MIR Assignment 2, Fall 2017 (10 pts)

The goal of this assignment is to familiarize you with monophonic pitch detection and simple audio feature extraction - namely the extraction of a centroid. This assignment requires some programming. Both of these operations are performed based on the Short Time Fourier Transform so you can reuse your code from the previous assignment. Basically you have to process the audio in short segments corresponding to 20-40 milliseconds of audio and using the DFT and the magnitude spectrum get what you need. You can also optionally try to use the *Marsyas* audio processing framework but that will require some effort to install it and learn it. Inside Marsyas you can use either the *Marsyas* scripting language, C++ code, or the Python bindings. Don't hesitate to contact me or the TA with any questions you might have.

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.

Hope you find it interesting, George Tzanetakis

1 Monophonic pitch estimation (5 points)

The goal in this question is to extract a fundamental frequency contour from an audio file. In order to test your code utilize the following audio input signals:

- The little melody you created in assignment 1
- The same melody hummed by you and recorded in Audacity. Use a single vowel like Ah for your singing.
- The qbhexamples.wav available under Resources in Connex.
- **(2 points) (*)** A simple fundamental frequency estimation is to compute the magnitude spectrum, select the highest peak, and return the corresponding frequency as the result. Processing the sound in windows will result in a time series of F0 estimates that can be plotted over time. Make plots of the estimated FO over time for the three input testing signals using a window of 2048 samples at 44100 sampling rate.

  (If you decide to use *Marsyas*):

  If you want to use *Marsyas* start from Question 2 (they two questions are independent) as it will help you understand *Marsyas* better. It is possible to work directly in C++, or using Python bindings in Python, or use the custom-designed *Marsyas* scripting language.

  In *Marsyas* audio processing is expressed by connecting basic processing blocks into networks similarly to block diagrams in Signal Processing. You will need to read the manual and tutorial for more details and me the the TA are happy to help. For this question the following *MarSystems* would need to be connected: *SoundFileSource*, *Windowing*, *Spectrum*, *PowerSpectrum*, *Peaker*, *Selector*, *MaxArgMax* and *Accumulator*. You can read the documentation to find how these work.

- **(2 point) (*)**

  An alternative approach is to compute the Autocorrelation of the signal. Either use an existing implementation of Autocorrelation, or write your own, or read about how it can be expressed using the DFT. The peaks in the AutoCorrelation function correspond to different time domain lags. Convert the lags to frequency and similarly to the previous question plot the resulting F0 contours for the three input signals.

  (If you decide to use *Marsyas*): Use the Autocorrelation *MarSystem* and the ones mentioned in the previous question.

- **(1 point) (**)** Compare the F0 contours of the DFT approach and the AutoCorrelation approach. Is one consistently better than the other ? What types of errors do you observe ? Change your code so that both the F0 estimates are computed for every window. Change your code so that both the F0 estimates are computed for every window and plot the sum of the two resulting F0 contours for each input signal.

  (If you decide to use *Marsyas*): The computation should be done in one pass through the audio file. In *Marsyas* use a *Fanout* after the *Windowing* followed by a *Sum*.

2 Centroid Sonification (5 points)

The purpose of this question is to experiment with the audio feature called *spectral centroid*. The centroid corresponds to a frequency and can easily be computed from the magnitude spectrum. Similarly to the previous question you will need process the audio in short chunks of 2048 samples and compute the *centroid* for each chunk.

2.1 (2/1 points) Sonification (**)

Plot the centroid over time for the three examples that you used for monophonic pitch estimation. In addition create audio files with a sine wave generator that is controlled by the centroid to sonify the result. Listen to the resulting audio and write some sentences about what you think is happening.

2.2 (2/1 points) (***) Experiment with music genres

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

Run two instances of your centroid sonification script at the same time (using two command-line windows) to compare the results of pairs of different input audio files:

- 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 replaces each centroid value with the average of the previous 20 centroid values and plot again the contours. Then sonify the resulting smoother contour. Provide a very brief commentary (no more than 3-4 sentences) of your observations on this sonification experiment.

2.3 (1 point) Using Marsyas (***)

Answer the sonification parts of the previous questions by writing the code in *Marsyas*. Download and install *Marsyas*, and learn the basics with the help of the online tutorial: `http://marsyas.info/tutorial/tutorial.html`. Don't hesitate to contact your course instructor or TA for help.

In the following steps you will create a *Marsyas* script that reads an audio file, computes the spectral centroid of each window of the short-term Fourier transform (STFT), and plays back both the original audio file and a sine wave that follows the spectral centroid frequency. The structure of the script will be very much like the example in the "Control links" section of the tutorial: `http://marsyas.info/tutorial/tutorial.html#control-links`. However, instead of applying the RMS energy to the amplitude of a noise generator, you will apply the spectral centroid to the frequency of a sine wave generator. It is a good idea to start with the example code and modify it to achieve the goal of this question.

Note that the same experimentation can be done with the *Marsyas* Python bindings or the *Marsyas* C++ library, and you are welcome to do so if you prefer.

To answer the question you will need to replace the RMS energy computation. First compute the STFT with a window size of 2048 samples and 1/2 window overlap. Do this by combining a ShiftInput, a Spectrum, and a PowerSpectrum MarSystem. Then add a Centroid MarSystem. Smooth the centroid stream by applying an "audio feature texture window": by computing the mean of the last 20 values using a *Memory* and a *Mean MarSystem*. Finally, convert the result to a control stream using a FlowToControl MarSystem.

Replaces the playback of noise that follows the RMS energy. Use a SineSource MarSystem to generate the sine wave. Create a link between the centroid control stream and the frequency control of the SineSource. However, the Centroid MarSystem produces a number between 0 and the number of power spectrum bins. This needs to be converted to a frequency value in Hz between 0 and the Nyquist frequency. Use a control expression to do the conversion using the current sample rate. The sample rate is available as the "israte" control of any MarSystem (in this case SineSource itself).

3 Reading about Yin (2 points) (ONLY FOR CSC575 students who get less points for question 2)

Read the article "YIN, a fundamental frequency estimator for speech and music" by Alain de Cheveigne and Hideki Kawahara (available on the MIR course webpage under Course/Readings) and answer the following questions:

- The authors propose several steps for improving the F0 estimation accuracy of a standard autocorrelation-based method. What step had the biggest effect ?
- What was your favorite figure in the paper ? If you don't have a favorite pick one at random. Explain what you learn from this figure ?
- Where is interpolation used in the algorithm and why is it used ?
- Which was the hardest subsection to read and understand ? Try to put more effort into understanding it (read it again, consult the internet, read related papers etc). Do you feel you now understand it ? Don't just yes or no but provide some context around your answer. Do you think the authors could have written that subsection in a more readable way ? If yes how ?