TRƯỜNG ĐẠI HỌC BÁCH KHOA TP.HCM

KHOA ĐIỆN – ĐIỆN TỬ

BỘ MÔN ĐIỀU KHIỂN VÀ TỰ ĐỘNG



BÁO CÁO ĐỒ ÁN

ỨNG DỤNG XỬ LÝ NHIỄU VÀ ÂM VANG

TRONG NHẬN DẠNG TIẾNG NÓI TỪ XA

Giáo viên hướng dẫn: Tiến sĩ Phạm Việt Cường

Sinh viên thực hiện: Võ Thanh Hải

MSSV: 1411084

*TP.HCM, Tháng 6/2017*

Mục lục

🙧🟐🙥

Nội dung Trang

|  |  |  |
| --- | --- | --- |
| 1. Mục tiêu tiểu luận | | 2 |
| 1. Giới thiệu thuật toán | | 3 |
|  | 1. Thuật toán mô phỏng nhiễu và âm vang | 3 |
|  | 1. Thuật toán xử lý nhiễu | 6 |
|  | 1. Thuật toán xử lý âm vang | 9 |
| 1. Lập trình thuật toán bằng Matlab | | 11 |
| 1. Lập trình mô phỏng nhiễu và âm vang | | 11 |
| 1. Lập trình xử lý âm vang | | 13 |
| 1. Lập trình xử lý âm nhiễu | | 16 |
| 1. Lập trình giao diện bằng Matlab GUIDE | | 19 |
| 1. Kết quả xử lý và tự đánh giá | | 24 |
| 1. Giáo viên hướng dẫn đánh giá | | 26 |

1. **MỤC TIÊU TIỂU LUẬN:**

Bên cạnh chữ viết, tiếng nói là một trong hai thành phần cơ bản nhất của ngôn ngữ tự nhiên. Tiếng nói hình thành trước chữ viết và phát triển trong suốt lịch sử phát triển của loài người. Tiếng nói đã, đang và sẽ luôn là phương tiện giao tiếp chủ yếu của con người bởi lẽ giao tiếp bằng tiếng nói là cách thức đơn giản, tự nhiên và đóng vai trò quan trọng trong đời sống con người.

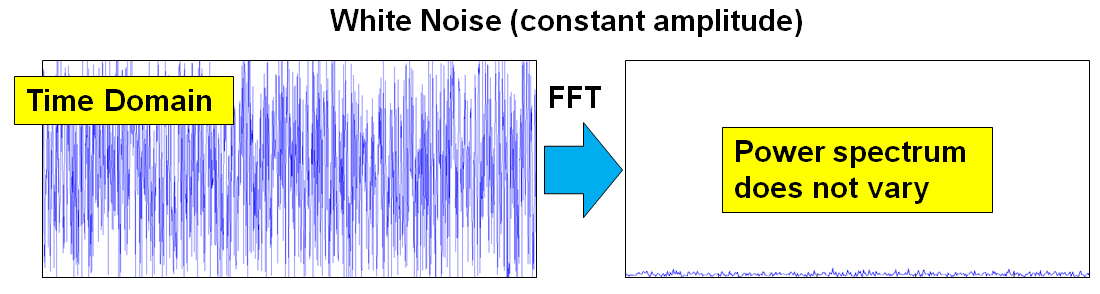
Ngày nay, với sự phát triển của khoa học kỹ thuật, máy móc đang dần thay thế các công việc cơ bản của con người nhưng việc giao tiếp giữa người với máy vẫn là những thao tác phức tạp và con người cần được đào tạo để trao đổi thông tin với máy. Do đó, việc nghiên cứu phát triển các loại thiết bị có khả năng giao tiếp với máy móc qua giọng nói đã và đang nhận được sự quan tâm của nhiều nhà khoa học trên thế giới, cũng như người tiêu dùng. Các ứng dụng này đã được áp dụng vào điện thoại di động, máy tính, … tuy nhiên mức độ phổ biến vẫn còn rất hạn chế.

Vào những năm 1970, ngành khoa học xử lý tiếng nói đã trở nên phổ biến và bùng nổ với nhiều thành công. Đến hiện tại, các thiết bị điện tử hỗ trợ cho cuộc sống con người đều được trang bị tính năng điều khiển bằng giọng nói để tương tác thuận tiện hơn với người dùng. Tuy nhiên, hầu hết trong số đó đều ghi âm nguồn gần; điều này đôi khi gây bất tiện cho người sử dụng. Điều này dẫn đến việc nghiên cứu và phát triển ứng dụng nguồn âm ở xa.

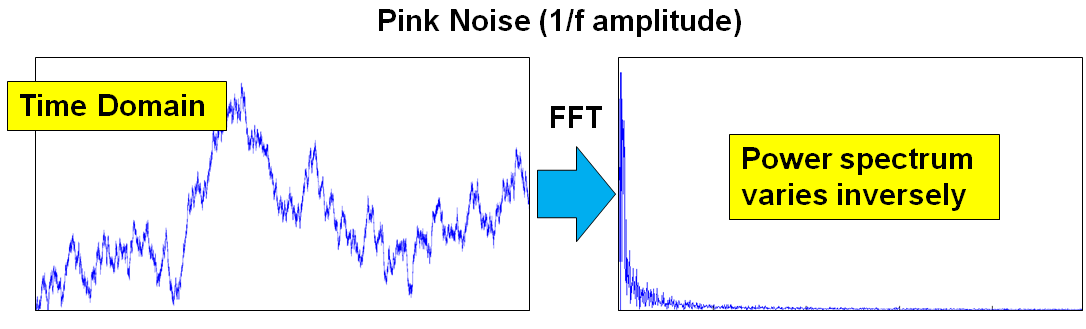
Mục tiêu đồ án này là nghiên cứu và lập trình các thuật toán xử lý nhiễu và âm vang để cải thiện chất lượng của tiếng nói thu được ở tầm xa. Bên cạnh đó, em cũng tìm hiểu thêm về quá trình hình thành nhiễu và âm vang để mô phỏng tín hiệu tiếng nói bị méo dạng do các nguyên nhân khác nhau vì hạn chế về các thiết bị ghi âm nguồn xa trong thực tế.

1. **GIỚI THIỆU THUẬT TOÁN:**
   * + 1. **Thuật toán mô phỏng nhiễu và âm vang:**
2. Mô phỏng nhiễu:

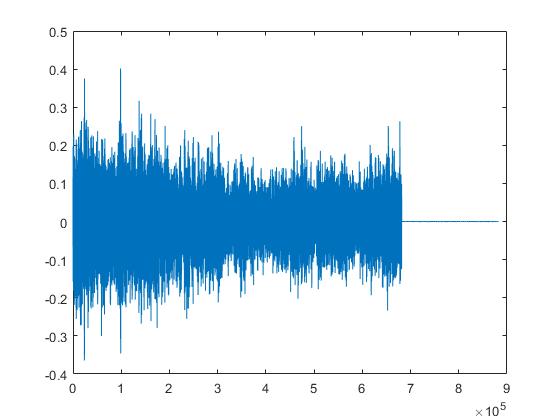
* Định nghĩa: Nhiễu là âm thanh không mong muốn và bị xem là tác nhân gây khó chịu hoặc gây rối cho các quá trình dựa vào thính giác. Từ quan điểm vật lý, nhiễu không thể phân biệt được với âm thanh vì cả hai đều là sự rung động của các phần tử trong môi trường truyền dẫn. Con người có khả năng phân biệt âm thanh với nhiễu nhờ vào các quá trình bộ não tiếp thu và học hỏi.
* Các loại nhiễu mô phỏng:
* ***Nhiễu trắng:*** là năng lượng truyền ngẫu nhiên trải dài trên dải tần số rộng với tất cả giá trị tần số.



* ***Nhiễu hồng:*** là tín hiệu có phổ tần số mà mật độ phổ năng lượng tỉ lệ nghịch với tần số của tín hiệu.



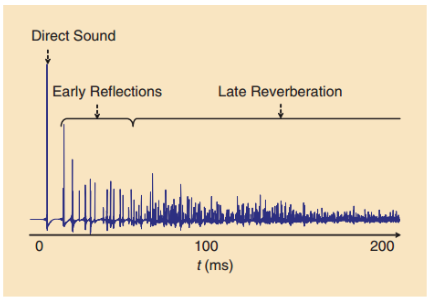
* ***Nhiễu lảm nhảm:*** xuất hiện chủ yếu ở các văn phòng mở, có tác dụng che phủ tín hiệu tiếng nói.



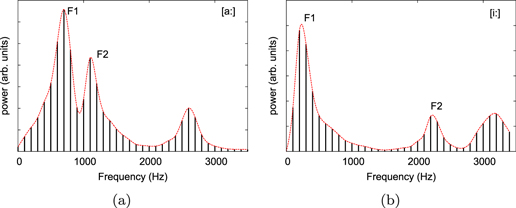
* Ảnh hưởng của nhiễu: làm méo dạng tín hiệu âm thanh dẫn đến làm sai lệch kết quả nhận dạng.

1. Mô phỏng âm vang:

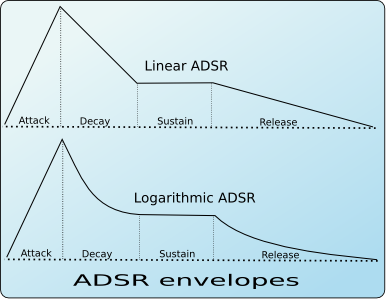
* ***Phòng vang:*** là phòng được thiết kế để tạo ra một trường âm thanh khuếch tán hoặc ngẫu nhiên (thường có phân bố đều về năng lượng và có hướng âm thanh khuếch tán ngẫu nhiên trong một thời gian ngắn).
* ***Âm vang:*** được tạo ra khi một âm thanh bị phản xạ tạo nên một số lượng lớn các tín hiệu phản chiếu và sau đó bị triệt tiêu bởi sự hấp thụ của bề mặt các vật thể trong không gian. Âm vang gồm 2 loại là ***tín hiệu phản xạ sớm*** và ***âm vang muộn***.



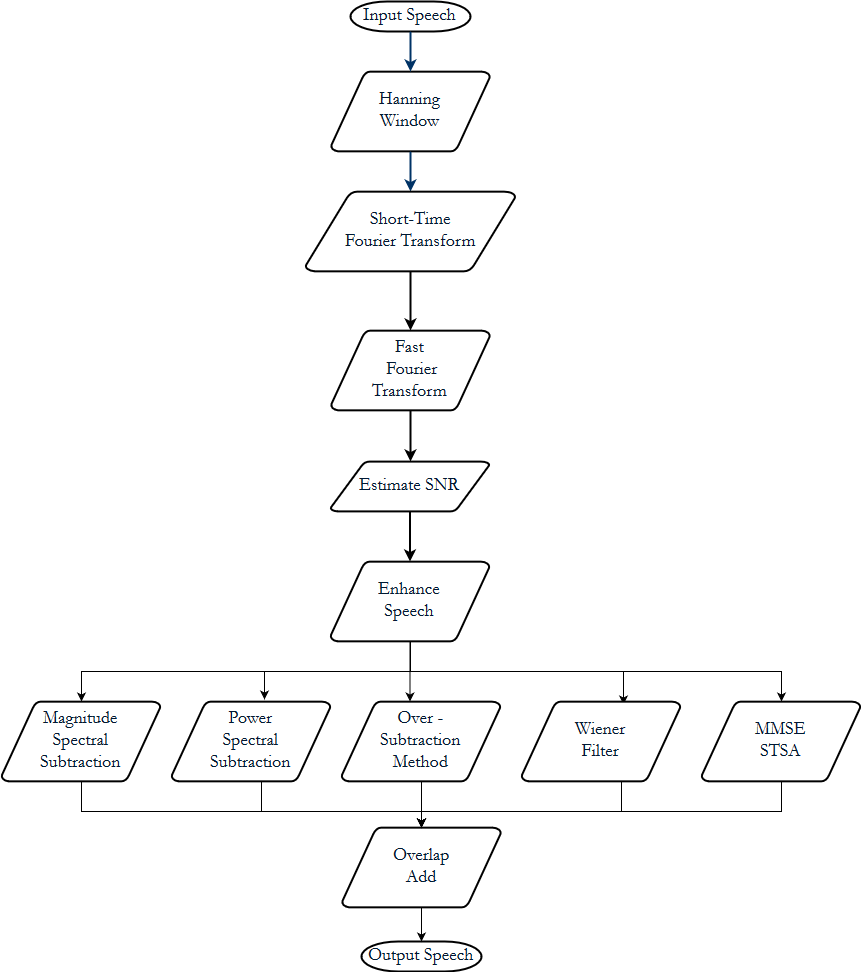
* Ảnh hưởng của ***tín hiệu phản xạ sớm***: làm tín hiệu âm thanh trong cả hai miền thời gian và tần số trở nên kém rõ ràng, làm phẳng đi sự thay đổi giữa các ***formant*** – các đỉnh của phổ tín hiệu âm thanh – khiến cho việc nhận dạng tiếng nói khó khăn hơn.



* Ảnh hưởng của ***âm vang muộn***: lấp đầy khoảng trống trong ***envelope* -** sự thay đổi biên độ của sóng âm thanh trong miền thời gian **–** của tiếng nói, giảm sự điều chế envelope ở tần số thấp.



* + - 1. **Thuật toán xử lý nhiễu:**
* Thuật toán nén nhiễu:



* Công thức ***Trừ Phổ Biên độ***:



* Công thức ***Trừ Phổ Năng lượng***:



* Công thức ***Trừ Phổ Năng lượng thích nghi***:



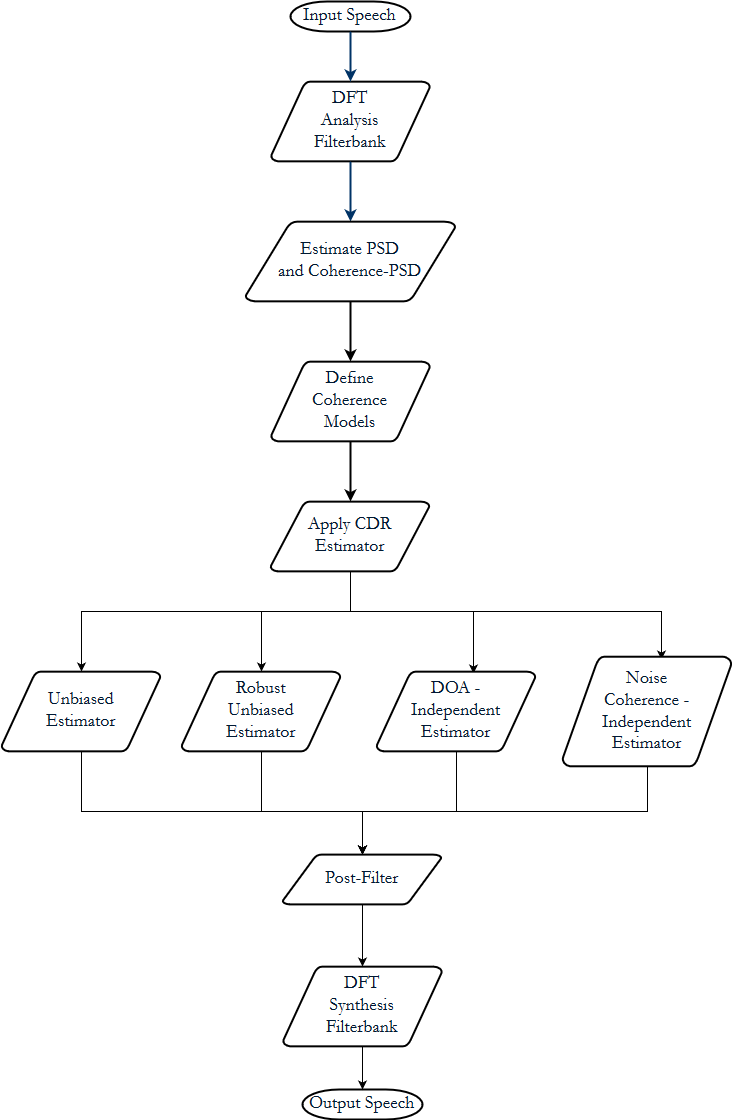
* Công thức ***Bộ lọc Wiener***:



* Công thức ***MMSE – STSA***:



* + - 1. **Thuật toán xử lý âm vang:**
* Thuật toán xử lý âm vang dựa vào ***Ước lượng năng lượng CDR***:



* Công thức ***Bộ ước lượng không thiên vị***:



* Công thức ***Bộ ước lượng không thiên vị tăng cường***:



* Công thức ***Bộ ước lượng không phụ thuộc DOA***:



* Công thức ***Bộ ước lượng không phụ thuộc tương quan với Nhiễu***:



1. **LẬP TRÌNH THUẬT TOÁN BẰNG MATLAB:**
   * + 1. **Lập trình mô phỏng nhiễu và âm vang:**

* Lập trình mô phỏng âm vang:
* Khai báo thông số:

SetupStruc.Fs = 16000; % sampling frequency in Hz

SetupStruc.c = 343; % propagation speed of acoustic waves in m/s

SetupStruc.room = [3 4 2.5]; % room dimensions in m

SetupStruc.mic\_pos = [1.6 1.0 1.3; % [x y z]

1.4 1.0 1.3; % positions of microphone array [m]

1.2 1.0 1.3;

1.4 1.2 1.3];

SetupStruc.src\_traj = [1.5 3 1.7];

SetupStruc.T60 = 0.2; % SetupStruc.T20 = 0.15;

SetupStruc.abs\_weights = [0.1 0.2 0.3 0.2 0.1 0.2];

* Tính toán đáp ứng xung:

beta = sqrt(1-alpha);

nMics = size(micpos,1); % number of microphones

nSPts = size(straj ,1); % number of source trajectory points

%% Compute RIR bank

RIR\_cell = cell(nMics,nSPts); % pre-allocate cell array

for mm = 1:nMics,

X\_rcv = micpos(mm,:);

% compute ISM room impulse response for each source-receiver combinations

for tt = 1:nSPts,

X\_src = straj(tt,:);

RIR\_cell{mm,tt} = ISM\_RoomResp(Fs,beta,rttype,rtval,X\_src,X\_rcv,room,

'SilentFlag',1,'c',cc,'Delta\_dB',Delta\_dB);

end

end

* Mô phỏng âm vang:

load(RIR\_cell); % load RIR bank 'RIR\_cell' from file

nSamp = length(SrcSignal); % total number of samples in the audio data

nMics = size(RIR\_cell,1); % number of mics

nFrames = size(RIR\_cell,2); % number of trajectory points

nSampPerFrame = ceil(nSamp/nFrames); % total number of frames in the source signal

% pre-determine max number of samples in the resulting mic signals

maxEndPt = 0;

for tt = 1:nFrames

if tt == nFrames

FrameStopInd = nSamp;

else

FrameStopInd = tt\*nSampPerFrame;

end

for mm = 1:nMics

RIRlen = length(RIR\_cell{mm,tt});

% endpoint of last source audio sample in current frame convolved with RIR

maxEndPt = max(maxEndPt,FrameStopInd+RIRlen-1);

end

end

%% Compute audio data

AuData = zeros(maxEndPt,nMics);

for tt = 1:nFrames

FrameStartInd = (tt-1)\*nSampPerFrame+1; % Start/end indices of the current frame

if tt == nFrames

FrameStopInd = nSamp;

else

FrameStopInd = tt\*nSampPerFrame;

end

FrameData = SrcSignal(FrameStartInd:FrameStopInd); % get one frame of data

% Compute the received signal by convolving the source signal with the RIR

for mm = 1:nMics,

hh = RIR\_cell{mm,tt};

RIRlen = length(hh); % length of current RIR

EndIndex = FrameStopInd+RIRlen-1; % max length for the current convolution.

AuData(FrameStartInd:EndIndex,mm) =

AuData(FrameStartInd:EndIndex,mm) + freq\_conv(hh,FrameData);

end

end

%% Truncate signals

if TruncateMicSig == 1

AuData = AuData(1:nSamp,:);

end

* Lập trình mô phỏng nhiễu:

noise\_file = 'Degradation/Noise/noise\_white.wav';

[noise, fs\_noise] = audioread(noise\_file);

if fs\_noise ~= fs

noise = resample(noise, fs, fs\_noise);

end

% Adjust input SNR using noise\_gain

noise\_gain = 0.7;

noise = noise\_gain\*noise(1:length(sig));

% Add noise n to signal x

noisySig = sig + noise;

* + - 1. **Lập trình xử lý âm vang:**
* Xử lý song kênh:

%% Configuration

% Filterbank

cfg.K = 512; % FFT size

cfg.N = 128; % frame shift

cfg.Lp = 1024; % prototype filter length

load('prototype\_K512\_N128\_Lp1024.mat');

% Algorithm and scenario configuration

cfg.fs = 16000; % sampling rate [Hz]

cfg.c = 342; % speed of sound [m/s]

cfg.d\_mic = 0.02; % mic spacing [m]

cfg.TDOA = 0.00e-04; % Time Difference of Arrival (ground truth)

cfg.nr.lambda = 0.68; % smoothing factor for PSD estimation

cfg.nr.mu = 1.3; % noise overestimation factor

cfg.nr.floor = 0.1; % minimum gain

cfg.nr.alpha = 2; cfg.nr.beta = 1; % Wiener filter

cfg.estimator = @estimate\_cdr\_robust\_unbiased; % unbiased, "robust" estimator

%% Signal Processing

% Analysis filterbank

X = DFTAnaRealEntireSignal(x, cfg.K, cfg.N, p);

% Estimate PSD and coherence

Pxx = estimate\_psd (X, cfg.nr.lambda);

Cxx = estimate\_cpsd(X(:,:,1), X(:,:,2), cfg.nr.lambda)./sqrt(Pxx(:,:,1).\*Pxx(:,:,2));

frequency = linspace(0, cfg.fs/2, cfg.K/2+1)'; % frequency axis

% Define coherence models

Css = exp(1j \* 2 \* pi \* frequency \* cfg.TDOA); % target signal coherence

Cnn = sinc(2 \* frequency \* cfg.d\_mic/cfg.c); % diffuse noise coherence

% Apply CDR estimator (=SNR)

SNR = cfg.estimator(Cxx, Cnn, Css);

SNR = max(real(SNR),0);

weights = spectral\_subtraction(SNR, cfg.nr.alpha, cfg.nr.beta, cfg.nr.mu);

weights = max(weights, cfg.nr.floor);

weights = min(weights, 1);

% Postfilter input is computed from averaged PSDs of both microphones

Postfilter\_input = sqrt(mean(abs(X).^2,3)) .\* exp(1j\*angle(X(:,:,1)));

% Apply postfilter

Processed = weights .\* Postfilter\_input;

% Synthesis filterbank

y = DFTSynRealEntireSignal(Processed,cfg.K,cfg.N,p);

* Xử lý tổng quát:

inSig(:,1) = degradedSig(:,1);

for channel = 1:col

inSig(:,2) = degradedSig(:,channel);

outSig = cdrDereverb(inSig);

derevSig(:,channel) = outSig/max(outSig);

end

* ***Bộ ước lượng không thiên vị***:

function CDR = estimate\_cdr\_unbiased(Cxx,Cnn,Css)

Css = bsxfun(@times, ones(size(Cxx)), Css);

Cnn = bsxfun(@times, ones(size(Cxx)), Cnn);

% limit the magnitude of Cxx to prevent numerical problems

magnitude\_threshold = 1-1e-10;

critical = abs(Cxx) > magnitude\_threshold;

Cxx(critical) = magnitude\_threshold .\* Cxx(critical) ./ abs(Cxx(critical));

CDR = real(exp(-1j\*angle(Css)).\*Cnn - (exp(-1i\*angle(Css)).\*Cxx))./...

(real(exp(-1i\*angle(Css)).\*Cxx) - 1);

% Ensure we don't get any negative or complex results due to numerical effects

CDR = max(real(CDR),0);

end

* ***Bộ ước lượng không thiên vị tăng cường***:

function CDR = estimate\_cdr\_robust\_unbiased(Cxx,Cnn,Css)

Css = bsxfun(@times, ones(size(Cxx)), Css);

Cnn = bsxfun(@times, ones(size(Cxx)), Cnn);

% limit the magnitude of Cxx to prevent numerical problems

magnitude\_threshold = 1-1e-10;

critical = abs(Cxx)>magnitude\_threshold;

Cxx(critical) = magnitude\_threshold .\* Cxx(critical) ./ abs(Cxx(critical));

CDR = 1./(-abs(Cnn-exp(1j\*angle(Css)))./(Cnn.\*cos(angle(Css))-1)).\*...

abs((exp(-1j\*angle(Css)).\*Cnn - (exp(-1i\*angle(Css)).\*Cxx))./...

(real(exp(-1i\*angle(Css)).\*Cxx) - 1));

% Ensure we don't get any negative or complex results due to numerical effects

CDR = max(real(CDR),0);

end

* ***Bộ ước lượng không phụ thuộc DOA***:

function CDR = estimate\_cdr\_nodoa(Cxx,Cnn,~)

Cnn = bsxfun(@times, ones(size(Cxx)), Cnn); % extend to dimension of Cxx

% limit the magnitude of Cxx to prevent numerical problems

magnitude\_threshold = 1-1e-10;

critical = abs(Cxx)>magnitude\_threshold;

Cxx(critical) = magnitude\_threshold .\* Cxx(critical) ./ abs(Cxx(critical));

CDR = ( -(abs(Cxx).^2 + Cnn.^2.\*real(Cxx).^2 - Cnn.^2.\*abs(Cxx).^2 - 2.\*Cnn.\*real(Cxx) + Cnn.^2).^(1/2) - abs(Cxx).^2 + Cnn.\*real(Cxx) )...

./(abs(Cxx).^2-1);

% Ensure we don't get any negative or complex results due to numerical effects

CDR = max(real(CDR),0);

end

* ***Bộ ước lượng không phụ thuộc tương quan với Nhiễu***:

function CDR = estimate\_cdr\_nodiffuse(Cxx,~,Css)

Css = bsxfun(@times, ones(size(Cxx)), Css);

% limit the magnitude of Cxx to prevent numerical problems

magnitude\_threshold = 1-1e-10;

critical = abs(Cxx) > magnitude\_threshold;

Cxx(critical) = magnitude\_threshold .\* Cxx(critical) ./ abs(Cxx(critical));

CDR = imag(Cxx)./(imag(Css) - imag(Cxx));

CDR(imag(Css)./imag(Cxx)<=1) = Inf;

CDR(imag(Css)./imag(Cxx)<=0) = 0;

% Ensure we don't get any negative or complex results due to numerical effects

CDR = max(real(CDR),0);

end

* + - 1. **Lập trình xử lý nhiễu:**
* Xử lý đơn kênh:

%% Parameters

data.denoise\_type = 'spec\_sub\_power';

%% Make window

win\_t = 0.03; % window size in seconds

win\_s = round(fs\*win\_t); % window size in samples

if (mod(win\_s, 2) == 0) % make odd

win\_s = win\_s - 1;

end

win = hann(win\_s);

% normalize it so the power is equal to its length

win = win\*sqrt(length(win)/sum(win.^2));

%% STFT of signal

hop\_size = (win\_s-1)/2; % hop size (half of window size)

num\_frames = floor(length(y)/hop\_size);

% over sample to prevent time aliasing of filters

nfft = 8\*win\_s;

data.est\_Sy = zeros(nfft, num\_frames); % estimate of clean speech spectrum

data.est\_Pn = zeros(nfft, num\_frames); % estimate of noise power spectrum

data.est\_Mn = zeros(nfft, num\_frames); % estimate of noise magnitude spectrum

for i = 1: num\_frames

data.iteration = i;

%% FFT

s = (i-1)\*hop\_size + 1; % start index

e = min(s+win\_s-1, length(y)); % end index

if (mod(e-s+1, 2) == 0)

e = e-1; % make length odd

end

l = e-s+1; % length of windowed signal

% zero-pad, zero-phase windowing

yzp = zpzpwin(y(s:e), win(1:l), nfft);

data.Sy = fft(yzp); % FFT

%% Estimate power spectrum of noise

data = noiseEstimationSNR(data);

[data, est\_Sy] = enhanceSpeech(data);

data.est\_Sy(:, i) = est\_Sy;

end

%% Overlap add

dc = sum(win(1:hop\_size:end));

yhat = real(invmyspectrogram2(data.est\_Sy, hop\_size, win\_s));

yhat = yhat/dc;

yhat = yhat(1:length(y));

%% Calculate SNR

snr = 10\*log10(sum(y.^2)/sum((y-yhat').^2));

* Xử lý tổng quát:

[row, col] = size(derevSig);

denoisedSig = zeros(row, col);

for channel = 1:col

denoisedSig(:,channel) = noiseReduction(derevSig(:,channel), fs);

denoisedSig(:,channel) = denoisedSig(:,channel)/max(denoisedSig(:,channel));

end

* ***Trừ Phổ Biên độ***:

estMagX = max(abs(data.Sy) - data.est\_Mn, 1e-10);

estAngleX = angle(data.Sy);

est\_Sx = estMagX.\*exp(complex(0,1)\*estAngleX);

* ***Trừ Phổ Năng lượng***:

estPowX = max(abs(data.Sy).^2 - data.est\_Pn, 1e-10);

estAngleX = angle(data.Sy);

est\_Sx = sqrt(estPowX).\*exp(complex(0,1)\*estAngleX);

* ***Trừ Phổ Năng lượng thích nghi***:

snr = 10\*log10(sum(abs(data.Sy).^2)./sum(data.est\_Pn));

if (snr >= -5 && snr <= 20)

alpha = -3/20\*snr + 4;

elseif (snr > 20)

alpha = 1;

else

alpha = 4.75;

end

beta = 0.002;

estPowX = max(abs(data.Sy).^2 - alpha\*data.est\_Pn, beta\*data.est\_Pn);

estAngleX = angle(data.Sy);

est\_Sx = sqrt(estPowX).\*exp(complex(0,1)\*estAngleX);

* ***Bộ lọc Wiener***:

gamma = abs(data.Sy).^2./data.est\_Pn;

% using inst\_snr

inst\_snr = max(gamma - 1, 1e-10);

% using decision directed approach

if data.iteration == 1

ksi = inst\_snr;

else

alpha = 0.98;

ksi = alpha\*abs(data.prevEstSx).^2./data.prevEstPn + (1-alpha)\*inst\_snr;

end

H = ksi./(1+ksi);

est\_Sx = H.\*data.Sy;

% save previous estimates

data.prevEstSx = est\_Sx;

data.prevEstPn = data.est\_Pn;

* ***MMSE*** – ***STSA***:

gamma = abs(data.Sy).^2./data.est\_Pn;

gamma = min(gamma, 100);

% using inst\_snr

inst\_snr = max(gamma - 1, 1e-10);

% using decision directed approach

if data.iteration == 1

ksi = inst\_snr;

else

alpha = 0.98;

ksi = alpha\*abs(data.prevEstSx).^2./data.prevEstPn + (1-alpha)\*inst\_snr;

end

nu = ksi./(1+ksi).\*gamma;

H = sqrt(pi)/2\*sqrt(nu)./gamma.\*exp(-nu/2).\*((1+nu).\*besseli(0,nu/2) + …

nu.\*besseli(1,nu/2));

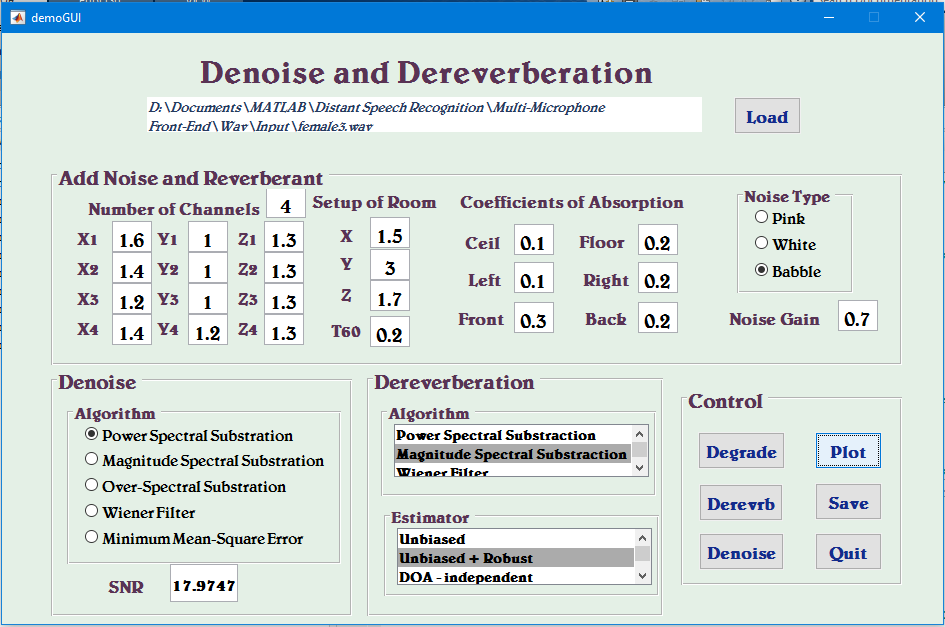
est\_Sx = H.\*data.Sy;

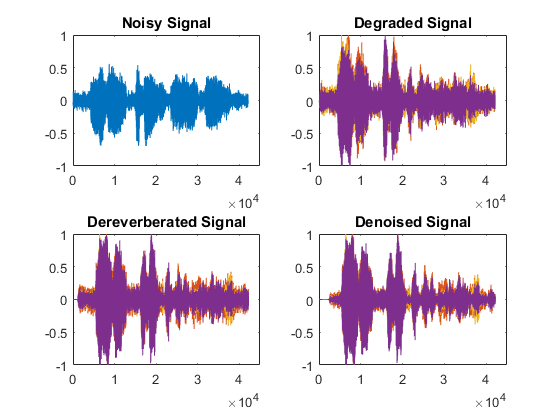
% save previous estimates

data.prevEstSx = est\_Sx;

data.prevEstPn = data.est\_Pn;

1. **CHƯƠNG TRÌNH GIAO DIỆN GIÁM SÁT TRÊN MÁY TÍNH:**
2. Giao diện chương trình Matlab GUIDE:





1. Chương trình giao diện:

* Đọc nút nhấn ***Load***:

function loadButton\_Callback(hObject, eventdata, handles)

[filename, pathname] = uigetfile({'\*.wav';'\*.mp3';'\*.\*'},...

'Select Audio File');

global SpeechFile

SpeechFile = strcat(pathname, filename);

set(handles.linkText, 'string', SpeechFile);

* Đọc nút nhấn ***Degrade***:

function degradeButton\_Callback(hObject, eventdata, handles)

%%% Read speech file %%%

global SpeechFile

[sig, fs] = audioread(SpeechFile);

if fs ~= 16000

[sig, fs] = resample(sig, 16000, fs);

end

% Check single-channel source

if size(sig, 2) ~= 1

sig = sig(:,1);

end

%%% Add noise %%%

pinkOption = get(handles.pinkButton, 'value');

whiteOption = get(handles.whiteButton, 'value');

babbleOption = get(handles.babbleButton, 'value');

if pinkOption

noiseType = 'pink';

elseif whiteOption

noiseType = 'white';

elseif babbleOption

noiseType = 'babble';

end

NoiseFile = strcat('Degradation/Noise/noise\_', noiseType, '.wav');

[noise, fs\_noise] = audioread(NoiseFile);

if fs\_noise ~= fs

noise = resample(noise, fs, fs\_noise);

end

global noiseGain

noise = noiseGain\*noise(1:length(sig));

% Add noise n to signal x

global noisySig

noisySig = sig + noise;

%%% \_Add reverberant %%%

% Add library

addpath('Degradation/Reverb');

% Setup

global x1 y1 z1 x2 y2 z2 x3 y3 z3 x4 y4 z4 ...

X Y Z T60 ...

Ceil Floor Front Back Left Right

SetupStruct.Fs = 16000; % sampling frequency in Hz

SetupStruct.c = 343; % propagation speed of acoustic waves in m/s

SetupStruct.room = [3 4 2.5]; % room dimensions in m

SetupStruct.mic\_pos = [x1 y1 z1; % [x y z]

x2 y2 z2; % positions of microphone array [m]

x3 y3 z3;

x4 y4 z4];

SetupStruct.src\_traj = [X Y Z];

SetupStruct.T60 = T60; % SetupStruc.T20 = 0.15;

SetupStruct.abs\_weights = [Ceil Floor Front Back Left Right];

% Execute

RIR\_cell = ISM\_RIR\_bank(SetupStruct);

global degradedSig

degradedSig = ISM\_AudioData(RIR\_cell, noisySig);

[~, col] = size(degradedSig);

for channel = 1:col

degradedSig(:,channel) = degradedSig(:,channel)/max(degradedSig(:,channel));

end

disp('Finish degrading')

* Đọc nút nhấn ***Dereverb***:

function derevbButton\_Callback(hObject, eventdata, handles)

addpath('DeReverb')

global degradedSig derevSig

algoOption = get(handles.algoDerevListBox, 'value');

switch algoOption

case 1

algorithm = 'power\_subs';

case 2

algorithm = 'magnitude\_subs';

case 3

algorithm = 'wiener';

end

estOption = get(handles.estDerevListBox , 'value');

switch estOption

case 1

estimator = 'unbiased';

case 2

estimator = 'unbiased\_robust';

case 3

estimator = 'DOA\_independent';

case 4

estimator = 'noise\_coherence\_independent';

end

[~, col] = size(degradedSig);

inSig(:,1) = degradedSig(:,1);

for channel = 1:col

inSig(:,2) = degradedSig(:,channel);

outSig = cdrDereverb(inSig,estimator,algorithm);

derevSig(:,channel) = outSig/max(outSig);

end

disp('Finish dereverberating')

* Đọc nút nhấn ***Denoise***:

function denoiseButton\_Callback(hObject, eventdata, handles)

addpath('DeNoise');

global derevSig denoisedSig

[row, col] = size(derevSig);

denoisedSig = zeros(row, col);

if get(handles.powSpecSubButton,'value')

algorithm = 'powSpecSub';

elseif get(handles.magSpecSubButton,'value')

algorithm = 'magSpecSub';

elseif get(handles.overSpecSubButton,'value')

algorithm = 'overSpecSub';

elseif get(handles.wienerButton,'value')

algorithm = 'wiener';

elseif get(handles.mmseButton,'value')

algorithm = 'mmse';

end

Fs = 16000;

for channel = 1:col

[denoisedSig(:,channel), snr(:)] = noiseReduction(derevSig(:,channel), Fs, algorithm);

denoisedSig(:,channel) = denoisedSig(:,channel)/max(denoisedSig(:,channel));

end

set(handles.snrEdit,'string',num2str(max(snr)));

disp('Finish denoising')

* Đọc nút nhấn ***Save***:

function saveButton\_Callback(hObject, eventdata, handles)

[path, user\_canceled] = imsave();

if user\_canceled

msgbox(sprintf('Error'),'Error','Error');

return

end

* Đọc nút nhấn ***Plot***:

function plotButton\_Callback(hObject, eventdata, handles)

global noisySig degradedSig denoisedSig derevSig

figure

subplot(221)

plot(noisySig)

title('Noisy Signal'), axis([0 4.5e4 -1 1])

subplot(222)

plot(degradedSig)

title('Degraded Signal'), axis([0 4.5e4 -1 1])

subplot(223)

plot(derevSig)

title('Dereverberated Signal'), axis([0 4.5e4 -1 1])

subplot(224)

plot(denoisedSig)

title('Denoised Signal'), axis([0 4.5e4 -1 1])

* Đọc nút nhấn ***Quit***:

function quitButton\_Callback(hObject, eventdata, handles)

option = questdlg('Would you like to quit?', ...

'Option', ...

'Yes','No','No');

% Handle response

switch option

case 'Yes'

close

case 'No'

end

1. **KẾT QUẢ VÀ TỰ ĐÁNH GIÁ KẾT QUẢ:**
2. **Xử lý nhiễu:**

* Kết quả:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **STT** | **Tên file** | | **SNR\_IN** | **SNR\_OUT** | | | | |
| **Âm thanh** | **Nhiễu** | ***Trừ phổ năng lượng*** | ***Trừ phổ biên độ*** | ***Trừ phổ biên độ thích nghi*** | ***Bộ lọc Wiener*** | ***MMSE \_ STSA*** |
| 1 | female1.wav | noise\_bable.wav | 7.1212 | 8.3477 | 8.4389 | 6.0834 | 7.3746 | **8.7679** |
| 2 | female2.wav | noise\_bable.wav | 10.0669 | 11.7101 | **13.0416** | 12.6285 | 12.9879 | 12.7685 |
| 3 | female3.wav | noise\_bable.wav | 9.8244 | 11.7564 | 12.9437 | 12.4689 | 13.3970 | **13.4508** |
| 4 | female4.wav | noise\_bable.wav | 7.0037 | 8.8978 | **10.1116** | 8.3729 | 9.7777 | 9.5673 |
| 5 | male1.wav | noise\_bable.wav | 2.0231 | 3.5667 | **4.5837** | 3.3525 | 3.8021 | 4.4255 |
| 6 | male2.wav | noise\_bable.wav | 4.1906 | 5.4714 | **5.6158** | 3.4651 | 4.7053 | 5.5456 |
| 7 | female1.wav | noise\_pink.wav | 14.1179 | 16.6630 | **17.8066** | 16.8034 | 17.7443 | 17.3300 |
| 8 | female2.wav | noise\_pink.wav | 16.9836 | 19.6060 | **21.2351** | 20.6538 | 20.9639 | 20.3694 |
| 9 | female3.wav | noise\_pink.wav | 16.8650 | 19.6287 | **20.7802** | 19.2332 | 20.3525 | 20.2365 |
| 10 | female4.wav | noise\_pink.wav | 14.0363 | 16.7203 | **17.9256** | 17.1137 | 17.3147 | 16.4095 |
| 11 | male1.wav | noise\_pink.wav | 9.0850 | 8.9598 | 8.4945 | 6.7223 | 7.4102 | 8.5372 |
| 12 | male2.wav | noise\_pink.wav | 11.1160 | 11.5459 | 10.0078 | 7.9739 | 8.7016 | 9.9206 |
| 13 | female1.wav | noise\_white.wav | 18.2138 | 20.8504 | **21.8264** | 20.2740 | 20.9103 | 20.7923 |
| 14 | female2.wav | noise\_white.wav | 21.0602 | 23.7189 | **24.8177** | 22.8585 | 23.6177 | 23.5541 |
| 15 | female3.wav | noise\_white.wav | 20.9297 | 23.9149 | **25.3625** | 23.7644 | 24.4252 | 24.1973 |
| 16 | female4.wav | noise\_white.wav | 18.1034 | 20.6272 | **21.2214** | 18.9850 | 19.7792 | 19.1093 |
| 17 | male1.wav | noise\_white.wav | 13.1527 | 10.6808 | 9.4928 | 8.0289 | 8.4506 | 9.7574 |
| 18 | male2.wav | noise\_white.wav | 15.1793 | 13.7571 | 11.1455 | 9.2726 | 9.6046 | 11.0391 |

* Tự đánh giá:
* Thuật toán ***Trừ Phổ Biên độ*** hoạt động ổn định và cho kết quả tốt nhất đối với các loại tiếng nói đầu vào có nhiễu khác nhau.
* Tiếng nói của đàn ông khi có ***nhiễu trắng*** hoặc ***nhiễu hồng*** thì gần như không thể xử lý được.
* Thời gian tính toán nhanh nên có thể ứng dụng vào việc tính toán online.

1. **Xử lý âm vang:**

* Kết quả:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ***STT*** | ***Tên File*** | ***Bộ ước lượng*** | ***DRR (dB)*** | | |
| ***In\_Left*** | ***In\_Right*** | ***Out*** |
| 1 | female1.wav | Unbiased | 1.0959 | 1.0157 | 2.3187 |
| 2 | female2.wav | Unbiased | -1.1693 | -1.2616 | -1.9986 |
| 3 | female3.wav | Unbiased | -4.0598 | -4.1921 | **-2.0844** |
| 4 | female4.wav | Unbiased | -5.7247 | -5.6882 | **-3.8033** |
| 5 | male1.wav | Unbiased | -2.8399 | 7.6117 | **7.6648** |
| 6 | male2.wav | Unbiased | -6.3274 | -6.3407 | **-5.662** |
| 7 | female1.wav | Unbiased + Robust | 1.0959 | 1.0157 | 1.1099 |
| 8 | female2.wav | Unbiased + Robust | -1.1693 | -1.2616 | -1.2471 |
| 9 | female3.wav | Unbiased + Robust | -4.0598 | -4.1921 | -4.0746 |
| 10 | female4.wav | Unbiased + Robust | -5.7247 | -5.6882 | -5.699 |
| 11 | male1.wav | Unbiased + Robust | -2.8399 | 7.6117 | -2.7635 |
| 12 | male2.wav | Unbiased + Robust | -6.3274 | -6.3407 | -6.3168 |
| 13 | female1.wav | DOA-Independent | 1.0959 | 1.0157 | **11.1708** |
| 14 | female2.wav | DOA-Independent | -1.1693 | -1.2616 | -1.1854 |
| 15 | female3.wav | DOA-Independent | -4.0598 | -4.1921 | -3.9245 |
| 16 | female4.wav | DOA-Independent | -5.7247 | -5.6882 | -5.7441 |
| 17 | male1.wav | DOA-Independent | -2.8399 | 7.6117 | -2.6646 |
| 18 | male2.wav | DOA-Independent | -6.3274 | -6.3407 | -5.895 |
| 19 | female1.wav | Noise-Coherent-Independent | 1.0959 | 1.0157 | 1.0565 |
| 20 | female2.wav | Noise-Coherent-Independent | -1.1693 | -1.2616 | -1.2477 |
| 21 | female3.wav | Noise-Coherent-Independent | -4.0598 | -4.1921 | -4.1224 |
| 22 | female4.wav | Noise-Coherent-Independent | -5.7247 | -5.6882 | -5.7073 |
| 23 | male1.wav | Noise-Coherent-Independent | -2.8399 | 7.6117 | -2.8155 |
| 24 | male2.wav | Noise-Coherent-Independent | -6.3274 | -6.3407 | -6.3326 |

* Tự đánh giá:
* ***Bộ ước lượng không thiên vị*** hoạt động ổn định và cho kết quả tốt nhất đối với các loại tiếng nói đầu vào có âm vang.
* ***Bộ ước lượng không phụ thuộc tương quan với nhiễu*** không hoạt động được với các tín hiệu đầu vào. Tương tự, ***Bộ ước lượng không thiên vị tăng cường*** hoạt động kém với các tín hiệu đầu vào.
* Thời gian tính toán nhanh nên có thể ứng dụng vào việc tính toán online.

1. **ĐÁNH GIÁ CỦA GIẢNG VIÊN HƯỚNG DẪN:**