Centro Universitário SENAI CIMATEC
Curso de Bacharelado em Engenharia Elétrica

Algoritmo para correção de tom por meio da análise em frequência

Cleber Couto Filho
Ícaro Nascimento Queiroz

Salvador-BA, 20 de outubro de 2018

Cleber Couto Filho
Ícaro Nascimento Queiroz

Algoritmo para correção de tom por meio da análise em frequência

Relatório apresentado como requisito parcial para obtenção de aprovação na disciplina Processamento Digital de Sinais, no centro universitário SENAI CIMATEC.

Docente: Leonardo Vasconsellos

Centro Universitário SENAI CIMATEC

Salvador-BA
20 de outubro de 2018

Lista de ilustrações

1 Introdução

Este relatório descreve os procedimentos e códigos utilizados para a criação de um sistema de processamento de áudio para correção de tons musicais.

Teoria Musical

Na música a notação utilizada, chamada diastemática, os sons são representados graficamente, de maneira que seja possível mensurar os intervalos de frequências, o que indica diferentes notas musicais.

Para representação de sons mais longos ou curtos surge então a notação da grandeza tempo, que diferentemente da convenção comum, não possui um valor fixo, portanto, cria-se uma referência com relação à duração da notas semibreve, sendo elas a mínima, semínima, colcheia, semicolcheia, fusa, semifusa.

Outro aspecto extremamente importante da teoria musical é a frequência, grandeza qual classifica o quão graves ou agudos são os sons. Sons que apresentam maiores frequências são mais agudos que os de frequências mais baixas

Diferentes instrumentos são capazes de gerar sons diferentes mesmo que as frequências fundamentais sejam idênticas, o que em outras palavras significa dizer que um Ré gerado por um instrumento de corda não tem o mesmo som de um Ré gerado por um instrumento de sopro ou da voz humana, Este fenômeno, que é chamado de timbre, se dá principalmente pela geometria do instrumento, que define quais serão as outras componentes senoidais adicionadas à fundamental.[1]

A evolução da música faz com que o tratamento do sinal se torne uma questão cada vez mais explorada. A busca de um som refinado e que soe bem fomentou o desenvolvimento de diversas técnicas para manipulação de sinais voltados para criação de novas experiências sonoras.

O intercâmbio cultural gerado pelo crescimento da internet, fez com que o nível de habilidade exigido dos artistas aumentasse, assim sendo necessário a exploração de novas características para o músico. Ferramentas para geração de sons digitais como os sintetizadores, utilizam as propriedades dos sinais para criar sons não encontrados normalmente na natureza.

O auto-tune é uma ferramenta conhecida por “consertar” a voz dos cantores, tendo a função de corrigir os tons de voz do cantor para uma escala específica, deixando a voz afinada. O nome pertence ao plugin criado em 1997 pela Antares Audio Technologies.

Desde então é utilizados em diversas músicas e de diferentes formas.

Proposta

Projetar um algoritmo capaz de realizar a afinação de um som emitido pela voz humana para uma frequência selecionada através das técnicas de processamento como transformadas e mudanças de tom. Para a realização dos algoritmos foram tomadas as seguintes decisões:

- Aplicação dos conceitos de Processamento digital de Sinais;
- Identificação das componentes de frequência da voz;
- Realizar a troca de tom através das técnicas de Pitch Shfit

2 Fundamentação Teórica

Processamento de Áudio

As gravações de áudio se dão por meio de microfones e captadores de áudio, os quais realizam a conversão por meio de um transdutor, o qual converte a variação da pressão acústica em uma variação de tensão elétrica correspondente. Este sinal é convertido em pequenas amostras individuais espaçadas no tempo de maneira regular, constituindo a aproximação da forma de onda original.

Este processo é conhecido como conversão analógico-digital. O número de amostras retiradas da onda original no período de um segundo, é chamada frequência de amostragem, e quanto mais elevado o seu valor, mais fiel será a representação do sinal no domínio digital. De acordo com o teorema de Nyquist, o limite mínimo para a frequência de amostragem de qualquer sinal é o dobro da sua frequência original. A representação do sinal de áudio no domínio digital é apresentada como uma sequência de palavras onde o número de bits determina a resolução em amplitude do sinal.

No processo de reprodução de áudio, acontece o inverso da situação original, onde o sinal digital é enviado a um conversor digital-analógico, responsável pela reconstrução do sinal para que ele possa ser reproduzido em alto-falantes, caixas de som, ou qualquer aparelho que possa reproduzir sinais de áudio.

Dentro do domínio digital, o sinal de áudio pode ser tratado utilizando todas as técnicas de processamento, tais como as FFTs, DFTs filtros digitais, técnicas de janelamento e filtros digitais. Sendo assim, torna-se extremamente interessante e necessário realizar diversos tratamentos nestes sinais no domínio digital para reduzir as interferências externas, ruídos e desafinação, mantendo assim uma afinação constante em uma música.

Uma das principais técnicas utilizadas para correção de afinação é a de pitch-shift, a qual consiste na mudança do tom de um sinal de áudio sem modificar o tamanho dele. A diferença de um deslocamento em frequência para o pitch-shift é justamente o fato de que num deslocamento em frequência, existe um deslocamento do espectro do som, enquanto um phase-shift dilata ou comprime o espectro de som.. O Pashe-Vocoder é uma técnica de pitch-shift muito utilizada por produtores musicais.[2][3]

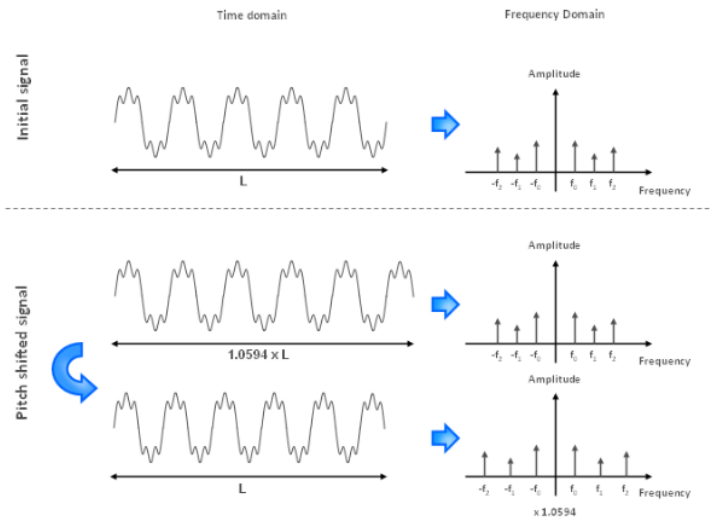

Figura 1 – Demonstração da técnica de Pitch Shift para sinais de guitarra

STFT (Short Time Fourier Transform)

Uma das técnicas mais utilizadas em processamento digital de sinais é a Transformada Discreta de Fourier (DFT), a qual é uma variação da Transformada de Fourier para sinais discretos. A transformada de Fourier define a redução de uma função periódica a um somatório de senos e cossenos. Este procedimento matemático gera uma representação de um sinal originalmente no domínio do tempo, em uma representação no domínio da frequência.[4]

$$f(x) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} \hat{f}(\omega) e^{i\omega x} \, d\omega.$$

Figura 2 – Transformada de Fourier para sinais contínuos

$$X(n) = \sum_{k=0}^{k=N-1} x(k) e^{-j(2\pi/N)nk}$$

Figura 3 – Transformada Discreta de Fourier

As transformadas de Fourier se aplicam somente a sinais de funções estacionárias, onde o espectro de frequência é fixo e não variam com o tempo.

Os sinais de áudio gerados pela voz humana se encontram dentro do espectro de frequências entre 50 a 3400 Hz, e a sua principal característica é a de que a sua frequência não é constante no tempo, o que dá ao sinal da voz humana a característica da não-ergodicidade (seu sinal não mantém as propriedades estatísticas ao longo do tempo), sendo assim, a utilização da STFT (Short Time Fourier Transform) se torna bastante eficaz em sinais dessa natureza.

A STFT é um algoritmo desenvolvido com base na transformada discreta de Fourier, diferenciando-se pela inclusão de uma função de janelamento w(t). Sua principal aplicação é para funções cujo o espectro de frequência varia com o tempo.[4]

$$\mathbf{STFT}\,\{x(t)\} \equiv X(\tau,\omega) = \int_{-\infty}^{\infty} x(t)w(t-\tau)e^{-j\omega t}\,dt$$

Figura 4 – Transformada de Fourier de Tempo Curto

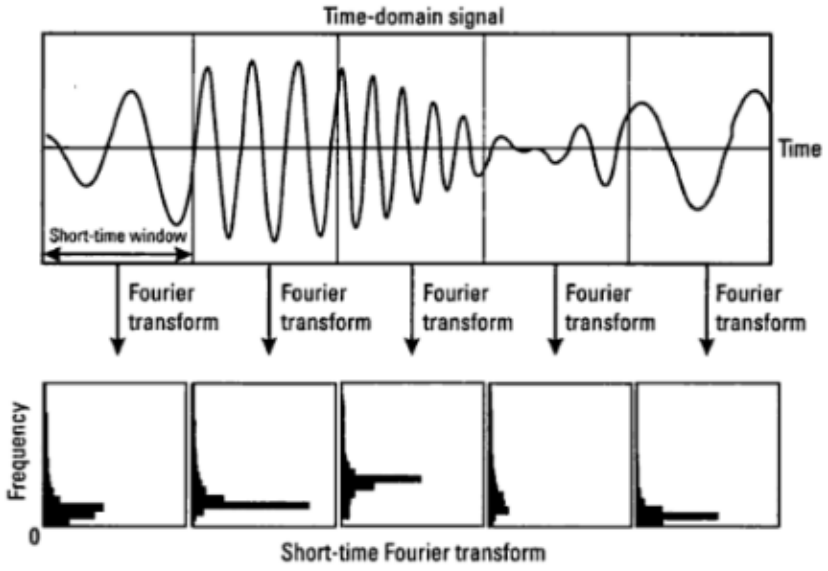

Figura 5 – Transformada de Fourier de Tempo Curto sendo aplicada em diversas frequências de um mesmo sinal

O principal propósito da utilização de uma STFT é separar o sinal em pequenos intervalos que possam ser tratados individualmente, obtidos através da janela que está inserida na transformada. Desta maneira a modificação de frequência se dá de forma independente, sem a alteração de tempo e vice versa.

Funções de Janelamento

Para aplicações que consistem na amostragem de sinais, a amostragem, por ser finita, resulta em uma forma de onda truncada com características diferentes do sinal original, consequentemente a influência do vazamento espectral torna-se maior para uma situação como esta, gerando uma perda de informação do sinal original. Para reduzir os efeitos das imperfeições de amostragem, melhorando a qualidade da reconstrução do sinal é a aplicação de uma função de janelamento.[5]

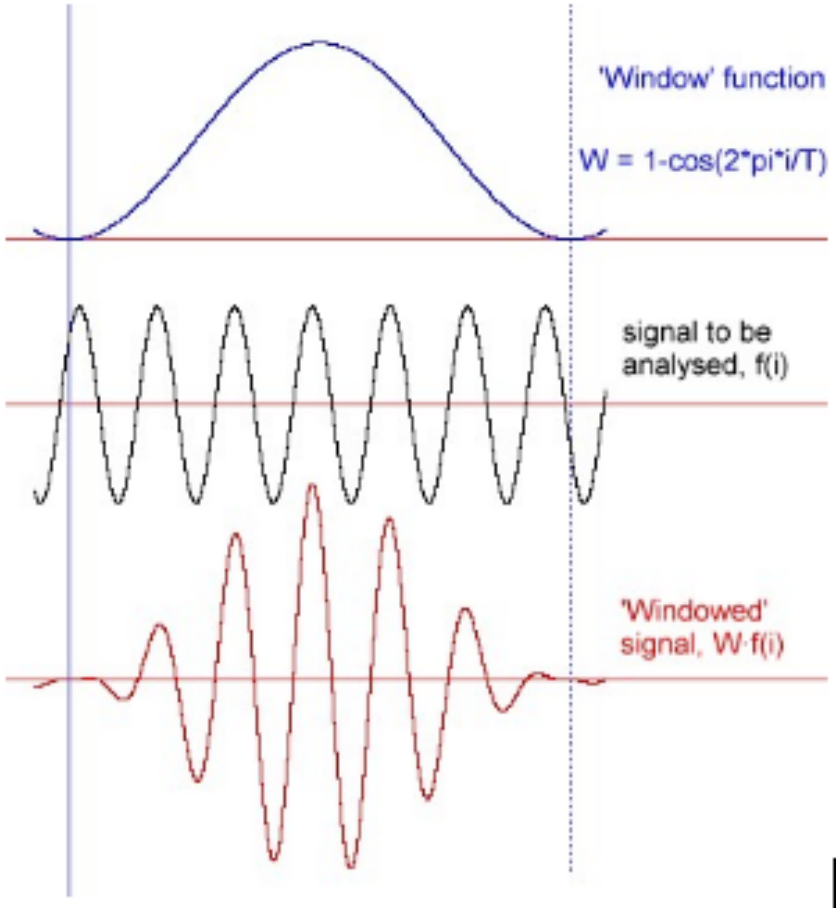

Figura 6 – Efeitos da função de janelamento em um sinal no espectro de frequência

A utilização de uma função de janelamento permite uma definição do período de observação do sinal, redução dos efeitos do vazamento espectral e a separação do sinal de pequena amplitude com frequências muito próximas. A aplicação de uma função de janelamento no tempo, consiste na multiplicação da função original pela função, o que equivale a uma convolução no domínio da frequência. Existem diversas funções de janelamento, as quais possuem diferentes características e aplicações dependendo principalmente dos parâmetros desejados do sinal original.

- Retangular
- Hanning
- Hamming
- Blackman
- Kaiser-Bessel

Janela Hanning

Dentre as funções de janelamento existentes, a função Hanning é a mais comumente utilizada na produção musical. O formato desta janela é similar ao de meio ciclo de uma onda cossenoidal. Suas características de baixo vazamento espectral e formato de onda bem similar ao formato cossenóide,. Torna-se recomendável utilizar portanto a janela para análises de sinais com transientes maiores que de duração da própria janela.

$$w[n] = 0.5\text{-}0.5\cos(2\pi n/N),\ n = 0, 1, 2, \ldots, N\text{-}1$$

Figura 7 – Função da janela Hanning

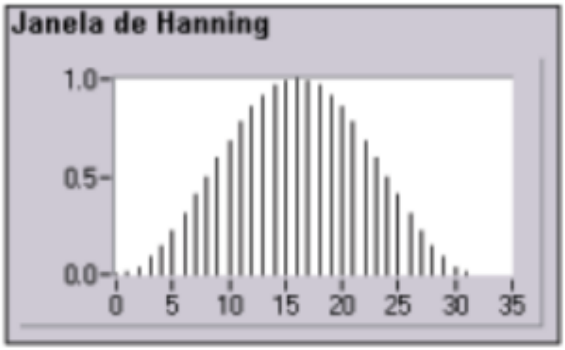

Figura 8 – Gráfico da janela Hanning

3 Metodologia

Funcionamento da solução

Utilizando o software MATLAB, uma solução computacional foi desenvolvida para realizar o procedimento de regulagem de tom.

O funcionamento do algoritmo se dá primeiramente com a captação do áudio por meio dos microfones. Logo em seguida é aplicada a STFT para detecção dos espectros de frequência presentes no sinal de áudio.

A detecção da frequência mais próxima se dá para que o Pitch-Shift seja realizado corretamente e logo em seguida o sinal seja reconstruído.

Para a reprodução do sinal é realizada a transformada inversa de Fourier para que o sinal seja colocado no tempo.

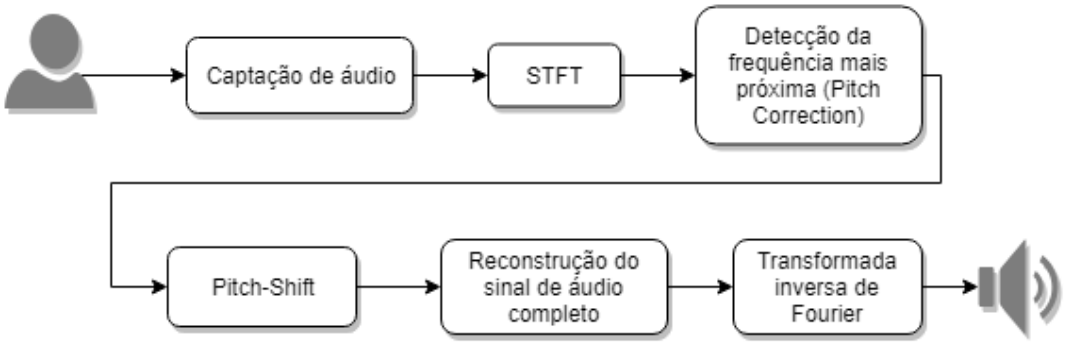

Figura 9 – Fluxograma de Funcionamento do algoritmo desenvolvido

Funções Utilizadas

gravaAudio

Função responsável por gravar o áudio à ser utilizado para afinação. Recebendo como parâmetros de entrada a frequência de amostragem fs e a duração do áudio. Retorna o aúdio gravado.

istft

Função responsável por realizar a transformada inversa da stft. Recebe como parâmetros de entrada o conjugado do sinal a ser convertido, o número de amostras da

stft, a quantidades de bins N da janela, e quantidade de bins que não sofrerão overlap. Retorna o sinal reconstruído

spectrogram

Função responsável por fazer a stft, recebendo como argumentos o sinal, a janela a ser utilizada, quantidade de amostras da FFT, quantidade de amostras que não sofrerão overlap e frequência de amostragem. Retornando um vetor contendo o sinal transformado, um vetor de frequências F, um vetor de tempo T e um vetor de densidade espectral de potência.

conj

Retorna o complexo conjugado dos elementos contidos no argumento.

mostraEspectograma

Mostra o spectrograma do sinal S, recebendo o vetor de frequência F e o vetor de tempo T.

tabela Do Maior

Retorna um vetor contendo todas as frequências da escala de Dó Maior.]

compareToPitches

A função recebe um tom e a tabela contendo a escola de tons, retornando o tom mais próximo da escala.

pitchCorrector

Função responsável por corrigir a stft, colocando os tons na escala fornecida. Recebe como entrada a STFT, seu espaço em frequência, o vetor de densidade espectral de potência P e tabela com a escala de tons. Retoranndo a stft corrigida.

sound

Toca o sinal de som fornecido, recebendo o som e a frequência de amostragem.

Referências

[1] DANTAS, Joseclécio Dutra, CRUZ, Sérgio da Silva ,Um olhar físico sobre a teoria musical, Universidade Federal de Campina Grande,2018

[2] BERNSSE, Stephan M., Time Stretching And Pitch Shifting of Audio Signals, 2005

[3] Gerhard, David Pitch Extraction and Fundamental Frequency: History and Current Techniques, 2003

[4] DIAZ, Joe, The fate of Auto-Tune, 2009

[5] Andrade, A. O. e Soares, A. B., Técnicas de Janelamento de Sinais, Universidade Federal de Uberlândia

ANEXO

Função GravaAudio

```
function audio = gravaAudio(fs,duration)
%Cria um objeto do tipo audio recorder com uma frequencia de amostragem fs, 16 bit, canal mono
recorder = audiorecorder(fs,16,1);
 record(recorder);
 pause(duration);

 audio = getaudiodata(recorder);
end
```

Função istft

```
%%Downloaded from http://www.ee.columbia.edu/~dpwe/e4810/matlab/s24/istft.m

function x = istft(d, ftsize, w, h)
% X = istft(D, F, W, H)                    Inverse short-time Fourier transform.
%       Performs overlap-add resynthesis from the short-time Fourier transform
%       data in D.  Each column of D is taken as the result of an F-point
%       fft; each successive frame was offset by H points (default
%      W/2, or F/2 if W==0). Data is hann-windowed at W pts, or
%      W = 0 gives a rectangular window (default);
%      W as a vector uses that as window.
%       This version scales the output so the loop gain is 1.0 for
%       either hann-win an-syn with 25% overlap, or hann-win on
%       analysis and rect-win (W=0) on synthesis with 50% overlap.
% dpwe 1994may24.  Uses built-in 'ifft' etc.
% $Header: /home/empire6/dpwe/public_html/resources/matlab/pvoc/RCS/istft.m,v 1.5 2010/08/12 20:39:42 dpwe Exp $

if nargin < 2; ftsize = 2*(size(d,1)-1); end
```

```
18 if nargin < 3; w = 0; end
19 if nargin < 4; h = 0; end  % will become winlen/2 later
20
21 s = size(d);
22 if s(1) ~= (ftsize/2)+1
23   error('number of rows should be fftsize/2+1')
24 end
25 cols = s(2);
26
27 if length(w) == 1
28   if w == 0
29     % special case: rectangular window
30     win = ones(1,ftsize);
31   else
32     if rem(w, 2) == 0   % force window to be odd-len
33       w = w + 1;
34     end
35     halflen = (w-1)/2;
36     halff = ftsize/2;
37     halfwin = 0.5 * ( 1 + cos( pi * (0:halflen)/halflen));
38     win = zeros(1, ftsize);
39     acthalflen = min(halff, halflen);
40     win((halff+1):(halff+acthalflen)) = halfwin(1:acthalflen);
41     win((halff+1):-1:(halff-acthalflen+2)) = halfwin(1:
          acthalflen);
42     % 2009-01-06: Make stft-istft loop be identity for 25% hop
43     win = 2/3*win;
44   end
45 else
46   win = w;
47 end
48
49 w = length(win);
50 % now can set default hop
51 if h == 0
52   h = floor(w/2);
53 end
54
55 xlen = ftsize + (cols-1)*h;
```

```
x = zeros(1,xlen);

for b = 0:h:(h*(cols-1))
  ft = d(:,1+b/h)';
  ft = [ft, conj(ft([((ftsize/2)):-1:2]))];
  px = real(ifft(ft));
  x((b+1):(b+ftsize)) = x((b+1):(b+ftsize))+px.*win;
end;
end
```

Função mostraEspectrograma

```
function Y = mostraEspectrograma(S,F,T)
%%Mostra o spectrograma do sinal S, recebendo o vetor de
    frequencia F e o vetor de tempo T

X = abs(S);
imagesc(T,F,S);
colorbar;
axis xy
xlabel('Time (s)')
ylabel('Freq (Hz)')

end
```

Função tabelaDoMaior

```
function pitches = tabelaDoMaior()
%%Retorna todas as frequencias da escala de Do maior

pitches = [16.352 18.354 20.602 21.827 24.500 27.500 30.868
    32.703 36.708 41.203 43.654 48.999 55.000 61.735 65.406
    73.416 82.407 87.307 97.999 110.000 123.470 130.800 146.830
    164.810 174.610 196.000 220.000 246.940 261.630 293.660
    329.630 349.230 392.000 440.000 493.880 523.250 587.330
    659.260 698.460 783.990 880.000 987.770 1046.500 1174.700
    1318.500 1396.900 1568.000 1760.000 1975.500 2093.000
    2349.300 2637.000 2793.800 3136.000 3520.000 3951.100
    4186.000 4698.600 5274.000 5587.700 6271.900 7040.000
```

```
    7902.100  8372.000  9397.300  10548.100  11175.300  12543.900
    14080.000  15804.300  16744.000  18794.500  21096.200  22350.600];

end
```

Função compareToPitches

```
%%A funcao recebe um tom e a tabela contadndo a escola de tons,
    retornando o tom mais proximo da escala
function exactpitch = compareToPitches(pitch, pitchtable)

%%Organiza a tabela em ordem crescente
sortedpitchtable = sort(pitchtable);

%%Cria vetor de zeros
midpitchtable = zeros(1,length(sortedpitchtable)-1);

%%Variavel que indica se o tom mais proximo foi encontrado
found = 0;

%%Gera uma tabela de tons na media dos tons da tabela de tons
    para tornar a comparacao
%%mais facil.
%%log2(0.5) e utilizado 0.5 afim de contabilizar para percepcao
    do som
for i = 1:(length(sortedpitchtable)-1)
    midpitchtable(i) = (sortedpitchtable(i+1) - sortedpitchtable
        (i))*log2(0.5) + sortedpitchtable(i);
end

%%Caso a frequencia seja menor que o menor tom da tabela, se
    retorna o menor tom
if (pitch <= midpitchtable(1))
    exactpitch = sortedpitchtable(1);
    found = 1;
end

%%Se o tom exato nao for encontrado ainda considera o maior tom
    da tabela
```

```
27 if (found==0)
28     if (pitch > midpitchtable(length(midpitchtable)))
29         exactpitch = sortedpitchtable(length(sortedpitchtable));
30         found = 1;
31     end
32 end
33
34 %%Se o tom exato nao foi encontrado ainda, compare o a todos os
       valores
35 %da midpitchtable e a saida corresponde ao tom na tabela de tom
36 i = 1;
37 while(found==0)
38     if(pitch > midpitchtable(i) && pitch <= midpitchtable(i+1))
39         exactpitch = sortedpitchtable(i+1);
40         found=1;
41     end
42     i = i+1;
43 end
44
45 end
```

Função pitchCorrector

```
1 %%A funcao recebe um tom e a tabela contadndo a escola de tons,
      retornando o tom mais proximo da escala
2 function exactpitch = compareToPitches(pitch, pitchtable)
3
4 %%Organiza a tabela em ordem crescente
5 sortedpitchtable = sort(pitchtable);
6
7 %%Cria vetor de zeros
8 midpitchtable = zeros(1,length(sortedpitchtable)-1);
9
10 %%Variavel que indica se o tom mais proximo foi encontrado
11 found = 0;
12
13 %%Gera uma tabela de tons na media dos tons da tabela de tons
      para tornar a comparacao
14 %%mais facil.
```

```
15 %%log2(0.5) e utilizado 0.5 afim de contabilizar para percepcao
       do som
16 for i = 1:(length(sortedpitchtable)-1)
17     midpitchtable(i) = (sortedpitchtable(i+1) - sortedpitchtable
           (i))*log2(0.5) + sortedpitchtable(i);
18 end
19
20 %%Caso a frequencia seja menor que o menor tom da tabela, se
       retorna o menor tom
21 if (pitch <= midpitchtable(1))
22     exactpitch = sortedpitchtable(1);
23     found = 1;
24 end
25
26 %%Se o tom exato nao for encontrado ainda considera o maior tom
       da tabela
27 if (found==0)
28     if (pitch > midpitchtable(length(midpitchtable)))
29         exactpitch = sortedpitchtable(length(sortedpitchtable));
30         found = 1;
31     end
32 end
33
34 %%Se o tom exato nao foi encontrado ainda, compare o a todos os
       valores
35 %da midpitchtable e a saida corresponde ao tom na tabela de tom
36 i = 1;
37 while(found==0)
38     if(pitch > midpitchtable(i) && pitch <= midpitchtable(i+1))
39         exactpitch = sortedpitchtable(i+1);
40         found=1;
41     end
42     i = i+1;
43 end
44
45 end
```

Código para Testes

```
fs = 44100; %% Frequencia de Amostragem
winsize = 4000; %% Tamanho da janela
NFFT = 4096; %% Numero de amostras da stft
noverlap = 2000; %Numero de amostras que nao sofrerao overlap
audio_duration = 3; %Duracao do audio
window = hanning(winsize); %Janela a ser utilizada na stft
signal = gravaAudio(fs,audio_duration);

[signal_transformed F T P] = spectrogram(signal, window,
    noverlap, NFFT, fs);

sound_corrected = pitchCorrector(signal_transformed, F, P,
    tabelaDoMaior());
sound_corrected = conj(sound_corrected); % Faz o conjugado
isignal = istft(sound_corrected,NFFT,winsize,noverlap); % Faz a
    transformada inversa do sinal

mostraEspectrograma(isignal,F,T); % Mostra o espectrograma

sound(isignal,fs)%toca o som
```