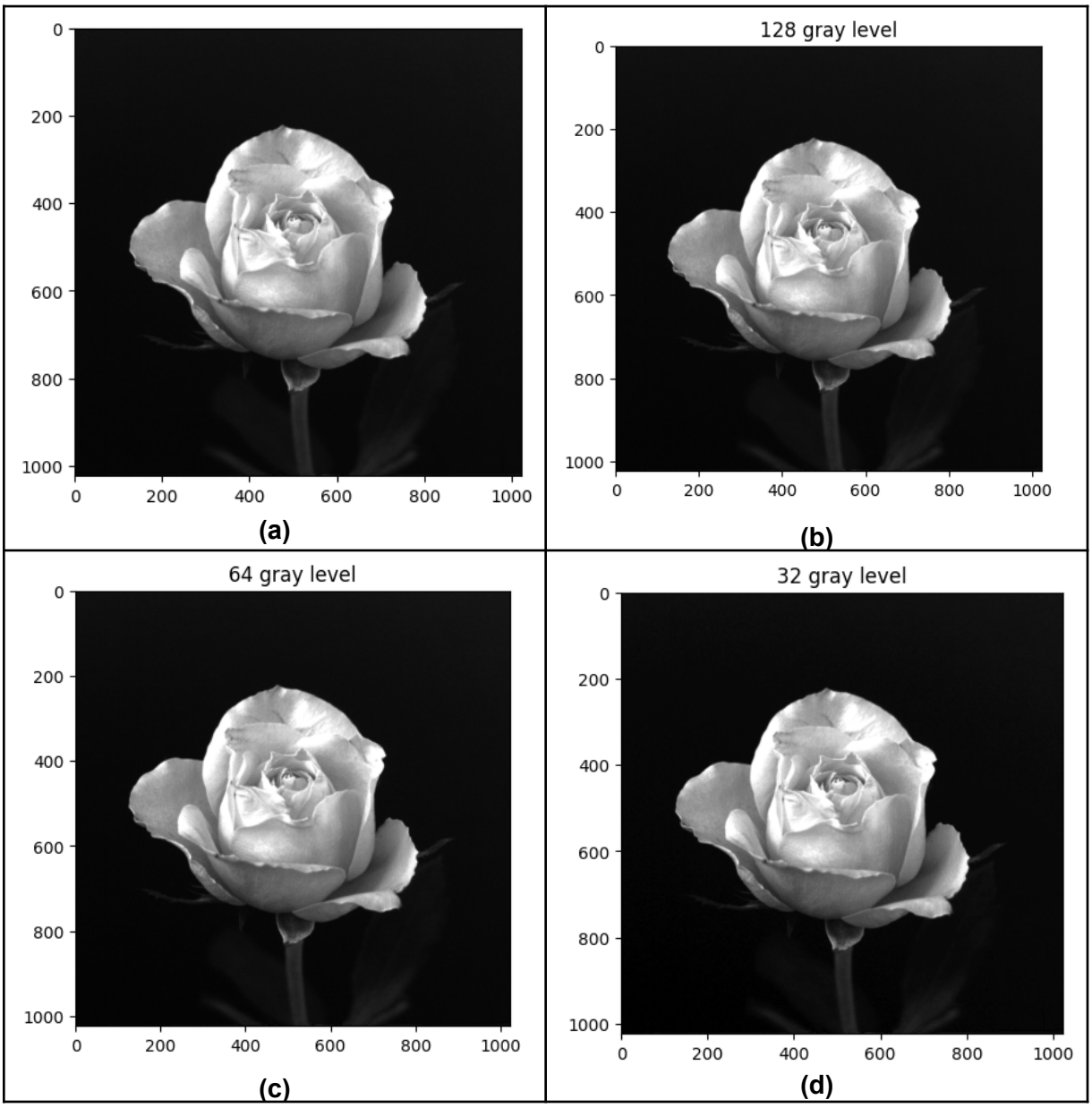


Figure 4: (a) Original grayscale image (rose.jpg) (b-d) Variations of original image with intensity level adjusted

2.2 Test Results for Image Upside-down

For each test the original image is through my own gray level changing function. This would take in the number of bits there should be. It would then create a new empty array for the new image and then loop through the input image. For each pixel in the original image a mask is created to find which bits we want to be discarded. Then a bitwise operation is down on the pixel to remove the lower bits that are no longer needed. And finally the bits are shifted to represent their new bit level and that pixel is added to the new image.

It is a bit hard to see the different intensity levels in these examples, but I tested this same function with a lower bit level like three and two and the difference there is much more noticeable.

3. DISCUSSION

From this assignment I learned simple down and up scaling. These were already concepts that I was familiar with but I had never implemented in a program. I also learned how to adjust the intensity level of a grayscale image. This concept I was not familiar with at all and had to do some research to learn how I would tackle this problem. I learned about creating masks and bitwise operations. I had heard of these before but never learned or researched them.

4. CODE

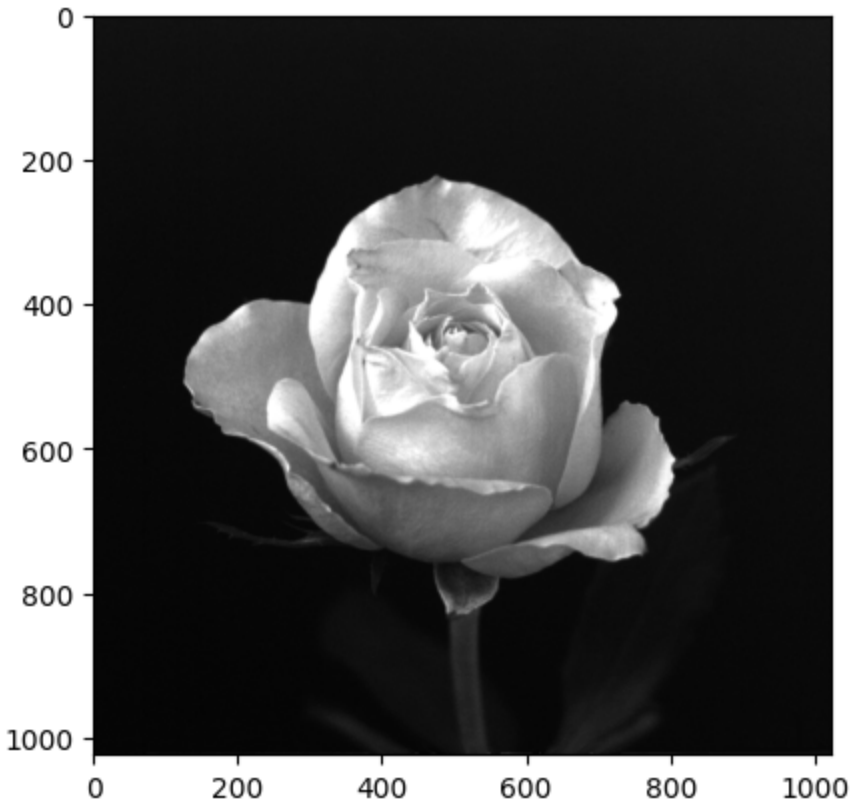
Max Haviv

Assignment 1 Image Resolution

Initial Image

In [186…

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

img = mpimg.imread('rose.jpg')
plt.imshow(img, cmap='gray')
plt.show()
```



Down Sample Function

In [187…

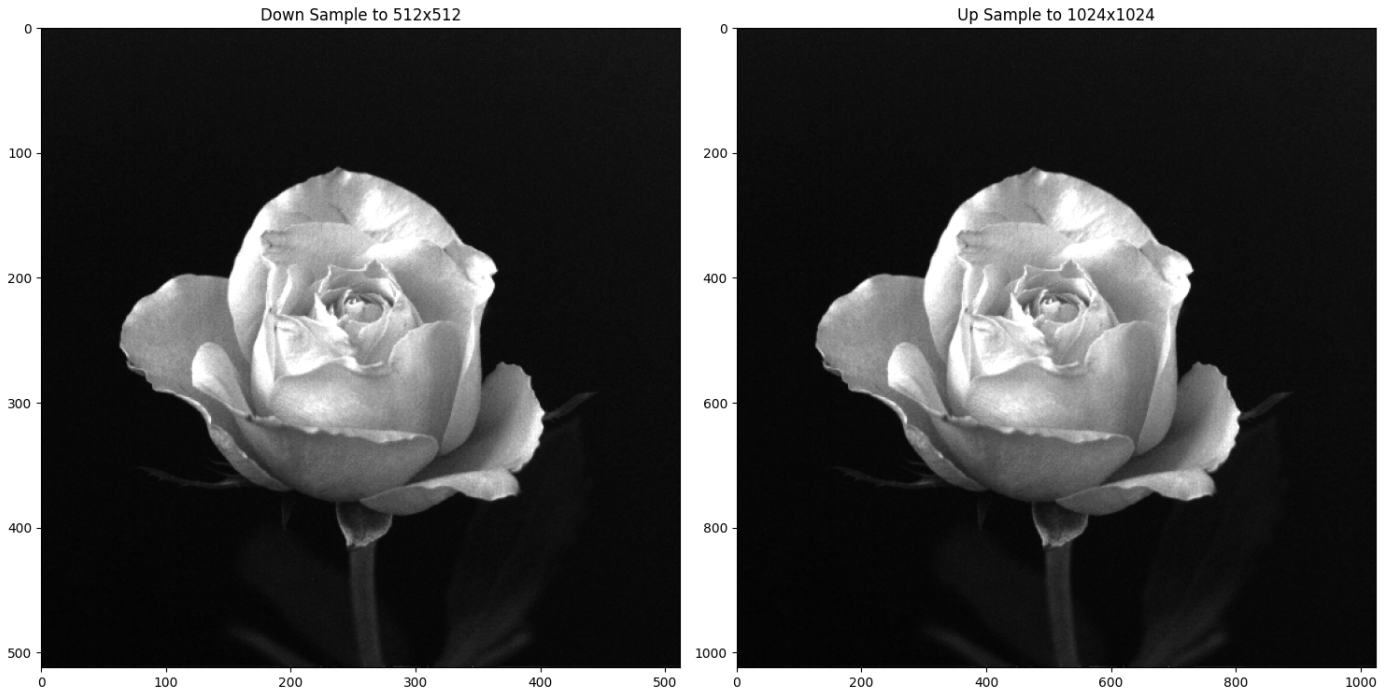
```
def down_sample(input_dimensions, output_dimensions, img):
  factor = int(input_dimensions / output_dimensions)
  down_sampled_img = np.zeros((output_dimensions, output_dimensions))

  # loop through the dimensions of the output to fill in the new image
  for row in range(output_dimensions):
    for col in range(output_dimensions):
      # select a pixel from the original image to set into the downscaled image
      # this pixel is the row and col multiplied by the factor that the image is being d
      selected_pixel= img[factor * row][factor * col]
      down_sampled_img[row,col] = selected_pixel

  return down_sampled_img
```

Up Sample Function

In [188…]
```
def up_sample(input_dimensions, output_dimensions, img):
  factor = int(output_dimensions / input_dimensions)
  up_sample_img = np.zeros((output_dimensions,output_dimensions))

  # loop through the input image
  for row in range(input_dimensions):
    for col in range(input_dimensions):
      selected_pixel = img[row][col]
      # spread the slected pixel through the rows and columns
      up_sample_img[row * factor:(row + 1) * factor,col * factor:(col + 1) * factor] = s
  return up_sample_img
```

512 Down and Up Sample

In [210…]
```
img_512 = down_sample(1024, 512, img)

img_512_upsample = up_sample(512, 1024, img_512)

fig, ax = plt.subplots(1,2, figsize=(15,10))

ax[0].imshow(img_512, cmap='gray', aspect='equal', extent=[0, img_512.shape[1], img_512.
ax[0].set_title(f"Down Sample to 512x512")
ax[0].axis('on')


ax[1].imshow(img_512_upsample, cmap='gray', aspect='equal', extent=[0, img_512_upsample.
ax[1].set_title(f"Up Sample to 1024x1024")
ax[1].axis('on')

plt.autoscale(False)
plt.tight_layout()
plt.show()
```



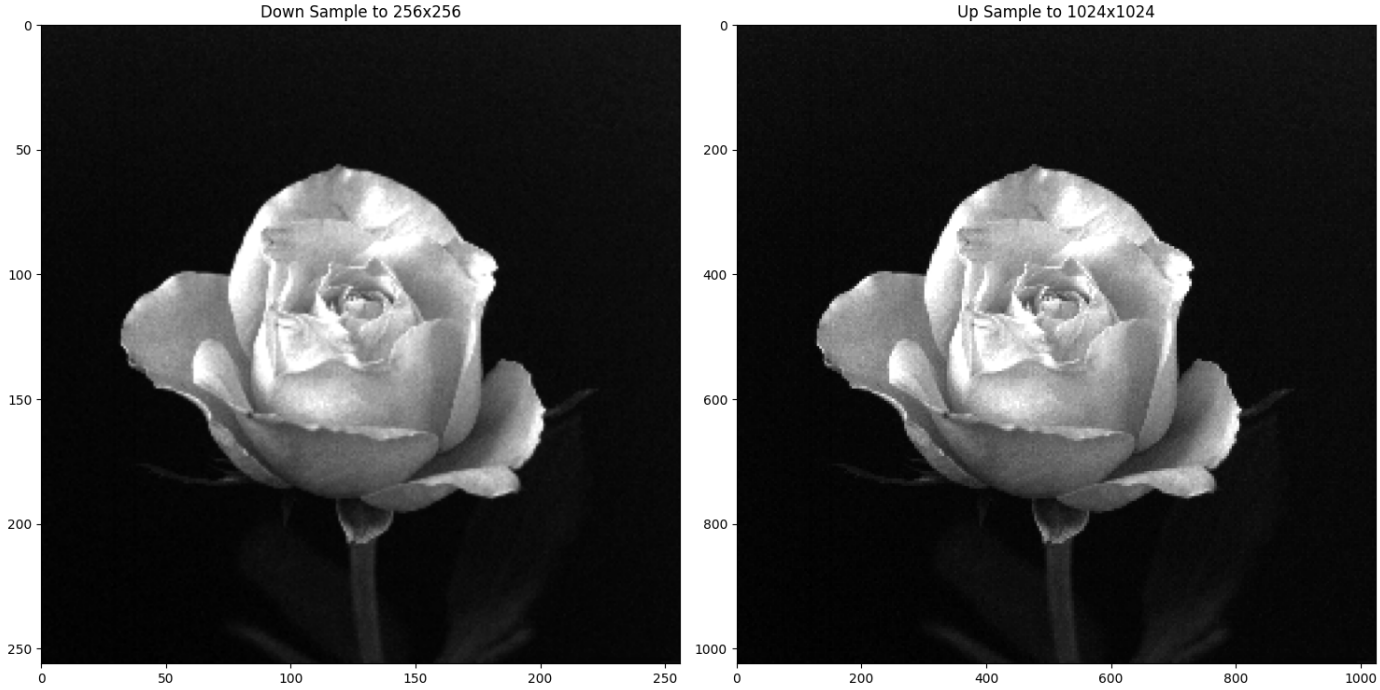
In [253…]
```
img1 = img
img2 = img_512
```

```
img3 = img_512_upsample

# Get image dimensions
img1_height, img1_width = img1.shape[:2]
img2_height, img2_width = img2.shape[:2]
img3_height, img3_width = img3.shape[:2]

spacing = 300

fig, ax = plt.subplots(figsize=(20,10))

ax.imshow(img1, cmap='gray', extent=(0, img1_width, 0, img1_height))
plt.text(img1_width / 2, img1_height + 10, 'Original Image', ha='center', fontsize=12)

ax.imshow(img2, cmap='gray', extent=(img1_width + spacing, img1_width + img2_width + spa
plt.text(img1_width + img2_width / 2 + spacing, max(img1_height, img2_height) + 10, '512

ax.imshow(img3, cmap='gray', extent=(img1_width + img2_width + 2 * spacing, img1_width +
plt.text(img1_width + img2_width + img3_width / 2 + 2 * spacing, max(img1_height, img3_h

# Adjust the axes limits to fit all three images and titles with spacing
ax.set_xlim(0, img1_width + img2_width + img3_width + 2 * spacing)
ax.set_ylim(0, max(img1_height, img2_height, img3_height) + 30)

ax.axis('off')
plt.show()
```

256 Down and Up Sample

In [211…

```
img_256 = down_sample(1024, 256, img)

img_256_upsample = up_sample(256, 1024, img_256)

fig, ax = plt.subplots(1,2, figsize=(15,10))

ax[0].imshow(img_256, cmap='gray', aspect='equal', extent=[0, img_256.shape[1], img_256.
ax[0].set_title(f"Down Sample to 256x256")
ax[0].axis('on')


ax[1].imshow(img_256_upsample, cmap='gray', aspect='equal', extent=[0, img_256_upsample.
ax[1].set_title(f"Up Sample to 1024x1024")
ax[1].axis('on')

plt.autoscale(False)
plt.tight_layout()
plt.show()
```



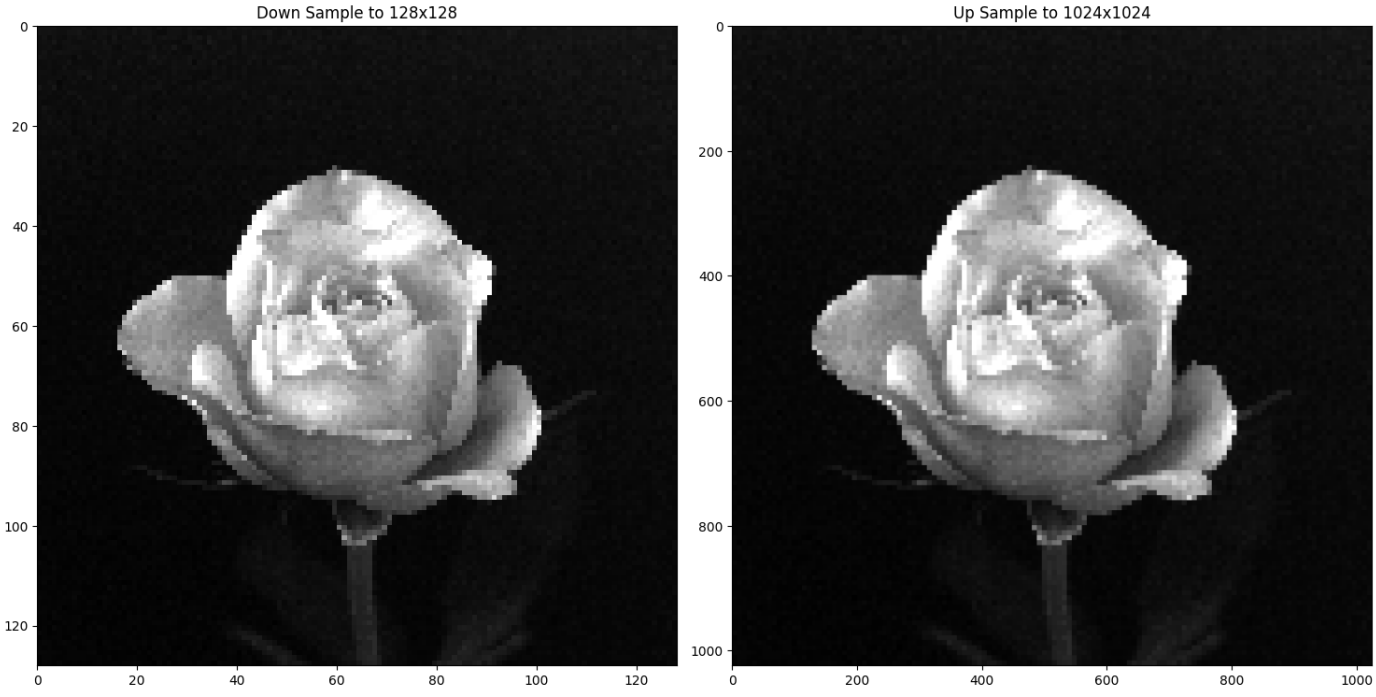
In [254…

```
img1 = img
img2 = img_256
img3 = img_256_upsample

# Get image dimensions
img1_height, img1_width = img1.shape[:2]
img2_height, img2_width = img2.shape[:2]
img3_height, img3_width = img3.shape[:2]

spacing = 300

fig, ax = plt.subplots(figsize=(20,10))

ax.imshow(img1, cmap='gray', extent=(0, img1_width, 0, img1_height))
plt.text(img1_width / 2, img1_height + 10, 'Original Image', ha='center', fontsize=12)

ax.imshow(img2, cmap='gray', extent=(img1_width + spacing, img1_width + img2_width + spa
plt.text(img1_width + img2_width / 2 + spacing, max(img1_height, img2_height) + 10, '256

ax.imshow(img3, cmap='gray', extent=(img1_width + img2_width + 2 * spacing, img1_width +
plt.text(img1_width + img2_width + img3_width / 2 + 2 * spacing, max(img1_height, img3_h

# Adjust the axes limits to fit all three images and titles with spacing
ax.set_xlim(0, img1_width + img2_width + img3_width + 2 * spacing)
ax.set_ylim(0, max(img1_height, img2_height, img3_height) + 30)

ax.axis('off')
plt.show()
```

128 Down and Up Sample

In [212…

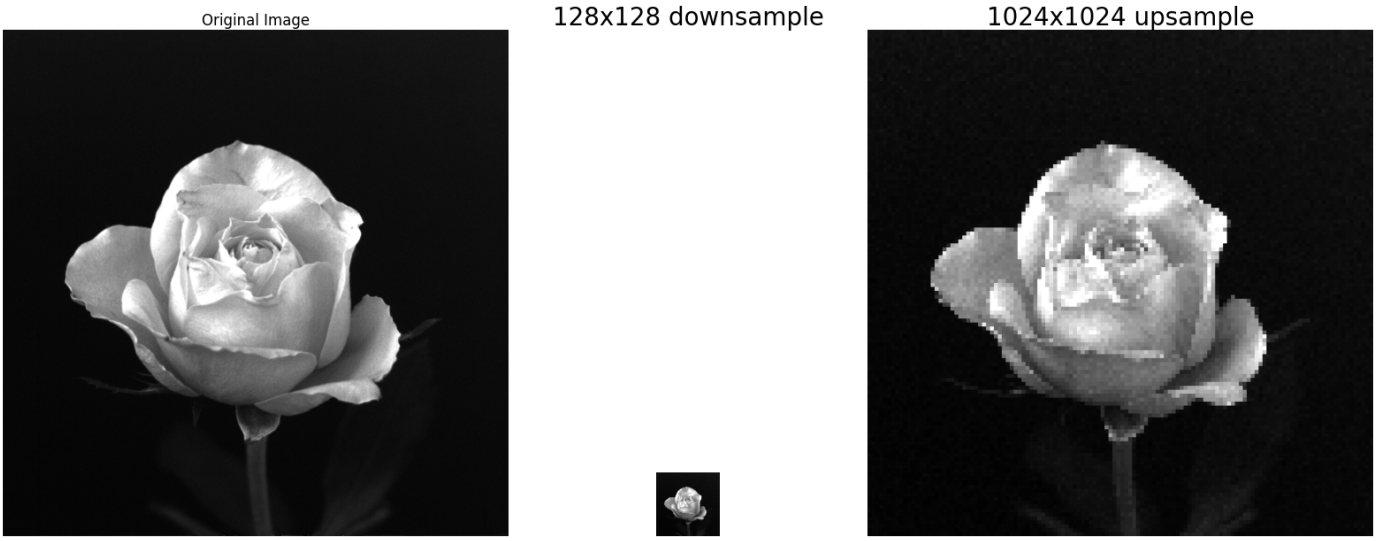
```
img_128 = down_sample(1024, 128, img)

img_128_upsample = up_sample(128, 1024, img_128)

fig, ax = plt.subplots(1,2, figsize=(15,10))

ax[0].imshow(img_128, cmap='gray', aspect='equal', extent=[0, img_128.shape[1], img_128.
ax[0].set_title(f"Down Sample to 128x128")
ax[0].axis('on')


ax[1].imshow(img_128_upsample, cmap='gray', aspect='equal', extent=[0, img_128_upsample.
ax[1].set_title(f"Up Sample to 1024x1024")
ax[1].axis('on')

plt.autoscale(False)
plt.tight_layout()
plt.show()
```



In [255…

```
img1 = img
img2 = img_128
```

```
img3 = img_128_upsample

# Get image dimensions
img1_height, img1_width = img1.shape[:2]
img2_height, img2_width = img2.shape[:2]
img3_height, img3_width = img3.shape[:2]

spacing = 300

fig, ax = plt.subplots(figsize=(20,10))

ax.imshow(img1, cmap='gray', extent=(0, img1_width, 0, img1_height))
plt.text(img1_width / 2, img1_height + 10, 'Original Image', ha='center', fontsize=12)

ax.imshow(img2, cmap='gray', extent=(img1_width + spacing, img1_width + img2_width + spa
plt.text(img1_width + img2_width / 2 + spacing, max(img1_height, img2_height) + 10, '128

ax.imshow(img3, cmap='gray', extent=(img1_width + img2_width + 2 * spacing, img1_width +
plt.text(img1_width + img2_width + img3_width / 2 + 2 * spacing, max(img1_height, img3_h

# Adjust the axes limits to fit all three images and titles with spacing
ax.set_xlim(0, img1_width + img2_width + img3_width + 2 * spacing)
ax.set_ylim(0, max(img1_height, img2_height, img3_height) + 30)

ax.axis('off')
plt.show()
```



Change the gray level of an 8-bit gray level image

define funtion

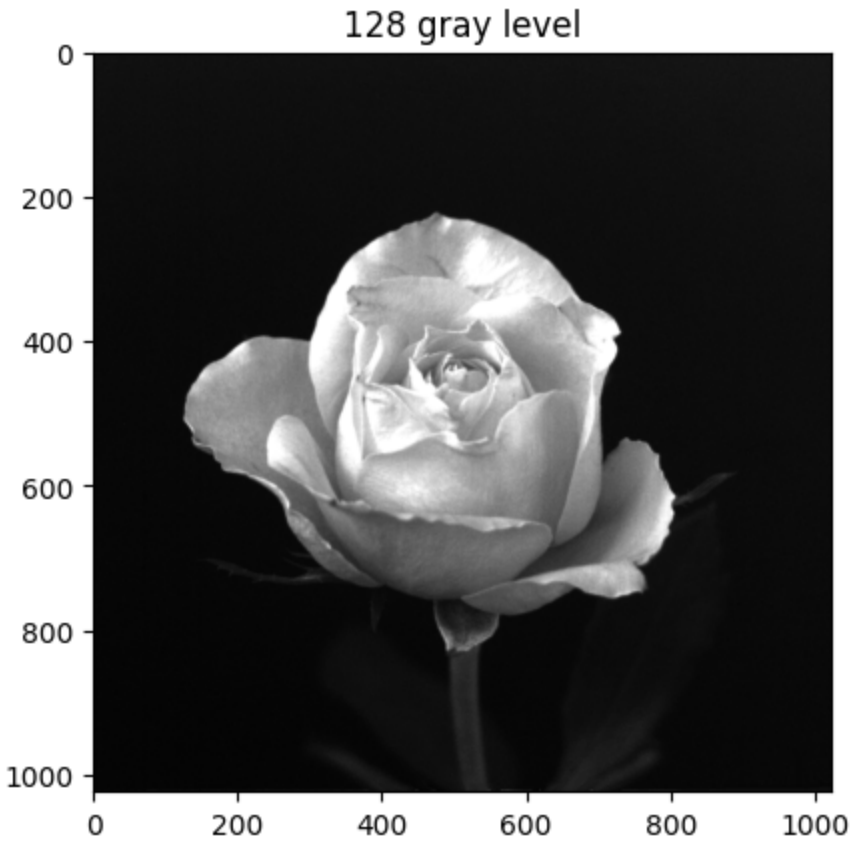
In [256…

```
def change_gray_level(bit):
  new_img = np.zeros((1024,1024))

  for row in range(1024):
    for col in range(1024):
      pixel = img[row,col]
      # create a mask for bits that we want to discard
      mask = 256 - (2 ** (8-bit))
      # bitwise operation clears the lower bits
      bitwise_pixel = pixel & mask
      # shift the bits from their original 8 bits to the bits provided
      final_pixel = bitwise_pixel >> (8 - bit)
```

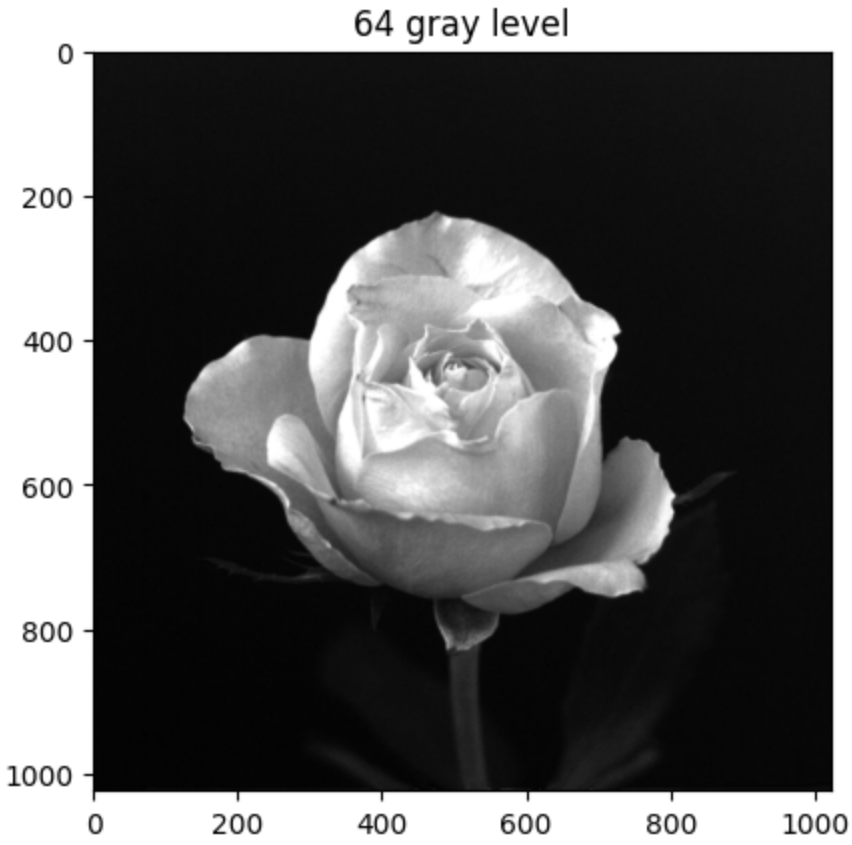
```
      new_img[row, col] = final_pixel

  return new_img
```

In [257…
```
img_128_gray = change_gray_level(7)
img_64_gray = change_gray_level(6)
img_32_gray = change_gray_level(5)
```

In [258…
```
plt.imshow(img_128_gray, cmap='gray')
plt.title("128 gray level")
plt.show()
```

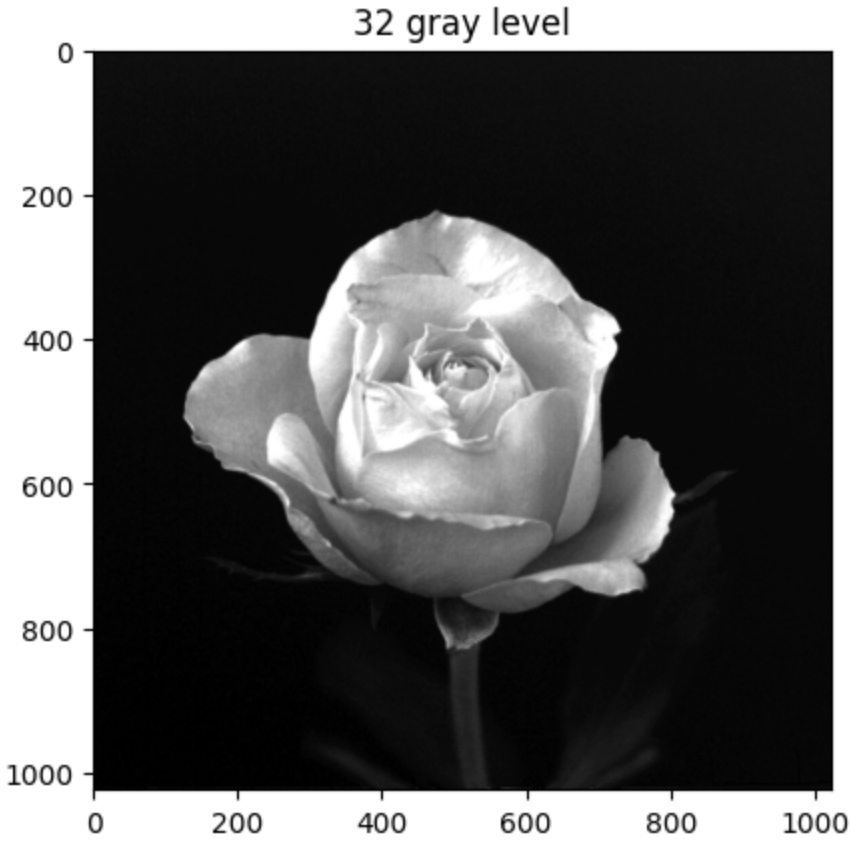


In [259…
```
plt.imshow(img_64_gray, cmap='gray')
plt.title("64 gray level")
plt.show()
```



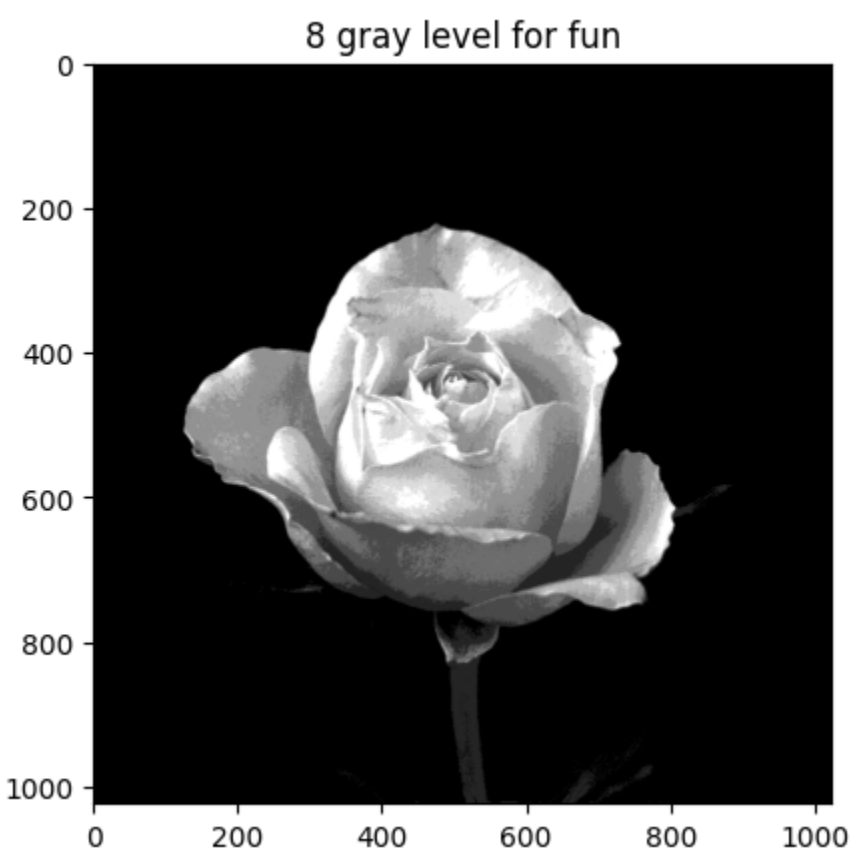
In [260…

```
plt.imshow(img_32_gray, cmap='gray')
plt.title("32 gray level")
plt.show()
```



In [262…

```
plt.imshow(change_gray_level(3), cmap='gray')
plt.title("8 gray level for fun")
plt.show()
```



In [261…

```
plt.imshow(change_gray_level(2), cmap='gray')
plt.title("4 gray level for fun")
plt.show()
```