CMPEN 455: Digital Image Processing

Project 1

Lab Introduction & Digital Image Quantization

Jonathan Lausch, Qiong Li, Zekai Liu

9/13/2019

A. Objectives

There are two objectives of this project:

1. Familiarize ourselves with MATLAB by importing digital images;
2. Perform quantization on digital image.

B. Methods

Question 1. First, we down-sampled the 512×512 image to 256×256, 128×128, and 32×32 pixels, effectively lowering the original sample rate. This was done by iterating through each pixel, and only saving every 2nd, 4th, or 16th. The same method can be applied to achieve different smaller resolutions as well. We then resized these images to have a resolution of 512x512 using nearest-neighbor interpolation. This method of interpolation repeats the value of a given pixel in a larger space, as seen in **Figure 1** below.

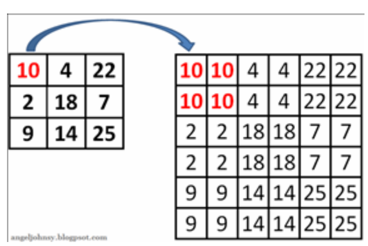

Figure 1. Illustration of nearest neighbor interpolation (from https://www.imageeprocessing.com/2017/11/nearest-neighbor-interpolation.html).

Question 2. This question called for the use of bilinear interpolation, as opposed to nearest-neighbor interpolation, like the last question.

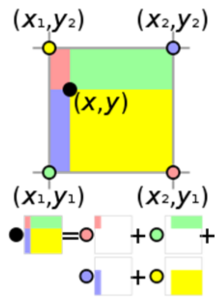

Figure 2. Bilinear interpolation.

The pixels (x1, y1), (x2, y1), (x1, y2), (x2, y2) are the values of down-sampled image. In this case, x2-x1 is equals to the up-sampling rate, which is 16.

We used the functions below to find the values of the unknown pixels (x,y).

$$f1 = \frac{Upsampling\ rate - distance\ x\ to\ (x1,y1)}{upsampling\ rate} \times (x1,y1) + \frac{distance\ x\ to\ (x1,y1)}{upsampling\ rate} \times (x2,y1)$$

$$f2 = \frac{Upsampling\ rate - distance\ x\ to\ (x1,y1)}{upsampling\ rate} \times (x1,y2) + \frac{distance\ x\ to\ (x1,y1)}{upsampling\ rate} \times (x2,y2)$$

$$(x,y) = f1 \times \frac{upsampling\ rate - distance\ y\ to\ (x1,y1)}{upsampling\ rate} + f2 \times \frac{distance\ y\ to\ (x1,y1)}{upsampling\ rate}$$

While the equations are quite lengthy, the process is nothing more than a weighted average of the known pixels in the x and y direction, creating a smoother visual transition from pixel to pixel in the new image.

Question 3. To start we declared arrays of zeroes, which would later be used to save the new values after truncation. The original image was converted to a double type, so we would not lose data when adjusting the gray-scale. To remove n bits from the grayscale, we divided the original data by 2^n and took the floor of the values. After that, we multiplied these pixels by 2^n, so we could utilize the full range of possible values. To demonstrate, ***Figure 3*** shows the original 8-bit grayscale pixels and ***Figure 4*** shows the processed 7-bit grayscale pictures.

	1	2	3
1	93	81	85
2	105	97	94
3	89	85	112

Figure 3. Example Original 8-bit Grayscale

	1	2	3
1	92	80	84
2	104	96	94
3	88	84	112

Figure 4. Example Quantized 7-bit Grayscale values

After the grayscale was quantized, we converted the values in the from type double to type uint8.

Question 4. In this part, we changed the spatial resolution of the original image to 256 × 256 pixels using the functions created for question 1, and lowered the gray-scale resolution from 8 to 6 bits/pixel using the function from question 3.

C. Results

Question 1. **Figure 6,** the up-sampled 256x256 image retains the most detail from the original image. **Figure 7**, the 128x128 image is somewhat blurry; the data loss is noticeable, however, **Figure 8**, the up-sampled 32×32 image lost most of the original's detail; it would be hard to say it was a bridge without knowing what the original looked like.

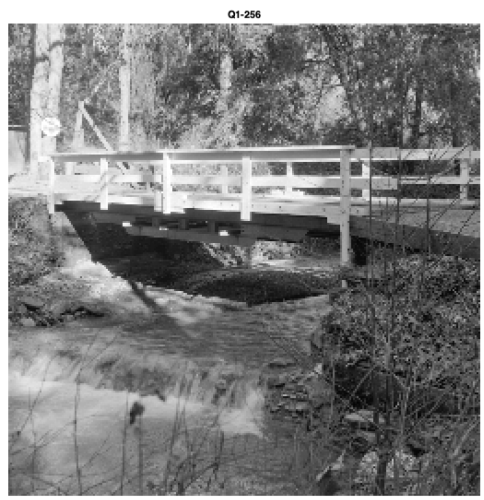

Figure 6. The up-scaled 256 x 256 image

Figure 7. The upscaled 128 x 128 image

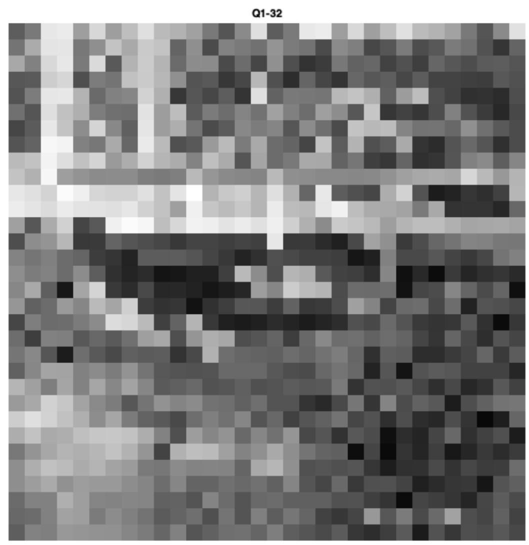

Figure 8. The upscaled 32 x 32 image

Question 2. We used bilinear interpolation on the 32x32 image. The results are shown below (**Figure 9**). Comparing Figure 9 with Figure 8, the image is more likely to be seen as a bridge. Bilinear interpolation acts as a sort of anti-aliasing to remove the artifacts introduced from down-sampling.

Figure 9. Bilinear interpolation of 32 X32 image

Question 3. We quantized the gray level of the 512×512 image (by reducing the number of bits per pixel). **Figure 10** shows a 7-bit gray scale, **Figure 11** a 6 bit scale, **Figure 12** a 5 bit scale, **Figure 13** a 4 bit scale, **Figure 14** a 3 bit scale, **Figure 15** a 2 bit scale, and a 1 bit scale in **Figure 16**. The image quality is positively correlated with the number of bits used per pixel. As seen in **Figures 11-16**, the bridge is still relatively distinguishable as the number of bits decrease, but detail in the environment and background is lost and becomes 'washed out' as the bit-count is lowered.

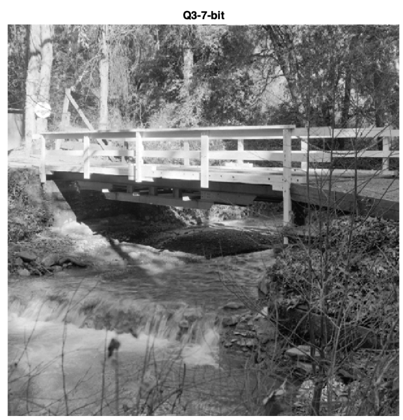

Figure 10. 7-bit image

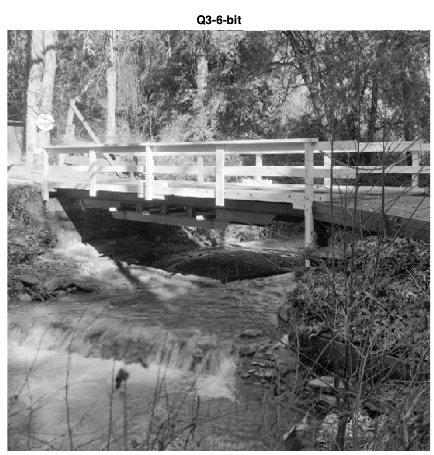

Figure 11. 6-bit image

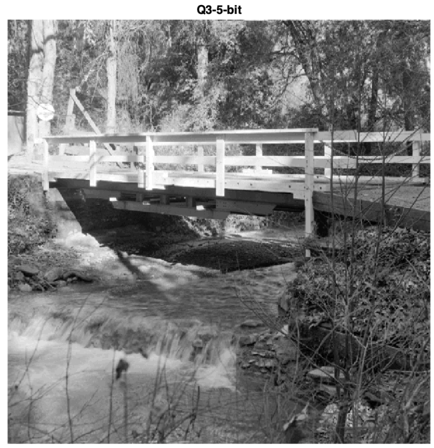

Figure 12. 5-bit image

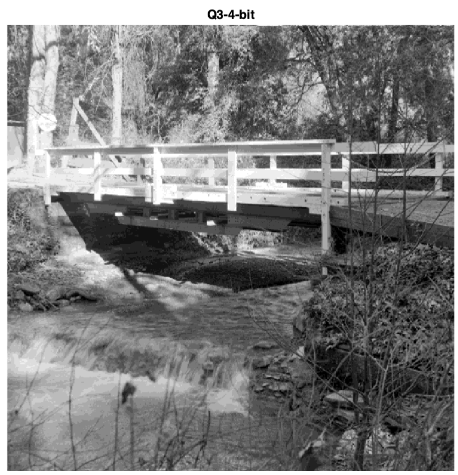

Figure 13. 4-bit image

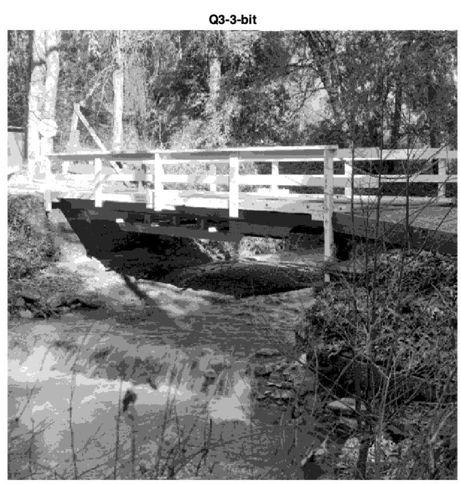

Figure 14. 3-bit image

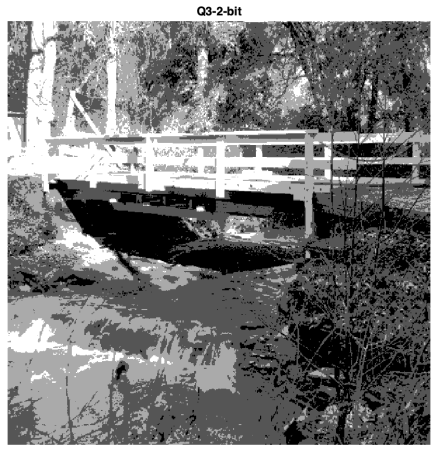

Figure 15. 2-bit image

Figure 16. 1-bit image

Question 4. This part combined the gray-scale quantization and down-sampling of the questions 1 & 3. We lowered the resolution to 256 × 256 and the gray-scale resolution to 6 bits per pixel (**Figure 17**). Compared to the original image, we can still find obvious distortion and more pronounced fine details.

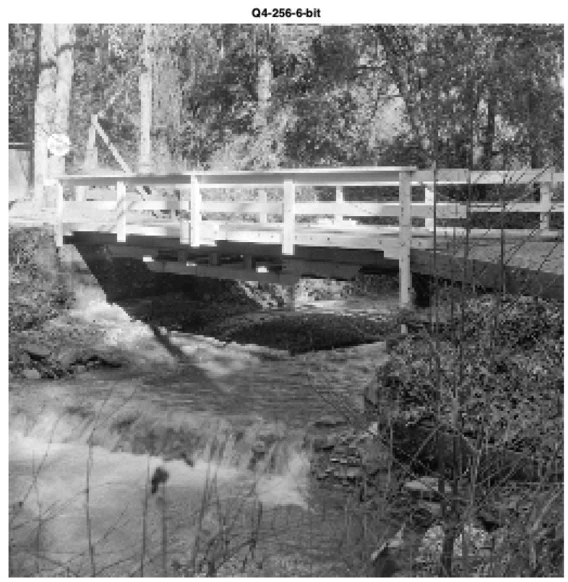

Figure 17. Processed image

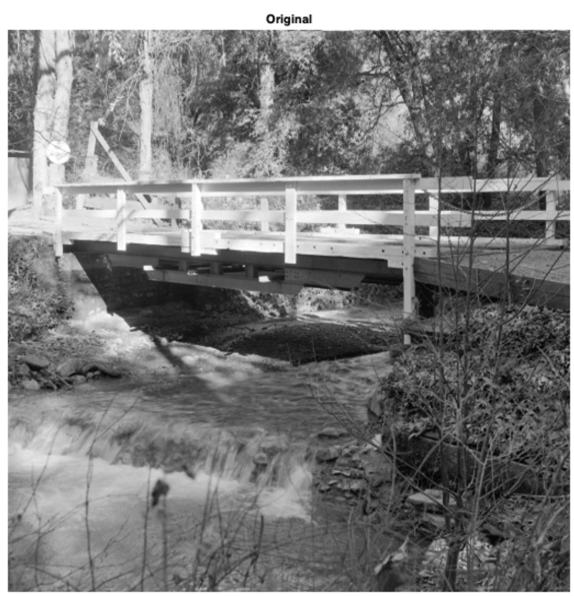

Figure 18. Original image

D. Conclusion

From using two different methods of interpolation, it is better to use a method where the known pixels are used to fill in the unknown pixels, as an image closer to the original will be produced. This seems to be a way of anti-aliasing, or way to correct the errors/artifacts created when down-sampling an image.

Quantizing the gray scale can really dramatize the lighting in an image, as well as mask many details of the image when quantized to the extreme.