Highlights from
matlab code for automatic speech recognition

Voice_Rec(sample_freq)

View all files

matlab code for automatic speech recognition

by shahista sayyed
26 Apr 2011

its a matlab code till daubechies wavelets done

Voice_Rec(sample_freq)

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%% Voice Record %%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%This program records the voice
function [norm_voice,h] = Voice_Rec(sample_freq)
option = 'n';
option_rec = 'n';
record_len = 1;         %Record time length in seconds
%sample_freq = 8192;    %Sampling frequency in Hertz
sample_time = sample_freq * record_len;

'Get ready to record your voice'
name = input('Enter the file name you want to save the file with: ','s');
file_name = sprintf('%s.wav',name);
option_rec = input('Press y to record: ','s');
if option_rec=='y'
    while option=='n',
        input('Press enter when ready to record--> ');
        record = wavrecord(sample_time, sample_freq);       %Records the input through the sound card
to the variable with specified sampling frequency
        input('Press enter to listen the recorded voice--> ');
        sound(record, sample_freq);
        option = input('Press y to save or n to record again: ','s');
    end
    wavwrite(record, sample_freq, file_name);  %Save the recorded data to a file with the specified
file name in .wav format
end
[voice_read,FS,NBITS]=wavread(file_name);
norm_voice = normalize(voice_read);
norm_voice = downsmpl(norm_voice, sample_freq);
le=32;
h=daubcqf(le,'min');


function vec = normalize(vec)

temp_vec = vec-mean(vec);
sum_temp_vec = sum(temp_vec.*temp_vec);
sqrt_temp_vec = sqrt(sum_temp_vec);
vec = (1/sqrt_temp_vec)*temp_vec;

function sampled = downsmpl(voice, freq)

x=freq;
y = freq/2;
z=1;
a=1;
sampled=0;
while z<freq,
    sampled(a) = sqrt(abs(voice(z)*voice(z+1)));
```



```
    a=a+1;
    z = z+2;
end
sampled = sampled';


function [h_0,h_1] = daubcqf(N,TYPE)
%    [h_0,h_1] = daubcqf(N,TYPE);
%
%    Function computes the Daubechies' scaling and wavelet filters
%    (normalized to sqrt(2)).
%
%    Input:
%       N    : Length of filter (must be even)
%       TYPE : Optional parameter that distinguishes the minimum phase,
%              maximum phase and mid-phase solutions ('min', 'max', or
%              'mid'). If no argument is specified, the minimum phase
%              solution is used.
%
%    Output:
%       h_0 : Minimal phase Daubechies' scaling filter
%       h_1 : Minimal phase Daubechies' wavelet filter
%
%    Example:
%       N = 4;
%       TYPE = 'min';
%       [h_0,h_1] = daubcqf(N,TYPE)
%       h_0 = 0.4830 0.8365 0.2241 -0.1294
%       h_1 = 0.1294 0.2241 -0.8365 0.4830
%
if(nargin < 2),
  TYPE = 'min';
end;
if(rem(N,2) ~= 0),
  error('No Daubechies filter exists for ODD length');
end;
K = N/2;
a = 1;
p = 1;
q = 1;
h_0 = [1 1];
for j  = 1:K-1,
  a = -a * 0.25 * (j + K - 1)/j;
  h_0 = [0 h_0] + [h_0 0];
  p = [0 -p] + [p 0];
  p = [0 -p] + [p 0];
  q = [0 q 0] + a*p;
end;
q = sort(roots(q));
qt = q(1:K-1);
if TYPE=='mid',
  if rem(K,2)==1,
    qt = q([1:4:N-2 2:4:N-2]);
  else
    qt = q([1 4:4:K-1 5:4:K-1 N-3:-4:K N-4:-4:K]);
  end;
end;
h_0 = conv(h_0,real(poly(qt)));
h_0 = sqrt(2)*h_0/sum(h_0);     %Normalize to sqrt(2);
if(TYPE=='max'),
  h_0 = fliplr(h_0);
end;
if(abs(sum(h_0 .^ 2))-1 > 1e-4)
  error('Numerically unstable for this value of "N".');
end;
h_1 = rot90(h_0,2);
h_1(1:2:N)=-h_1(1:2:N);
```