UNIVERSIDADE FEDERAL DO PARANÁ

SETOR DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

GISELE TAM

GRR20102167

PHASE VOCODER

Trabalho realizado para a disciplina de Processamento Digital de Sinais 2 para obtenção de nota parcial.

CURITIBA

2015

INTRODUÇÃO

O som é uma vibração mecânica que se propaga em meios físicos e pode ser representado por um sinal elétrico analógico ou por um sinal digital. No formato digital, ele é descrito por uma série de números, o que possibilita a aplicação de operações matemáticas que o modificarão ao ser reproduzido novamente.

O *phase vocoder* é um algorítmo para dilatação ou compressão do tempo que costuma apresentar grande fidelidade. Ele é uma variação da Transformada de Fourier de Tempo Curto (STFT – *short-time Fourier Transform*) e tem muito potencial musical.

OBJETIVO

O objetivo deste trabalho é implementar um programa que aplique o *phase vocoder* em um arquivo sonoro e observar as variações de tempo e frequência do som.

DESENVOLVIMENTO

O código é composto de 4 etapas e foi implementado no Matlab. A transcrição completa encontra-se no Apêndice deste documento.

1. Análise

Inicialmente, o arquivo sonoro é especificado e os parâmetros “high” e “low” podem ser escolhidos (caso eles sejam iguais, não haverá mudança no tempo nem na frequência). Estes parâmetros definem o quanto o tempo será comprimido (low > high) ou dilatado (low < high).

Aqui, o som original é separado em vários segmentos menores e eles são multiplicados por uma janela Hanning (foi escolhida a função *hann* do Matlab). Todos os segmentos têm tamanho de 1024 pontos (windowLen) e eles se sobrepõem, ou seja, o segmento seguinte inicia 256 pontos depois (nJump) do anterior. Em seguida,

a transformada rápida de Fourier (função *fft* do Matlab) é aplicada em cada segmento e os valores são armazenados numa matriz. Assim, cada coluna desta matriz é a representação espectral de um segmento do som original.

2. Mudança do Espectro

Nesta etapa, os valores high e low são usados para criar um fator de modificação que será utilizado para alterar a velocidade do som por meio da mudança da distância das colunas da matriz criada na etapa de análise.

Com a variação na distância entre as colunas, a fase do sinal pode ser alterada, o que influenciará a frequência (por definição, a frequência é a derivada da fase no tempo). Portanto, é preciso calcular a difrença de fase entre as colunas e somar à fase da coluna seguinte para que não haja diferença na frequência do sinal.

3. Síntese

Após a mudança do sinal é possível realizar sua síntese com a transformada inversa de Fourier (função ifft do Matlab) e a soma dos segmentos. O sinal final terá sido comprimido ou dilatado no tempo.

4. Reamostragem do sinal

Para alterar a frequência do sinal compensa-se a dilatação/compressão do tempo com a reamostragem do sinal (função *resample* do Matlab). Desta forma, é possível alterar a frequência do sinal e manter o tempo original. Se o arquivo teve o tempo dilatado, a frequência será mais aguda. Por outro lado, se o tempo foi comprimido, a frequência será mais grave.

CONCLUSÃO

O programa foi testado com arquivos sonoros disponíveis na internet e funcionou corretamente. Contudo, inicialmente o trabalho era para ser aplicado em arquivos de voz. Uma possível complementação seria criar uma função auxiliar para gravação de uma amostra de voz, aplicar do *phase vocoder* nesse arquivo e observar se as alterações provocadas serão igualmente satisfatórias.

REFERÊNCIAS

CASTILHO, S. **Análise e Ressíntese de Sinais Musicais.** 2008. Florianópolis. Disponível em: < https://repositorio.ufsc.br/xmlui/handle/123456789/91876>. Acesso em: 01 out. 2015.

ELLIS, D. P. W. **A Phase Vocoder in Matlab**. 2002. Disponível em: <http://labrosa.ee.columbia.edu/matlab/pvoc/> Acesso em 30 set. 2015.

Matlab Documentation. **Pitch Shifiting and Time Dilatation Using a Phase Vocoder**. Disponível em: <http://www.mathworks.com/help/dsp/examples/pitch-shifting-and-time-dilation-using-a-phase-vocoder.html>. Acesso em: 30 set. 2015.

SETHARES, W. A. **A Phase Vocoder in Matlab**. Disponível em: <http://sethares.engr.wisc.edu/vocoders/phasevocoder.html>. Acesso em: 25 set. 2015

HARDIE, R. C. **Sound Processing in Matlab**. Disponível em: <http://homepages.udayton.edu/~hardierc/ece203/sound.htm>. Acesso em: 22 set. 2015.

APÊNDICE – Código MATLAB

```
%% Phase Vocoder
% Trabalho 01 de Processameto Digital de Sinais 2
% Gisele Tam - GRR20102167
%
% Este código foi baseado no código escrito por D.P.W. Ellis em 2002, que
% está disponível em http://labrosa.ee.columbia.edu/matlab/pvoc/

[original,fs] = wavread('singing.wav');
sound(original,fs)
display('Som original')

windowLen = 1024; %tamanho da janela;
nJump = (windowLen)/4; %passo da janela

% Parâmetros para compressão/dilatamento no tempo e mudança de frequência
% Para mais agudo: high>low;
% Para mais grave: high<low.
low = 3;
high = 4;
%% Aplicação da STFT (Short-time Fourier Transform)

sizeO = length(original);
vAnalise = original(:,1)';

mSTFT = zeros((1+windowLen/2),1+floor(sizeO-windowLen)/nJump);
win = (hann(windowLen))'; %janela Hanning
col = 1; %inicialização do valor da coluna
for ind = 0:nJump:(sizeO-windowLen)
    aux = win.*vAnalise((ind+1):(ind+windowLen));
    aux = fft(aux);
    mSTFT(:,col) = (aux(1:(1+windowLen/2)))'; %usa metade para retirar os
pontos redundantes
    col = col+1;
end


%% Novo spectograma

[r,c] = size(mSTFT);
factor = low/high;
vChange = 0:factor:(c-2);
N = 2*(r-1);

mNew = zeros(r,length(vChange)); %inicialização do vetor de interpolação

% Acumulador de fase
ph = angle(mSTFT(:,1));

% Matriz auxiliar equivalente a mNew com uma coluna de segurança de zeros
% no final caso seja um número ímpar de colunas
mAux = [mSTFT, zeros(r,1)];

col = 1; %auxiliar para coluna no loop
for fac = vChange
    mAuxC = mAux(:,floor(fac)+[1 2]);
    ff = fac - floor(fac);
```

```
    mMag = (1-ff)*abs(mAuxC(:,1)) + ff*(abs(mAuxC(:,2)));
    % Cálulo do avanço de fase
    dp = angle(mAuxC(:,2)) - angle(mAuxC(:,1));
    % Deixar os valores entre -pi e pi
    dp = dp -2*pi*round(dp/(2*pi));
    % Nova matriz
    mNew(:,col) = mMag.*exp(j*ph);
    % Acumulação da fase
    ph = ph + dp;
    col = col+1;
end

%% Aplicação da ISTFT (STFT Inversa)

[rows, cols] = size(mNew);

winISTF = win;
lengthISTF = windowLen + (cols-1)*nJump;
vISTF = zeros(1,lengthISTF);

for ind = 0:nJump:(nJump*(cols-1))
    i_aux = (mNew(:,1+ind/nJump))';
    i_aux = [i_aux, conj(i_aux([(windowLen/2):-1:2]))];
    vecI = real(ifft(i_aux));
    vISTF((ind+1):(ind+windowLen)) =
vISTF((ind+1):(ind+windowLen))+vecI.*winISTF;
end

vISTF = vISTF;

%%
% Comprimido/dilatado no tempo
sound(vISTF,fs)
display('Som após PhaseVocoder')

% Mudança de frequêmcia
pShift = resample(vISTF,low,high);
sound(pShift,fs)
display('Som após mudança de frequência')
```