CS 576 – Assignment 1
Instructor: Parag Havaldar

Assigned on Monday 01/22/2024,
Solutions due on Monday 02/12/24 by midday 12 pm noon
Late Policy: None. Please don't submit multiple assignments. We will consider only the first one submitted.

PART 1: Theory Questions for practice (solutions will be provided)

Q.1 Suppose a camera has 450 lines per frame, 520 pixels per line, and 25 Hz frame rate. The color sub sampling scheme is 4:2:0, and the pixel aspect ratio is 16:9. The camera uses interlaced scanning, and each sample of Y, Cr, Cb is quantized with 8 bits.

- What is the bitrate produced by the camera?
- Suppose we want to store the video signal on a hard disk, and, 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

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.
- How many bits in total do you need to transmit it?

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. Assume that the tire rotates without skidding.

- If you are watching this projected movie in a theatre, what do you perceive the rate of tire rotation to be in rotations/sec?
- 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
- If you use an NTSC camera with 30 fps, what is the maximum speed that the car can go at so that you see no aliasing in the recording.

Programming Part (*100 points*)

This assignment will help you gain a practical understanding of *Spatial/Temporal Sampling* and *Filtering* in terms of how these processes affect visual media types like images and video. First, you need to know how to display images in the RGB format. We have provided starter code to read and display an image in three ways – a cross platform C++ visual studio code project, a windows Microsoft Visual Studio C++ project and cross platform java project. You are free to start with any one of your choice, a github pointer for these will be posted along with this assignment section. Since the intent of the assignments is for students to understand and implement details, we have a policy of not making use of programming environments that support libraries (such matlab, python, javascript etc.). Use of external libraries, including APIs that make the algorithmic implementation details a simple library function call are not allowed.

Your task is to start with an input image of size 512x512 and generate a video which will be rotating and zooming either into the image or out of the image. But the display size of your output will be just 512x512 while the content will change depending on the following four parameters as explained below:

- 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 of size 512 x 512. The .rgb file format stores the image in a particular order – first comes the red channel in scan line order, followed by the green and blue channels. Please refer to the reference code if needed.
- The second parameter will be a ***Z***oom speed value indicating how fast you are zooming in. This will give a zoom factor per second as a floating-point number, operates uniformly on width and height. Typical values may be.
    - 1.0 indicating no zoom per second.
    - 1.25 indicating zooming into the image at 1.25 magnification per second.
    - 0.5 indicating the image is reducing in size to 0.5 of the original per second.
- The third parameter will be a ***R***otation speed value in degrees per second. This will control by how much the image is rotating as the video zoom in (or out). It is a floating-point value which may be 0 (no rotation), positive (clockwise rotation) or negative (anti clockwise rotation). Typical values may be:
    - 0 indicating no rotation while zooming in (or out).
    - 15 indicating 15 degrees of clockwise rotation per second.
    - -25 indicating 25 degrees of anti-clockwise rotation second.
- The fourth parameter will be ***F***rames per second to display your video. This will be an integer value.

Expectations and Examples:
All parameters will have reasonable values. You should expect the following:

1. $0.50 < \boldsymbol{Z} < 2.00$
2. $-180.00 < \boldsymbol{R} < 180.00$
3. $1 < \boldsymbol{F} < 30$

Some things to note regarding your implementation output:

- We don't expect to have an "end" state to your video, which means the output is generated and displayed successively until your program is terminated. If rotating and zooming out, we expect the content to get smaller and smaller until it maps to a single center pixel or disappears. If rotating and zooming in, we expect the content to get bigger and bigger until you just see one big pixel.
- We expect your output frames to be anti-aliased using simple averaging filters. This mostly applies to resampled cases while zooming out and resolutions get smaller.
- When zooming in or out, you may have "empty" or "undefined" areas in frames where the original square image does not have a mapped definition. You may initialize these areas to all black or all white. See image examples below.

To invoke your program, we will compile it and run it at the command line as

YourProgram.exe C:/myDir/myImage.rgb Z R F

where *Z R F* are the parameters as described above. Example invocations and their expected outputs are explained below which should give you a fair idea about what your input parameters do and how your program will be tested.

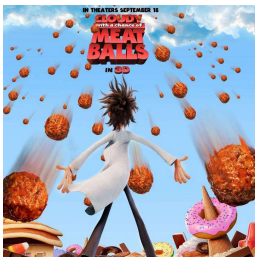
1. *YourProgram.exe C:/myDir/myImage.rgb 1.0 0.0 30*

Here the video will be formed with a zoom speed of 1.0 (no zoom), rotation speed of 0 degrees/second (no rotation) and displaying at 30 fps. Essentially you will see a video with an unchanging image.

2. *YourProgram.exe C:/myDir/myImage.rgb 1.25 30.0 1*

Here the video will be formed with a zoom speed of 1.25 per second, rotation speed of 30.0 degrees per second and displaying at 1 fps. Essentially you will see a video where the image is updating every second, each time rotated by 30 degrees clockwise and magnified by 1.25 of the original. , an example of the first **three frames** is given below, rest will continue in the same pattern

Frame 1 at 0 seconds



Frame 2 at 1 second, note how corners appear white because there is no data.

Frame 3 at 2 seconds

3. *YourProgram.exe C:/myDir/myImage.rgb 0.8 -45.0 10*

Here the video will be formed with a zoom speed of 0.8 (zooming out), rotating at -45.0 degrees per second and displaying at 10 fps. Essentially you will see a video where the image is updating 10 frames every second, each frame rotating by 4.5 degrees anti clockwise when compared to the previous and zoomed out by 0.02 of the original at each successive frame. An example of frames at second 0, 1, 2 etc. are shown below.

Frame 1 at 0 seconds

…

Frame 11 at 1 second, note how parts of the image are white because of undefined data.

…

Frame 21 at 2 seconds

Details of implementation:

Your goal is to display video with changing images each of size 512x512. Every image to be displayed is of size 512x512 and will need to be created and then shown on screen (rendered) depending on the input parameters. Each image completely defined or there may be pixels not defined eg when zooming out. These pixels may be given a default color (all black or all white).

Every pixel has an *RGB* value and a [*x, y*] pixel coordinate. These coordinates start at [0,0] in the upper left corner and end at [511, 511] lower right corner. For a new image at

a new frame, you will need to compute the RGB value for all the pixels [x,y] The matrices to zoom and rotate are given below, where w and h are the width and height of the image, z is the zoom factor and θ is the angle of rotation. Remember that your image's coordinates start from the upper left corner, you will need to offset the origin to the center of the image before you rotate and scale, else you will be rotating about the left corner and not the center of the image. So, if x, y are the coordinate of any pixel, the transformation looks as such

$$\begin{bmatrix} x\_original \\ y\_original \end{bmatrix} = \begin{bmatrix} x - w/2 \\ y - h/2 \end{bmatrix}$$

$$\begin{bmatrix} x\_transformed \\ y\_transformed \end{bmatrix} = \begin{bmatrix} z & 0 \\ 0 & z \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x\_origin \\ y\_origin \end{bmatrix}$$

$$\begin{bmatrix} x\_new \\ y\_new \end{bmatrix} = \begin{bmatrix} x\_transformed + w/2 \\ y\_transformed + h/2 \end{bmatrix}$$

To create anti-aliased smooth outputs (when zooming out and resolutions are becoming smaller), the output pixel values can be generated by averaging a 3x3 window of original source samples. Make sure you take care of corner cases accordingly.
Give some thought to your implementation. The original image has 512x512 pixels to begin with on the first frame. But the transformations on every subsequent frame may not map to areas outside of 512x512 (when zooming in) and or there may be pixels in a 512x512 frame that don't have define values, in which case initialize them to white or black.

What should you submit?

- Your source code, and your project file needed to compile the code. ***Please do not submit any binaries or images***. We will compile your program and execute our tests accordingly.
- If you need to include a readme.txt file with any special instructions on compilation, that is fine too.
- Please submit only one copy using DEN.