Written Part:

Q.1 - Suppose a camera has 450 lines per frame, 520 pixels per line with a color sub sampling scheme is 4:2:0. The camera has a frame rate of 25 Hz, and each sample of Y, Cr, Cb is quantized with 8 bits

- What is the bit-rate produced by the camera? (2 points)

>> Bit Rate = sampling * quantization
For subsampling scheme 4:2:0 => 4*8 = 32 bits for Y, 2*8 = 16 bits for U and 0*8 = 0 bits for V.
Number of bits per pixel is (32+16+0)/4 = 12
Bit rate = 450 * 520 * 12 * 25 = 70,200,000 = 70.2 MBPS

- Suppose we want to store the video signal on a hard disk, and, in order to save space, re-quantize each chrominance (Cr, Cb) signals with only 6 bits per sample. What is the minimum size of the hard disk required to store 10 minutes of video (3 points)

>> For subsampling scheme 4:2:0 and 6 bits per sample for Cr and Cb => 4*8 = 32 bits for Y, 2*6 = 12 bits for U and 0*6 = 0 bits for V.
Number of bits per pixel is (32+12+0)/4 = 11
Space required for 10 minutes of Video = 450*520*11*25 * (60*10) = 38610000000 bits

Q.2 The following sequence of real numbers has been obtained sampling an audio signal: 1.8, 2.2, 2.2, 3.2, 3.3, 3.3, 2.5, 2.8, 2.8, 2.8, 1.5, 1.0, 1.2, 1.2, 1.8, 2.2, 2.2, 2.2, 1.9, 2.3, 1.2, 0.2, -1.2, -1.2, -1.7, -1.1, -2.2, -1.5, -1.5, -0.7, 0.1, 0.9 Quantize this sequence by dividing the interval [-4, 4] into 32 uniformly distributed levels (place the level 0 at -3.75, the level 1 at -3.5, and so on. This should simplify your calculations).

- Write down the quantized sequence. (4 points)

>> Quantized sequence :
2.0, 2.25, 2.25, 3.25, 3.5, 3.5, 2.5, 3.0, 3.0, 3.0, 1.5, 1.0, 1.25, 1.25, 2.0, 2.25, 2.25, 2.25, 2.0, 2.5, 1.25, 0.25, -1.0, -1.0, -1.5, -1.0, -2.0, -1.5, -1.5, -0.5, 0.25, 1.0

- How many bits do you need to transmit it? (1 points)

>> We divide the interval [¬4,4] into 32 uniformly distributed levels, so 5 bits are required to represent a level i.e. 5 bits to transmit a sequence and a total of 32x5 = 160 bits to transmit the sequence.

Q.3 Temporal aliasing can be observed when you attempt to record a rotating wheel with a video camera. In this problem, you will analyze such effects. Assume there is a car moving at 36 km/hr and you record the car using a film, which traditionally record at 24 frames per second. The tires have a diameter of 0.4244 meters. Each tire has a white mark to gauge the speed of rotation.

- If you are watching this projected movie in a theatre, what do you perceive the rate of tire rotation to be in rotations/sec? (5 points)

>> Speed = 36 km/hr = 10 m/s, Diameter (D) = 0.4244 m, Distance covered in 1 rotation (circumference) = Pi * D = 3.142 * 0.4244 = 1.333 m
Rate of tire rotation in rotations/second = 10/1.333 = 7.5 rotations/second

According to Nyquist theorem, minimum frame rate needed to capture same rotation is 2x7.5 = 15 fps. Recording was done at 24 fps > 15 fps.
Thus, we will perceive the rate of tire rotation as 7.5 rotations/second.

- If you use your camcorder to record the movie in the theater and your camcorder is recording at one third film rate (ie 8 fps), at what rate (rotations/sec) does the tire rotate in your video recording (5 points)

>> film rate is now at 8 fps < 15 fps given by Nyquist's theorem. Hence there will be aliasing.
The aliased frequency is given by = |R * n - fs|

R (film rate) = 8 fps
n = the closest integer multiple of the film rate and the rate of the signal being aliased = (8/7.5) = 1
fs = rate of signal being aliased = 7.5

Aliased frequency = | 8 x 1 - 7.5| = 0.5 rotations/sec.

Programming Part:

Java source Filename: imageReader.java
Compilation: javac imageReader.java
Invocation: java imageReader Image.rgb Y U V Q

Assumptions:
Height and Width of the image have been set to 288 and 352 respectively.

Classes created:

1. imageReader: main class used for reading input, subsampling, quantization, YUV/RGB conversion and Display functions.
2. YUV_Pixel: Inner helper class to store YUV values of a given pixel
3. RGB_Pixel: Inner helper class to store RGB values of a given pixel

Program Steps and Logic:

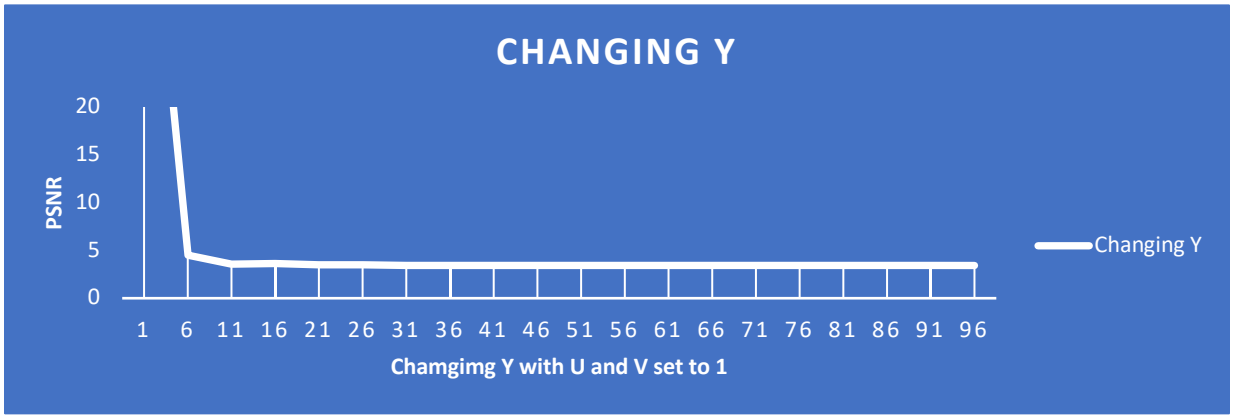
1.Read Input Image - Code is already given as part of the assignment. Modified to read R,G,B values and stores in a 2D array of RGB objects
2. Convert to YUV Space – Convert RGB to YUV from the given conversion matrix. Y,U and V values are stored in double data type
3. Process YUV Subsampling – Depending on the input parameters subsampling is done. If the input value is K, then i retain 1 value for every K values of Y,U and V if (J mod K) is equal to 0. We throw the rest of the Y, U and V values.
4. Upsampling – Since we are throwing away samples during subsampling, i filled the thrown values as follows - Pixel in position (I,J) is filled by the pixel value in the (I,J-k) if (J mod K) is not equal to 0. This is a filter which will have large artifacts. But a better algorithm will be used as part of Analysis Q2.
5. Convert back to RGB space – Convert YUV to RGB from the given conversion matrix.
6. Quantization of RGB channels – if Q is m then the function is designed such that there can be only m possible values for each of R, G and B. For example if Q = 8 => 3 bits per channel => 8 possible values => 0,31,63,95,,127,159,,191,223
7. Display both input and modified RGB image – Code already given as part of the assignment

Analysis Questions:

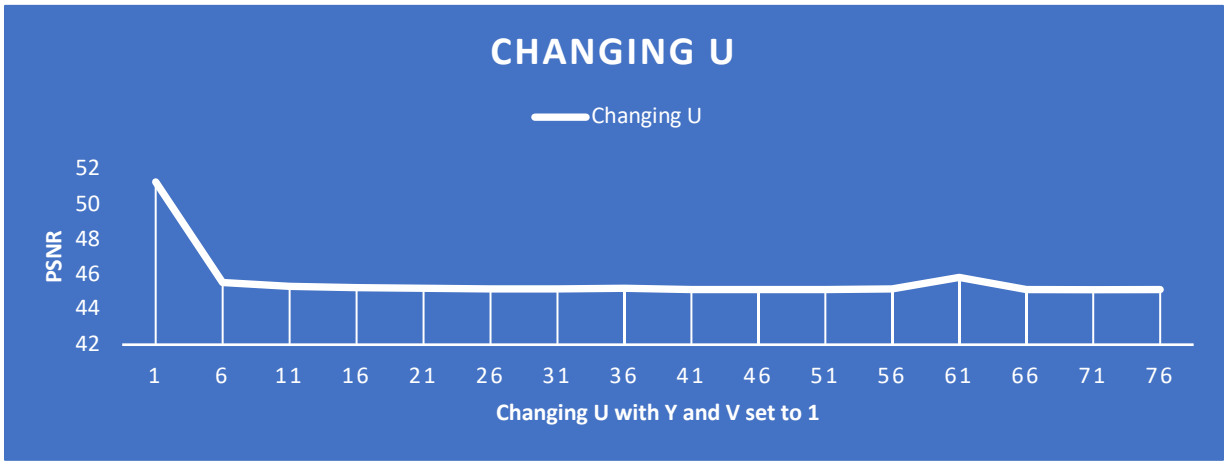
Q1: Subsampling obviously degrades the image quality. Here you are going to analyze this degradation. Can you formulate a way to quantify this degradation as an error number or error percentage? Explain how you are measuring your error. Compute this error value for output images where each input Y, U, V is individually varying while the other two remain constant. (Assume Q is constant at 256). Plot your error metric values for each.
>>
Reference: 1. http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/VELDHUIZEN/node18.html
2. https://stackoverflow.com/questions/26210055/psnr-of-image-using-matlab

I have used PSNR to analyze the degradation. PSNR is measured in decibels (dB). In most of the cases a higher PSNR generally indicates reconstruction with higher quality.

• Keep U and V constant at 1, and vary Y to compute different image outputs. For each compute your error metric and finally plot all these values to get a distortion curve.

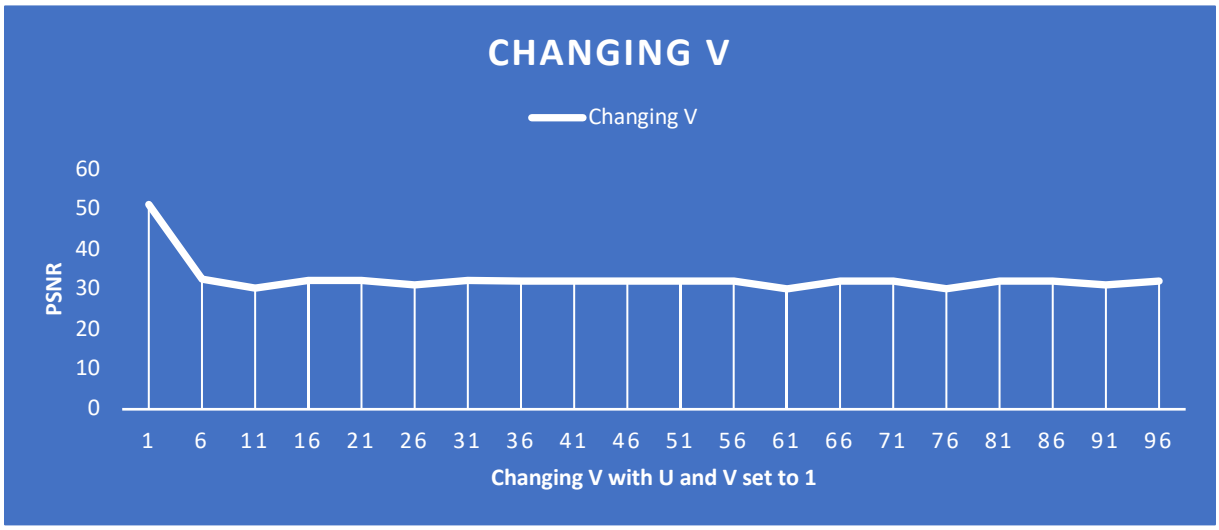


• Repeat the above process to compute your distortion curve keeping Y and V constant at 1 and vary U



.

• Repeat the above process to compute your distortion curve keeping Y and U constant at 1 and vary V.



Do you see any patterns in the curves? What conclusion(s) can you draw from your analysis?

>> As seen in above plots, PSNR drops rapidly as we increase the value of Y, indicating that the image gets blurred at a faster rate as we increase Y than increase in U and V.

Q2: Keeping Y constant at 1 (no subsampling), as you increase the value of U and V, the outputs generated by filtering in values for the missing Us and Vs start to produce visible artifacts.

• What artifacts do you see and explain what causes them?

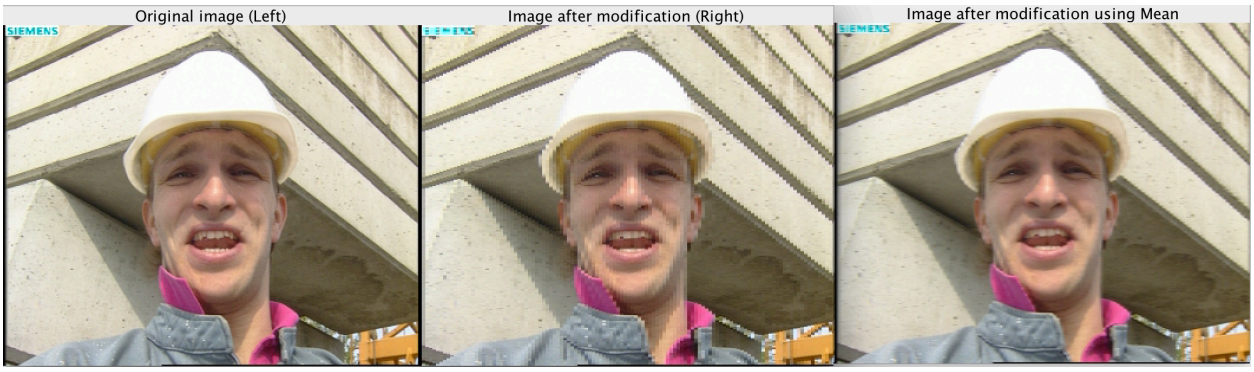
>> Image quality reduces when values of U and V are increased. Image is blurry. Aliasing is clearly Visible to human eye.

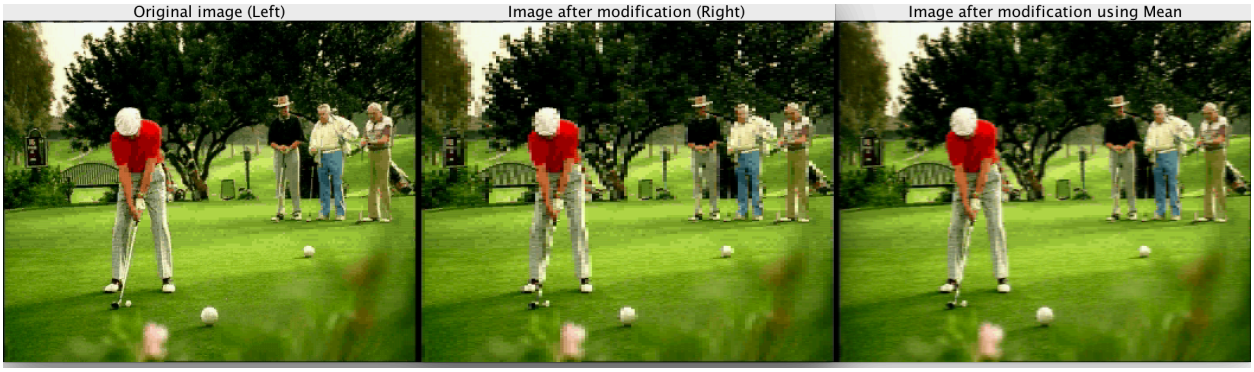
• Suggest a method to rectify or minimize these artifacts explaining your algorithm.

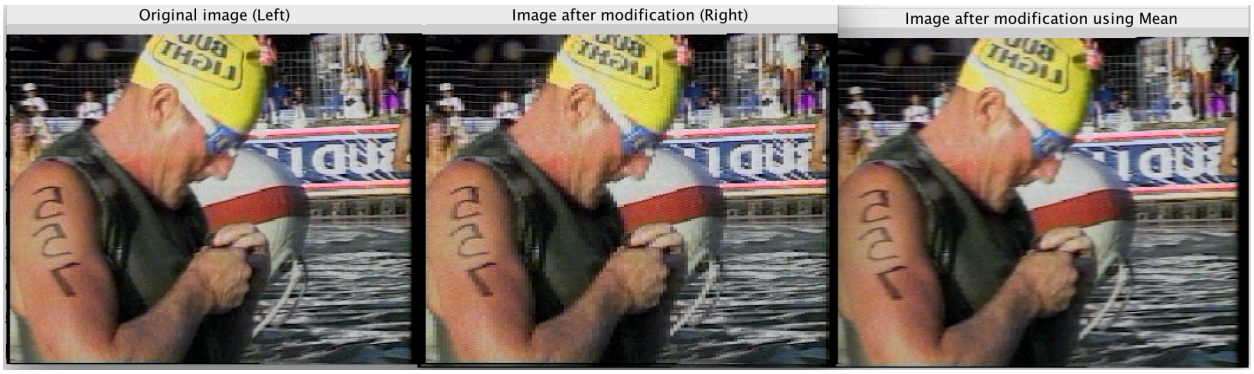
>> I have modified the subsampling algorithm and filled the thrown values by calculating the average of the neighborhood values. Pixel in position (I,J) is filled by ((I,J-1)+(I,J+1))/2 if (J mod K) is not equal to 0.

• Write a program to implement your idea and show results. You don't have to submit your program code but attach outputs for the given image data sets showing the images before your change and with the artifacts alongside images after your change which minimized the artifacts.

>> Below are all the three data sets – Left-> Original Image, Middle -> Subsampling without Mean, Right -> Interpolating missing values with mean of the neighborhood. Clearly Image quality is significantly improved when mean is used to fill the missing values.







Q3: For an input of Y=1, U=1, V=1 and Q=2, you have no sampling in Y U and V but have 2 values per color channel resulting in 8 distinct colors. However, the method suggested in the programming section produces colors which might not be the best causing the output image colors to look very different compared to the original colors. Research into and implement an algorithm to better the final quantized colors.

• Explain your algorithm or method of choice. Remember your algorithm needs to work for all possible values of Q, not just Q=2.

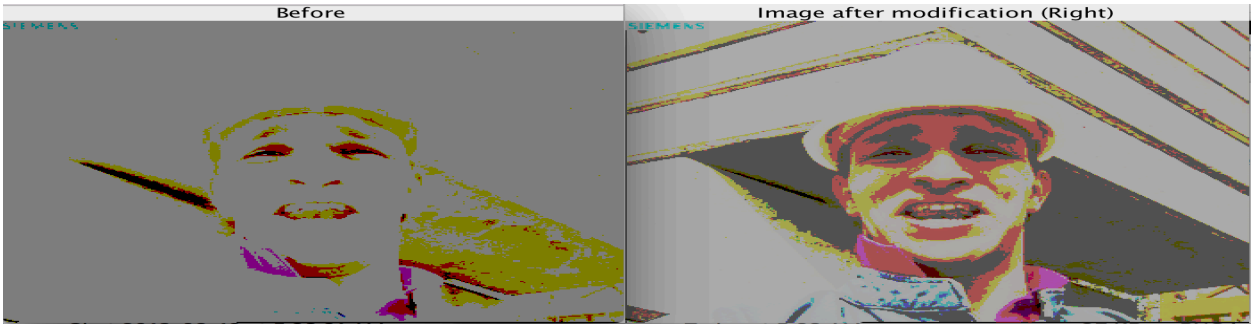
>> Algorithm:

Reference: 1. https://web.cs.wpi.edu/~matt/courses/cs563/talks/color_quant/CQindex.html
2. https://en.wikipedia.org/wiki/Median_cut

The median cut method attempts to put roughly equal numbers of pixels in each color cell. It repeatedly divides the space in planes perpendicular to one of the color axes. The plane is chosen to divide the largest side and to leave about half the pixels in each of the resulting regions.

Suppose we have an image with an arbitrary number of pixels and want to generate a palette of 16 colors. Put all the pixels of the image (that is, their RGB values) in a bucket. Find out which color channel (red, green, or blue) among the pixels in the bucket has the greatest range, then sort the pixels according to that channel's values. For example, if the blue channel has the greatest range, then a pixel with an RGB value of (32, 8, 16) is less than a pixel with an RGB value of (1, 2, 24), because 16 < 24. After the bucket has been sorted, move the upper half of the pixels into a new bucket. Repeat the process on both buckets, giving you 4 buckets, then repeat on all 4 buckets, giving you 8 buckets, then repeat on all 8, giving you 16 buckets. Average the pixels in each bucket and you have a color-set of 16 colors.

• Modify your code using your idea to quantize your output to the best colors choice of colors. Again, you don't have to submit your program code, but attach outputs for the given image examples showing the images before your change and after your implementation showing better colors. Do this for different values of Q, Q = 2, 3, 4
>>
For Q = 2, below is the output before and after the changes.



For Q =3,