EE 301

PROJECT REPORT

Monaural voiced speech segregation based on elaborate harmonic grouping strategies

INSTRUCTOR: Dr. Rajesh Hegde

GROUP Members:

Gaurav Solanki	Y9231
Shouvik Ganguly	Y9558
Shiv Prakash	Y9551
Prashant Khokhar	Y9427
Kundan Kanwaria	Y9300
Rakesh Meena	Y9474

ACKNOWLEDGEMENTS

We sincerely express our gratitude to our instructor-in-charge **Dr. Rajesh Hegde sir** for his valuable support and advice in making this project. Without his moral support and inspiration we would not have been able to complete this effortful task. Overall we thank our instructor for providing us with the opportunity to make something creative of which we could not even think of doing before actually doing it....

Thank You!!!

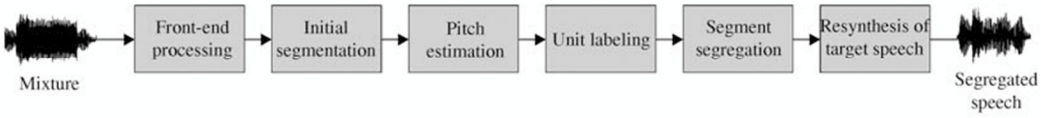
INTRODUCTION

Separating target voices from their mixture with noises is important for automatic speech/speaker recognition, hearing aid design and other fields. In our project we try to implement the algorithm mentioned in the paper having the title of our project. Main achievements of the algorithm used lies in three aspects. Firstly, the algorithm classifies the time-frequency (T-F) units into resolved and unresolved ones by carrier-to-envelop energy ratio. Secondly, resolved T-F units are grouped together according to minimum amplitude principle, which has been verified to exist in human perception, as well as the harmonic principle. Finally, "enhanced" envelope autocorrelation function is employed to detect amplitude modulation rates, which helps a lot in reducing half-frequency error in grouping of unresolved units.

System structure and algorithm description

System Structure

The algorithm proceeds in six main steps, as shown in Figure



Improved algorithm based on elaborate harmonic grouping strategies

Front-end processing

The input signal is decomposed into 128 channels by a gamma tone filter bank. Centre frequencies of the filters range from 80 to 5000 Hz in a quasi-logarithmical way and the bandwidth of each filter is set by equivalent rectangular bandwidth (ERB). Gamma tone filter bank can successfully simulate the frequency response feature of the basilar membrane, and thus it often serves as a standard acoustic filtering model [14]. Outputs of the filter bank are further transformed into neural firing rate by a hair cell model. Besides, Hilbert envelope of each channel is computed, and this envelope is further sent into a bandpass filter with passband of 50–550 Hz, whose output can be used to compute the amplitude modulation rates of the T-F units.

In each channel, the window length is set to 20 ms with 10 ms shift. Afterwards, CER (R_{eng}), autocorrelation function (A_H), autocorrelation function of the filtered envelope (A_E), cross-channel correlation function (C_H), cross-channel correlation function of the filtered envelope (C_E) are computed for each T-F unit by formulas provided in the paper, given below is the code for the front-end processing:

```
function[g,e]=front_end(xd)

n = -1600:1600;          % accounting for all 3201 sample points

fmin = 80;               % given
fmax = 5000;             % given
```

```
Ts=1/16000;
% Total number of centre frequencies is 128.
f0 = fmin*((fmax/fmin).^((0:127)/127)); % dividing quasi logarithmically into 128 parts
w0 = 2*pi*f0;
% calculating ERB with applied correction
b = (16/(5*pi))*(((6.23*1e-6)*(f0.^2))+((93.39*1e-3)*f0)+(28.52*ones(1,128)));

w = -pi:(2*pi/4000):pi;

Xd = dtft(xd,n,w);          % calculating dtft of signal

% The factor 10^10 is used to increase the magnitude *only*.
 G=1e10*((3*ones(128,length(w))./(((2*pi*b'*ones(1,length(w)))+(1j*((ones(128,1)*w/Ts)-
((w0'*ones(1,length(w))))))).^4))+(3*ones(128,length(w))./(((2*pi*b'*ones(1,length(w)))+(1j*((ones(128,1)*w/
Ts)+((w0'*ones(1,length(w))))))).^4)));

Yd = (ones(128,1)*Xd).*G;

g = idtft(Yd(1:128,:),w,n,2*pi/4000);

A=2*(Yd.*(((ones(128,1)*w)>=2*pi*50*Ts*ones(128,length(w)))&((ones(128,1)*w)<=2*pi*550*Ts*ones(128
,length(w)))));
% Taking inverse DTFT
e = idtft(A(1:128,:),w,n,2*pi/4000);


% 1.) We need to store the values of g and e according to further use.
% 2.) First we store the transverse of the matrices storing square of
% modulus of g and e.
% 3.) If the values are written successfully, the values of the status
% variables will become '1' from '0'.
```

Initial Segmentation

Code for finding carrier to envelop ratios:

```
function [Reng] =carrier_to_envelop(g,e)


absg=abs(g).^2;
abse=abs(e).^2;


Reng = zeros(size(absg,1), floor(size(absg,2)/160));
% Algorithm: We take a row of g and break it into frames of 320 samples
% each, with an overlap of 160 samples between adjacent frames. We compute
% sum of modulus of g squared over each frame and divide it by the sum of
% modulus of e squared over the same frame and take the logarithm


for a = 1:size(absg,1)
   for b = 1:floor(size(absg,2)/160)
      Reng(a,b) = log((sum(absg(a,160*(b-1)+1:160*b+1)))/sum(abse(a,160*(b-1)+1:160*b+1)));
   end
end
```

Unit Labelling

1. Resolved units

If the carrier-to-envelope ratio is greater than 1.82, the TF unit is labelled as ‘resolved’, otherwise ‘unresolved’.

Function to compute target pitch

```
function [tau_final]= target_pitch(Ah,g,In)

tau=0:200;
Fs=16000;
Ts=1/Fs;
m=(size(g,2)-1)/160;
A2 = zeros(128, m, length(tau));   % resolved and unresolved

for t = 1:length(tau)
   A2(:, :, t) = Ah(:, :, t).*In;
end

A = sum(A2);
Am = zeros(1, m, length(tau));
for t = 3:199
   Am(:,:,t) = ((A(:,:,t) > A(:,:,t-2)) & (A(:,:,t) > A(:,:,t-1)) & (A(:,:,t) > A(:,:,t+1)) & (A(:,:,t) > A(:,:,t+2)));
end

tau1 = (ones(m, 1))*tau;
tau2 = zeros(m, length(tau));

for t = 1:length(tau)
   for k = 1:(size(g,2)-1)/160
      tau2(k, t) = (tau1(k, t))*(Am(1, k, t) > 0);
   end
end

% tau1 = min(tau2, [], 2);



tau_1 = 2e-3;      % given tau lies between 2ms to 12.5ms
temp = Fs*tau_1;
% Algorithm to find the tau_final:
% We take a temporary variable sum(i) for i'th row and add the value of the
% next column unless the value of sum(i) becomes nonzero(say n'th column).
% Then the tau_final(i) is the value tau2(i,n). This algorithm follows
% Order(n) complexity.
sum1 = zeros(m,1);
tau_final = zeros(m,1);
for k = 1:m
   for n = temp+1:length(tau)
      if (sum1(k)==0)
         sum1(k) = sum1(k) + tau2(k,n);
      else break;
      end
```

```
      tau_final(k) = tau2(k,n);
   end
end
```

2. Unresolved units

```
function [harmo1]=target_voice(Ae,tau1,Reng)

m=length(tau1);


harmo1=zeros(size(Ae,1),m,m);
for c = 1:size(Ae,1)
   for m2 = 1:m-1
      for m3=1:m-1
          harmo1(c,m2,m3) = (((Ae(c,m2,tau1(m3)))/(max(Ae(c,m2,32:201))))>0.85)&(Reng(c,m2)<=1.82);

      end
   end
end
```

Segment segregation

Speech separation is done at the segment level. Minimum amplitude principle is also considered in the grouping process of resolved segments, depending on the labels of the units in each segment. Segments are also classified into resolved segments (formed by resolved T-F units that are continuous in time and frequency) and unresolved segments (consisting of unresolved T-F units that are located in consecutive frames and channels). For the segment numbered n, the matching degree of this segment with the pitch in frame m is

$$M(n,m)/\{N(n,m) + a*K(n,m)\} >1 \text{ ----------(!!!!!)}$$

where $M(n, m)$ is the number of T-F units that meet the harmonic principle and the minimum amplitude principle simultaneously for segment n in frame m; $N(n, m)$ is the number of units that cannot meet the harmonic relations; $K(n, m)$ is the number of units that cannot satisfy the minimum amplitude principle; α is a constant, which equals 3 here.

If formula (!!!!!) is satisfied, then segment n is said to match with the pitch in frame m. If the number of the frames of this kind exceeds half of the length of the segment, then the segment is classified as foreground. Otherwise it is classified as background. The code for segment segregation is given below:

```
function [segments_foreground]= segregate(g,e)


Reng=carrier_to_envelop(g,e);
[Ah,Ae,Ch,Ce,In]=auto_cross_correlation(g,e,Reng);
tau1=target_pitch(Ah,g,In);
harmo=harmonic(Ah,tau1,Reng);
Ae1=(Ae(:,:,1:201)-(Ae(:,:,ceil((1:201)/2)))).*(Ae(:,:,1:201)-(Ae(:,:,ceil((1:201)/2)))>zeros(size(Ae)));

harmo1=target_voice(Ae1,tau1,Reng);

minamp = integrated_minimum_amplitude(g,e);

In_res = uint8(In);          % converting logical matrix to matrix with integer values
m=size(In,2);
```

```
% graythresh(In) computes a global threshold (level) that can be used to
% convert an intensity image to a binary image with im2bw

BW = im2bw(In_res,graythresh(In_res));

% bwconncomp finds connected components in a binary image. The second
% input is the connectivity; in this case it is '4'. It gives as output a
% data structure that stores values in the variables: Connectivity,
% ImageSize, NumObjects, and PixelIdxList

cc = bwconncomp(BW, 8);

segments_resolved = zeros(size(BW));
segments_resolved_foreground=segments_resolved;
for k=1:cc.NumObjects

   segments_resolved(cc.PixelIdxList{k}) = k*uint8(true);
end


In_unres=uint8(~In);
BW1 = im2bw(In_unres,graythresh(In_unres));
cc1 = bwconncomp(BW1, 8);

segments_unresolved = zeros(size(BW1));
segments_unresolved_foreground=segments_unresolved;
for k=1:cc1.NumObjects

   segments_unresolved(cc1.PixelIdxList{k}) = k*uint8(true);
end


M=zeros(cc.NumObjects,m);
N=M;
K=N;
for k1=1:cc.NumObjects
   for k2=1:m
      M(k1,k2)=
sum(sum(((segments_resolved==k1*ones(size(segments_resolved)))&(harmo(:,:,k2)>zeros(128,m))&(minamp(
:,:,k2)>zeros(128,m)))));
      N(k1,k2)=
sum(sum(((segments_resolved==k1*ones(size(segments_resolved)))&(harmo(:,:,k2)==zeros(128,m)))));
      K(k1,k2)=
sum(sum(((segments_resolved==k1*ones(size(segments_resolved)))&(minamp(:,:,k2)==zeros(128,m)))));
   end
end
alpha=3;
matching_degree=M./(N+(alpha*K));
matching=(sum((matching_degree>=.001*ones(size(matching_degree))),2));

foreground_resolved=zeros(1,cc.NumObjects);
for k1=1:cc.NumObjects

   foreground_resolved(k1) =
matching(k1)>=.5*max(sum(segments_resolved==k1*ones(size(segments_resolved)),2));
end
for k=1:cc.NumObjects

   segments_resolved_foreground(cc.PixelIdxList{k}) = uint8(foreground_resolved(k));
```

```
end

for k1=1:128
   for k2=1:m
      segments_unresolved_foreground(k1,k2)=harmo1(k1,k2,k2)*(segments_unresolved(k1,k2)>0);
   end
end

segments_foreground=segments_resolved_foreground+ segments_unresolved_foreground;
```

In the above code, we have used the Image Processing toolbox to solve the problem of forming resolved and unresolved segments.

Segments belonging to the foreground are used to synthesize target speech.

Re-synthesis of target Speech

The cochlear model filterbank output of the sum of the two speech sounds (before compression and half wave rectification) is used as the basis of the resynthesis process. Each frequency-time region is multiplied by the per cent of the sound ($\frac{A1\hat{}}{AS}$) that belongs to this sound source, and a frequency gain (to compensate for the spectral tilt from the cochlear model). This separated cochlear output is then time reversed, and passed through a backwards original cascade filterbank. When the output of the backwards filterbank is time reversed again,, the resulting waveform is the resynthesized output. The reason for time reversing different waveforms is to compensate for the time delay imposed by cochlear model in the low frequency regions.

The re-synthesis of the separated output using this method has an important disadvantage. An example which exhibits this difficulty is the case of a sine wave of 110 Hz. The resulting is a sine wave of 105 Hz amplitude modulated at 10 Hz. If the separation system worked perfectly, it would correctly estimate that the amplitude of the 100 Hz sine wave was equal to the amplitude of 110 Hz sine wave. However, when the original sum is multiplied by the constant amplitude fraction ($\frac{A1}{AS}$), the resynthesized output sounds like a softer version of the original sum waveform, and not like the original 100 Hz sine waveform.

Following is the code for the re-synthesis process:

```
function [yd] = processed_output(g,segments_foreground)


g_out=zeros(size(g));
g_out_normalised=g_out;
for c=1:128
   for m1=1:((size(g,2)-1)/160)-1
      g_out(c,160*(m1-1)+(1:321))=(g(c,160*(m1-1)+(1:321)))*segments_foreground(c,m1);
   end
end

for c=1:128
   g_out_normalised(c,:)=g_out(c,:)*(sum(g_out(c,:)))/(sum(sum(g_out)));
end
G_out=zeros(128,4001);
for c=1:128
   G_out(c,:)=dtft(reverse(g_out_normalised(c, :)),-1600:1600,-pi:(2*pi/4000):pi);
end

Ts = 1/16000;
```

```
fmin = 80;              % given
fmax = 5000;            % given


f0 = fmin*((fmax/fmin).^((0:127)/127));
w0 = 2*pi*f0;

b = (16/(5*pi))*(((6.23*1e-6)*(f0.^2))+((93.39*1e-3)*f0)+(28.52*ones(1,128)));

w=-pi:(2*pi/4000):pi;
H=1e10*((3*ones(128,length(w))./(((2*pi*b'*ones(1,length(w)))+(1j*((ones(128,1)*w/Ts)-
((w0'*ones(1,length(w))))))).^4))+(3*ones(128,length(w))./(((2*pi*b'*ones(1,length(w)))+(1j*((ones(128,1)*w/
Ts)+((w0'*ones(1,length(w))))))).^4)));

Y_out=G_out./H;

Yd=sum(Y_out);

yd=reverse(idtft(Yd,w,-1600:1600,2*pi/4000));
```

System evaluation and experiment results

SCOPES FOR IMPROVEMENT

1. We could have implemented the hair cell model in a more realistic way.

2. With higher availability of memory, runtime could have been reduced by processing all the data together instead of running the code 50 times.

3. The envelope autocorrelation function could have been enhanced more (for the purpose of determining target voice from amongst unresolved segments).

4. Response frequency was determined by treating each frame as a pure sinusoid as determining the average distance between peaks. This method introduces some error which could have been lessened by taking the Fourier Transform of the frame and finding the average frequency from there. This was not feasible due to shortage of time, as well as speed issues.

A SPECIAL WORD OF THANKS

So, all's well that ends well. **We have projected ourselves the best we could have projected**. It is a compiled result of our

thought, our hard work, all of your motivation, and the grace of The Almighty. Once again thanks for all of those who contributed to this project directly or indirectly, named or unnamed. Thank you all…

References

- Monaural voiced speech segregation based on elaborate harmonic grouping strategies-LIU WenJu, ZHANG XueLiang, JIANG Wei, LI Peng2 & XU Bo; December 2011 Vol. 54 No. 12: 2471–2480
- A theory and computational model of auditory monaural sound separation, by Mitchel Weintraub

- A time-domain digital cochlear model, J ame s M. Kates, Senior Member, IEEE ;IEEE TRANSACTIONS ON S I G N A L PROCESSING, VOL 39. NO. 12, DECEMBER 1991
- Monaural Speech Separation, by Hu and Wang
- A Tandem Algorithm for Pitch Estimation and Voiced Speech Segregation, Guoning Hu, Member, IEEE, and DeLiang Wang, Fellow, IEEE; IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, VOL. 18, NO. 8, NOVEMBER 2010
- Implementing a Gamma Tone Filter Bank, John Holdsworth, Ian Nimmo-Smith, Roy Patterson, Peter Rice, 26th February 1988
- Suggested formulae for calculating auditory-filter bandwidths and excitation patterns, Moore BC, Glasberg BR; 1983 Sep; 74(3):750-3.