MIR Assignment 3. Spring 2014 (100 pts)

This assignment requires some programming. You can choose any programming language/environment you like but some of the questions are specifically designed for *Marsyas* and will be easier if you use the framework. You can choose to use the C++ code, the Python bindings or the scripting language of *Marsyas*. Do not hesitate to contact me with any questions you might have. The assignment is due on March 13 (soft deadline).

IMPORTANT Some questions have additional deliverables for the graduate students taking the course. These are marked with [CSC575]. Undergraduate students are welcome to do them but they are optional and will not affect their grade. The assignment is worth 10% of the final grade. There is some variance in the amount of time each question probably will require. Therefore dont expect them to be equally difficult even though they are all worth the same number of points. Please provide your answers in a SINGLE PDF file. There is no need to copy the assignment specification. Also please answer the questions in order and explicitly mention if you have decided to skip a question. For questions that involve programming provide a listing of the relevant code but not all the details that are not directly relevant to the question. You can either hand me a paper copy of your submission in class or email me an electronic copy.

Hope you find it interesting, George Tzanetakis

Question 1 (35 points)

The first step will be to download and compile the latest version of Marsyas from github. Instructions as well as documentation can be found at the Marsyas user manual: `http://marsyas.info/docs/manual/marsyas-user/index.html` This step is worth **10 points** and will probably require some interaction with me. Don't hesitate to contact me if you have questions/issues with this step.

The mudbox application (the code is in src/apps/mudbox) contains a large number of small simple "toy" functions mainly used for prototyping and experimentation. Your task will be to modify the *toy_with_centroid function*.

Currently the code calculates and outputs to standard output the running spectral centroid of one sound file. You will need to modify the code to output to standard output the spectral centroid of two sound files given as arguments to the *toy_with_centroid* function. To accomplish that you will need to create a Fanout Composite and add two branches (each one a Series Composite with the processing steps for each sound file). You will need to change the updctrl messages to reflect the new dataflow layout.

You can see an example of how the Fanout Composite can be used in the *toy_with_sine()* function in mudbox.cpp. As your final task copy the code of the SineSource playback network in *toy_with_sine()* to *toy_with_centroid* and add audible feedback to the centroid output (5 points). Each centroid is between 0 and 1 where 0 corresponds to the lowest FFT frequency bin and 1 corresponding to the highest FFT frequency bin. The input to the FFT is a buffer of 512 time domain samples. You will need to convert the centroid values of the centroid network to Hertz, update the corresponding SineSource frequency controls and hopefully hear the sine output following the centroids around. Basically you will have two dataflow networks (the centroid calculation one and the sine playback one that you will tick at the same time in the processing loop).

Most likely you can do these modification just by understanding and changing the existing source code but in case you want to read some more information you can look at the Marsyas documentation as well as papers published regarding Marsyas:

http://marsyas.info/documentation http://marsyas.info/about/publications

Download one or two classical music files and metal music files from the corresponding folder of: http://marsyas.cs.uvic.ca/sound/genres/

Feed the following combinations into your centroid sonification system:

- classical and classical

- classical and metal
- metal and metal

To some extent this would be what an automatic genre classification system based purely on the Spectral Centroid would "hear" to make a decision of classical or metal. FYI using just the mean of the spectral centroid a trained classifier makes the right decision 75% of the time. This is impressive given that decisions are made every 20 milliseconds so even better results could be obtained by some majority voting/filtering over the entire file.

To see the influence of the texture window comment out the the MarSystems Memory and Mean in both branches.

In your answer please include the modified code as well as very brief commentary (no more than 3-4 sentences) of your observations on this sonification experiment. The centroid computation for two files is worth **10 points**, the sonification part is also worth **10 points** and the short commentary is worth **5 points**.

It is also straightforward to do the same experiment using either the *Marsyas* Python bindings or the *Marsyas* scripting language and you are welcome to do so.

[CSC575] Extend the code to so that in addition to playing the sonified centroid it also plays the original sound files.

Question 2 (30 points)

Download the 1.2 GB genre classification dataset from:
`http://marsyas.info/download/data_sets`
You will only need 1.2 GB of space for download but after that you can pick any three genres out of the 10 genres for your experiments. Alternatively if you don't have enough space you can download individual files for 3 genres (at least 20 tracks for each genre) from:
`http://marsyas.cs.uvic.ca/sound/genres/`
Read the instructions in Chapter 3 of the Marsyas User Manual (Tour - Command Line Tools) and use bextract to extract features for the 3 genres you selected. Load the extracted .arff file into Weka and report on the classification accuracy of the following classifiers: ZeroR, NaiveBayesSimple, J48, and SMO.

Your deliverable will be the list of command you used and the classification accuracy + confusion matrix for each classifier for the 3-genre experiment.

[CSC575] The command-line options to bextract -ws, -hs control the analysis window size and hop size for the audio feature calculation. Create a table showing how classification accuracy is affected by that choice. Investigate all combinations of the following values 256, 512, 1024, 2048, 4096 for each. For example 256 hop size and 1024 window size, or 1024 hop size and 1024 window size.

Question 3 (35 points) Read the paper "Semantic Annotation and Retrieval of Music and Sound Effects"
`http://cosmal.ucsd.edu/~lbarring/pubs/Turnbull,Barrington,Torres&Lanckriet-SemanticAnnotation&RetrievalOfMusic.pdf`
which is one of the first published works exploring automatic music tagging and answer the following questions:

1. **(15 points)** In Section IIIB the authors formulate the annotation problem using probabilistic notation. Come up with a concrete example with made up numbers where each song has a sequence of 4 feature vectors of dimension 2.

2. **(10)** The authors describe three parameter estimation techniques: direct estimation, model averaging, and mixture hierachies estimation. For which category of tags does model averaging work better than mixture hiearchies ? Which category was modeled most successfully using the proposed approach ?

3. **(10)** Describe the differences between the music data set and the sound effect dataset. For which one does auto-tagging seem to work better ? Speculate why this is the case.

4. [**CSC475**] Read about the EM-algorithm and how it can be used for estimating the parameters of a Gaussian mixture model. Prepare a short (2-3 paragraphs) description targeted to undergraduate students for helping them understand how it works. Try to go beyond a dry mathematical description using one or more of the following techniques: 1) code example, 2) animation, 3) static visualization, 4) concrete specific example.