CMPT 365 Programming Assignment 2 Spring 2020

Due Date: Mar. 3, 2020 (Tues.)

Objective: Learn fundamentals for multimedia manipulation (using libraries such as OpenCV) and user interface design.

You are encouraged to work in pairs, with a singe submission for two students; however you may work individually if you like.

1. Assignment:

You are to construct a video-based program to help blind people. Assuming the blind are not hearing-impaired, we shall use the sense of hearing to help the brain construct a picture of what a sense of vision would convey.

Firstly, it is known that each ear can process only about 15kB/s of information, so we'll have to reduce the resolution of each video frame, and temporally subsample our video down to only one image per second or even less.

We shall use a small image of size only 64×64 pixels. To obtain this image from each video frame, create a luminance image via the (very rough) NTSC guide:

$$Y' \;=\; 0.299 * R' \;+\; 0.587 * G' \;+\; 0.114 * B'$$

where R', G', B' are gamma-corrected (i.e., ordinary) camera signals. We'll simply use this grey image Y', and abandon colour.

Now subsample your image down to 64×64 pixels.

Since the ear is limited in loudness and frequency, let's also reduce the intensity resolution of our image frame down to just 4 bits, making an image with only 16 levels. We'll tie the image *intensity* to the sound *loudness* that we generate.

We're trying to make a sound picture, and a simple way to do this is to associate *up* the y-axis with higher frequencies, and *down* with lower ones. Each column of our image has 64 pixels. So for the top pixel, use the highest frequency; the bottom pixel will have the lowest frequency.

To make a sound picture, proceed across the image by columns, 64 columns per second. For each column, produce a "chord" of sound, with each frequency sounded tied to the pixel position, and the loudness given by the pixel intensity. Note that this means that if 30 pixels of the column are black, say, then those 30 frequencies will not be present in the chord sounded.

At the end of every frame, sound an audible "click", to signal a new image starting.

An interesting extra feature would be to have the sound *pan* from left to right as you progress across the image, assuming we have two stereo audio channels.

Note: As a starting place, you could just use still images, or a sequence of still images, and then progress to video once you've developed your code. More and simpler possibilities are listed below in Section 2.

1.1. Pseudocode

As an example, here's some Matlab code for carrying out the assignment on a single frame — treat this as pseudocode.

```
im = imread('frame1.bmp');
im = rgb2gray(im);
im = imresize(im, [64, 64]);
im = im2double(im); % now in 0 .. 1.0
im = fix(im*16);    % truncate to int 0..15
im = im/16;         % back to float -- in 0 .. 1.0

Fs=8000; % sampling frequency
freq = zeros(64,1); % 64-vector
freq(32) = 440.0; % center frequencies around "A":
for m = 33 : 64
    freq(m) = freq(m-1) * 2^(1.0/12.0);
end
for m = 31:-1:1
    freq(m) = freq(m+1) * 2^(-1.0/12.0);
end


% If the sampling rate is Fs=8000Hz, and we wish each signal to play
% for only 1/64 s, then we must use only Fs/64=125 samples.  However,
% this sounds very short, so instead let's use:

N = 500; % so, 500/8000 = 0.0625 s;  and 64 columns will take 4 s.

tt=(1:N)/Fs; % time. == samples.

% Sound every column as a chord:
for col = 1 : 64
     signal = zeros(1,N);
         for row = 1:64

           m = 64-row+1; % row=1 is at the top, but that is "high" frequency.
           ss = sin(2*pi*freq(m)* tt );
           signal = signal + im(row,col) * ss;

     end % for row
     signal = signal / 64;  % since im(row,col) could be 1.0 -- is this the best idea?
     sound(signal, Fs);
end % for col

click = sin(2*pi*50* tt );
sound(click, Fs);
```

2. Stages in your answer

Remembering our "*Remarks on marking in CMPT365*"
http://www2.cs.sfu.ca/CourseCentral/365/mark/material/marking.html

you can approach this problem as a set of possible stages — the idea is that, starting from a mark of zero, you can work up toward a perfect score depending on how much you accomplish. Plus, as you work up you'll have your current effort all ready to hand in by the due date.

So in rough order, you might answer this assignment by

1. Make sure you clearly understand what the pseudocode is supposed to do. You could start by generating some test images using a tool like Photoshop, then run the code in MATLAB for simplicity and to run the pseudocode (MATLAB is free for SFU students).

2. The TA has already provided you with a skeleton program and a tutorial (in the .RAR file in "Skeleton Code" on our home page). Read the tutorial carefully. This will help you set up the programming environment we need for this assignment.

3. Once you become familiar with the environment, modify the skeleton program so that it works for our Hearing for the Blind problem.

4. There are some optional components in this assignment, as described in this document as well as the tutorial. If you are interested in coding, I recommend you implement a MIDI encoder (without using 3rd party libraries) and output the audio to a MIDI file. To clarify, my rationale is that you'd really learn something if you didn't use a canned library: so I was thinking you could use libraries that manipulate bits and bytes. For example, java.nio (https://docs.oracle.com/javase/7/docs/api/java/nio/package-summary.html).

   There is a package in the Java standard library
   (https://docs.oracle.com/javase/7/docs/api/javax/sound/midi/package-summary.html)
   that allows you to create MIDI files directly, so the idea would be not to use that. Of course, to make the code work do what you have to do finish on time. But here I'm making a hierarchy from simplest to most effort.

   If you are interested in UI design, I recommend you rebuild the GUI so that it looks better.

   If you are interested in theory, I recommend you change some of the acoustical parameters, e.g. make real music instead of simple freqencies.

3. Programming

The skeleton program for this assignment is written in Java. Java is one of the most important languages in use. It is reported (in 2013) that there are 9 million Java developers worldwide, more than twice the population of BC, so you should definitely learn Java.
(http://www.oracle.com/technetwork/articles/java/afterglow2013-2030343.html)

3.1 Libraries

Assignment 2 presents you with three basic problems: how to create a GUI; how to retrieve pixels from (decode) a video and process them as you wish; and how can you make a computer play the sound you wish to play. There are several solutions, some of which are introduced below and discussed comprehensively in the tutorial.

3.1.1 Video

We can use OpenCV (Open Source Computer Vision) and FFMPEG in this assignment.

As in Chapter 10 and 11 of the text, modern video encoding standards are very sophisticated. What this means is that it is very difficult to reconstruct the pixels and the frames from bits stored in a video file. Fortunately, some very clever coders from FFMPEG have already implemented the code that decodes the video. In this assignment, you can use FFMPEG to read in video files.

As you will see in the skeleton program, once you provide FFMPEG with a filename, it will generate a (sequence of) OpenCV matrices. Each matrix corresponds to a frame in the video, and each element in the matrix stores the colour of a pixel. You can use OpenCV to perform operations (resize, etc.) on these matrices.

In case you are using a Mac, the following is useful:
http://opencv-java-tutorials.readthedocs.io/en/latest/01-installing-opencv-for-java.html#install-opencv-3-x-under-macos

3.1.2 Sound

We are going to use Java Sound API in the skeleton file for this assignment.

As you have already learned in Chapter 6, a digital, single-channel audio file is composed of a sequence of samples. The index of a point determines when it will be played, and the sample value of a point determines its amplitude. To make a computer play the sound you wish it to play, you have to provide Java Sound API with that sequence of sampling points.

3.1.3 GUI

We are going to use JavaFX in the skeleton file for this assignment.

How can we create interactive components, such as buttons, in our program? In the past, one drew a square on the screen (with something like Borland Graphics Interface, graphics.h). Once the program detects (through interrupts) that the user clicks the mouse inside the square, it knows that the button is pressed.

Few people nowadays create a button like this. Instead, many people use JavaFX, which allows you to describe the GUI in XML, and register the logic that will be automatically executed once the user interacts with the GUI. Writing XML is not very straightforward, so we are going to use Scene Builder to help us.

3.2 Details / Sample Code

You are provided with a skeleton program written in Java. The skeleton program has a basic GUI. The user is able to open an image and play it. The tutorial attached to the skeleton program contains step-by-step instructions about how to set up the environment (JDK, eclipse, OpenCV, JavaFX, and JavaFX Scene Builder). It also teaches you how to modify the GUI and how to use OpenCV to process videos.

You don't have to use the skeleton program in this assignment, if you are not happy with it. However, please understand that the instructor and the TA may be unfamiliar with the particular technology (library) you choose. What this means is that we may not be able to provide you with all the help you need.

4. What to hand in

Hand in a short (2 – 3 page) written description in file cmpt365a2.pdf summarizing what you have done for your assignment, with material showing how your system works. Your assignment will be marked on (a) the functionality of your code; (b) the design of your user interface; and, to a lesser extent, (c) the discussions in cmpt365a2.pdf and the quality of optional features you implemented.

√	Yes	No
I used the skeleton program.		

√	Yes	No
My program can loads a video.		

√	Yes	No
The user is able to select whatever video she wants.		

√	Yes	No
There is a "click" sound between every two frames.		

√	Yes	No
I implemented some optional features.		

Please specify the optional features you implemented:

The CourSys submission configuration has 2 more components: a zipfile for your code; and, if any, an URL for a video if you like.

Since you may be submitting as a pair, only one set of submission components is uploaded per

group: make the fact that you are submitting as a pair clear on the title page of your report.

5. References

If you are interested in this subject applied "in-the-wild", have a listen to the interview after "To find out more about this technology and how it works..." in http://refractiveindex.wordpress.com/2011/03/18/sounds-of-science-seeing-with-sound/ , and see the commercial website http://www.seeingwithsound.com/ That website points to some sample YouTube videos:

`http://www.youtube.com/watch?v=v--bE_C-DXk&feature=related`

and

`http://www.youtube.com/watch?v=8xRgfaUJkdM`