Table of Contents

%%%

Matlab Implementation of MPM pitch detection algorithm %%

Version V1.0 Date: Feb 15, 2015 Author: Zichao Wang

Documentation

This Function implements the MPM pitch detection algorithm by Dr. Pilip McLeod. You can read more about this algorithm in his paper titled "a smarter way to find pitch". The algorithm first calculates a normalized square difference function, and then find all local maxima, then threshold them, and use the first maxima as the key maxima. Its index is recorded as the pitch period. Using this pitch period, we are able to find the pitch of the sample.

Input

filename: The file name of your music. t: start time of the sample. Should not be longer than the total samples in your music file. W: Window size to perform MPM algorithm. Typically value is 2048. You can also use 512, 1024, 4096.

Output

f: detected frequency (pitch) note: detected pitch in MIDI scale (MIDI scale: http://newt.phys.unsw.edu.au/jw/notes.html) Also a plot of NSDF is generated.

Example:

filename = '153597__carlos-vaquero__violin-g-5-tenuto-non-vibrato.wav'; t = 1; W = 2048; [f,note] = MPM_pitch_detection('153597__carlos-vaquero__violin-g-5-... tenuto-non-vibrato.wav', 10000, 2048)

%%%

```
function [f,note] = MPM_pitch_detection(filename, t, W)

    [violin,fs] = audioread(filename);

Error using MPM_pitch_detection (line 37)
Not enough input arguments.
```

initialize parameters and vectors

```
%W = 2048; % window size
%t = 50000; % start time of sample
r_tau = zeros(1,W); % initialize ACF
m_tau = zeros(1,W); % initialize SDF
n_tau = zeros(1,W); % initialize NSDF
```

get a sample

```
x = violin(t:W+t-1);
```

calculate NSDF (normalized square difference function)

```
for tau = 0:W-1
   for j = 1:1+W-tau-1
      r_tau(tau+1) = r_tau(tau+1) + x(j)*x(j+tau); % calculate ACF
      m_tau(tau+1) = m_tau(tau+1) + (x(j)^2+ x(j+tau)^2);%calculate SDF
      n_tau(tau+1) = 2*r_tau(tau+1)/m_tau(tau+1); % calculate NSDF
   end
end
```

find local maxima

```
MAX = max(n_tau); % maximum correlation in NSDF (usually 1)
k = 0.8; % threshold parameter
th = MAX*k; % threshold for selecting key maximum
idx = 2; % starting index, excluding the first data point, which is 1
max_idx = 0; % the index of the key maximum. To be changed later
temp = find(n_tau<0); % for finding index of first negative element
local_max = zeros(1,W); % local maximum in NSDF
while idx < length(n_tau)-1 % for each sample
    temp_max = 0;
    while n_tau(idx) > 0 && idx < W && idx > temp(1)
    % for sample starting from the second arising pattern
       if n_tau(idx+1) > n_tau(idx)
           if temp_max < n_tau(idx+1)
               temp_max = n_tau(idx+1);
           end
       end
       idx = idx + 1;
    end
```

```
            max_idx = find(n_tau==temp_max);
            local_max(max_idx) = temp_max;
            idx = idx + 1;
        end
```

find pitch and fundamental frequency

```
        tau = find(local_max>th); % pitch period
        if ~isempty(tau) % check if there is a key maximum
            f = fs / tau(1); % corresponding frequency
            note = log10(f/27.5)/log10(2^(1/12));
        else
            f = -1;
            note = -1;
        end
        if note < 0 % deal with noise / no sound
            f = -1;
            note = -1;
        end
        plot(n_tau);

    end
```

Published with MATLAB® R2014b