CS 2108 Project

Parameter list:

Freq_Step	100 Hz
Start_freq	9000 Hz
End_freq	8900 Hz
Each word period	0.05s
Start period	0.15s
End period	0.15s

Algorithm introduction:

For encode part, I use a MATLAB map to assign each character with one frequency. And then I generate encode music. After that, I use low pass filter to delete the high frequency in source music and replace it with my encode music. After getting the combined music, I start to decode the message.

During encoding,

Insert 0 frequency between each word during encoding in order to solve the synchronization problem.

```
for i = 1: length(readMsg)

    period = 0.0025;
    t = [0 : timeStep : period-timeStep];
    freq = 0;
    y = sin(2*pi*freq*t);
    encode_music = [encode_music,y];

    %disp(readMsg(i))
    period = 0.045;
    char = readMsg(i);
    freq = encode_dic(char);
    t = [0 : timeStep : period-timeStep];
    y = sin(2*pi*freq*t);
    encode_music = [encode_music,y];

    period = 0.0025;
    t = [0 : timeStep : period-timeStep];
    freq = 0;
    y = sin(2*pi*freq*t);
    encode_music = [encode_music,y];
end
```

During combine message,

Try to insert the music in the middle of the source music so that source music is much louder than encoded music.

```
%source: https://in.mathworks.com/matlabcentral/answers/223712-making-two-different-vect
maxlen = max(length(source_music), length(encode_music));

encode_music_c = source_music;
start_point = int32(length(source_music)/3);
encode_music_c(1:start_point) = 0;
encode_music_c(start_point+1:start_point+length(encode_music)) = encode_music;
encode_music_c(end+1:maxlen) = 0;

combined_music = 0.012*encode_music_c+source_music;
cm = audioplayer(combined_music, Fs_source);
audiowrite(strcat("combined music.mp4"),combined_music,Fs_source);
%play(cm)
```

During decode,

I save my music into array with every 0.05s time slot

```
%split the music array
freq_sample = 44100;
period = 0.05;
chunk_len = int32(freq_sample*period);
chunk_num = idivide(length(record_mu),chunk_len);
```

Use the High pass filter to give away low frequency so that I can decode high frequency easier.

```
%highpass
highpass(record_mu,10000,freq_sample);
record_mu_filter = highpass(record_mu,10000,freq_sample);

%check the music after hpf
figure
plot(record_mu_filter)
```

Solve Synchronization Problem,

Using Three methods:

1. Give 0 frequency between each word when encoding (mentioned above)
2. Use the sliding window to find a start point

   The accuracy is the best when we find the point where the last point with frequency among 9000 and 9010 minus 199 points is the best time. 199 points mean 0.0199s here and can be different for different time interval. In my project, the time interval is 0.05s.

```
%sliding window to get the point where decode starts
if(0)
    value = 0;
    for k = 0:150000
        time_interval = 0.0001;
        incre = int32(freq_sample*time_interval*k);
        sound = record_mu_filter(1+incre:incre+chunk_len);
        sound_fft = fft(sound, 2^nextpow2(length(sound)));
        mx = max(abs(sound_fft));
        idx = find(abs(sound_fft)==mx);
        delFreq = freq_sample / length(sound_fft);
        start_freq = (idx(1)-1).*delFreq;
        if start_freq >= 9000 & start_freq <= 9010
            value = k-199;
        end

    end
end
```

3. Choose the center of each time slot
   When we decode, for each time slot we discard the right and left ends data and only use the center.

```
sound = record_mu_filter(start_chunk+1+(i-1)*chunk_len+400:start_chunk+(i)*chunk_len-400)
sound_fft = fft(sound, 2^nextpow2(length(sound)));
```

Find 3 Peak and calculate their frequency. Choose one if previous peak's frequency has no corresponding value in decoding dictionary.

```
[max_v, index] = maxk(abs(sound_fft),6);

mx = max(abs(sound_fft));
idx = find(abs(sound_fft)==mx);
delFreq = freq_sample / length(sound_fft);

fTemp = (idx(1)-1).*delFreq;
fTemp = FreqStep*round(fTemp/FreqStep);

fTemp_2 = (index(3)-1).*delFreq;
fTemp_2 = FreqStep*round(fTemp_2/FreqStep);

fTemp_3 = (index(5)-1).*delFreq;
fTemp_3 = FreqStep*round(fTemp_3/FreqStep);
```

Detect start and end.

```
    if fTemp == 9000
        flag = 1;
    end

    if fTemp == 8900
        flag = 0;
        decode_word = decode_dic(fTemp);
        deccode_mssg = [deccode_mssg decode_word];
        break
    end
```

Decode the message,
If start skip.
If the repeat word exists, first check whether they should be different and just a synchronization problem. If yes, then use the second peak's frequency.

```
if(flag)
    if any(decode_freq(:) == fTemp)
        if fTemp == 9000
            continue
        end
        decode_word = decode_dic(fTemp);
        if length(idx_detect)>2 & deccode_mssg(end) == decode_word
            decode_word = decode_dic(fTemp_2);
            deccode_mssg = [deccode_mssg decode_word];
            continue
        end
        deccode_mssg = [deccode_mssg decode_word];
    elseif any(decode_freq(:) == fTemp_2)
        decode_word = decode_dic(fTemp_2);
        deccode_mssg = [deccode_mssg decode_word];
    elseif any(decode_freq(:) == fTemp_3)
        decode_word = decode_dic(fTemp_3);
        deccode_mssg = [deccode_mssg decode_word];
```

How to display:

1. Encode the message using "proj_encode.m"
2. Combine the encoded music and source music using "combine_music.m"
3. Record the music using "record_music.m"
4. Decode the message using "decode_music.m"

Note:
My classmate Ma Jianheng and I have cooperated work on this project.