Joshua Lai
804-449-134

1) Write an algorithm (pseudocode) capable of reducing the number of gray levels in an image from 256 to 2, in integer powers of 2. The desired number of gray levels needs to be a variable input to your code.

```
%import image
image = imread('image_name.png')

%convert image to to grayscale image
grayImage = rgb2gray(image)

%grab user desired number of gray levels
prompt = 'Desired number of gray levels?';
grayLevels = input(prompt)

%use switch statement for each gray level desired and definition of ranges for gray levels
switch grayLevels
    case 128
        for %each pixel in image
            if %value at pixel is 0 or 1
                %assign this gray level
            elseif %value at pixel is 2 or 3
                %assign this gray level
            elseif %value at pixel is 4 or 5
                %assign this gray level
            %etc
    case 64
        for %each pixel in image
            if %value at pixel falls in range
                %assign this gray level
            elseif %value at pixel is another range
                %assign this gray level
            %etc
    case 32
    %etc
end
```

Another algorithm that could have been used as demonstrated by the TA was to divide the image by the number of gray levels requested:

$$reducedImage = \frac{importedImage}{grayLevels} \tag{1}$$

This results in the image being split by the number of bits in the gray level requested.

2a) Give a continuous function for implementing the contrast stretching transformation shown in Fig. 3.2(a). In addition to k, your function must include a parameter, E, for controlling the slope of the function as it transitions from low to high gray-level values. Your function should be normalized (between 0 and 1 or 0 and 255).

From the textbook, we can see the following:

1. If $r < k$, then the slope will be increasing.
2. If $r = k$, then the slope is infinite and $s = \frac{1}{2}$
3. If $r > k$, then the slope will be decreasing.

Based on the hint that was given by the TA, we can deduce that the continuous function for implementing the contrast stretching transformation is:

$$s = \frac{1}{1 + (\frac{k}{r})^E} \tag{2}$$

2b) Sketch (plot) a family of transformations function of E, for a fixed value k = L/2, where L is the number of gray levels in the image. Include the transformation that will output a binary image.

3) Suppose that a digital image is subjected to histogram equalization. Show that a second pass of histogram equalization will produce exactly the same result as the first pass.

Let $n_k$ be the number of pixels in the input image associated with the gray level $r_k$ and let **M** be the number of rows and **N** be the number of columns in the image. Let $k = [0, L-1]$. From lecture, we know that the discrete form of histogram equalization is:

$$\begin{aligned} s_k &= T(r_k) \qquad (3) \\ &= (L-1)\sum_{j=0}^{k} p_r(r_j) \qquad (4) \\ &= \frac{(L-1)}{\mathbf{MN}}\sum_{j=0}^{k} n_j \qquad (5) \end{aligned}$$

where **MN** is the total number of pixels in the image and $n_j$ is the number of pixels at gray level $r_j$. Next, we apply the histogram equalization on $s_k$. We will label the result as $x_k$.

$$\begin{aligned} x_k &= T(s_k) \qquad (6) \\ &= (L-1)\sum_{l=0}^{k} p_r(r_l) \qquad (7) \\ &= \frac{(L-1)}{\mathbf{MN}}\sum_{l=0}^{k} n_l \qquad (8) \end{aligned}$$

By comparing the two results, we can see $s_k = x_k$. Therefore, applying histogram equalization a second time will produce the same result as being applied once.

4a) Write a computer program for computing the histogram of an image.

```
%%Import image
importImage = imread('Homework_1.jpg');

%Find the size of the image
[M N] = size(importImage);

vector = zeros(256, 1);

for i = 0:255
    for j = 1:M
        for k = 1:N
            if i == importImage(j, k)
                vector(i + 1) = vector(i + 1) + 1;
            end
        end
    end
end

figure, plot(vector), title('Imported Image (Non-Equalized)');
xlabel('Gray Level');
ylabel('# of Pixels');
```

4b) Implement the histogram equalization technique discussed in the class and in Section 3.3.1 of the textbook.

Continuing from the previous code:

```
transition = vector/numel(importImage);

cSum = cumsum(transition);
histogram_eq = cSum(importImage + 1);
histogram_eq = uint8(histogram_eq * 255);

figure, plot(histogram_eq), title('Imported Image (Equalized)');
```

4c) Download Fig. 3.8(a) and perform histogram equalization on it (the MRI of a fractured human spine).

**Please see attached sheets with images and results.**