CM3106: Multimedia
Tutorial/Lab Class 2 (Week 3)
Coursework Handout
Time-Frequency Analysis
(Short Time Fourier Transform)

Prof David Marshall

dave.marshall@cs.cardiff.ac.uk

CARDIFF UNIVERSITY
PRIFYSGOL CAERDYDD

School of Computer Science & Informatics
Cardiff University, UK

All Lab Materials available at:

http://www.cs.cf.ac.uk/Dave/Multimedia/PDF/tutorial.html

You should develop an **Interactive Fourier-based Synthesiser** in MATLAB:

The inspiration for the work is a piece of audio software called *Iris* by Izotope Inc.

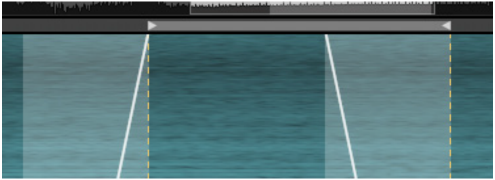
What is Iris?

Izotope **Iris**

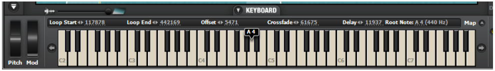
Iris is an innovative sampling **Fourier-based** re-synthesiser:

- You can input up to 3 waveforms and dissect and process them in many ways.
- Using *Iris*'s *spectrogram display* and easy drawing/selection tools to spotlight the most interesting spectral characteristics you can blend and layer your modified samples with some otherwise unrealisable filters.
- The sounds can then subsequently processed with other audio effects.

- Looping of samples:



- Keyboard Control:



See Izotope Iris manual:
http://www.izotope.com/products/audio/iris/help/Iris English Help PDF.pdf and download the demo code to understand Iris's full potential.

- Layers, Audio Effects and ADSR and LFO control:

Coursework Requirements

You are required to create a MATLAB program that implements the **basic spectrogram editing** and playback functionality of Iris with some **additional additional audio processing**.

A simple version of this!

Basic Coursework Requirements

What do I need to do to pass the coursework?

- Input audio file, compute time-frequency **short-term Fourier transform** and **spectrogram**.
- Provide an **interactive means of editing** this time-frequency form via its displayed spectrogram.
- The resulting edited audio then needs to be **played back**: **inverse short-term Fourier transform**
  - Playback need only be **monophonic**.
  - Not necessarily real time.
- Implement some way to trigger a sequence of notes.

Basic Coursework Requirements (Cont.)

Some more . . .

- Some **additional audio processing**:
  - You should implement some form of **volume shaping or envelope shaping** to control or modulate the basic sounds synthesised.
  - You should provide some **additional audio effects** that are applied to the newly synthesised waveforms to provide a wider sound palette. The obvious example here would be some form of equalisation, chorus/phaser/flanger or reverb, although other forms of processing could be provided.
  - **fixed** effects pipeline
  - Stay tuned for future lectures/labs and MATLAB code!

Additional Requirements

What do I need to do to get a High Mark:

Add**two** novel extensions or additional features.

Suggestions but feel free to be innovative!:

- **Multi-layer sample playback** using more than one audio source, like IRIS.
- Advanced playback functionality to allow for **looping** of sections of audio, **reversing** sections of audio *etc.*.
- Some **further** digital audio effects.
- Provide a **user-friendly editor** for the audio and/or to enter musical data.
  - GUI elements to control the synthesis, filtering/modulation, effects and sound output may be provided.
- Provide support for **polyphonic output**.
- Provide **MIDI support** for data input.
- Provide **additional** methods of **digital synthesis**. Granular Synthesis?
- A **modular synthesis/effects pipeline**.

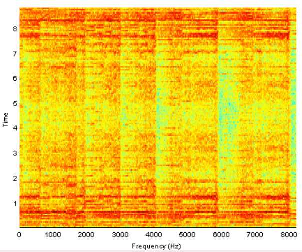
Demonstrating your system

You must prove it all works (and makes a sound)

- You will be required to demonstrate your final system to the lab tutor in order to **verify** the extent to which the programs work according to specification. The tutor is only **guaranteed** to be available to sign at Multimedia Laboratory Sessions.
- For the demo you need only play a short number of sounds/notes. This will be enough to demonstrate that you can make interesting atmospheric and/or musical sounds!
- If you have any additional features in your system, it will be appropriate to demonstrate how they work and that they function accordingly.

Short-Time Fourier Transform (STFT):

STFTs as well as standard **Fourier transforms** (and other tools) are frequently used to analyse audio.



Visual information about an audio sample, **for example**:

- to locate the frequencies of specific noises (especially when used with greater frequency resolution)
- to find frequencies which may be more or less resonant in the space where the signal was recorded.

This representation can be used for **equalisation**, **tuning temporal shifting** and other audio effects.

Short-Time Fourier Transform (STFT)

Forward Short-Time Fourier Transform (STFT)

$$X(\tau, u) = \int_{-\infty}^{\infty} f(t)w(t-\tau)\mathbf{e}^{-\mathbf{2\pi itu}}\, dt.$$

where $w(t)$ is the window function, commonly a **Hann window** or **Gaussian window** **bell centered** around **zero**, and $f(t)$ is the signal to be transformed.

- $X(\tau, u)$ is essentially the Fourier Transform of $t(t)w(t-\tau)$, a complex function representing the phase and magnitude of the signal over time and frequency.

Inverse defined similarly

Short-Time Fourier Transform MATLAB Code

There are many implementations of the STFT.

Short-Time Fourier Transform MATLAB Code

- stft.m: Forward STFT
- istft.m: Inverse STFT

Part of a simple Phase Vocoder Toolbox (Useful: **More soon**)

```
function D = stft(x, f, w, h, sr)
% D = stft(X, F, W, H, SR)     Short-time Fourier transform.
% Returns some frames of short-term Fourier transform of x.  Each
% column of the result is one F-point fft (default 256); each
% successive frame is offset by H points (W/2) until X is exhausted.
% Data is hann-windowed at W pts (F), or rectangular if W=0, or
% with W if it is a vector.
```

See also MATLAB Central implementation with Pitch detection

Using stft.m and istft.m

Simple Example

```
load handel; % Get some audio

% stft parameters (can vary)
n = 512;
nhop = n/4;
Y = stft(y,n,n,nhop);

yback = istft(Y,n,n, nhop);
%should be same as y!
```

Spectrogram

Cooking your own Spectrogram, stft_spectrogram.m:

```
load handel; % Get some audio

% stft parameters (can vary)
n = 512;
nhop = n/4;
Y = stft(y,n,n,nhop);

% Make Spectrogram
specy = abs(Y)/n;

% set left-hand coordinate origin
imshow(flipud(255*specy));
colormap(hsv); %color display
```

Phase Vocoder

Phase Vocoder

An algorithm for timescale modification of audio.

- Basically we can stretch or compress the time-base of a spectrogram to change the temporal characteristics of a sound while retaining its short-time spectral characteristics;
  - **Narrowband spectrogram** — analysis window longer than a pitch cycle — preserving the pitch but change speed/tempo.
  - **Wideband spectrogram** — change pitch in a controlled way.

Phase Vocoder MATLAB Code

A Basic Phase Vocoder in MATLAB

- pvoc.m — the top-level routine
- pvsample.m — interpolate/reconstuct the new STFT on the modified timebase
    - `pvsample()` routine could also support arbitrary timebase variation (freezing, reversal, modulation) with simple modification — **useful for Coursework!**.

Requires: stft.m and istft.m

Original Code here: Phase Vocoder Toolbox

Phase Vocoder Tempo Change

Phase Vocoder Tempo Change Code, pvoc_speed.m

```
% Get some audio
load handel;

% Half Speed
yslow =pvoc(y,.5,1024);
% Compare original and resynthesis
sound(y,Fs);
sound(yslow,Fs);


% Twice as Fast
yfast =pvoc(y,2,1024);
% Compare original and resynthesis
sound(y,Fs);
sound(yfast,Fs);
```

Phase Vocoder Pitch Change

Phase Vocoder Pitch Change Code, pvoc_pitch.m

```
% Get some audio
load handel;

% Pitch up a Fifth
ypvoc =pvoc(y, 0.66666);
ypitch = resample(ypvoc,2,3); % NB: 0.666 = 2/3
sound(y,Fs);
sound(ypitch, Fs);
sound(y(1:length(ypitch)) + ypitch, Fs);

% Pitch up an octave
ypvoc =pvoc(y, 0.5);
ypitch = resample(ypvoc,1,2);
...


% Pitch down an octave
ypvoc =pvoc(y, 2);
ypitch = resample(ypvoc,2,1);
...
```

Phase Vocoder Pitch Change Explanation

Pitch Change Code Fragment, pvoc_pitch.m

```
% Pitch up a Fifth
ypvoc =pvoc(y, 0.66666);
ypitch = resample(ypvoc,2,3); % NB: 0.666 = 2/3
......
```

Pitch Change Explanation:

- Extending/compress duration with the phase vocoder:
  - `pvoc(y, 0.66666)` : Need to know appropriate fraction of pitch shift
  - Look up here: *Meantone intervals*. (**Octave shifts are obvious**.)
- Resampling to the original length
  - `resample(ypvoc,2,3)`:
  - **Note**: New sample length won't be same as sample of original pitch.

References:

Useful web links:

- Izotope Iris main web page: *http://www.izotope.com/products/audio/iris/*
- Izotope Iris manual: *http://www.izotope.com/products/audio/iris/help/Iris English Help PDF.pdf*
- A **demo** version of Iris is freely available: *http://www.izotope.com/products/audio/iris/download.asp*
- *http://www.jyu.fi/musica/miditoolbox/* — MATLAB MidiToolbox. Reads/Writes Midi files, converts midi between note number, musical notes/pitches and frequencies. See also *http://www.cs.cf.ac.uk/Dave/Multimedia/exercises_BSC/* for local copy of the MidiToolbox.
- *http://labrosa.ee.columbia.edu/matlab/* — MATLAB Audio Processing Examples
- *http://www.harmony-central.com/articles/tips/pitch_vs_frequency/* — Musical pitches v frequency relationship.
- *http://en.wikipedia.org/wiki/List_of_meantone_intervals*: List of pitch/tone intervals as ratios. (Useful for the Phase Vocoder)