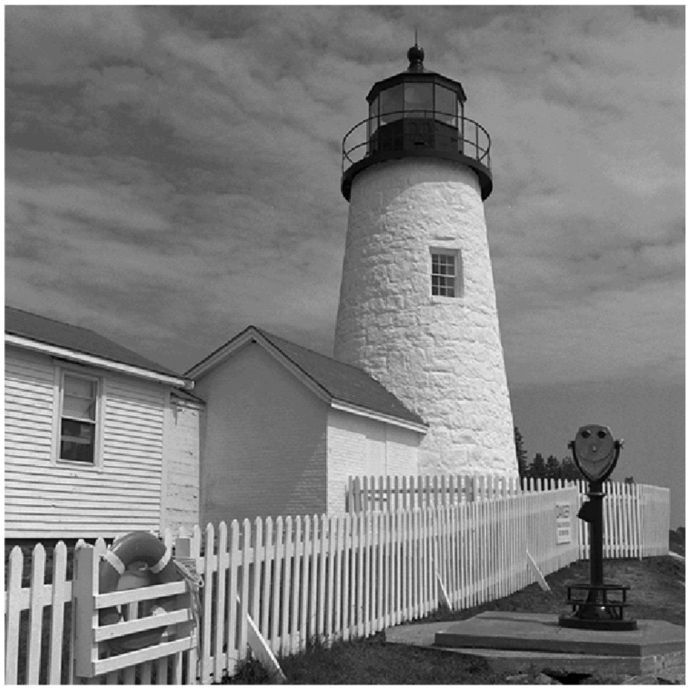
Image Processing Applications of Probability Theory

Qudsia Sultana

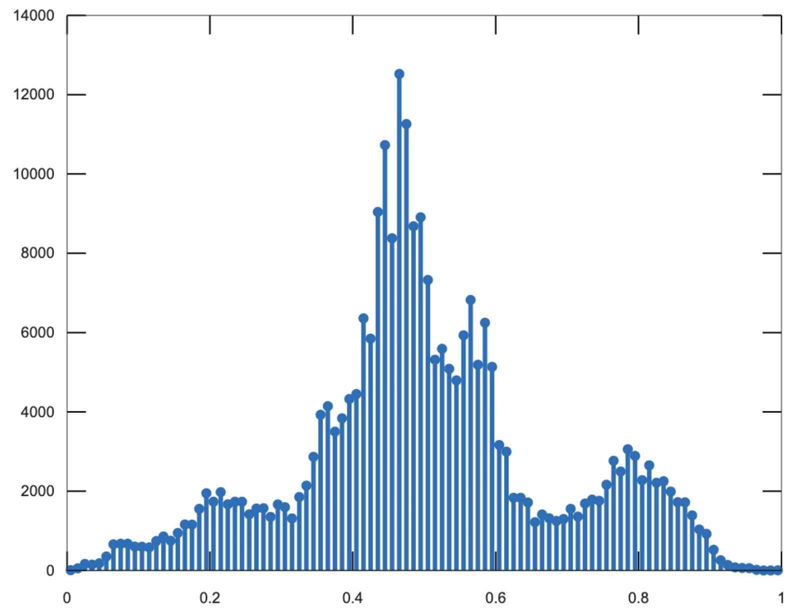
a) Graph the image that was provided (type "load image"; the image is returned in matrix A

```
octave:2> load image
octave:3> imshow(A)
```



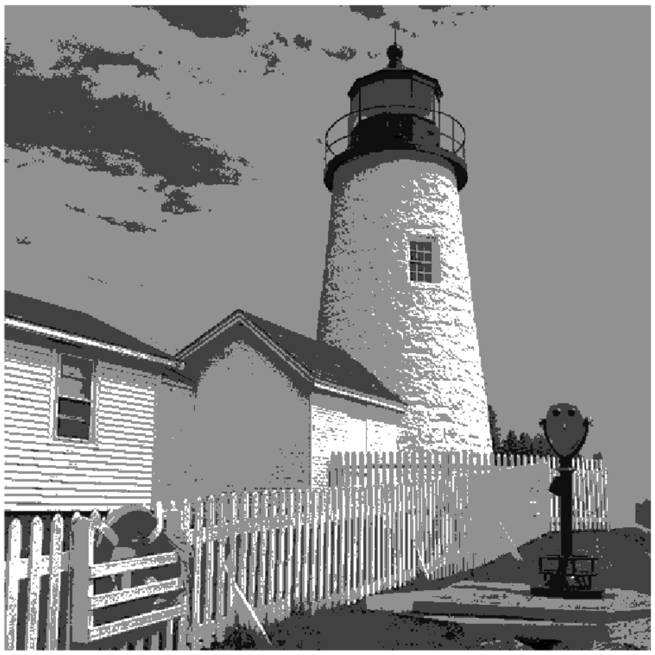
b) Graph the histogram of the original image.

```
octave:4> Histogram(A)
```



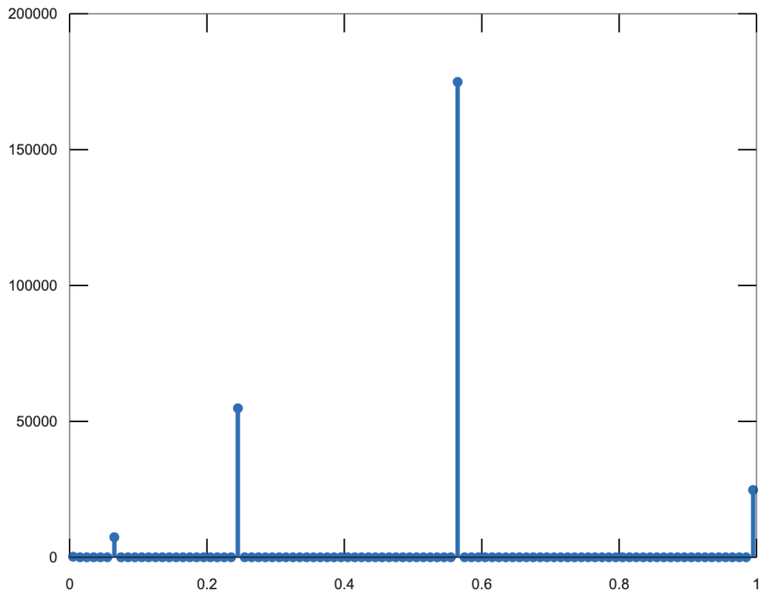
c) Quantize the original image using amplitude levels {0, 1/16, 1/4, 9/16, 1} (rounding-tonearest method). Graph the quantized image.

```
octave:5> Aq = Quantize(A, [0, 1/16, 1/4, 9/16, 1]);
octave:6> imshow(Aq)
```



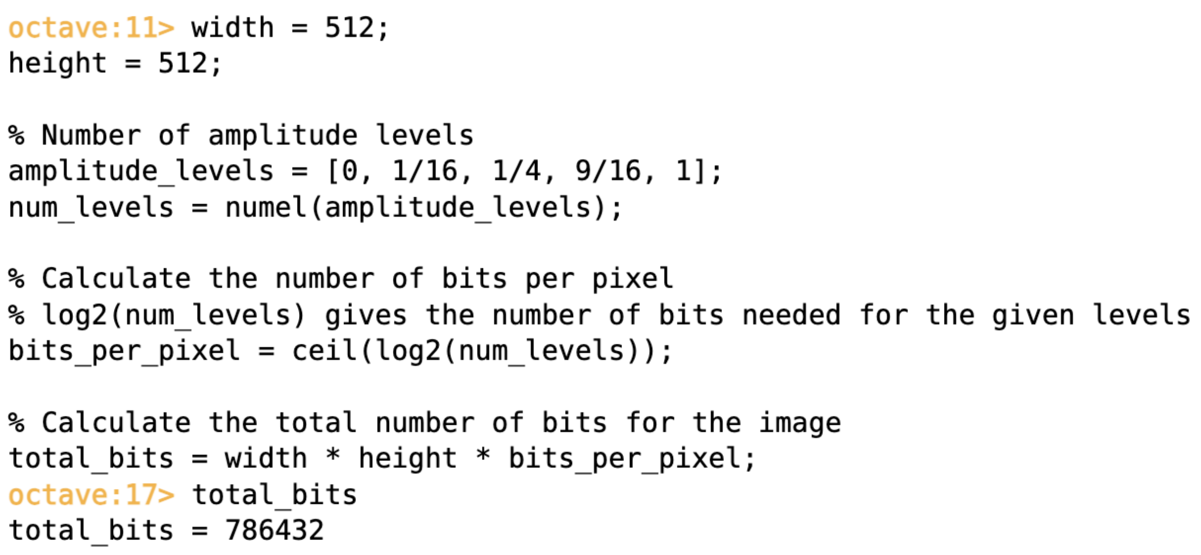
d) Graph the histogram of the quantized image.

```
octave:8> Histogram(Aq)
```



e) How many bits are required to store the quantized image using a fixed-length code to represent each pixel? Note that the image size is 512 × 512.

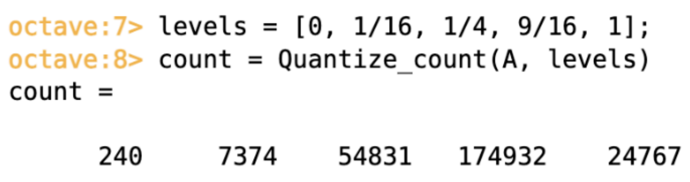
The total bits required to store the quantized images are 786,432.



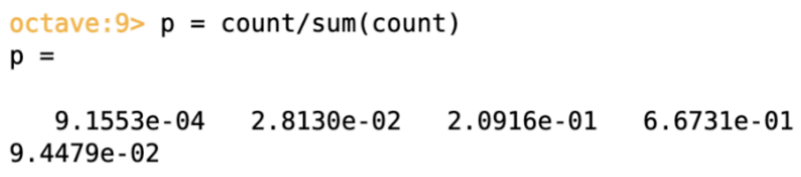
```
octave:11> width = 512;
height = 512;

% Number of amplitude levels
amplitude_levels = [0, 1/16, 1/4, 9/16, 1];
num_levels = numel(amplitude_levels);

% Calculate the number of bits per pixel
% log2(num_levels) gives the number of bits needed for the given levels
bits_per_pixel = ceil(log2(num_levels));

% Calculate the total number of bits for the image
total_bits = width * height * bits_per_pixel;
octave:17> total_bits
total_bits = 786432
```

f) How many bits are required to store the quantized image using a Huffman code to represent each pixel? Show the tree diagram used to calculate Huffman codewords. (For this you will need to estimate the probability of occurrence of each amplitude level in the quantized image.)

Estimate the probabilities of each quantization level:



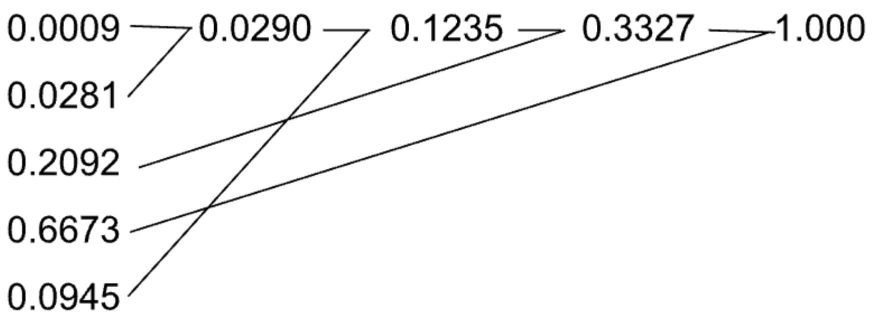
```
octave:7> levels = [0, 1/16, 1/4, 9/16, 1];
octave:8> count = Quantize_count(A, levels)
count =

      240     7374    54831   174932    24767
```



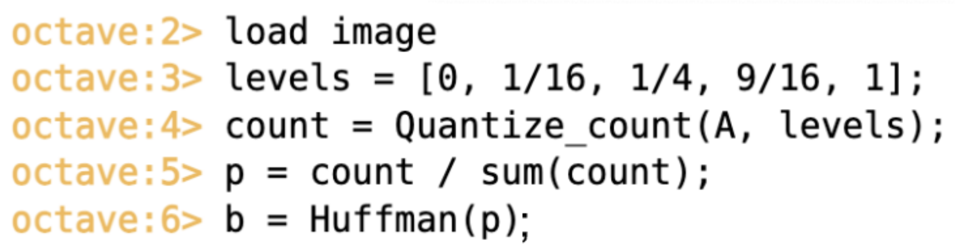
```
octave:9> p = count/sum(count)
p =

   9.1553e-04   2.8130e-02   2.0916e-01   6.6731e-01
9.4479e-02
```

Huffman Tree diagram to determine the number of codeword bits per amplitude:

# codebits	codeword
4	0000
4	0001
2	01
1	1
3	001



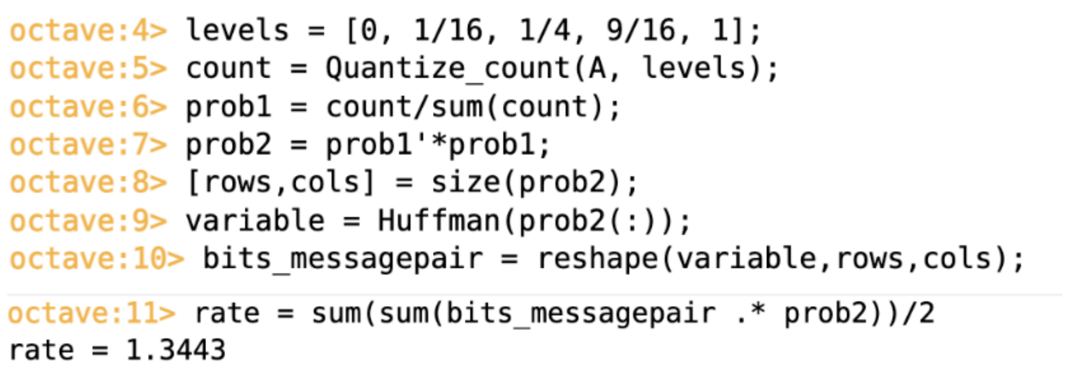
The average number of bits per Huffman codeword:
0.009(4) + 0.0281(4) + 0.2092(2) + 0.6673(1) + 0.0945(3) = 1.4852 bits



```
octave:2> load image
octave:3> levels = [0, 1/16, 1/4, 9/16, 1];
octave:4> count = Quantize_count(A, levels);
octave:5> p = count / sum(count);
octave:6> b = Huffman(p);
```

The number of bits required to store a quantized image using a Huffman code:
1.4852 x 512 x 512 = 389,337 bits

g) How many bits are required to store the quantized image using a Huffman code to represent pixel pairs? In this case, no tree diagram is required – find a solution using the provided Huffman function



```
octave:4> levels = [0, 1/16, 1/4, 9/16, 1];
octave:5> count = Quantize_count(A, levels);
octave:6> prob1 = count/sum(count);
octave:7> prob2 = prob1'*prob1;
octave:8> [rows,cols] = size(prob2);
octave:9> variable = Huffman(prob2(:));
octave:10> bits_messagepair = reshape(variable,rows,cols);
octave:11> rate = sum(sum(bits_messagepair .* prob2))/2
rate = 1.3443
```

.

The bits required to store the quantized image using a Huffman code to represent pixel pairs is
Rate x 512 x 512 = 1.3443 x 512 x 512 = 352,401 bits

h) If one assumes that the quantized image found in (c) is the result of sampling a discrete random variable X, what is the entropy of X (in bits)? For this, you must estimate the probability of occurrence of each amplitude level in the quantized image.

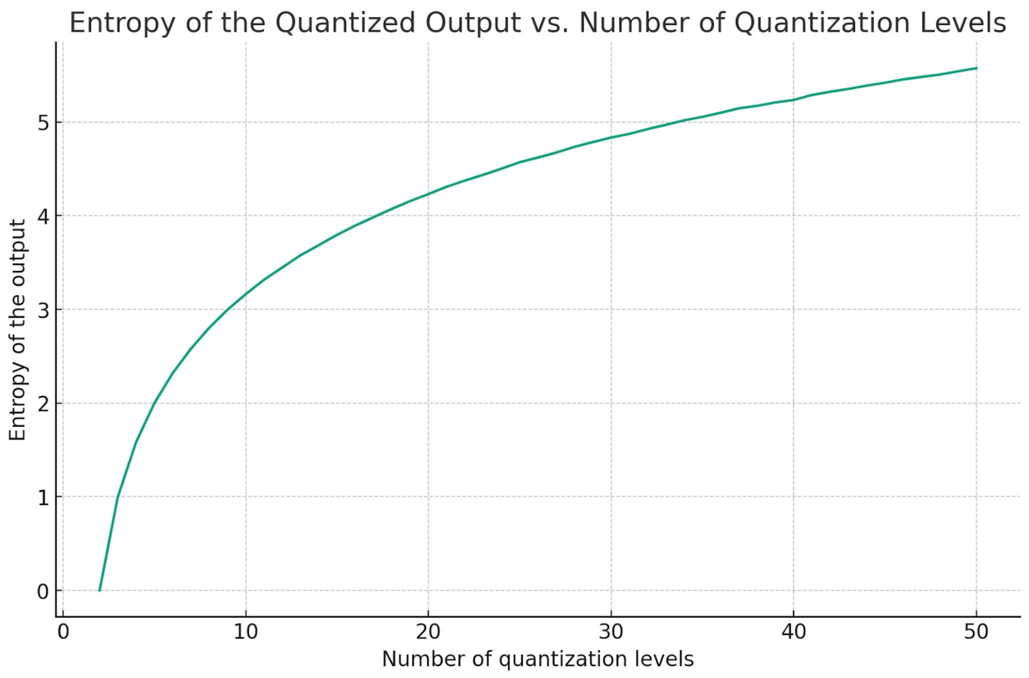
```
octave:35> % Given probabilities for
each quantization level
p = count/sum(count);
octave:36> % Calculate the entropy
H = -sum(p .* log2(p))
H = 1.3373
```

i) Summarize the results of parts (e)-(h) in a table.

	Description	# bits needed for 512 x 512 image
e	total bits required to store the quantized image	786,432
f	bits required to store the quantized image using a Huffman code to represent each pixel	389,337
g	bits required to store the quantized image using a Huffman code to represent pixel pairs?	352,401
h	Information content	350,566

j) Following the approach in part (h), calculate the "entropy of the quantized image" H (in bits) vs. number of quantization levels L, which are found using MATLAB function linspace(0,1,L), for $2 \le L \le 50$. Plot a graph of H(L) vs. L.

```
octave:24> levels = [0, 1/16, 1/4, 9/16, 1];
count = Quantize_count(A, levels);
prob1 = count/sum(count);
prob2 = prob1'*prob1;
[rows,cols] = size(prob2);
variable = Huffman(prob2(:));
bits_messagepair = reshape(variable,rows,cols);
rate = sum(sum(bits_messagepair .* prob2))/2;
H = [];
l = 2:50;
for n = 1:length(l)
    levels = linspace(0,1,l(n));
    count = Quantize_count(A, levels);
    p = count/sum(count);
    H = [H, Entropy(p)];
end
plot (l, H)
xlabel('# of quantization levels')
ylabel('entropy of the output')
```



Conclusion:

In this lab, I explored the application of probability theory to image processing by working with a 512×512 grayscale image in Octave-online. I started by displaying the original image and its histogram, then quantized the image to simplify its grayscale values. This step was essential for understanding the effects of quantization on image quality and data representation. I compared fixed-length and Huffman coding to assess the efficiency of different compression methods, discovering the superiority of Huffman coding in reducing storage size. Additionally, I calculated the entropy of the quantized image to understand the theoretical limits of compression. By examining the relationship between entropy and the number of quantization levels, I gained insights into balancing image fidelity and compression efficiency. This lab reinforced my understanding of image processing and compression, highlighting the practical applications of probability theory in this field.