Image Processing: Sampling & Quantization Using MATLAB

DATA COMMUNICATION SYSTEMS (ELC 2709)
PROF OSAMA EL GHANDOUR
TA BASMA DIAA

Prepared by:
Usama Mohammed Mohammed Abdulgwad

TABLE OF CONTENTS

Introduction

Digital image processing is a key area of study and application in computer science and engineering, with broad implications across a range of fields such as medical imaging, satellite imagery analysis, robotics, and multimedia applications. Central to many image processing tasks are the fundamental processes of image sampling and quantization, which play pivotal roles in determining the quality, size, and computational complexity of digital images.

The objective of this project is to conduct a comprehensive analysis of image sampling and quantization techniques using MATLAB. Specifically, the project aims to explore the effects of sampling and quantization on image quality and file size, and to compare the results of different techniques to understand their trade-offs and implications in image processing workflows. This project only concerns about the sampling and quantization as the quintessential facets of the image begin processed.

Image sampling is the process of converting a continuous image into a discrete representation by selecting a subset of pixels. This process is akin to how a digital camera captures an image, where the sensor records discrete samples of the continuous scene. The resolution of the sampled image directly affects its quality, with higher resolutions capturing more detail but requiring more storage space and processing power.

Quantization, on the other hand, involves reducing the number of intensity levels in an image. In digital images, each pixel is typically represented by a certain number of bits, which determines the number of possible intensity levels. Quantization reduces this number, leading to a loss of information and a decrease in image quality. However, quantization is essential for reducing file sizes and simplifying image processing operations.

The project will begin by resizing the original image to different resolutions using MATLAB's imresize function. Images will be resized to resolutions such as 1024x1024, 256x256, 128x128, and so on, to observe the effects of sampling on image quality. Subsequently, quantization will be applied to the sampled images to reduce the number of intensity levels. This will be achieved using the imquantize function, with thresholds selected to quantize the images to 4 and 8 intensity levels.

The sampled and quantized images will be visually inspected and compared to the original image. Through this project, we aim to provide valuable insights into image processing workflows and the trade-offs involved in optimizing image quality and file size.

Theory & Background

Digital image processing involves manipulating digital images using computer algorithms. Two fundamental processes in digital image processing are image sampling and quantization, which are essential for various image processing tasks such as compression, enhancement, and analysis.

1. Image Sampling: Sampling is the process of converting a continuous image into a discrete representation. In digital images, each pixel represents a sample of the original continuous image. The resolution of a digital image is determined by the number of pixels in the image. Higher resolutions result in more detailed images but also require more storage space.
2. Image Quantization: Quantization is the process of reducing the number of intensity/threshold levels in an image. In digital images, each pixel is represented by a certain number of bits, which determines the number of possible intensity/threshold levels. Quantization reduces the number of these levels, leading to a loss of information and a decrease in image quality. However, quantization is necessary for reducing file sizes and simplifying image processing operations.

These processes are crucial in image processing for several reasons:

- Memory Efficiency: Sampling and quantization help reduce the amount of memory required to store images. By reducing the resolution and intensity levels, the size of the image file can be significantly reduced without losing essential information.
- Computational Efficiency: Images with lower resolutions and fewer intensity levels are computationally less complex to process. This makes image processing operations such as filtering, edge detection, and segmentation faster and more efficient.
- Transmission Efficiency: In applications where images need to be transmitted over networks with limited bandwidth, such as in remote sensing or video streaming, sampling and quantization help reduce the amount of data that needs to be transmitted, improving transmission efficiency.
- Visual Quality: While sampling and quantization lead to a loss of information, they can be optimized to minimize perceptible differences in image quality. Various techniques, such as dithering and error diffusion, can be used to improve the visual quality of quantized images.

Understanding these processes and their implications is essential for optimizing image processing workflows and achieving the desired balance between image quality, file size, and computational complexity.

Methodology & Outcomes

Methodology:

- **Image Sampling:** The project starts by loading the original image (The new logo of Helwan University, Faculty of Engineering), using MATLAB's imread function. The image will then be sampled at different resolutions using the imresize function. Resolutions such as 1024x1024, 256x256, 128x128, and so on, will be used to observe the effects of sampling on image quality. Each sampled image will be saved for further analysis and will be displayed as figure 1 by using the subplot function.

```
img = imread('SampleImage.jpg');

img1 = imresize(img, [1024 1024]); % n = 10
img2 = imresize(img1, [256 256]); % n = 8
img3 = imresize(img2, [128 128]); % n = 7
img4 = imresize(img3, [64 64]); % n = 6
img5 = imresize(img4, [32 32]); % n = 5
img6 = imresize(img5, [16 16]); % n = 4
img7 = imresize(img6, [4 4]); % n = 2
img8 = imresize(img7, [2 2]); % n = 1

figure(1)
axis off
subplot(3, 3, 1);imshow(img, []); title('Orginal Image')
subplot(3, 3, 2);imshow(img1, []); title('Image[1024, 1024]')
subplot(3, 3, 3);imshow(img2,[]); title('Image[256, 256]')
subplot(3, 3, 4);imshow(img3, []); title('Image[128, 128]')
subplot(3, 3, 5);imshow(img4,[]); title('Image[64, 64]')
subplot(3, 3, 6);imshow(img5, []); title('Image[32, 32]')
subplot(3, 3, 7);imshow(img6,[]); title('Image[16, 16]')
subplot(3, 3, 8);imshow(img7, []); title('Image[4, 4]')
subplot(3, 3, 9);imshow(img8,[]); title('Image[2, 2]')
```

Figure 1 is a code snippet for the sampling part

- **Image Quantization:** After sampling, the sampled images will undergo quantization to reduce the number of intensity levels. At first, we measure the threshold/intensity levels of the sample image to have an idea of where we are at in terms of quality. The original threshold/intensity levels are measured using the multithresh function and stored in thresh variable. After that the thresholds levels are reduced to 8 and then to 4 levels only and are displayed as minimum interval value and maximum interval value for each reduction. This will be done using the imquantize function in MATLAB and will be displayed using the montage function.

```
thresh = multithresh(img, 1);
disp(['Original Threshold Value: ', num2str(thresh)]);

Max_Quantization_Level = max(img(:));

%Quantization Level Reduction to 8
thresh8 = multithresh(img, 7);
valuesMax8 = [thresh8 max(img(:))];
valuesMin8 = [min(img(:)) thresh8];
[quant8_I_Max, Index] = imquantize(img, thresh8, valuesMax8);
quant8_I_Min = valuesMin8(Index);

%Quantization Level Reduction to 4
thresh4 = multithresh(img, 3);
valuesMax4 = [thresh4 max(img(:))];
valuesMin4 = [min(img(:)) thresh4];
[quant4_I_Max, index] = imquantize(img, thresh4, valuesMax4);
quant4_I_Min = valuesMin4(index);

%Displaying
figure(2)
multi = cat(4,quant8_I_Min,quant8_I_Max,quant4_I_Min,quant4_I_Max);
montage(multi, 'Size', [2, 2]);
title('Minimum Interval Value                Maximum Interval Value')
ylabel('L = 4                                          L = 8')
```

Figure 2 is a code snippet for the quantization part

Outcomes:

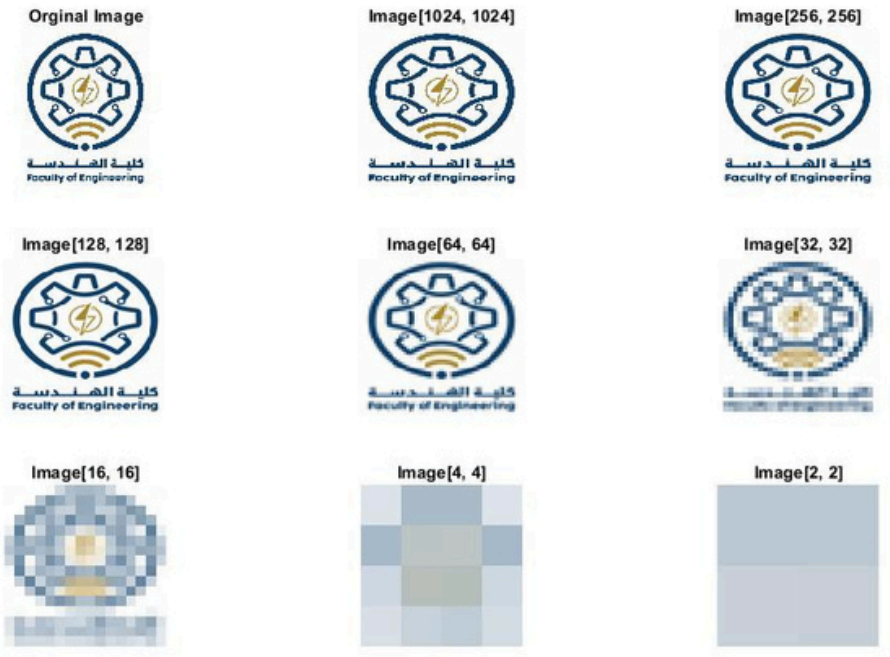

Figure 3 shows the outcome of the sampling part

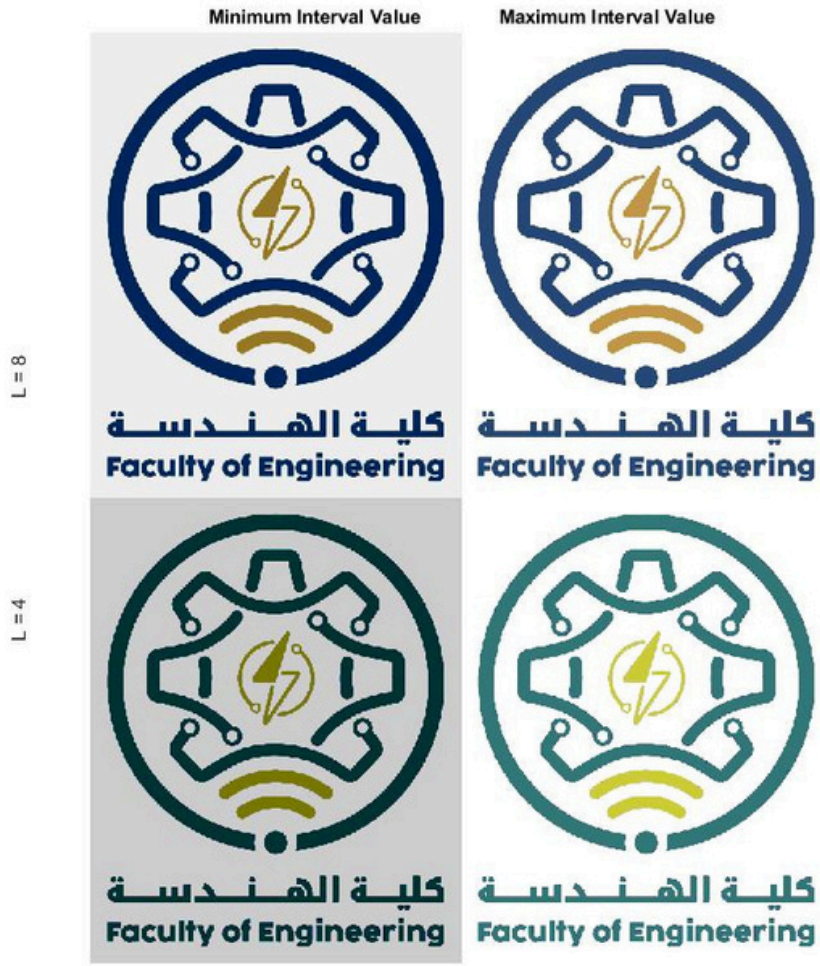

Figure 4 shows the outcome of the quantization part

Conclusion

Through this project, the significance of image sampling and quantization in digital image processing and their practical applications in various domains is thoroughly invesitgated. The project provides valuable insights into optimizing image processing workflows for efficiency and effectiveness.