CS 576 – Assignment 1
Instructor: Parag Havaldar

Assigned on Mon August 29, 2022,
Solutions due on Mon Sept 19 - by 4:00 pm afternoon
Late policies – None

Theory Part (30 points)

Question 1: (5 *points*)
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 by placing the level 0 at -3.75, the level 1 at -3.5, and so on and level 31 at 4.00. Also, remember that quantization should result in least error

- Write down the quantized sequence. *(4 points)*
- How many bits do you need to transmit it? *(1 points)*

Question 2: (*10 points*)
A high-definition film color camera has 1080 lines per frame, 1920 pixels per line, with a 24 Hz capture frame rate. Each pixel is quantized with 12 bits per channel during the quantization process. The capture pipeline employs the follow sequence

1. YUV 4:2:0 color subsampling scheme
2. An optional feature, to the signal to standard definition CIF (352x288)
3. An obligatory MPEG2 compression phase
4. Disk write with a varying disk write speed (12 to 36 Mbytes per second).

Answer the following questions

- If the second optional feature is off, what minimal compression ratio needs be achieved by the third compression step process? *(4 points)*
- If the second optional feature is turned on to produce CIF format, how does your previous answer change? *(3 points)*
- If original pixels were square, how do the pixel stretch with the second optional feature turned on.? *(3 points)*

The original image HD with an aspect ratio of 16:9. With the second optional feature turned on, this original image is converted to an image with a 11:9 aspect. In the conversion, how does the pixel aspect ratio change?

Question 3: (*15 points*)
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 records 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. (15 points)

- If you are watching this projected movie in a theater, what do you perceive the rate of tire rotation to be in rotations/sec? (*3 points*)
- 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 (*6 points*)
- The driver decides to participate in race, and buys tires that safely allow a max speed of 180 km/hr. What must be the diameter of the tire if no temporal aliasing needs to be witnessed in the recording? (*6 points*)

Programming Part (*120 points*)

This assignment will help you gain a practical understanding of Quantization and Subsampling to analyze how it affects visual media types like images and video.
We have provided you with a Microsoft Visual C++ project and a java class to display two images side by side (original on the left and a processed output on the right). Currently both left and right correspond to the same input image. You are free to any of these as a start.

Your program will be invoked using seven parameters where

YourProgram.exe C:/myDir/myImage.rgb Y U V S_w S_h *A*

- The first parameter is the name of the image, which will be provided in an 8 bit per channel RGB format (Total 24 bits per pixel). You may assume that all images will be of the same size for this assignment (HD size = 1920wx1080h), more information on the image format will be placed on the class website
- The next three parameters are ***integers*** control the subsampling of your Y U and V spaces respectively. For sake of simplicity, we will follow the convention that subsampling occurs only along the width dimension and not the height. Each of these parameters can take on values from *1* to *n* for some *n*, *1* suggesting no sub sampling and n suggesting a sub sampling by *n*
- The next two parameters are ***single precision floats*** S_w and S_h which take positive values < 1.0 and control the scaled output image width and height independently.
- Finally a integer *A* (0 or 1) to suggest that antialiasing (prefiltering needs to be performed). 0 indicates no antialiasing and vice versa

Example invocations shown below should give you a fair idea about what your input parameters do and how your program will be tested.

1. *YourProgram,exe image1.rgb 1 1 1 1.0 1.0 0*

No subsampling in the Y, U or V, and no scaling in w and h and no antialiasing, which implies that the output is the same as the input

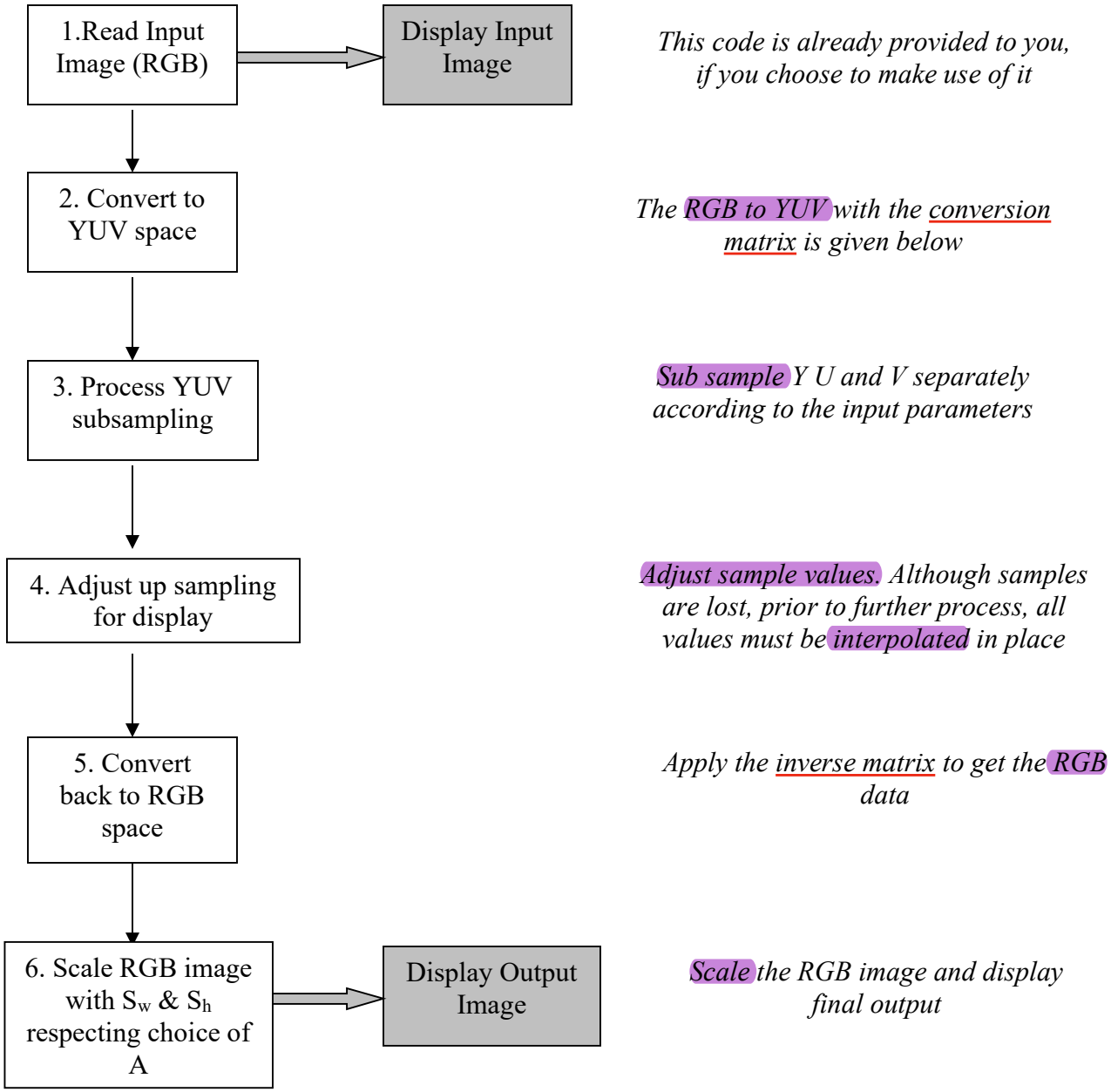
2. *YourProgram,exe image1.rgb 1 1 1 0.5 0.5 1*

No subsampling in Y, U or V, but the image is one fourth its original size (antialiased)

3. *YourProgram,exe image1.rgb 1 2 2 1.0 1.0 0*

The output is not scaled in size, but the U and V channels are subsampled by 2. No subsampling in the Y channels.

Now for the details - In order the display an image on a display device, the normal choice is an RGB representation. Here is the dataflow pipeline that illustrates all the steps.



Conversion of RGB to YUV
Given R, G and B values the conversion from RGB to YUV is given by a matrix multiplication

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312` \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Remember that if RGB channels are represented by n bits each, then the YUV channels are also represented by the same number of bits.

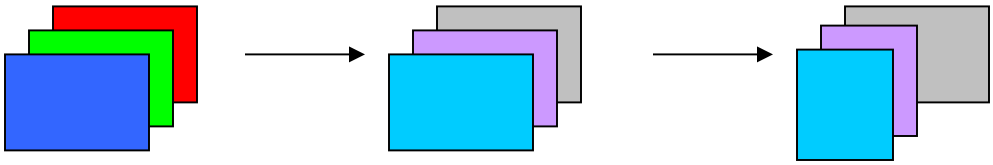
Conversion of YUV to RGB
Given R, G and B values the conversion from RGB to YUV is given by the inverse matrix multiplication

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.621 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ U \\ V \end{bmatrix}$$

Sub sampling of YUV & processing
Sub sampling, as you know will reduce the number of samples for a channel.
Eg for the input parameters

YourProgram.exe image1.rgb 1 2 2 256

In this example, the YUV image is not subsampled in Y, but by 2 in U and by 2 in V resulting in



When converting back to the RGB space, all the YUV channels have to be of the same size. However the sampling throws away samples, which have to be filled in appropriately by average the neighborhood values. For example, for the above case a local image area would look like

$Y_{11}U_{11}V_{11}$	Y_{12}	$Y_{13}U_{13}V_{13}$	Y_{14}	. . . . . line 1
$Y_{21}U_{21}V_{21}$	Y_{22}	$Y_{23}U_{23}V_{23}$	Y_{24}	. . . . . line 2
$Y_{31}U_{31}V_{31}$	Y_{32}	$Y_{33}U_{33}V_{33}$	Y_{34}	. . . . . line 3
$Y_{41}U_{41}V_{41}$	Y_{42}	$Y_{43}U_{43}V_{43}$	Y_{44}	. . . . . line 4

The missing values may be filled in using filters. Here is an example

$U_{12} = (U_{11} + U_{13})/2$ $\quad\quad$ $V_{12} = (V_{11} + V_{13})/2$
$U_{14} = (U_{13} + U_{15})/2$ $\quad\quad$ $V_{14} = (V_{13} + V_{15})/2$

Or you may choose to invent your own filter using appropriate valid neighborhood samples

$Y_{11}U_{11}V_{11}$	$Y_{12}U_{12}V_{12}$	$Y_{13}U_{13}V_{13}$	$Y_{14}U_{14}V_{14}$	line 1
$Y_{21}U_{21}V_{21}$	$Y_{22}U_{22}V_{22}$	$Y_{23}U_{23}V_{23}$	$Y_{24}U_{24}V_{24}$	line 2
$Y_{31}U_{31}V_{31}$	$Y_{32}U_{32}V_{32}$	$Y_{33}U_{33}V_{33}$	$Y_{34}U_{34}V_{34}$	line 3
$Y_{41}U_{41}V_{41}$	$Y_{42}U_{42}V_{42}$	$Y_{43}U_{43}V_{43}$	$Y_{44}U_{44}V_{44}$	line 4

Note the samples that you take to fill in values will change depending on the subsampling parameters. The YUV components can now be converted to RGB space.

Scaling with Antialiasing
Your output image width and height will change to a new (smaller) value depending on scale factors S_w and S_h. You will need to create the output image by resampling the input image. This can be achieved by inverse mapping all destination pixel indexes [i,j] to their source location indexes. Depending on whether you need to perform antialiasing the destination resampled pixel value can be the value of your inverse mapped source pixel (A=0) or the average of small neighborhood around the inverse mapped source pixel (A=1). To compute the average, you may use a 3x3 kernel.

What should you submit ?

- Your source code, and your project file or makefile, if any, using the submit program. ***Please do not submit any binaries or data files***. We will compile your program and execute our tests accordingly.
- Along with the program, also submit an electronic document (word, pdf, pagemaker etc format) using the submit program that answers the fore-mentioned analysis questions. You may use any (or all) input images for this analysis.