Project-- Speech Processing

1.Present the program code, the methods you used and the results

(1)SpeechProcessing.m(主程式)

Code:

% part1: Waveform

[data,fs]=audioread('voice.wav'); % read .wav file

% info=audioinfo("voicd.wav");

%data=data/abs(max(data)); % normalize data

%sound(data,fs);

figure;

t = [0 : 1/fs : length(data)/fs]; % time in sec

t = t(1:end - 1);

subplot(5,1,1);

plot(t,data);

title('Waveform');

% part2: Energy contour

% do framing

f\_d=0.025;

frames = framing(data, fs, f\_d);% it is like 0% overlap with rectangular window

% calculate frames energy

[r, c] = size(frames);

ste = 0;

for i = 1 : r

ste(i) = sum(frames(i,:).^2);

end

%ste = ste./max(ste); % normalize the data

f\_size = round(f\_d \* fs); % how many samples in a frame

ste\_wave = 0;

for j = 1 : length(ste)

l = length(ste\_wave);

ste\_wave(l : l + f\_size) = ste(j);

end

% plot the STE with Signal

t = [0 : 1/fs : length(data)/fs]; % time in sec

t = t(1:end - 1);

t1 = [0 : 1/fs : length(ste\_wave)/fs];

t1 = t1(1:end - 1);

subplot(5,1,2);

plot(t1,ste\_wave,'r','LineWidth',1);

title('Energy contour');

% part3: Zero-crossing rate contour

x=frames(110,:);

ZCR = sum(abs(diff(x > 0)));

%[r, c] = size(frames); % finding ZCR of all frames

for i = 1 : r

x = frames(i, :);

ZCRf(i) = sum(abs(diff(x > 0)));

end

% calculating rate

ZCRr = ZCRf/length(x);

%ZCRr = ZCRr/max(ZCRr2);

zcr\_wave = 0;

for j = 1 : length(ZCRr)

l = length(zcr\_wave);

zcr\_wave(l : l + f\_size) = ZCRr(j);

end

% plot the ZCR with Signal

t1 = [0 : 1/fs : length(zcr\_wave)/fs];

t1 = t1(1:end - 1);

subplot(5,1,3);

plot(t1,zcr\_wave\*length(x),'r','LineWidth',1);

title('Zero-crossing rate contour');

% part5: Pitch contour

[f0,idx] = pitch(data,fs, ...

'Method','PEF', ...

'WindowLength',round(fs\*0.08), ...

'OverlapLength',round(fs\*(0.08-0.01)), ...

'Range',[60,1000], ...

'MedianFilterLength',3);

%figure;

t0 = (idx - 1)/fs;

subplot(5,1,5);

plot(t0,f0,'r','LineWidth',1);

xlabel('time(s)');

title('Pitch contour');

%part4: End point detection

frameSize = 256;

overlap = 128;

y=data-mean(data); % zero-mean substraction

frameMat=enframe(y, frameSize, overlap); % frame blocking

frameNum=size(frameMat, 2); % no. of frames

volume=frame2volume(frameMat); % volume

volumeTh=median(volume)\*6; % volume threshold

index = find(volume>volumeTh);

endPoint=frame2sampleIndex(index, frameSize, overlap);

subplot(5,1,4);

t=(1:length(y))/fs;

plot(t, y);

title('End Point Detection');

axis([-inf inf -1 1]);

line(t(endPoint( 1))\*[1 1], [-1, 1], 'color', 'k');

line(t(endPoint(end))\*[1 1], [-1, 1], 'color', 'k');

**程式碼說明:**

1. Waveform:

 data為聲音訊號的向量，fs 為取樣頻率，其值為16000，表錄製此音訊檔案(voice.wav)時，每秒記錄16000個聲音訊號的取樣值。 t 為對應在時間軸的向量，將 data 對 t 做圖，即可得到在**Waveform** in Time Domain。

1. Energy contour:

音量表聲音的強度，又稱為能量（Energy），因音訊在短時間內是相對穩定的，所以將Energy以一個音框內的訊號震幅大小來類比。先用framing function將音訊切成一個一個音框(frame)，f\_d為音框長度(frame duration，設為0.025，即一個frame是0.025秒)，f\_size為每個音框所含點數。計算每個frame中每個取樣點的絕對值總和即為所求，單位為焦耳(未換成分貝)。

1. Zero-crossing rate contour:

過零率是指一個信號的符號變化的比率，可用每個音框中，音訊通過零點的次數來判斷(前後兩點相乘為負即可判斷)，我是利用sum ,abs ,diff函式組合實作。

Ex:

x = [1 2 -3 4 5 -6 2 -6 2];

(1)令y=x>0:

y = [1 1 0 1 1 0 1 0 1];

(2)對y取diff(取該數列元素間之差異):

ans=[0 -1 1 0 -1 1 -1 1];

(3)對ans取abs(絕對值):

result=[0 1 1 0 1 1 1 1 ];

(4)對result取sum(總和)

Num=6(即為通過零的次數)

1. End point detection:

只要音量小於某個門檻值，我們就認定是靜音或是雜訊，先訂出三個門檻值(ITU、ITL、IZCT即高能量、低能量和過零率門檻)，再利用能量及過零率來判斷聲音的起點和終點，Overlap為音框重疊量，即音框之間重疊的點數。

1. Pitch contour:

音高為頻率高低，頻率為週期的倒數，先求出音訊的週期後再做倒數，即可得到答案。

(2)framing.m(SpeechProcessing.m(主程式)所用到之function):

把音訊分成n\_f個frame，f\_size為每個音框所含點數，n\_f為frame個數

Code:

function [frames] = framing(x,fs,f\_d)

f\_size=round(f\_d\*fs);% how many samples in a frame

l\_s=length(x);

n\_f=floor(l\_s/f\_size);% no. of frames

temp = 0;

for i = 1 : n\_f

frames(i,:) = x(temp + 1 : temp + f\_size);

temp = temp + f\_size;

end

end

(3) enframe.m(SpeechProcessing.m(主程式)在End point detection所用到之function):

使用現有函式，將音訊切分為frames，每行為一個frame

Code:

function varargout=enframe(varargin)

%

% For calling details please see v\_enframe.m

%

% This dummy routine is included for backward compatibility only

% and will be removed in a future release of voicebox. Please use

% v\_enframe.m in future and/or update with v\_voicebox\_update.m

%

% Copyright (C) Mike Brookes 2018

% Version: $Id: enframe.m 10863 2018-09-21 15:39:23Z dmb $

%

if nargout

varargout=cell(1,nargout);

[varargout{:}]=v\_enframe(varargin{:});

else

v\_enframe(varargin{:});

end

(4) v\_enframe.m(enframe.m所用到之function):

使用現有函式，將音訊切分為(overlapping) frames，每列為一個frame

Code:

function [f,t,w]=v\_enframe(x,win,hop,m,fs)

nx=length(x(:));

if nargin<2 || isempty(win)

win=nx;

end

if nargin<4 || isempty(m)

m='';

end

nwin=length(win);

if nwin == 1

lw = win;

w = ones(1,lw);

else

lw = nwin;

w = win(:).';

end

if (nargin < 3) || isempty(hop)

hop = lw; % if no hop given, make non-overlapping

elseif hop<1

hop=lw\*hop;

end

if any(m=='a')

w=w\*sqrt(hop/sum(w.^2)); % scale to give unity gain for overlap-add

elseif any(m=='s')

w=w/sqrt(w\*w'\*lw);

elseif any(m=='S')

w=w/sqrt(w\*w'\*lw/hop);

end

if any(m=='d') % scale to give power/energy densities

if nargin<5 || isempty(fs)

w=w\*sqrt(lw);

else

w=w\*sqrt(lw/fs);

end

end

nli=nx-lw+hop;

nf = max(fix(nli/hop),0); % number of full frames

na=nli-hop\*nf+(nf==0)\*(lw-hop); % number of samples left over

fx=nargin>3 && (any(m=='z') || any(m=='r')) && na>0; % need an extra row

f=zeros(nf+fx,lw);

indf= hop\*(0:(nf-1)).';

inds = (1:lw);

if fx

f(1:nf,:) = x(indf(:,ones(1,lw))+inds(ones(nf,1),:));

if any(m=='r')

ix=1+mod(nf\*hop:nf\*hop+lw-1,2\*nx);

f(nf+1,:)=x(ix+(ix>nx).\*(2\*nx+1-2\*ix));

else

f(nf+1,1:nx-nf\*hop)=x(1+nf\*hop:nx);

end

nf=size(f,1);

else

f(:) = x(indf(:,ones(1,lw))+inds(ones(nf,1),:));

end

if (nwin > 1) % if we have a non-unity window

f = f .\* w(ones(nf,1),:);

end

if any(lower(m)=='p') % 'pP' = calculate the power spectrum

f=fft(f,[],2);

f=real(f.\*conj(f));

if any(m=='p')

imx=fix((lw+1)/2); % highest replicated frequency

f(:,2:imx)=f(:,2:imx)+f(:,lw:-1:lw-imx+2);

f=f(:,1:fix(lw/2)+1);

end

elseif any(lower(m)=='f') % 'fF' = take the DFT

f=fft(f,[],2);

if any(m=='f')

f=f(:,1:fix(lw/2)+1);

end

end

if nargout>1

if any(m=='E')

t0=sum((1:lw).\*w.^2)/sum(w.^2);

elseif any(m=='A')

t0=sum((1:lw).\*w)/sum(w);

else

t0=(1+lw)/2;

end

t=t0+hop\*(0:(nf-1)).';

end

(5) frame2volume.m(SpeechProcessing.m(主程式)在End point detection所用到之function):

使用現有函式，將frame矩陣轉為音量

Code:

function volume = frame2volume(frameMat, opt);

if nargin<1, selfdemo; return; end

% ====== Set the default options

if ischar(frameMat) & strcmp(lower(frameMat), lower('defaultOpt'))

volume.method='absSum';

volume.meanCurvePolyOrder=0;

return

end

if nargin<2, opt=feval(mfilename, 'defaultOpt'); end

[frameSize, frameNum]=size(frameMat);

% ====== Use polyfit to find the mean curve

if opt.meanCurvePolyOrder==0

for i=1:frameNum

frameMat(:,i)=frameMat(:,i)-mean(frameMat(:,i));

end

else

frameMat=frameZeroMean(frameMat, opt.meanCurvePolyOrder);

end

% ====== Volume computation

volume=zeros(1, frameNum);

switch lower(opt.method)

case lower('absSum')

for i=1:frameNum

frame=frameMat(:,i);

% frame=frame-median(frame);

volume(i)=sum(abs(frame));

end

case lower({'squaredSum', 'decibel'})

for i=1:frameNum

frame=frameMat(:,i);

% frame=frame-mean(frame);

volume(i)=sum(frame.^2);

end

if strcmp(opt.method, 'decibel')

volume=10\*log10(volume+realmin); % add realmin to avoid log(0) warning

end

otherwise

error('Unknown method!');

end

% ====== Self demo

function selfdemo

mObj=mFileParse(which(mfilename));

strEval(mObj.example);

(6) frame2sampleIndex.m(SpeechProcessing.m(主程式)在End point detection所用到之function):

使用現有函式，將frame index 轉為sample index

Code:

function sampleIndex=frame2sampleIndex(frameIndex, frameSize, overlap)

sampleIndex=(frameIndex-1)\*(frameSize-overlap)+round(frameSize/2);

end

(7) pitch.m(SpeechProcessing.m(主程式)在Pitch contour所用到之function):

使用現有函式但稍作修改

Code:

function [f0,sampleStamp] = pitch(x, fs,varargin)

validateRequiredInputs(x,fs)

defaults = struct( ...

'Method', 'NCF', ...

'Range', cast([50,400],'like',x), ...

'WindowLength', cast(round(fs.\*0.052),'like',x), ...

'OverlapLength', cast(round(fs\*(0.052-0.01)),'like',x), ...

'MedianFilterLength',cast(1,'like',x), ...

'SampleRate', cast(fs,'like',x), ...

'NumChannels', cast(size(x,2),'like',x), ...

'SamplesPerChannel', cast(size(x,1),'like',x));

params = matlabshared.fusionutils.internal.setProperties(defaults, nargin-2, varargin{:});

validateOptionalInputs(x,fs,params)

% Determine pitch

f0 = stepMethod(x,params);

% Create sample stamps corresponding to pitch decisions

hopLength = params.WindowLength - params.OverlapLength;

numHops = cast(floor((size(x,1)-params.WindowLength)/hopLength),'like',x);

sampleStamp = cast(((0:numHops)\*hopLength + params.WindowLength)','like',x);

% Apply median filtering

if params.MedianFilterLength ~= 1

f0 = movmedian(f0,params.MedianFilterLength,1);

end

% Trim off zero-padded last estimate

f0 = f0(1:(numHops+1),:);

end

% -------------------------------------------------------------------------

% Validate required inputs

% -------------------------------------------------------------------------

function validateRequiredInputs(x,fs)

validateattributes(x,{'single','double'}, ...

{'nonempty','2d','real','nonnan','finite'}, ...

'pitch','audioIn')

validateattributes(fs,{'single','double'}, ...

{'nonempty','positive','scalar','real','nonnan','finite'}, ...

'pitch','fs')

end

% -------------------------------------------------------------------------

% Validate optional input

% -------------------------------------------------------------------------

function validateOptionalInputs(x,fs,userInput)

N = size(x,1);

validateattributes(userInput.Range,{'single','double'}, ...

{'nonempty','increasing','positive','row','ncols',2,'real'}, ...

'pitch','Range')

coder.internal.errorIf(userInput.WindowLength < 1, ...

'audio:pitch:BadWindowLength', ...

'WINDOWLENGTH','[1,size(x,1)]','x','round(fs\*0.052)');

validateattributes(userInput.WindowLength,{'single','double'}, ...

{'nonempty','integer','positive','scalar','real','<=',192000}, ...

'pitch','WindowLength')

validateattributes(userInput.OverlapLength,{'single','double'}, ...

{'nonempty','integer','scalar','real'}, ...

'pitch','OverlapLength')

validateattributes(userInput.MedianFilterLength,{'single','double'}, ...

{'nonempty','integer','positive','scalar','real'}, ...

'pitch','MedianFilterLength')

coder.internal.errorIf(sum(strcmp(userInput.Method,{'NCF','PEF','CEP','LHS','SRH'}))~=1, ...

'audio:pitch:BadMethod', ...

'NCF','PEF','CEP','LHS','SRH');

coder.internal.errorIf(userInput.WindowLength > N, ...

'audio:pitch:BadWindowLength', ...

'WINDOWLENGTH','[1,size(x,1)]','x','round(fs\*0.052)');

coder.internal.errorIf(userInput.OverlapLength >= userInput.WindowLength, ...

'audio:pitch:BadOverlapLength', ...

'OVERLAPLENGTH', 'WINDOWLENGTH');

switch userInput.Method

case 'NCF'

coder.internal.errorIf(fs/userInput.Range(1) >= userInput.WindowLength, ...

'audio:pitch:BadSpecifications', ...

'NCF','fs/RANGE(1) < WINDOWLENGTH');

coder.internal.errorIf(fs/2<userInput.Range(2), ...

'audio:pitch:BadSpecifications', ...

'NCF','fs/2 >= RANGE(2)');

case 'PEF'

coder.internal.errorIf((userInput.Range(1)<=10) || (userInput.Range(2)>=min(4000,fs/2)), ...

'audio:pitch:BadSpecifications', ...

'PEF','RANGE(1) > 10 && RANGE(2) < min(4000,fs/2)');

case 'CEP'

coder.internal.errorIf((userInput.Range(2)>=fs/2), ...

'audio:pitch:BadSpecifications', ...

userInput.Method,'RANGE(2) < fs/2');

coder.internal.errorIf(round(fs/userInput.Range(1))>2^nextpow2(2\*userInput.WindowLength-1), ...

'audio:pitch:BadSpecifications', ...

userInput.Method,'round(fs/RANGE(1)) <= 2^nextpow2(2\*WINDOWLENGTH-1)');

case 'LHS'

coder.internal.errorIf(((userInput.Range(2)+1)\*5>=fs), ...

'audio:pitch:BadSpecifications', ...

userInput.Method,'(RANGE(2)+1)\*5 < fs');

case 'SRH'

coder.internal.errorIf(((userInput.Range(2)+1)\*5>=fs), ...

'audio:pitch:BadSpecifications', ...

userInput.Method,'(RANGE(2)+1)\*5 < fs');

end

% -------------------------------------------------------------------------

end

function f0 = stepMethod(x,params)

oneCast = cast(1,'like',x);

r = cast(size(x,1),'like',x);

c = cast(size(x,2),'like',x);

hopLength = params.WindowLength - params.OverlapLength;

numHopsFinal = ceil((r-params.WindowLength)/hopLength) + oneCast;

% The SRH method uses a fixed-size intermediate window and hop

% length to determine the residual signal.

if isequal(params.Method,'SRH')

N = round(cast(0.025\*params.SampleRate,'like',x));

hopSize = round(cast(0.005\*params.SampleRate,'like',x));

else

N = cast(params.WindowLength,'like',x);

hopSize = cast(hopLength,'like',x);

end

numHops = ceil((r-N)/hopSize) + oneCast;

% Convert to matrix for faster processing

y = zeros(N,numHops\*c,'like',x);

for channel = 1:c

for hop = 1:numHops

temp = x(1+hopSize\*(hop-1):min(N+hopSize\*(hop-1),r),channel);

y(1:min(N,numel(temp)),hop+(channel-1)\*numHops) = temp;

end

end

% Run pitch detection algorithm

extraParams = struct('NumCandidates',1,'MinPeakDistance',1);

switch params.Method

case 'SRH'

f0 = audio.internal.pitch.SRH(y,params,extraParams);

case 'PEF'

f0 = audio.internal.pitch.PEF(y,params,extraParams);

case 'CEP'

f0 = audio.internal.pitch.CEP(y,params,extraParams);

case 'LHS'

f0 = audio.internal.pitch.LHS(y,params,extraParams);

otherwise %'NCF'

f0 = audio.internal.pitch.NCF(y,params,extraParams);

end

% Force pitch estimate inside band edges

bE = params.Range;

f0(f0<bE(1)) = bE(1);

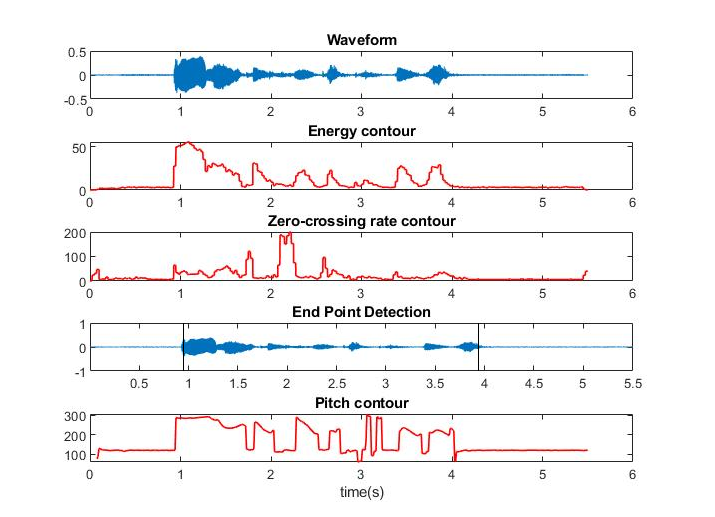
f0(f0>bE(end)) = bE(end);

% Reshape to multichannel

f0 = reshape(f0,numHopsFinal,c);

end

**Result:**



1. Waveform圖中，可明顯看出說話的地方震幅較大: 多(約0.9 ~1.2秒) 、媒(約1.5秒)、體(約1.9秒)、系(約2.4秒)、統(約2.6秒)、與(約3.1秒)、應(約3.5秒)、用(約3.8秒)。
2. Energy contour圖中， 用SAP Toolbx 裡面的 endPointDetect.m 函數(此函數即是使用音量和過零率來決定端點) 也可觀察說話的秒數都有相對高峰，即能量較強。
3. Zero-crossing rate contour圖中，對照Energy contour圖的高峰，通常波峰跟波峰之間過零率較易出現微小高峰。
4. End Point Detection圖中，兩條黑線分別代表錄音開始和結束，有嘗試使用用SAP Toolbx 裡面的 endPointDetect.m 函數來找出這八個音的開始和結束位置，自己的函式則無法。
5. Pitch contour圖中，可以看出錄製”應用”兩字的頻率較低。