PROJETO 02 - DESENVOLVIMENTO DE FERRAMENTAS COMPUTACIONAIS PARA ANÁLISE TEMPO-FREQUÊNCIA

Davi de Alencar Mendes - 16/0026415

Engenharia Eletrônica, UnB-FGA, Brasília, Brasil

1. INTRODUÇÃO

A utilização de ferramentas como espectrograma (baseados na Transformada de Fourier de Tempo Curto - *STFT*) e escalograma (baseado em transformada Wavelet Discreta - *DWT*) provê informações de interesse a respeito do comportamento dinâmico do sinal. Essas informações podem ser utilizada para extração de características. O presente trabalho aborda o desenvolvimento computacional dessas ferramentas no ambiente de programação MATLAB.

2. PROJETO

2.1. Espectrograma

O espectrograma desenvolvido emprega a *STFT* para analisar o comportamento em frequência em versões janeladas do sinal. O algoritmo desenvolvido permite ainda a utilização de janelas configuráveis e sobreposição de janelas bem como variar a quantidade de pontos utilizados para cálculo da FFT. O algoritmo desenvolvido é apresentado anexo ao trabalho.

Sinal do tipo Chirp Linear Inicialmente, foi realizado um teste com um sinal do tipo Chirp que apresentação uma variação constante da frequência de acordo com $f(t) = 20 \times t$.

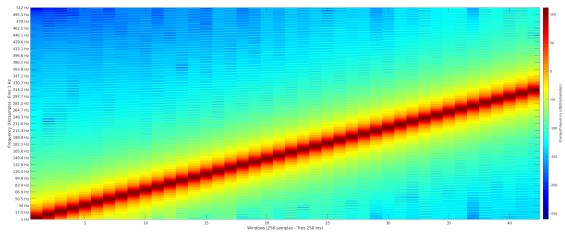

Fig. 1. Espectrograma para sinal do tipo Chirp Linear

Esteganografia - Gatos! Esteganografia é o estudo e uso de técnicas para ocultação de uma mensagem dentro de outra. Trata-se de uma forma de segurança por obscurantismo. Uma dessas abordagens consiste em esconder uma imagem em um segmento de aúdio de maneira que a imagem possa ser recuperada por meio visualização do espectrograma.

A música *Look* produzida pelo artista Venetian Snares faz o uso da técnica citada para esconder imagens de gatos no espectrograma [1]. A análise do conteúdo espectral da música contida em um arquivo de áudio MP3 revela os felinos escondidos!

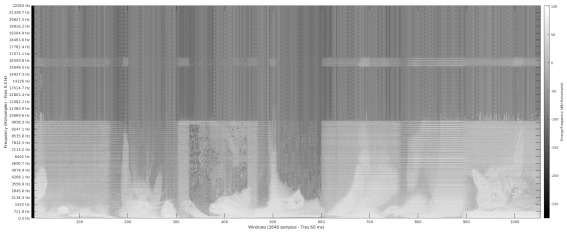

Fig. 2. Espectrograma para Venetian Snares - *Look*

O procedimento para recuperação da imagem consistiu em isolar um canal do áudio em estéreo para análise via espectrograma. A taxa de amostragem do sinal foi de 44.1 kHz, típica para sinais de áudio codificados em MP3. Ademais, o *bitrate* é de 95 kbps valor para o qual é obtida uma qualidade similar a de uma rádio FM [2]. Usando janelas de 60 ms com 50% de sobreposição do tipo Hamming foi possível os resultados mostrados na figura 2.

Em especial, percebe-se que metade do espectrograma apresenta baixa concentração de energia. Possivelmente a codificação em MP3, que impõe perdas, levou a redução da largura de banda do sinal causando grandes perdas no conteúdo da imagem. Entretanto, ainda é possível observar os gatos presentes nas regiões de baixa frequência (vide figura 3).

Fig. 3. Recorte do Espectrograma no qual observa-se um gato

3. ESCALOGRAMA

Com a aplicação de técnicas multiresolução é possível obter representações tempo-frequência para as quais há uma melhor divisão do plano tempo-frequência conforme a figura 7.

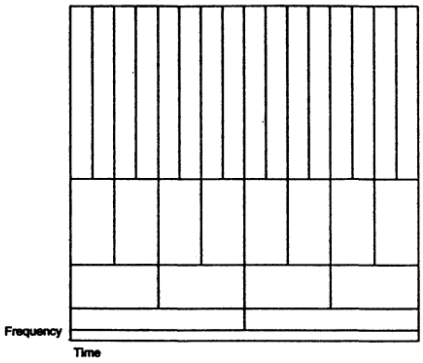

Fig. 4. Divisão Idealizada do plano Tempo-Frequência com wavelets - O tamanho da janela varia de maneira que em baixa frequências há melhor representação do tempo e em altas frequências há melhor resolução em frequência. Retirado de: *The World According to Wavelets*

O escalograma desenvolvido emprega transformada discreta de Wavelets para compor uma visualização bastante similar ao espectrograma. A visualização permite observar quanto cada coeficiente é representativo em energia percentual para toda a escala mostrada. O número de escalas é determinado de acordo com o número de níveis da decomposição QMF utilizada. Como a decomposição QMF gera uma transformada com coeficientes de diferentes comprimentos foi empregada uma interpolação linear usando a técnica do vizinho mais próximo para que os níveis com menos coeficientes adequem-se ao grid presente na imagem de acordo com a divisão em termos da frequência de amostragem.

Para testar foi utilizado um sinal do tipo $x(t) = t^2 \times -1^{fs \times t} + sen(10 \times t)$. Nota-se que há uma componente oscilatória de baixa frequência e uma componentes de alta frequência no sinal $x(t)$ mostrado na figura 5.

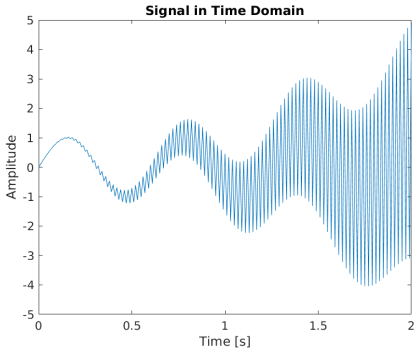

Fig. 5. sinal $x(t)$ representado no tempo

Utilizando uma decomposição de 2 níveis com a Wavelet de Haar é possível obter o escalograma mostrado na figura 6. Na menor escala nota-se o comportamento de alta frequência apresentado no sinal em domínio temporal. Para as demais escalas, nota-se um comportamento oscilatório que é também característico do sinal.

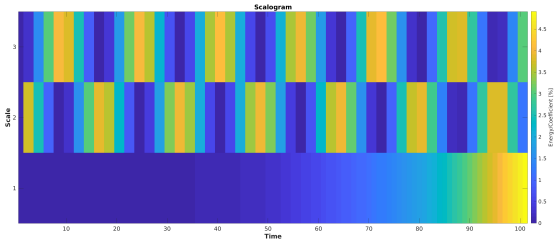

Fig. 6. Escalograma para x(t)

4. EXTRAÇÃO DE CARACTERÍSTICAS

Por último, foi desenvolvido uma algoritmo para extração de características de um sinal. A extração de características é executada em uma janela de tempo do sinal, podendo haver

sobreposição entre janelas. As características temporais escolhidas foram: (i) Valor Eficaz (RMS); (ii) Valor Médio Retificado (RMV); (iii) Desvio Padrão (STD). As características espectrais escolhidas são aplicadas a cada uma das bandas definidas em frequência, podendo haver sobreposição entre bandas adjacentes. Sendo elas: (i) Energia; (ii) Frequência Média; (iii) Frequência Mediana; (iv) Frequência Modal.

Os testes para avaliar a coerência do extrator de características foram realizados com a comparação entre uma gravação do set A e outra do set E. É evidente que haverá uma grande separação das características já que tratam-se de pacientes saudáveis *vs.* acometidos com epilepsia. Os resultados obtidos são mostrados nas figuras a seguir.

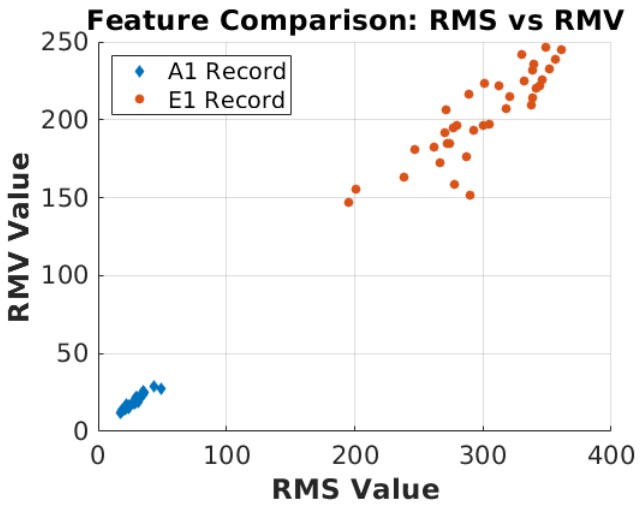

Fig. 7. Comparação de características temporais

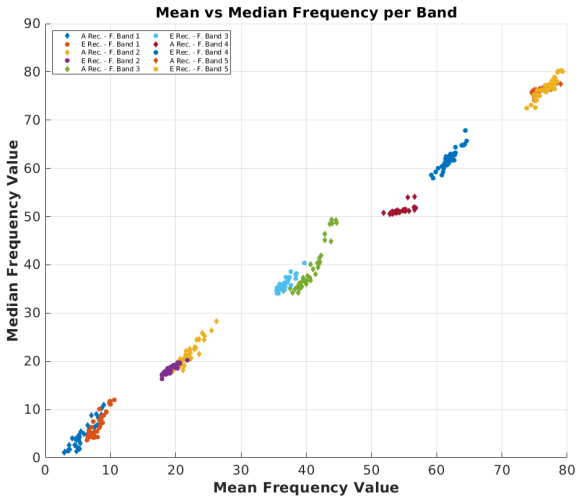

Fig. 8. Comparação de características espectrais

5. REFERENCES

[1] Bastwood Blog: The Aphex Face - `http://www.bastwood.com/?page_id=10`

[2] Sayood, K. (2017). Introduction to Data Compression 5th Ed. (Chapter 17: Audio Coding) Morgan Kaufmann.

Anexos

Espectrograma - STFT

```
%% stftSpectrogram: Spectrogram using Short-Time FT
function [varargout] = stftSpectrogram(x, fs, win, loverlap, nfft, zeropad)
        % Treat Inputs
        if ~exist('loverlap', 'var') || isempty(loverlap)
                loverlap = 0;
        end
        x = double(x(:));
        win = double(win(:));
        lwin = length(win);
        if ~exist('nfft', 'var') || isempty(nfft)
                nfft = lwin;
        end
        if ~exist('zeropad', 'var')
                zeropad = false;
        end

        % Reshape Signal without Zero Padding
        x = reshapeOverlap(x, lwin, loverlap, zeropad);

        % Split Apply Function
        spect_vals = @(col) 20*log10(abs(fftshift(fft(win.*col, nfft))).^2);
        x = splitapply(spect_vals, x, 1:size(x,2));
        % Remove the bottom half of the spectrogram
        x = x(1:floor(end/2),:);

        % Provide Outputs (Plot or Matrix)
        switch nargout
                case 0 % Plot Spectrogram
                        figure('units','normalized','outerposition',[0 0 1 1]);
                        imagesc(x);
                        colormap jet;
                        hcb = colorbar;
                        ylabel(hcb, 'Energy/Frequency (dB/(Hz/sample))');
                        set(get(gca,'YLabel'),'String', ['Frequency (Hz/sample) -
                            Fres ' num2str(fs/nfft, 2) ' Hz']);
                        ticks = linspace(1, size(x,1), 32);
                        yticks('manual');
                        yticks(ticks);
                        fv_label = arrayfun(@num2str, round(flip(ticks)*fs/nfft, 1),
                             'UniformOutput', false);
                        labelf = @(freq) [freq ' Hz'];
                        fv_label = cellfun(labelf, fv_label, 'UniformOutput', false)
                            ;
                        yticklabels('manual');
                        yticklabels(fv_label);
                        set(get(gca,'XLabel'),'String',['Windows (' int2str(lwin) '
                            samples - Tres ' num2str(1000*lwin/fs, 5) ' ms)']);
                case 1 % Return Spectrogram as matrix
                        varargout{1} = x;
                otherwise
```

```
                        error('Error in varargout! Provide the right number of
                            outputs.');
        end
end
```

Espectrograma - Teste com Sinal Chirp & Esteganografia

```
clear all; close all; clc;

% Generate Chirp Signal
fs = 1024;
fchirp = @(t) 20*t;
[x, t] = chirpGenerator(10, 8, fs, fchirp);

% Plot STFT Spectrogram
win = hann(256);
lwin = length(win);
stftSpectrogram(x, fs, win, lwin/4, 1024);
% saveas(gcf, 'linear_chirp_spectrogram.png');

% Test for Venetian Snares - Look (Hidden Cats in Spectrogram)
[y, fs] = audioread('venetian-snares_look.mp3');
stftSpectrogram(y(1:end-3.5*fs,2), fs, hamming(0.06*fs), 0.03*fs, 2*fs);

%% chirpGenerator: Generate Chirp Signals
function [x, t] = chirpGenerator(amp, duration, fs, chirp_fun)
        t = 0: 1/fs : duration;
        x = amp*sin(2*pi*chirp_fun(t).*t);
end
```

Escalograma - DWT

```
%% dwtScalogram: function description
function [varargout] = dwtScalogram(x, fs, filter_type, levels)
        % Treat Inputs
        if class(filter_type) == 'char'
                [h0, h1, ~, ~] = wfilters(filter_type);
        elseif class(filter_type) == 'cell'
                h0 = filter_type{1};
                h1 = filter_type{2};
        else
                error('Error in filter_type! Provide {h0, h1} as a cell array or use
                    MATLAB std filters!');
        end

        % Perform Decomposition
        [~, xdc, ~] = qmf_decomposition(x, h0, h1, levels);

        % Interpolate frame
        len_xdc = cellfun(@length, xdc);
        max_len = max(len_xdc(:));
        frame = zeros(levels+1,max_len);
        for l = 1:levels+1
                xi = linspace(0, length(x)*fs, length(xdc{l}));
                xq = linspace(0, length(x)*fs, max_len);
```

```
                y = xdc{l}.^2;
                y = 100 * y/sum(y);
                frame(l,:) = interp1(xi, y, xq, 'nearest');
        end

        figure('units','normalized','outerposition',[0 0 1 1]);
        imagesc(frame);
        title('Scalogram');
        colormap parula;
        hcb = colorbar;
        ylabel(hcb, 'Energy/Coefficient [%]');
        set(get(gca,'XLabel'),'String','Time');
        set(gca,'YTick', 1:levels+1);
        fv_label = arrayfun(@num2str, levels+1:-1:1, 'UniformOutput', false);
        yticklabels('manual');
        yticklabels(fv_label);
        set(get(gca,'YLabel'),'String','Scale');
        set(findall(gcf,'type','text'), 'FontSize', 22, 'fontWeight', 'bold');
        set(gca,'FontSize',16);

        % Provide Outputs
        if nargout == 1
                varargout{1} = frame;
        elseif nargout > 1
                error('Error with varargout outputs!');
        end
end
```

Escalograma - Rotina de Teste

```
clear all; close all; clc;

% Add Functions to PATH
addpath('../filterbanks/');
rehash path;

% Generate WEIRD Frequency Mod. Signal to test scalogram
fs = 100;
t = 0:1/fs:2;
x = (t.^2).*real((-1).^(fs.*t)) + sin(10*t);

% Plot Signal
figure;
plot(t,x);
title('Signal in Time Domain');
set(get(gca,'XLabel'),'String','Time [s]');
set(get(gca,'YLabel'),'String','Amplitude');
% saveas(gcf, 'scalogram_signal.png');

% Plot Scalogram
levels = 2;
sc = dwtScalogram(x, fs, 'haar', levels);
% saveas(gcf, 'scalogram.png');

% Plot Energy vs Time per Scale
```

```
figure;
legends = cell(1,levels+1);
for s = 1:levels+1
        plot(sc(s,:), 'LineWidth', 2);
        hold on;
        legends{s} = ['Scale' int2str(s)];
end
grid on;
legend(legends);
set(get(gca,'YLabel'),'String','Energy/Coefficient [%]');
set(get(gca,'XLabel'),'String','Wavelet Coefficients');
title('Energy/Coefficient per Scale');
% saveas(gcf, 'energy_scalogram.png');
```

Extração de Características

```
%% featureExtractor: function description
function [temporal_features, frequency_features] = featureExtractor(data, fs, win,
    loverlap, zeropad, iirfilter, bands, nbands, boverlap, maxfreq, fmaxorder, ftol,
    fnum, fden)

        if ~exist('iirfilter', 'var') || isempty(iirfilter)
                iirfilter = @butter;
        end
        if ~exist('fmaxorder', 'var') || isempty(fmaxorder)
                fmaxorder = 15;
        end
        if ~exist('boverlap', 'var') || isempty(boverlap)
                boverlap = 0.25;
        end
        if ~exist('ftol', 'var') || isempty(ftol)
                ftol = 1e-5;
        end
        if ~exist('maxfreq', 'var') || isempty(maxfreq)
                maxfreq = fs/2;
        end
        if ~exist('zeropad', 'var') || isempty(zeropad)
                zeropad = false;
        end
        lwin = length(win);
        % Reshape Signal
        data = reshapeOverlap(data, lwin, loverlap, zeropad);
        s_data = size(data);
        % Perform Window Multiplication
        win_matrix = repelem(win, 1, s_data(2));
        data = data.*win_matrix;

        % Extract temporal features
        rMeanVal = @(w) mean(abs(w)); % Rectified Mean Value
        RMS = splitapply(@rms, data, 1:s_data(2));
        STD = splitapply(@std, data, 1:s_data(2));
        RMV = splitapply(rMeanVal, data, 1:s_data(2));

        %% Extract frequency features
        % Evaluate Filtering bands
```

```
        if ~exist('bands', 'var') || isempty(bands)
                maxnfreq = maxfreq / fs;
                bands = [ (0 : maxnfreq / nbands : maxnfreq * (1 - 1/nbands)).' (
                    maxnfreq / nbands : maxnfreq / nbands : maxnfreq).'];
                bands(2:end,1) = bands(2:end,1) - boverlap / 2 * maxnfreq / nbands;
                bands(1:end-1,2) = bands(1:end-1,2) + boverlap / 2 * maxnfreq /
                    nbands;
        end
        % Design Filters
        if (~exist('fnum', 'var') || isempty(fnum)) && (~exist('fnum', 'var') ||
            isempty(fnum))
                fnum = cell(1, nbands);
                fden = cell(1, nbands);
                for b = 1 : nbands
                        [fnum{b}, fden{b}] = iir_design(iirfilter, fmaxorder, ftol,
                            bands(b, :));
                end
        end
        % Perform Filtering
        nfft = 2^(nextpow2(lwin));
        freq = linspace(0,fs/2, nfft);
        getEnergy = @(w) norm(w)^2;
        magfft = @(w) abs(fft(w, nfft));
        fdata = zeros(s_data(1), s_data(2), nbands);
        fftdata = zeros(nfft, s_data(2), nbands);
        E = zeros(s_data(2), nbands);
        MEDFREQ = zeros(s_data(2), nbands);
        MEANFREQ = zeros(s_data(2), nbands);
        MODALFREQ = zeros(s_data(2), nbands);
        for b = 1:nbands
                fdata(:,:,b) = filter(fnum{b}, fden{b}, data, [], 1); % Filtering
                    along the cols
                E(:,b) = splitapply(getEnergy, fdata(:,:,b), 1:s_data(2)); % Obtain
                    Energy
                MEANFREQ(:,b) = meanfreq(fdata(:,:,b), fs); % Obtain Mean Frequency
                MEDFREQ(:,b) = medfreq(fdata(:,:,b), fs); % Obtain Median Frequency
                fftdata(:,:,b) = splitapply(magfft, fdata(:,:,b), 1:s_data(2)); %
                    Obtain Frequency Response
                [~,I] = max(fftdata(:,:,b), [], 1);
                MODALFREQ(:,b) = freq(I);
        end

        table_names = {'RMS', 'RMV', 'STD', 'E', 'MEDFREQ', 'MEANFREQ', 'MODALFREQ'
            };
        temporal_features = table(RMS, RMV, STD, 'VariableNames', table_names(1:3));
        frequency_features = table(E, MEDFREQ, MEANFREQ, MODALFREQ, 'VariableNames',
            table_names(4:end));
end

% iir_design: function description
function [num, den] = iir_design(iir_type, maximum_order, filter_tol, bands)
        not_finished = 1;
        order = maximum_order;
        bands_ = bands;
```

```
        mode = 'bandpass';
        if bands(1) == 0
                bands_ = bands(2);
                mode = 'low';
        end
        if bands(2) == 0.5
                bands_ = bands(1);
                mode = 'high';
        end
        while not_finished
                [num, den] = iir_type(order, bands_ * 2, mode);
                not_finished = any(abs(roots(den)) > 1 - filter_tol);
                order = order - 1;
        end
end
```

Extração de Características - Teste com sinais de EEG

```
clear all; close all; clc;

A1 = loadSetRecords('A', 1, '../eeg_signals/');
E1 = loadSetRecords('E', 1, '../eeg_signals/');
fs = 173.61;
lwin_sec = 0.8;
win = hamming(round(lwin_sec*fs));
loverlap = round(lwin_sec*fs/5);
nbands = 5;

% [features_table] = featureExtractor(data, fs, win, loverlap, zeropad, iirfilter,
   bands, nbands, boverlap, maxfreq, fmaxorder, ftol, fnum, fden)
[A_tf, A_ff] = featureExtractor(A1, fs, win, loverlap, false, [], [], nbands);
[E_tf, E_ff] = featureExtractor(E1, fs, win, loverlap, false, [], [], nbands);

% Plot Results
figure;
scatter(A_tf.RMS, A_tf.RMV, 'filled', 'd');
hold on;
grid on;
scatter(E_tf.RMS, E_tf.RMV, 'filled', 'o');
legend('A1 Record', 'E1 Record', 'Location', 'northwest');
set(get(gca,'XLabel'),'String','RMS Value');
set(get(gca,'YLabel'),'String','RMV Value');
title('Feature Comparison: RMS vs RMV');
set(findall(gcf,'type','text'), 'FontSize', 22, 'fontWeight', 'bold');
set(gca, 'FontSize', 16);

figure;
legend_titles = {};
for b = 1:nbands
        scatter(A_ff.MEANFREQ(:,b), A_ff.MEDFREQ(:,b), 'filled', 'd');
        hold on;
        scatter(E_ff.MEANFREQ(:,b), E_ff.MEDFREQ(:,b), 'filled', 'o');
        legend_titles{end+1} = ['A Rec. - F. Band ' int2str(b)];
        legend_titles{end+1} = ['E Rec. - F. Band ' int2str(b)];
end
```

```
l = legend(legend_titles, 'Location', 'northwest', 'NumColumns', 2);
set(get(gca,'XLabel'),'String','Mean Frequency Value');
set(get(gca,'YLabel'),'String','Median Frequency Value');
title('Mean vs Median Frequency per Band');
set(findall(gcf,'type','text'), 'FontSize', 22, 'fontWeight', 'bold');
set(gca, 'FontSize', 16);
l.FontSize = 8;
grid on;

% legend('A1 Record', 'E1 Record', 'Location', 'northwest');
% set(get(gca,'XLabel'),'String','RMS Value');
% set(get(gca,'YLabel'),'String','RMV Value');
% title('Feature Comparison: RMS vs RMV');
% set(findall(gcf,'type','text'), 'FontSize', 22, 'fontWeight', 'bold');
% set(gca,'FontSize',18);



%% loadSetRecords: Load records from specified dataset
function [recs] = loadSetRecords(rset, nrecord, dataPath)
        % Treat Inputs
        if isempty(nrecord)
                nrecord = 1:100;
        end
        if ~exist('dataPath', 'var')
                dataPath = [];
        end

        % Load Signals
        recs = zeros(4097, length(nrecord));
        file_prefix = {'Z', 'O', 'N', 'F', 'S'};
        for n = 1:length(nrecord)
                rec_fname = sprintf('%sset%c/%c%.3d.txt', dataPath, rset,
                   file_prefix{rset-'A'+1}, nrecord(n));
                recs(:,n) = load(rec_fname, '-ascii');
        end
end
```