LAMPIRAN A
LISTING PROGRAM

program.m

```
% --------------------------------------------------------------------------------
% Program Aplikasi Pembuka Software berbasis Pengenalan Suara
% Matlab Window Programming
% Oleh : Anugrah Endy (0322028)
% --------------------------------------------------------------------------------
clear all;
clc;

win1=figure(...
   'units','points',...
   'position',[130 190 400 200],...
   'color',[.8 .8 .9],...
   'menubar','none',...
   'resize','off',...
   'numbertitle','off',...
   'name','Aplikasi Pembuka Software berbasis Pengenalan Suara');

label1=uicontrol('parent',win1,...
   'units','points',...
   'position',[30 150 300 30],...
   'backgroundcolor',[.8 .8 .9],...
   'style','Text',...
   'string','Aplikasi Pembuka Software berbasis Pengenalan Suara',...
   'fontname','arial',...
   'fontsize',12,...
   'fontweight','bold',...
   'foregroundcolor',[0 0 0]);

 frame1=uicontrol('parent',win1,...
   'units','points',...
```

```
    'position',[0 0 500 60],...
    'backgroundcolor',[.3 .3 .4],...
    'style','Frame');

tomstartrecognition=uicontrol('parent',win1,...
    'units','points',...
    'position',[200 60 180 30],...
    'style','pushbutton',...
    'callback','recognition',...
    'string','Start Recognition',...
    'fontname','arial',...
    'fontsize',20);

tomclear=uicontrol('parent',win1,...
    'units','points',...
    'position',[30 20 80 15],...
    'style','pushbutton',...
    'string','Clear',...
    'fontname','arial',...
    'fontsize',10,...
    'callback','bersih');

tomtutup=uicontrol('parent',win1,...
    'units','points',...
    'position',[270 20 80 15],...
    'style','pushbutton',...
    'string','Tutup',...
    'fontname','arial',...
    'fontsize',10,...
    'callback','close');
```

recognition.m

```
%Input masuk
fs = 11025;
x = wavrecord(1*fs,fs,'double');

%lakukan proses Preemphasis ,Frame blocking & windows
xprhm = preham(x, 1);
wx=xprhm;

%menentukan orde LPC
order = 12;

% Metoda autokorelasi LPC
[lpcoefs, errorPow] = lpc(wx, order);

%KOnversi Parameter LPC ke koefisien

% Fit the original LPC model (high-order)
[a,g,e] = lpcfit(wx,20);

% Warp pole-pole
% (warppoles memodifikasi setiap frame - baris pada a - pada waktu sama)

alpha = 0.5;
[bhat, ahat] = warppoles(a, alpha);

% Resynthesize with the new LPC
% (fortunately, bhat is the same for all frames)
dw = filter(bhat(1,:), 1, lpcsynth(ahat, g, e));
Z = dw';
```

```
load('newdatabase');
[p l] = size(x);
%Dynamic Time Warping ( DTW)
M = simmx(Z,x);
[p,q,d] = dp(M);
Banding
```

banding.m

```
D1=(d(:,1)+d(:,2)+d(:,3)+d(:,4)+d(:,5))/5;
D2=(d(:,6)+d(:,7)+d(:,8)+d(:,9)+d(:,10))/5;
D3=(d(:,11)+d(:,12)+d(:,13)+d(:,14)+d(:,15))/5;
D=[D1 D2 D3];

number = 1;
Temp = abs(D)
min=Temp(:,1);
for i=2:3
   if Temp(:,i)<=min
      min = Temp(:,i);
      number = i;
   end;
end;

switch number
   case 1
      disp('jalankan program word');
      winopen('C:\Program Files\Microsoft Office\OFFICE11\winword.exe');
   case 2
       disp('jalankan program paint');
      winopen('c:\windows\paint.exe')
```

```
   case 3
      disp('jalankan program notepad');
      winopen('c:\windows\notepad.exe')

end;
```

simmx.m

```
function M = simmx(A,B)
% M = simmx(A,B)
%    calculate a sim matrix between specgram-like feature matrices A and B.
%    size(M) = [size(A,2) size(B,2)]; A and B have same #rows.
% 2003-03-15 dpwe@ee.columbia.edu

% Copyright (c) 2003 Dan Ellis <dpwe@ee.columbia.edu>
% released under GPL - see file COPYRIGHT

EA = sqrt(sum(A.^2));
EB = sqrt(sum(B.^2));

%ncA = size(A,2);
%ncB = size(B,2);
%M = zeros(ncA, ncB);
%for i = 1:ncA
%  for j = 1:ncB
%    % normalized inner product i.e. cos(angle between vectors)
%    M(i,j) = (A(:,i)'*B(:,j))/(EA(i)*EB(j));
%  end
%end

% this is 10x faster
M = (A'*B)./(EA'*EB);
```

dp.m

```
function [p,q,D] = dp(M)
% [p,q] = dp(M)
%    Use dynamic programming to find a min-cost path through matrix M.
%    Return state sequence in p,q
% 2003-03-15 dpwe@ee.columbia.edu

% Copyright (c) 2003 Dan Ellis <dpwe@ee.columbia.edu>
% released under GPL - see file COPYRIGHT

[r,c] = size(M);

% costs
D = zeros(r+1, c+1);
D(1,:) = NaN;
D(:,1) = NaN;
D(1,1) = 0;
D(2:(r+1), 2:(c+1)) = M;
 % traceback
phi = zeros(r,c);
 for i = 1:r;
  for j = 1:c;
    [dmax, tb] = min([D(i, j), D(i, j+1), D(i+1, j)]);
    D(i+1,j+1) = D(i+1,j+1)+dmax;
    phi(i,j) = tb;
  end
end
 % Traceback from top left
i = r;
j = c;
```

```
p = i;
q = j;
while i > 1 & j > 1
  tb = phi(i,j);
  if (tb == 1)
    i = i-1;
    j = j-1;
  elseif (tb == 2)
    i = i-1;
  elseif (tb == 3)
    j = j-1;
  else
    error;
  end
  p = [i,p];
  q = [j,q];
end
 % Strip off the edges of the D matrix before returning
D = D(2:(r+1),2:(c+1));
```

lpcfit.m

```
function [a,g,e] = lpcfit(x,p,h,w,ov)
% [a,g,e] = lpcfit(x,p,h,w,ov)  Fit LPC to short-time segments
%    x is a stretch of signal.  Using w point (2*h) windows every
%    h points (128), fit order p LPC models.  Return the successive
%    all-pole coefficients as rows of a, the per-frame gains in g
%    and the residual excitation in e.
%    ov nonzero selects overlap-add of window-length
%    residuals, otherwise successive hop-sized residuals are concatenated
%    for independent near-perfect reconstruction with lpcsynth.
%    (default is 1)
```

```
% 2001-02-25 dpwe@ee.columbia.edu $Header:
/homes/dpwe/matlab/columbiafns/RCS/lpcfit.m,v 1.1 2004/03/30 20:55:52 dpwe
Exp $
 if nargin < 2
  p = 12;
end
if nargin < 3
  h = 128;
end
if nargin < 4
  w = 2*h;
end
if nargin < 5
  ov = 1;
end
 if (size(x,2) == 1)
  x = x';  % Convert X from column to row
end
 npts = length(x);
 nhops = floor(npts/h);
 % Pad x with zeros so that we can extract complete w-length windows
% from it
x = [zeros(1,(w-h)/2),x,zeros(1,(w-h/2))];
 a = zeros(nhops, p+1);
g = zeros(nhops, 1);
if ov == 0
  e = zeros(1, npts);
else
  e = zeros(1, (nhops-1)*h+w);
end
 % Pre-emphasis
pre = [1 -0.9];
```

```
x = filter(pre,1,x);
 for hop = 1:nhops
  % Extract segment of signal
  xx = x((hop - 1)*h + [1:w]);
  % Apply hanning window
  wxx = xx .* hanning(w)';
  % Form autocorrelation (calculates *way* too many points)
  rxx = xcorr(wxx);
  % extract just the points we need (middle p+1 points)
  rxx = rxx(w+[0:p]);
  % Setup the normal equations
  R = toeplitz(rxx(1:p));
  % Solve for a (horribly inefficient to use full inv())
  an = inv(R)*rxx(2:(p+1))';
  % Calculate residual by filtering windowed xx
  aa = [1 -an'];
  if ov == 0
   rs = filter(aa, 1, xx((w-h)/2 + [1:h]));
  else
   rs = filter(aa,1,wxx);
  end
  G = sqrt(mean(rs.^2));
  % Save filter, gain and residual
  a(hop,:) = aa;
  g(hop) = G;
  if ov == 0
   e((hop - 1)*h + [1:h]) = rs'/G;
  else
   e((hop - 1)*h + [1:w]) =  e((hop - 1)*h + [1:w]) + rs/G;
  end
end
 % Throw away first (win-hop)/2 pts if in overlap mode
```

```
% for proper synchronization of resynth
if ov ~= 0
  e = e((1+((w-h)/2)):end);
end
```

lpcsynth.m

```
function d = lpcsynth(a,g,e,h,ov)
% d = lpcsynth(a,g,e,h,ov)   Resynthesize from LPC representation
%    Each row of a is an LPC fit to a h-point (non-overlapping)
%    frame of data.  g gives the overall gains for each frame and
%    e is an excitation signal (if e is empty, white noise is used;
%    if e is a scalar, a pulse train is used with that period).
%    ov nonzero selects overlap-add of reconstructed
%    windows, else e is assumed to consist of independent hop-sized
%    segments that will line up correctly without cross-fading
%    (matching the ov option to lpcfit; default is ov = 1).
%    Return d as the resulting LPC resynthesis.
% 2001-02-25 dpwe@ee.columbia.edu $Header:
/homes/dpwe/matlab/columbiafns/RCS/lpcsynth.m,v 1.1 2004/03/30 20:56:04
dpwe Exp $

if nargin < 3
  e = [];
end
if nargin < 4
  h = 128;
end
if nargin < 5
  ov = 1;
end
```

```
w = 2*h;

[nhops,p] = size(a);

npts = nhops*h;
% Excitation needs extra half-window at the end if in overlap mode
nepts = npts + ov*(w-h);

if length(e) == 0
  e = randn(1,nepts);
elseif length(e) == 1
  pd = e;
  e = sqrt(pd) * (rem(1:nepts,pd) == 0);
else
  nepts = length(e);
  npts = nepts - ov*(w-h);
end

% Try to make sure we don't run out of e (in ov mode)
e = [e, zeros(1, w)];

d = zeros(1,npts);

for hop = 1:nhops

  hbase = (hop-1)*h;

  oldbit = d(hbase + [1:h]);
  aa = a(hop,:);
  G = g(hop);
  if ov == 0
    newbit = G*filter(1, aa, e(hbase + [1:h]));
```

```
  else
    newbit = G*filter(1, aa, e(hbase + [1:w]));
  end
  if ov == 0
    d(hbase + [1:h]) = newbit;
  else
    d(hbase + [1:w]) = [oldbit, zeros(1,(w-h))] + (hanning(w)'.*newbit);
  end

end

% De-emphasis (must match pre-emphasis in lpcfit)
pre = [1 -0.9];
d = filter(1,pre,d);
```

enframe.m

```
function f=enframe(x,win,inc)
%ENFRAME split signal up into (overlapping) frames: one per row.
F=(X,WIN,INC)
%
%   F = ENFRAME(X,LEN) splits the vector X up into
%   frames. Each frame is of length LEN and occupies
%   one row of the output matrix. The last few frames of X
%   will be ignored if its length is not divisible by LEN.
%   It is an error if X is shorter than LEN.
%
%   F = ENFRAME(X,LEN,INC) has frames beginning at increments of INC
%   The centre of frame I is X((I-1)*INC+(LEN+1)/2) for I=1,2,...
%   The number of frames is fix((length(X)-LEN+INC)/INC)
%
%   F = ENFRAME(X,WINDOW) or ENFRAME(X,WINDOW,INC) multiplies
```

```
%   each frame by WINDOW(:)

%   Copyright (C) Mike Brookes 1997
%
%      Last modified Tue May 12 13:42:01 1998
%
%   VOICEBOX home page:
http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%
%   This program is free software; you can redistribute it and/or modify
%   it under the terms of the GNU General Public License as published by
%   the Free Software Foundation; either version 2 of the License, or
%   (at your option) any later version.
%
%   This program is distributed in the hope that it will be useful,
%   but WITHOUT ANY WARRANTY; without even the implied warranty of
%   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See
the
%   GNU General Public License for more details.
%
%   You can obtain a copy of the GNU General Public License from
%   ftp://prep.ai.mit.edu/pub/gnu/COPYING-2.0 or by writing to
%   Free Software Foundation, Inc.,675 Mass Ave, Cambridge, MA 02139, USA.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%

nx=length(x);
nwin=length(win);
```

```
if (nwin == 1)
   len = win;
else
   len = nwin;
end
if (nargin < 3)
   inc = len;
end
nf = fix((nx-len+inc)/inc);
f=zeros(nf,len);
indf= inc*(0:(nf-1)).';
inds = (1:len);
f(:) = x(indf(:,ones(1,len))+inds(ones(nf,1),:));
if (nwin > 1)
    w = win(:)';
    f = f .* w(ones(nf,1),:);
end
```

warppoles.m

```
function [B,A] = warppoles(a,alpha)
%  [B,A] = warppoles(a,alpha)  warp an all-pole polynomial by substitution
%    Warp all-pole polynomials defined by rows of a by the first-order
%    warp factor alpha.  Negative alpha shifts poles up in frequency.
%    Output polynomials have zeros too, hence B and A.
% 2003-12-10 dpwe@ee.columbia.edu

% Construct z-hat^-1 polynomial
d = [-alpha 1];
c = [1 -alpha];

[nrows,order] = size(a);
```

```
A = zeros(nrows, order);
B = zeros(nrows, order);

B(:,1) = a(:,1);
A(:,1) = ones(nrows,1);

dd = d;
cc = c;

% This code originally mapped zeros.  I adapted it to map
% poles just by interchanging b and a, then swapping again at the
% end.  Sorry that makes the variables confusing to read.

for n = 2:order

 for row = 1:nrows

  % add another factor to num, den
  B(row,1:order) = conv(B(row,1:(order-1)), c);

 end

 % accumulate terms from this factor
 B(:,1:length(dd)) = B(:,1:length(dd)) + a(:,n)*dd;

 dd = conv(dd, d);
 cc = conv(cc, c);

end

% Construct the uniform A polynomial (same for all rows)
```

```
AA = 1;
for n = 2:order
  AA = conv(AA,c);
end
A = repmat(AA, nrows, 1);

% Exchange zeros and poles
T = A; A = B; B = T;
```

coba.bat

```
C:\windows\notepad.exe
```