main.m:

function varargout = main(varargin)

% MAIN MATLAB code for main.fig

% MAIN, by itself, creates a new MAIN or raises the existing

% singleton\*.

%

% H = MAIN returns the handle to a new MAIN or the handle to

% the existing singleton\*.

%

% MAIN('CALLBACK',hObject,eventData,handles,...) calls the local

% function named CALLBACK in MAIN.M with the given input arguments.

%

% MAIN('Property','Value',...) creates a new MAIN or raises the

% existing singleton\*. Starting from the left, property value pairs are

% applied to the GUI before main\_OpeningFcn gets called. An

% unrecognized property name or invalid value makes property application

% stop. All inputs are passed to main\_OpeningFcn via varargin.

%

% \*See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one

% instance to run (singleton)".

%

% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help main

% Last Modified by GUIDE v2.5 07-Aug-2014 21:18:20

% Begin initialization code - DO NOT EDIT

gui\_Singleton = 1;

gui\_State = struct('gui\_Name', mfilename, ...

'gui\_Singleton', gui\_Singleton, ...

'gui\_OpeningFcn', @main\_OpeningFcn, ...

'gui\_OutputFcn', @main\_OutputFcn, ...

'gui\_LayoutFcn', [] , ...

'gui\_Callback', []);

if nargin && ischar(varargin{1})

gui\_State.gui\_Callback = str2func(varargin{1});

end

if nargout

[varargout{1:nargout}] = gui\_mainfcn(gui\_State, varargin{:});

else

gui\_mainfcn(gui\_State, varargin{:});

end

% End initialization code - DO NOT EDIT

% --- Executes just before main is made visible.

function main\_OpeningFcn(hObject, eventdata, handles, varargin)

% This function has no output args, see OutputFcn.

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% varargin command line arguments to main (see VARARGIN)

% Choose default command line output for main

handles.output = hObject;

% Update handles structure

guidata(hObject, handles);

textIn=get(handles.text1,'String');

set(handles.text2,'Visible','off');

set(handles.text3,'Visible','off');

set(handles.id,'Visible','off');

set(handles.name,'Visible','off');

set(handles.submit,'Visible','off');

set(handles.back,'Visible','off');

% UIWAIT makes main wait for user response (see UIRESUME)

% uiwait(handles.figure1);

% --- Outputs from this function are returned to the command line.

function varargout = main\_OutputFcn(hObject, eventdata, handles)

% varargout cell array for returning output args (see VARARGOUT);

% hObject handle to figure

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure

varargout{1} = handles.output;

% --- Executes on button press in add.

function add\_Callback(hObject, eventdata, handles)

% hObject handle to add (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

set(handles.text2,'Visible','on');

set(handles.text3,'Visible','on');

set(handles.id,'Visible','on');

set(handles.name,'Visible','on');

set(handles.add,'Visible','off');

set(handles.test,'Visible','off');

set(handles.submit,'Visible','on');

set(handles.back,'Visible','on');

% --- Executes on button press in test.

function test\_Callback(hObject, eventdata, handles)

% hObject handle to test (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

recog();

set(handles.back,'Visible','off');

function id\_Callback(hObject, eventdata, handles)

% hObject handle to id (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of id as text

% str2double(get(hObject,'String')) returns contents of id as a double

% --- Executes during object creation, after setting all properties.

function id\_CreateFcn(hObject, eventdata, handles)

% hObject handle to id (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

function name\_Callback(hObject, eventdata, handles)

% hObject handle to name (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of name as text

% str2double(get(hObject,'String')) returns contents of name as a double

% --- Executes during object creation, after setting all properties.

function name\_CreateFcn(hObject, eventdata, handles)

% hObject handle to name (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.

% See ISPC and COMPUTER.

if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))

set(hObject,'BackgroundColor','white');

end

% --- Executes on button press in submit.

function submit\_Callback(hObject, eventdata, handles)

% hObject handle to submit (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

global ID NAME

ID=get(handles.id,'string');

NAME=get(handles.name,'string');

if (isempty(ID) || isempty(NAME))

msgbox('Enter all Values');

else

add1(ID,NAME);

end

% --- Executes on button press in back.

function back\_Callback(hObject, eventdata, handles)

% hObject handle to back (see GCBO)

% eventdata reserved - to be defined in a future version of MATLAB

% handles structure with handles and user data (see GUIDATA)

set(handles.text2,'Visible','off');

set(handles.text3,'Visible','off');

set(handles.id,'Visible','off');

set(handles.name,'Visible','off');

set(handles.submit,'Visible','off');

set(handles.add,'Visible','on');

set(handles.test,'Visible','on');

2. Adding Voice to Database:

function ADD=add1(ID,NAME)

NB=8;

FS=22050;

if (exist('voice\_database.dat','file')==2)

load('voice\_database.dat','-mat');

sound\_number = sound\_number+1;

else

sound\_number =1;

data{sound\_number,1} = [];

data{sound\_number,2} = [];

data{sound\_number,3} = [];

save('voice\_database.dat','data','sound\_number','FS','NB');

end

% ID=input('Enter a id');

% duration=input('Enter the duration of recording(in seconds): ');

duration=3;

% if isempty(duration)

% error('Invalid Duration');

% end

recording=audiorecorder(FS,NB,1);

msgbox('Speak');

% display('Speak');

record(recording,duration);

while(isrecording(recording)==1)

close

msgbox('Recording...')

% disp('recording...');

pause(0.5);

end

% display('Recording stopped');

close

msgbox('Recording stopped');

D1=getaudiodata(recording);

st=strcat('u',num2str(sound\_number));

wavwrite(D1,FS,NB,st);

D1 =getaudiodata(recording, 'uint8');

if size(D1,2)==2

D1=D1(:,1);

end

D1 = double(D1);

data{sound\_number,1} = D1;

data{sound\_number,2} = ID;

data{sound\_number,3} = NAME;

save('voice\_database.dat','data','sound\_number','-append');

display('Voice added to database');

end

3. For Voice Recognition:

function ADD=add1(ID,NAME)

NB=8;

FS=22050;

if (exist('voice\_database.dat','file')==2)

load('voice\_database.dat','-mat');

sound\_number = sound\_number+1;

else

sound\_number =1;

data{sound\_number,1} = [];

data{sound\_number,2} = [];

data{sound\_number,3} = [];

save('voice\_database.dat','data','sound\_number','FS','NB');

end

% ID=input('Enter a id');

% duration=input('Enter the duration of recording(in seconds): ');

duration=3;

% if isempty(duration)

% error('Invalid Duration');

% end

recording=audiorecorder(FS,NB,1);

msgbox('Speak');

% display('Speak');

record(recording,duration);

while(isrecording(recording)==1)

close

msgbox('Recording...')

% disp('recording...');

pause(0.5);

end

% display('Recording stopped');

close

msgbox('Recording stopped');

D1=getaudiodata(recording);

st=strcat('u',num2str(sound\_number));

wavwrite(D1,FS,NB,st);

D1 =getaudiodata(recording, 'uint8');

if size(D1,2)==2

D1=D1(:,1);

end

D1 = double(D1);

data{sound\_number,1} = D1;

data{sound\_number,2} = ID;

data{sound\_number,3} = NAME;

save('voice\_database.dat','data','sound\_number','-append');

display('Voice added to database');

end

4. Supporting functions:

Mfcc.m:

function R=mfcc(y,fs)

N=256;

M=100;

stepsize=N-M;

nframes=(length(y)/stepsize)-1;

s1=1;

s2=N;

frame=[];

for i=1:nframes

fr=y(s1:s2);

frame=[frame,fr];

s1=s1+stepsize;

s2=s2+stepsize;

end

h=hamming(256);

H=diag(h)\*frame;

for i=1:nframes

F(:,i)=fft(H(:,i));

end

m=melfb(20,N,fs);

n2=1+floor(N/2);

z=m\*abs(F(1:n2,:)).^2;

R=dct(log(z));

t=N/2;

tm=length(y)/fs;

end

disteu.m:

function d = disteu(x, y)

[M, N] = size(x);

[M2, P] = size(y);

if (M ~= M2)

error('Matrix dimensions do not match.')

end

d = zeros(N, P);

% if (N < P)

% copies = zeros(1,P);

% for n = 1:N

% d(n,:) = sum((x(:, n+copies) - y) .^2, 1);

% end

% else

% copies = zeros(1,N);

% for p = 1:P

% d(:,p) = sum((x - y(:, p+copies)) .^2, 1)';

% end

% end

% d = d.^0.5;

for ii=1:N

for jj=1:P

%d(ii,jj)=sum((x(:,ii)-y(:,jj)).^2).^0.5;

d(ii,jj) = mydistance(x(:,ii),y(:,jj),2);

end

end

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

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

end

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

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

%% mydistance function

% x and y are input 1D vectors

% out is the measured distance

% Euclidean distance

function [out] = mydistance(x,y,tipo)

if tipo == 0

out = sum((x-y).^2).^0.5;

end

% Distance sum | x -y |

if tipo == 1

out = sum(abs(x-y));

end

% Weighted distance

if tipo == 2

pesi = zeros(size(x));

pesi(1) = 0.20;

pesi(2) = 0.90;

pesi(3) = 0.95;

pesi(4) = 0.90;

pesi(5) = 0.70;

pesi(6) = 0.90;

pesi(7) = 1.00;

pesi(8) = 1.00;

pesi(9) = 1.00;

pesi(10) = 0.95;

pesi(11:13) = 0.30;

out = sum(abs(x-y).\*pesi);

%out = sum(pesi.\*(x-y).^2).^0.5;

end

end

melfb.m:

function m = melfb(p, n, fs)

% MELFB Determine matrix for a mel-spaced filterbank

%

% Inputs: p number of filters in filterbank

% n length of fft

% fs sample rate in Hz

%

% Outputs: x a (sparse) matrix containing the filterbank amplitudes

% size(x) = [p, 1+floor(n/2)]

%

% Usage: For example, to compute the mel-scale spectrum of a

% colum-vector signal s, with length n and sample rate fs:

%

% f = fft(s);

% m = melfb(p, n, fs);

% n2 = 1 + floor(n/2);

% z = m \* abs(f(1:n2)).^2;

%

% z would contain p samples of the desired mel-scale spectrum

%

% To plot filterbanks e.g.:

%

% plot(linspace(0, (12500/2), 129), melfb(20, 256, 12500)'),

% title('Mel-spaced filterbank'), xlabel('Frequency (Hz)');

f0 = 700 / fs;

fn2 = floor(n/2);

lr = log(1 + 0.5/f0) / (p+1);

% convert to fft bin numbers with 0 for DC term

bl = n \* (f0 \* (exp([0 1 p p+1] \* lr) - 1));

b1 = floor(bl(1)) + 1;

b2 = ceil(bl(2));

b3 = floor(bl(3));

b4 = min(fn2, ceil(bl(4))) - 1;

pf = log(1 + (b1:b4)/n/f0) / lr;

fp = floor(pf);

pm = pf - fp;

r = [fp(b2:b4) 1+fp(1:b3)];

c = [b2:b4 1:b3] + 1;

v = 2 \* [1-pm(b2:b4) pm(1:b3)];

m = sparse(r, c, v, p, 1+fn2);

vqlbg.m:

function r = vqlbg(d,k)

e = .01;

r = mean(d, 2);

dpr = 10000;

for i = 1:log2(k)

r = [r\*(1+e), r\*(1-e)];

while (1 == 1)

z = disteu(d, r);

[m,ind] = min(z, [], 2);

t = 0;

for j = 1:2^i

r(:, j) = mean(d(:, find(ind == j)), 2); %#ok<FNDSB>

x = disteu(d(:, find(ind == j)), r(:, j)); %#ok<FNDSB>

for q = 1:length(x)

t = t + x(q);

end

end

if (((dpr - t)/t) < e)

break;

else

dpr = t;

end

end

end

end