CSCI 576 Assignment 2
Instructor: Parag Havaldar
Assigned on 09/24/2018
Solutions due 10/15/2018 at 12:00 pm afternoon
Late policy – None!

Theory Part *(40 points)*

Question 1: Color Theory *(10 points)*

1. The chromaticity diagram in (x, y) represents the normalized color matching functions X, Y and Z. Prove that *(2 points)*

$$Z = [\ (1-x-y)/y\ ]\ Y$$

2. Here you are tasked with mapping the gamut of a printer to that of a color CRT monitor. Assume that gamuts are not the same, that is, there are colors in the printer's gamut that do not appear in the monitor's gamut and vice versa. So, in order to print a color seen on the monitor you choose the nearest color in the gamut of the printer. *(8 points)*

- Comment (giving reasons) whether this algorithm will work effectively? *(2 points)*
- You have two images – a cartoon image with constant color tones and a real image with varying color tones? Which image will this algorithm perform better – give reasons. *(2 points)*
- Can you suggest improvements rather than just choosing the nearest color? *(4 points)*

Question 2: Generic Compression *(10 points)*

Consider a source that emits an alphabet with three symbols A, B, C having probability distributions as follows - P(A) = 0.625, P(B) = 0.125, P(C) = 0.25

- Construct a Huffman code and calculate its average code length. *(2 points)*
- For this three-symbol vocabulary, how many Huffman codes are possible. What are they? *(2 points)*
- Is the code you have defined optimal - give reasons! If not, what can you do to improve upon this. Show a solution by an example computing the avg. code length. *(6 points)*

This study resource was shared via CourseHero.com

Question 3: Entropy Coding *(10 points)*

Consider a communication system that gives out only two symbols X and Y. Assume that the parameterization followed by the probabilities are $P(X) = x^k$ and $P(Y) = (1-x^k)$

- Write down the entropy function and plot it as a function of x for k=2. (*1 points)*
- From your plot, for what value of x with k=2 does H become a minimum? (*1 points)*
- Your plot visually gives you the minimum value of x for k=2, find out a generalized formula for x in terms of k for which H is a minimum *(3 points)*.
- From your plot, for what value of x with k=2 does H become a maximum? (*1 points)*
- Your plot visually gives you the maximum value of x for k=2, find out a generalized formula for x in terms of k for which H is a maximum *(4 points)*.

Question 4: DCT Coding *(10 points)*

In this question you will try to understand the working of DCT in the context of JPEG. Below is an 8x8 luminance block of pixel values and its corresponding DCT coefficients.

188	180	155	149	179	116	86	96
168	179	168	174	180	111	86	95
150	166	175	189	165	101	88	97
163	165	179	184	135	90	91	96
170	180	178	144	102	87	91	98
175	174	141	104	85	83	88	96
153	134	105	82	83	87	92	96
117	104	86	80	86	90	92	103

- Using the 2D DCT formula, compute the 64 DCT values. Assume that you quantize your DCT coefficients uniformly with Q=100. What does your table look like after quantization? *(3 points)*
- In the JPEG pipeline, the quantized DCT values are then further scanned in a zigzag order. Show the resulting zigzag scan when Q=100, *(1 point)*.
- For this zigzag AC sequence, write down the intermediary notation *(2 points)*
- Assuming these are luminance values, write down the resulting JPEG bit stream. For the bit stream you may consult standard JPEG VLC and VLI code tables. You will need to refer to the code tables from the ITU-T JPEG standard which also uploaded with your assignment. *(2 points)*
- What compression ratio do you get for this luminance block? *(2 points)*

This study resource was shared via CourseHero.com

https://www.coursehero.com/file/35042605/Fall2018-Assignment2pdf/

Programming Part on Vector Quantization *(160 points)*

This assignment will increase your understanding of image compression. We have studied JPEG compression in class, which works by transforming the image to the frequency domain and quantizing the frequency coefficients in that domain. Here you will implement a common but contrasting method using "vector quantization". Quantization or more formally scalar quantization, as you know, is a way to represent (or code) one sample of a continuous signal with a discrete value. Vector quantization on the contrary codes a group or block of samples (or a vector of samples) using a single discrete value or index.

Why does this work, or why should this work? Most natural images are not a random collection of pixels but have very smooth varying areas – where pixels are not changing rapidly. Consequently, we could pre-decide a codebook of vectors, each vector represented by a block of two pixels (or four pixels etc.) and then replace all similar looking blocks in the image with one of the code vectors. The number of vectors, or the length of the code book used, will depend on how much error you are willing to tolerate in your compression. More vectors will result in larger coding indexes (and hence less compression) but results are perceptually better and vice versa. Thus, vector quantization may be described as a lossy compression technique where groups of samples are given one index that represents a code word. In general, this can work in *n* dimensions, we will start your implementation to two dimensions to perform vector quantization on an image and later extend it to higher dimensions. Your program will be invoked as follows

MyCompression.exe myImage.ext N mode

Your result should present images side by side – original and output. The output would be computed depending on three parameters:

- An input image, all images are standard CIF size 352x288 and supplied in two formats – *myImage.raw* (if it is a single channel 8 bit per pixel image) or *myImage.rgb* (if it is a color three channel 24 bits per pixel image). Please refer to the readme with the images for the format. Your code in assignment 1 can be used to display these images.
- *N* which gives you several vectors for quantization, so that each vector in the input can ultimately use $log_2 N$ bits. Expect this input to be a power of 2.
- *Mode,* which suggests how pixels should be grouped to form vectors. For this assignment, these are the following values the variable can take
    - *1 – suggesting two side by side pixels (whether gray or color) form a vector*
    - *2 – suggesting a 2x2 block of pixels (whether gray or color) form a vector*
    - *3 – suggesting a 4x4 block of pixels (whether gray or color) form a vector*

For the sake of understanding, the following description outlines in detail the working for a 2-dimensional vector where each vector is composed of two side by side pixels for a gray image. i.e. for the case:

MyCompression.exe myImage.raw 16 1

Here are the steps that you need to implement to compress an image for such a condition.

1. Understanding your two pixel vector space to see what vectors your image contains
2. Initialization of codewords - select N (16 in this case) initial codewords

This study resource was shared via CourseHero.com

https://www.coursehero.com/file/35042605/Fall2018-Assignment2pdf/

3. Clustering vectors around each code word
4. Refine and Update your code words depending on outcome of step 3.

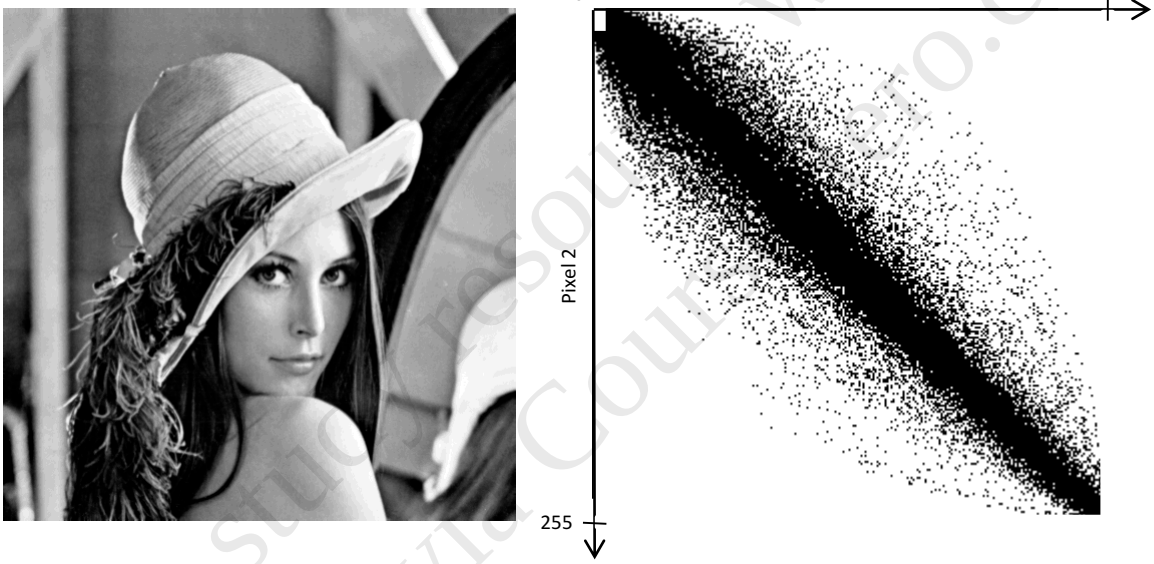
Repeat steps 3 and 4 until code words don't change or the change is very minimal.

5. Quantize input vectors to produce output image

Each of these steps are further explain below

Step 1 - *Understanding your vector space:*

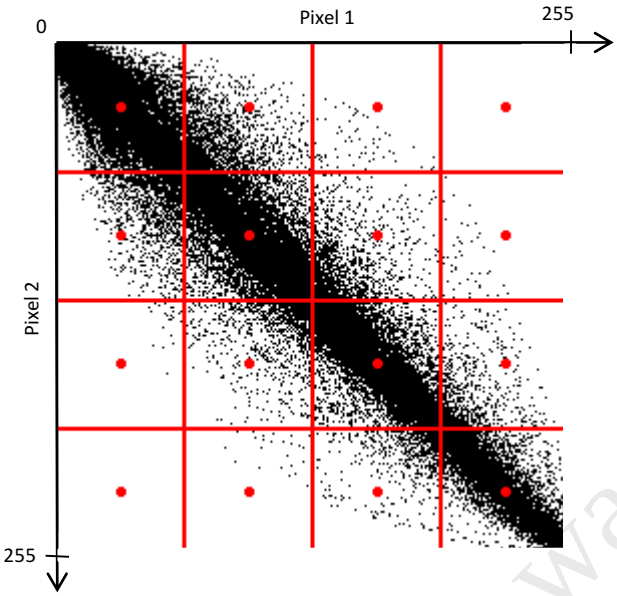
Let's look at the traditional Lena image below on the left. When creating a code book, we need to decide two things – the size of the vector and the number of vectors. For this example, we have a vector space defined by two adjacent pixels and the number of codebook vectors is 16. The space of all possible two-pixel vectors (where each pixel is 8 bits) is 256x256 vector space. The right image shows a plot of which 2 pixels vectors are present in the Lena image. Here a black dot shows a vector [pixel1, pixel2] being used in the image. This naturally is a sparse set because all combinations of [pixel1, pixel2] are not used.



Step 2 - *Initialization of codewords*

Next, we need to choose the *N* vectors of two pixels each that will best approximate this data set, noting that a possible best vector may not necessarily be present in the image but is part of the space. We could initialize codewords using a heuristic, which may help cut down the number of iterations of the next two steps or we may choose our initial codewords in a random manner. The figure below shows a uniform initialization for N=16 where the space of vectors is broken into 16 uniform cells and the center of each cell is chosen as the initial codeword.

https://www.coursehero.com/file/35042605/Fall2018-Assignment2pdf/



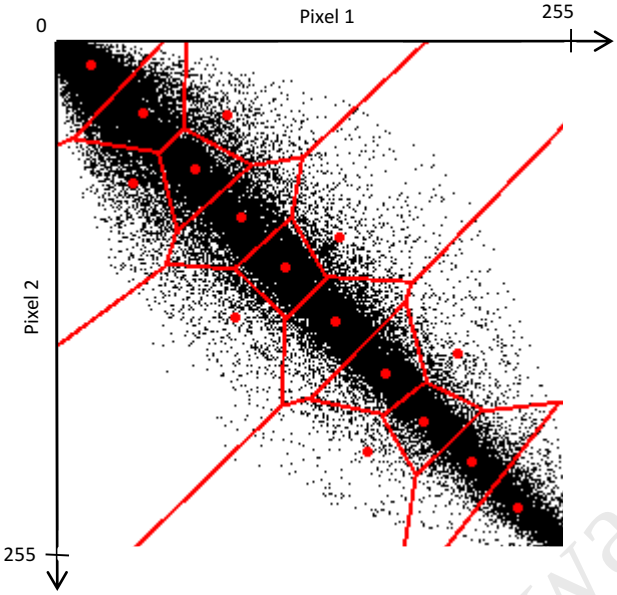
Every pair from the input image pixels would be mapped to one of these red dots during the quantization. In other words - all vectors inside a cell would get quantized to the same codebook vector. Thus, compression is achieved by mapping 2 pixels in the original image to an index which needs log(N) bits or 4 bits in this example. In other words, we are compressing down to 2 bits per pixel.

While the compression ratio is good, we see why this quantization is very inefficient: Two of the cells are completely empty and four other cells are very sparsely populated. The codebook vectors in the six cells adjacent to the x = y diagonal are shifted away from the density maxima in their cells, which means that the average quantization error in these cells will be unnecessarily high. Thus, six of the 16 possible pairs of pixel values are wasted, six more are not used efficiently and only four seem probably well used. This results in large overall quantization error for all codewords, also known as the mean quantization error. The next steps aim to reduce this overall mean quantization error.

Step 3 and 4:

The figure below shows a much better partitioning and assignment of codewords, which is how vector quantization should perform. The cells are smaller (that is, the quantization introduces smaller errors) where it matters the most—in the areas of the vector space where the input vectors are dense. No codebook vectors are wasted on unpopulated regions, and inside each cell the codebook vector is optimally spaced with regard to the local input vector density.

This study resource was shared via CourseHero.com



This is done iteratively performing steps 3 and 4 together. Normally, no matter what your starting state is, uniformly appointed codewords or randomly selected codewords, this step should converge to the same result.

In step 3, you want to clusterize all the vectors, ie assign each vector to a codeword using the Euclidean distance measure. This is done by taking each input vector and finding the Euclidean distance between it and each codeword. The input vector belongs to the cluster of the codeword that yields the minimum distance.

In step 4 you compute a new set of codewords. This is done by obtaining the average of each cluster. Add the component of each vector and divide by the number of vectors in the cluster. The equation below gives you the updated position of each codeword y_i as the average of all vectors x_{ij} assigned to cluster i, where m is the number of vectors in cluster i.

$$y_i = \frac{1}{m}\sum_{j=1}^{m} x_{ij}$$

Steps 3 and 4 should be repeated, updating the locations of codewords and their clusters iteratively until the codewords don't change or the change in the codewords is small.

Step 5 – *Quantize input vectors to produce output image:*

Now that you have your code vectors, you need to map all input vectors to one of the codewords and produce your output image. Be sure to show them side by side.

This study resource was shared via CourseHero.com

How will you be graded:
We will be conducting many tests. For each test we may use a raw image or a color rgb image. For each setup we will be changing N, but you may assume it to be a power of 2. Also for each setup we will change the mode to control the vectors.
For gray images, you will have to produce an output that displays the best N gray vector blocks (whether side by side, 2x2 or 4x4). Note that all the pixels in a vector need not have the same gray values.
For color images, you will have to produce an output that displays the best N color vector blocks (whether side by side, 2x2 or 4x4). Note that all the pixels in a vector need not have the same color.

As output please show two images – the original and your quantized result – side by side.

Example invocations include:

MyCompression.exe myImage.rgb 16 1
MyCompression.exe myImage.raw 32 2
MyCompression.exe myImage.rgb 8 1
MyCompression.exe myImage.rgb 16 2
MyCompression.exe myImage.raw 64 3

What should you submit?

- Your source code, and your project file or makefile. Please confirm submission procedure from the TAs. Please do not submit any binaries or data sets. We will compile your program and execute our tests accordingly.
- Along with the program, also submit an electronic document (word, pdf, pagemaker etc) for the written part and any other extra credit explanations.

This study resource was shared via CourseHero.com

https://www.coursehero.com/file/35042605/Fall2018-Assignment2pdf/