Digital Signal Processing

Mini-Project:

An automatic speaker recognition system

February 3, 2003

Responsible: Vladan Velisavljevic
Authors: Christian Cornaz
Urs Hunkeler

1 Table of Contents

IMPORTANT REMARK

- All Matlab code can be found on the CD-Rom. Most pictures were prepared using the demo script.
- The code snippets in this document are not always complete. We took the liberty to remove commands, which are used for beautifying the output, and change variable names to clarify the code in this document. For a detailed view of the code, please consult the scripts on the CD-Rom.

2 Question 1: Human speaker recognition

Play each sound file in the TRAIN folder. Can you distinguish the voices of those eight speakers? Now play each sound in the TEST folder in a random order without looking at the file name (pretending that you do not know the speaker) and try to identify the speaker using your knowledge of their voices that you just learned from the TRAIN folder. This is exactly what the computer will do in our system. What is your (human performance) recognition rate? Record this result so that it can be used later to be compared with the computer's performance of our system.

Both of us seem to be unable to recognise random people just by listening at their voice. Our success rates for the provided samples were 1 person out of 8 each. However, for the samples we used in question 10, we were easily able to recognise each speaker. This is probably because we knew all the persons.
We also realized that we did not identify speakers by the frequencies they use to talk, but rather by other characteristics, like accent, speed, etc.

3 Question 2: Technical data of samples

Read a sound file into Matlab. Check it by playing the sound file in Matlab using the function: sound. What is the sampling rate? What is the highest frequency that the recorded sound can capture with fidelity? With that sampling rate, how many msecs of actual speech are contained in a block of 256 samples?

We analysed the sample "train\s1.wave" with the following command:

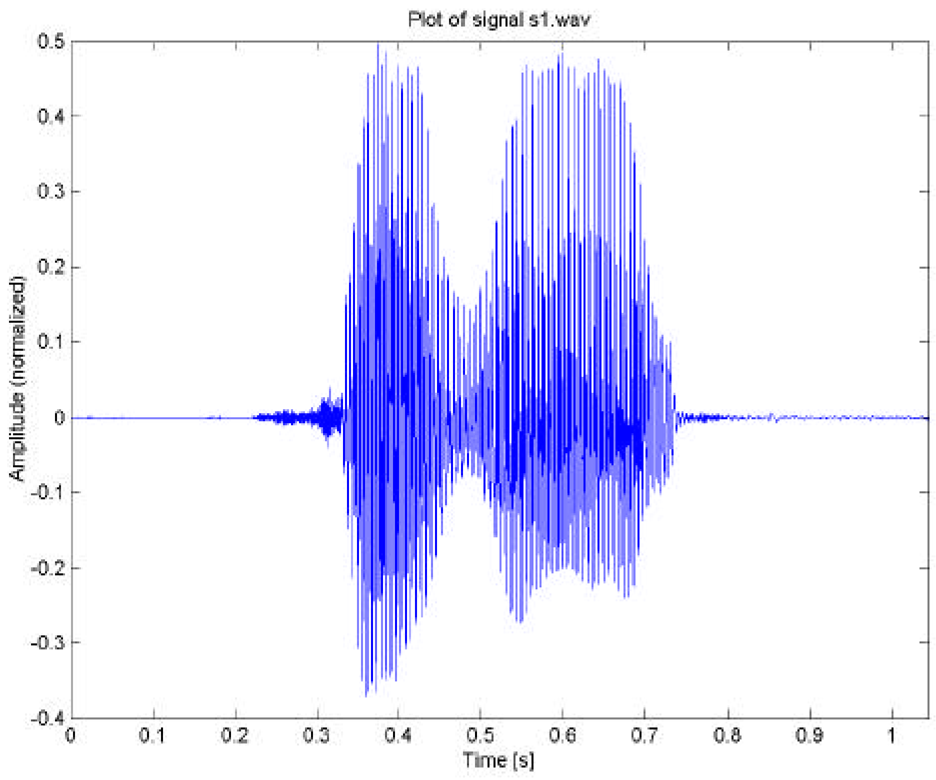
```
[s1, fs1] = wavread('train\s1.wav');
```

This command puts the signal data into the vector s1. The variable fs1 contains the sampling frequency, which is 12'500 [Hz] (or 12'500 samples per second).

According to the Shannon-Whithake theorem:
$W_s = 2 * W_o \Rightarrow f_s = 2 * f_o \Rightarrow f_o = f_s / 2 = 12'500 / 2 = 6'250$ [Hz]

One sample represents 1 / f_s [s]. 256 samples therefore contain 256 / f_s [s] of the actual signal: 256 / 12'500 = 0.02048 [s] = 20.48 [ms]

Plot the signal to view it in the time domain. It should be obvious that the raw data in the time domain has a great amount of data and for this reason it is difficult to analyse the voice characteristic. So the purpose of this step (speech feature extraction) should be clear now!



4 Intermediate Steps

4.1 Frames blocking phase

To obtain a matrix M containing all the frames, we used the following script:

```
l = length(s1);
m = 100;
n = 256;

nbFrame = floor((l - n) / m) + 1;

for i = 1:n
    for j = 1:nbFrame
        M(i, j) = s1(((j - 1) * m) + i);
    end
end
```

That way we obtain the 256 x 129 matrix M.

4.2 Windowing phase

We create the Hamming matrix and transform the matrix M into the new matrix M2, where the column vectors of M2 are the original frame vectors transformed by the Hamming filter.

```
h = hamming(n);
M2 = diag(h) * M;
```

4.3 FFT Phase

We create a new matrix M3 where the column vectors are the FFTs of the column vectors of M2.

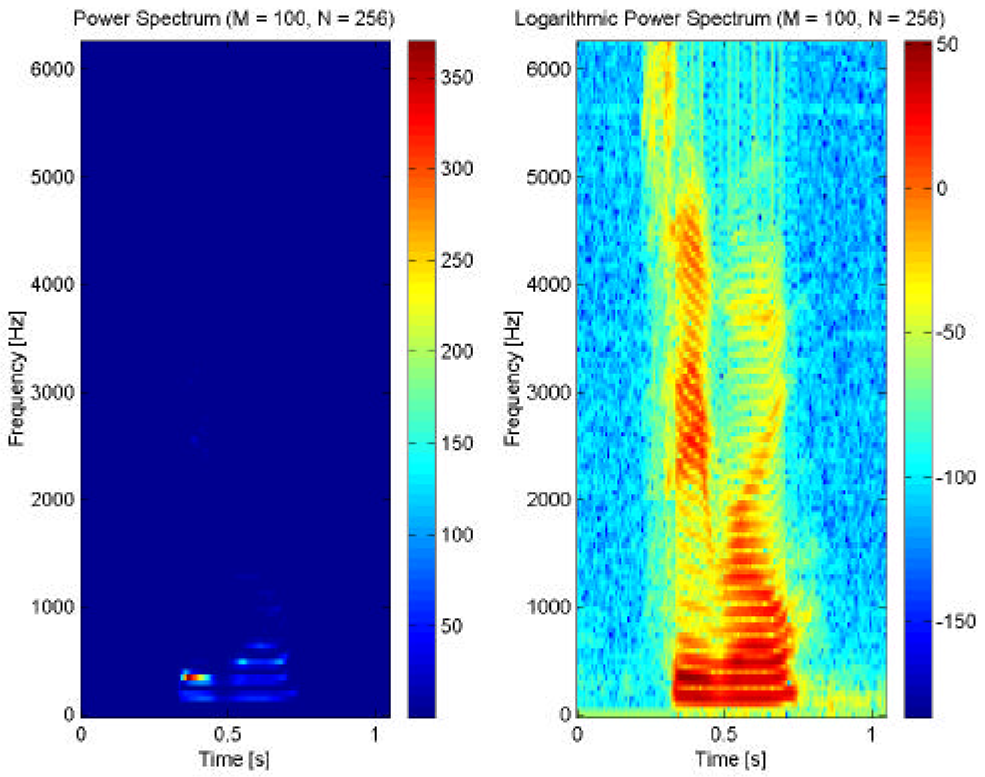
```
for i = 1:nbFrame
    M3(:, i) = fft(M2(:, i));
end
```

5 Question 3: Power Spectrum

After successfully running the preceding process, what is the interpretation you can make of the result obtained? Compute the power spectrum and plot it out using the imagesc command. Note that it is better to view the power spectrum on the log scale. Locate the region in the plot that contains most of the energy. Translate this location into the actual ranges in time (msec) and frequency (in Hz) of the input speech signal.

As stated before, the columns matrix M3 contain the frames of the original signal, filtered by the Hamming filter and transformed with the FFT. The elements of M3 are complex numbers and symmetrical because FFT was used to transform the data.
Each column in M3 is a power spectrum representation of the original signal.

To plot the power spectrum, we take the absolute values of the matrix elements. Since the spectrum is symmetric, we only plot half of it. Note that we plot simultaneously with the linear and the logarithmic spectrum scale.
The result obtained is the following plot:

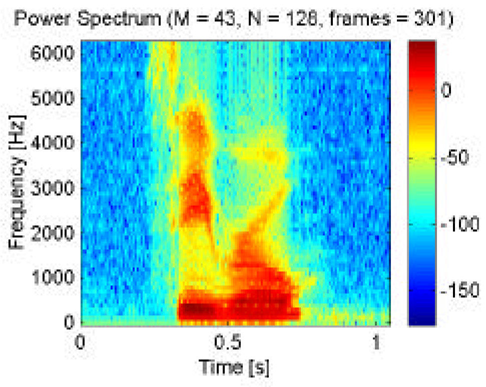


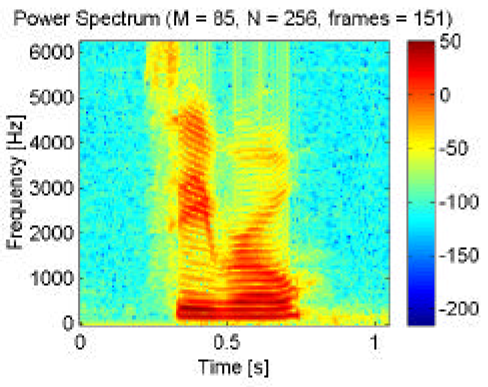
In this plot, the areas containing the highest level of energy are displayed in red. As we can see on the plot, the red area is located between 0.3 and 0.7 seconds. The plot also shows that most of the energy is concentrated in the lower frequencies (between 50 Hz and 1 kHz). For a more detailed plot run the demo script on the CD-Rom.

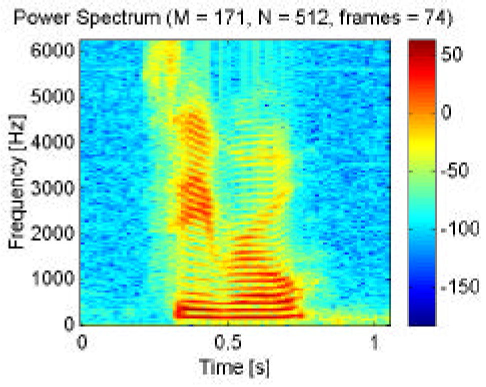
6 Question 4: Power Spectrum with different M and N

Compute and plot the power spectrum of a speech file using different frames sizes: for example N = 128, 256 and 512. In each case, set the frame increment M to be about N/3. Can you describe and explain the differences between these spectra?

See the demo script on the CD-Rom to see how we obtained the following plots.







For N = 128 we have a high resolution of time. Furthermore each frame lasts a very short period of time. This result shows that the signal for a frame doesn't change its nature (i.e. it will be for the same vowel or consonant). On the other hand, there are only 65 distinct frequencies samples. This means that we have a poor frequency resolution.

For N = 256 we have a compromise between the resolution in time and the frequency resolution.

For N = 512 we have an excellent frequency resolution (256 different values) but there are lesser frames, meaning that the resolution in time is strongly reduced.

It seems that a value of 256 for N is an acceptable compromise. Furthermore the number of frames is relatively small, which will reduce computing time.

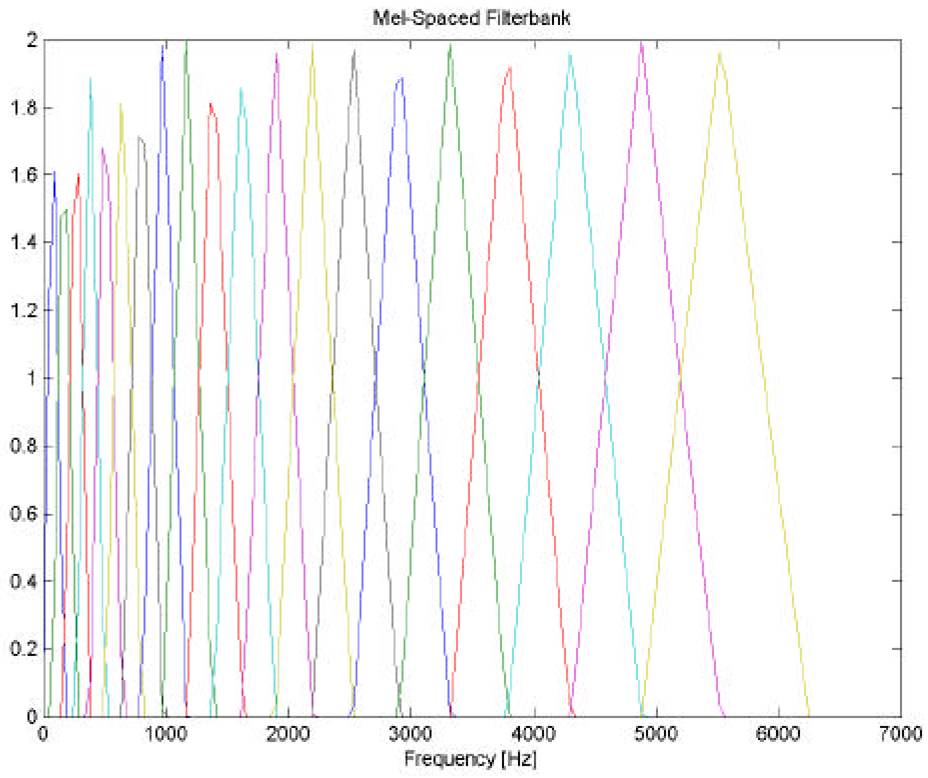
7 Question 5: Mel-Spaced Filter Bank

Type 'help melfb' at the Matlab prompt for more information about this function. Follow the guidelines to plot out the mel-spaced filter bank. What is the behaviour of this filter bank? Compare it with the theoretical part.

To plot this filter bank, we use the following command:

```
plot(linspace(0, (fs1/2), 129), (melfb(20, 256, fs1))');
```

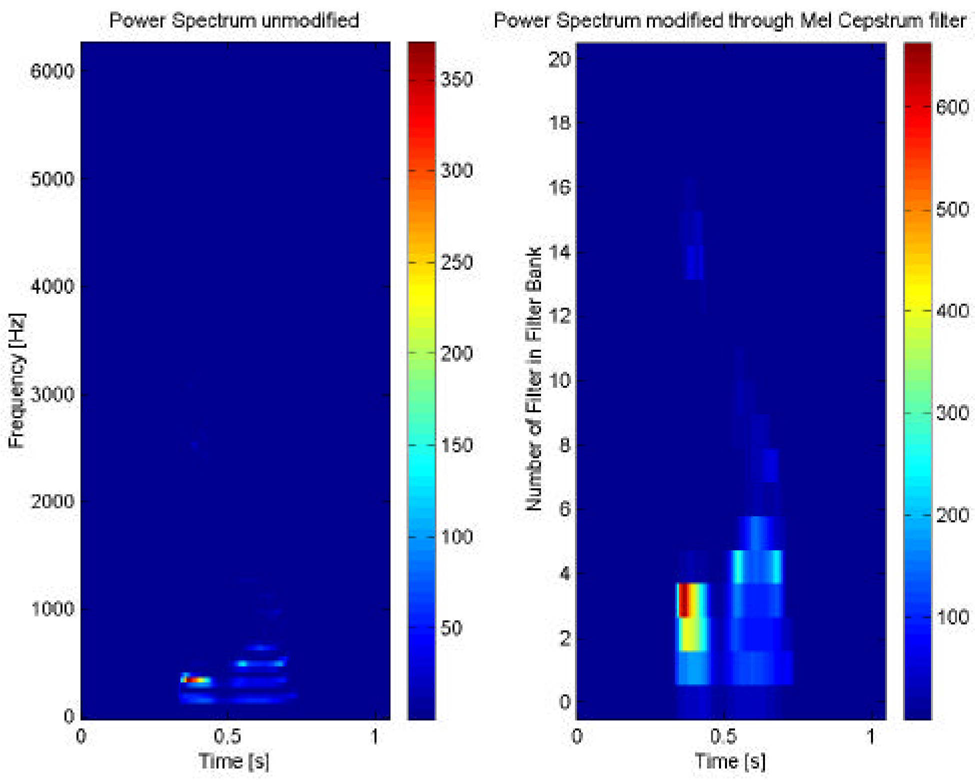
Here is the plot we obtained:



This filter bank behaves like a succession of histograms on the spectrum. Each filter of the filter bank has a triangular frequency response. It quantifies the zone of the frequency spectrum.
The filter bank is used to transform the spectrum of a signal into a representation which reflects more closely the behaviour of the human ear. As the human ear (or the associated neurons) favours low frequencies for analysing speech, the filters are denser for the lower frequencies. To mimic the human ear, the filters are linearly distributed for low frequencies (below 1kHz). For higher frequencies (above 1 kHz) the distribution of the filters is logarithmic.
This plot is basically the same figure as in chapter 3.2.4 of the project guidelines. However we calculated 20 filters (instead of 12).

8 Question 6: Spectrum before and after Mel-Frequency wrapping

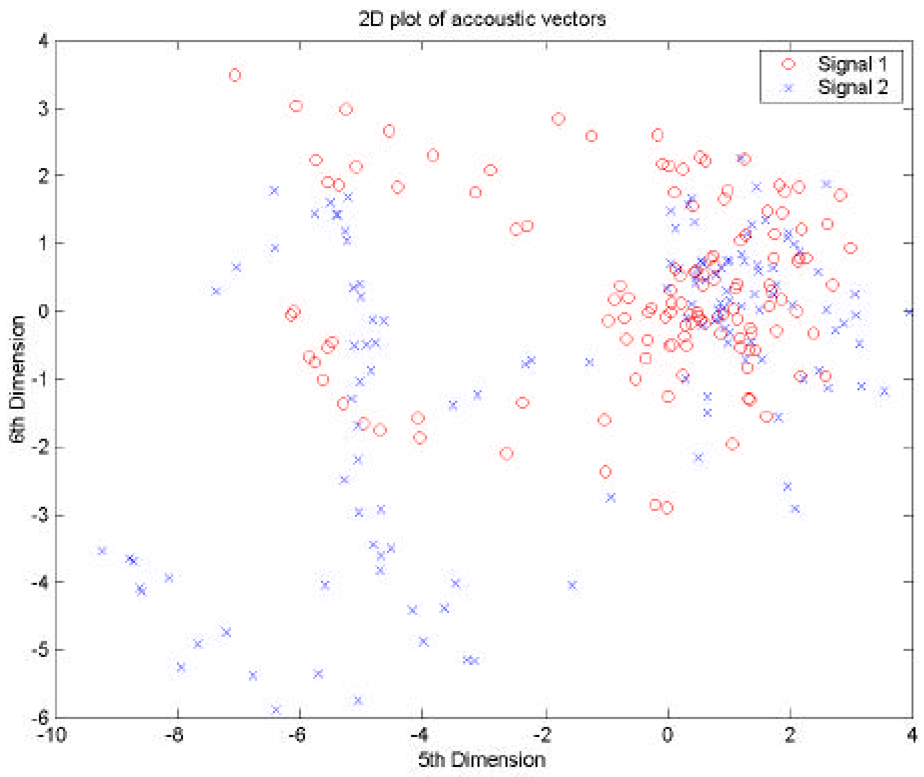
Compute and plot the spectrum of a speech file before and after the mel-frequency wrapping step. Describe and explain the impact of the melfb program.



As we can see in the first plot, most of the information is contained in the lower frequencies. This information has been extracted and amplified in the second plot. The second plot therefore shows the main characteristics of the speech signal.
Note that the transformation produced an acoustic vector of 20 dimensions.

9 Question 7: 2D plot of acoustic vectors

To inspect the acoustic space (MFCC vectors) we can pick any two dimensions (say the 5th and the 6th) and plot the data points in a 2D plane. Use acoustic vectors of two different speakers and plot data points in two different colours. Do the data regions from the two speakers overlap each other? Are they in clusters?



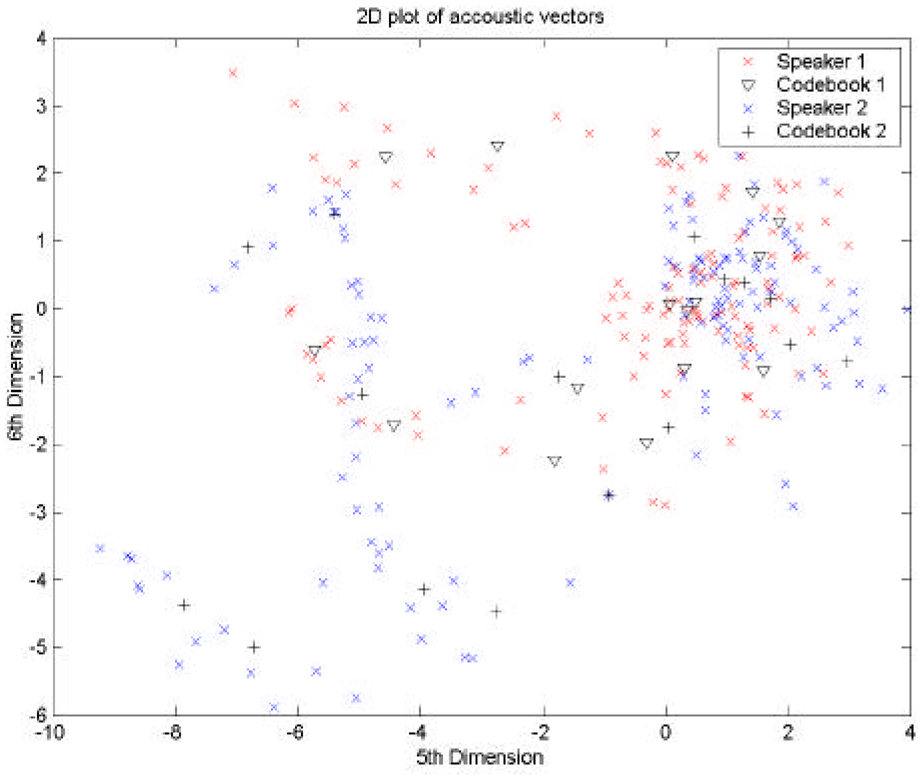
This graph was obtained with the following code (please refer to the demo script on the CD-Rom for more details):

```
c1 = mfcc(s1, fs1);
c2 = mfcc(s2, fs2);
plot(c1(5, :), c1(6, :), 'or');
hold on;
plot(c2(5, :), c2(6, :), 'xb');
```

Mostly the two areas overlap. But certain regions seem to be used exclusively by one or the other speaker. This is what will allow us to distinguish the different speakers.
The points don't form actual clusters, but there are areas where the density of points is higher.
Please note that this is only a two dimensional plot. The actual vector contains 20 dimensions.

10 Question 8: Plot of VQ codewords

Plot the data points of the trained VQ codeword using the two dimensions previously used for the plot of the last question. Compare this with Figure 5.



For a **nicer** plot of the codeword, please use the demo script provided on the CD-Rom.

The example in figure 5 is an ideal case for teaching. In reality however the clusters cannot be that easily distinguished. But again, this is a two dimensional representation of a 20 dimensions space.

11 Question 9: Recognition rate of the computer

What is the recognition rate that our system can perform? Compare this with the human's performances.

Observe as well if the system makes errors on the speech samples. If so, re-listen to the speech files and try to come up with some explanations.
We used the following commands to test the system:

```
code = train('train\', 8);
test('test\', 8, code);
```

The result is:

```
Speaker 1 matches with speaker 1
Speaker 2 matches with speaker 2
Speaker 3 matches with speaker 7
Speaker 4 matches with speaker 4
Speaker 5 matches with speaker 5
Speaker 6 matches with speaker 6
Speaker 7 matches with speaker 7
Speaker 8 matches with speaker 8
```

Our system was able to recognise 7 out of 8 speakers. This is an error rate of 12.5%. The recognition rate of our system is much better than the one of a human's recognition rate. However you must be aware that this test is not really representative of the computer's efficiency to recognise voices because we only tested on 8 persons, with only one training session and with only one word.

12 Question 10: Test with other speech files

You can also test the system with your own speech files. Use the Windows program Sound Recorder to record your own voice and your friends' voices! Each new speaker needs to provide one speech file for training and one for testing. Can the system recognize your voice? Enjoy!

Thanks to the participation of the following persons, we could test our system with real life voices:

• Justin Désiré Ngomsi Nene	alias s1
• Edouard de Moura Presa	alias s2
• Claude Richard	alias s3
• Emanuel Corthay	alias s4
• Giselle Toedtli	alias s5
• Christian Cornaz	alias s6
• Stefan Pelimon	alias s7
• Urs Hunkeler	alias s8
• Prof. Juan Mosig	alias s9
• Javier Bonny	alias s10

Here is the result:

```
Speaker 1 matches with speaker 1
Speaker 2 matches with speaker 10
Speaker 3 matches with speaker 3
Speaker 4 matches with speaker 4
Speaker 5 matches with speaker 5
Speaker 6 matches with speaker 6
Speaker 7 matches with speaker 7
Speaker 8 matches with speaker 8
Speaker 9 matches with speaker 8
Speaker 10 matches with speaker 10
```

In this second test, our system was able to recognise 8 out of 10 speakers. Actually we expected the error rate to increase because with more people there is also a bigger chance for mistakes. And of course the recording conditions were not optimal (background noise, etc).

13 Matlab code

13.1 blockFrames.m

```
function M3 = blockFrames(s, fs, m, n)
% blockFrames: Puts the signal into frames
%
% Inputs: s  contains the signal to analize
%         fs is the sampling rate of the signal
%         m  is the distance between the beginnings of two frames
%         n  is the number of samples per frame
%
% Output: M3 is a matrix containing all the frames
%
%
%%%%%%%%%%%%%%%%%%
% Mini-Project: An automatic speaker recognition system
%
% Responsible: Vladan Velisavljevic
% Authors:     Christian Cornaz
%              Urs Hunkeler
    l = length(s);
    nbFrame = floor((l - n) / m) + 1;

    for i = 1:n
        for j = 1:nbFrame
            M(i, j) = s(((j - 1) * m) + i);
        end
    end

    h = hamming(n);
    M2 = diag(h) * M;

    for i = 1:nbFrame
        M3(:, i) = fft(M2(:, i));
    end
```

13.2 mfcc.m

```
function r = mfcc(s, fs)
% MFCC
%
% Inputs: s  contains the signal to analize
%         fs is the sampling rate of the signal
%
% Output: r contains the transformed signal
%
%
%%%%%%%%%%%%%%%%%%
% Mini-Project: An automatic speaker recognition system
%
% Responsible: Vladan Velisavljevic
% Authors:     Christian Cornaz
%              Urs Hunkeler

m = 100;
n = 256;
l = length(s);

nbFrame = floor((l - n) / m) + 1;

for i = 1:n
    for j = 1:nbFrame
        M(i, j) = s(((j - 1) * m) + i);
    end
end

h = hamming(n);

M2 = diag(h) * M;
```

```
for i = 1:nbFrame
    frame(:,i) = fft(M2(:, i));
end

t = n / 2;
tmax = l / fs;

m = melfb(20, n, fs);
n2 = 1 + floor(n / 2);
z = m * abs(frame(1:n2, :)).^2;

r = dct(log(z));
```

13.3 vqlbg.m

```
function r = vqlbg(d,k)
% VQLBG Vector quantization using the Linde-Buzo-Gray algorithme
%
% Inputs: d contains training data vectors (one per column)
%         k is number of centroids required
%
% Output: r contains the result VQ codebook (k columns, one for each centroids)
%
%
%%%%%%%%%%%%%%%%%%
% Mini-Project: An automatic speaker recognition system
%
% Responsible: Vladan Velisavljevic
% Authors:     Christian Cornaz
%              Urs Hunkeler

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);
            x = disteu(d(:, find(ind == j)), r(:, j));
            for q = 1:length(x)
                t = t + x(q);
            end
        end
        if (((dpr - t)/t) < e)
            break;
        else
            dpr = t;
        end
   end
end
```

13.4 demo.m

```
% Demo script that generates all graphics in the report
% and demonstrates our results.
%
%
%%%%%%%%%%%%%%%%%%
% Mini-Project: An automatic speaker recognition system
%
% Responsible: Vladan Velisavljevic
% Authors:     Christian Cornaz
%              Urs Hunkeler

disp('Mini-Projet: Automatic Speaker Recognition');
disp('By Christian Cornaz & Urs Hunkeler (SC)');
disp('Responsible Assistant:  Vladan Velisavljevic');
disp(' ');
disp(' ');
[s1 fs1] = wavread('train\s1.wav');
[s2 fs2] = wavread('train\s2.wav');

%Question 2
disp('> Question 2');
t = 0:1/fs1:(length(s1) - 1)/fs1;
plot(t, s1), axis([0, (length(s1) - 1)/fs1 -0.4 0.5]);
title('Plot of signal s1.wav');
xlabel('Time [s]');
ylabel('Amplitude (normalized)')

pause
close all

%Question 3 (linear)
disp('> Question 3: linear spectrum plot');
M = 100;
N = 256;
frames = blockFrames(s1, fs1, M, N);
t = N / 2;
tm = length(s1) / fs1;
subplot(121);
imagesc([0 tm], [0 fs1/2], abs(frames(1:t, :)).^2), axis xy;
title('Power Spectrum (M = 100, N = 256)');
xlabel('Time [s]');
ylabel('Frequency [Hz]');
colorbar;

%Question 3 (logarithmic)
disp('> Question 3: logarithmic spectrum plot');
subplot(122);
imagesc([0 tm], [0 fs1/2], 20 * log10(abs(frames(1:t, :)).^2)), axis xy;
title('Logarithmic Power Spectrum (M = 100, N = 256)');
xlabel('Time [s]');
ylabel('Frequency [Hz]');
colorbar;
D=get(gcf,'Position');
set(gcf,'Position',round([D(1)*.5 D(2)*.5 D(3)*2 D(4)*1.3]))

pause
close all

%Question 4
disp('> Question 4: Plots for different values for N');
lN = [128 256 512];
u=220;
for i = 1:length(lN)
    N = lN(i);
    M = round(N / 3);
    frames = blockFrames(s1, fs1, M, N);
    t = N / 2;
    temp = size(frames);
    nbframes = temp(2);
    u=u+1;
    subplot(u)
    imagesc([0 tm], [0 fs1/2], 20 * log10(abs(frames(1:t, :)).^2)), axis xy;
    title(sprintf('Power Spectrum (M = %i, N = %i, frames = %i)', M, N, nbframes));
    xlabel('Time [s]');
    ylabel('Frequency [Hz]');
    colorbar
end
D=get(gcf,'Position');
set(gcf,'Position',round([D(1)*.5 D(2)*.5 D(3)*1.5 D(4)*1.5]))
pause
close all

%Question 5
disp('> Question 5: Mel Space');
plot(linspace(0, (fs1/2), 129), (melfb(20, 256, fs1))');
title('Mel-Spaced Filterbank');
xlabel('Frequency [Hz]');

pause
close all

%Question 6
disp('> Question 6: Modified spectrum');
M = 100;
N = 256;
frames = blockFrames(s1, fs1, M, N);
n2 = 1 + floor(N / 2);
m = melfb(20, N, fs1);
z = m * abs(frames(1:n2, :)).^2;
t = N / 2;
tm = length(s1) / fs1;
subplot(121)
imagesc([0 tm], [0 fs1/2], abs(frames(1:n2, :)).^2), axis xy;
title('Power Spectrum unmodified');
xlabel('Time [s]');
ylabel('Frequency [Hz]');
colorbar;
subplot(122)
imagesc([0 tm], [0 20], z), axis xy;
title('Power Spectrum modified through Mel Cepstrum filter');
xlabel('Time [s]');
ylabel('Number of Filter in Filter Bank');
colorbar;
D=get(gcf,'Position');
set(gcf,'Position',[0 D(2) D(3)*2 D(4)])
pause
close all

%Question 7
disp('> Question 7: 2D plot of accustic vectors');
c1 = mfcc(s1, fs1);
c2 = mfcc(s2, fs2);
plot(c1(5, :), c1(6, :), 'or');
hold on;
plot(c2(5, :), c2(6, :), 'xb');
xlabel('5th Dimension');
ylabel('6th Dimension');
legend('Signal 1', 'Signal 2');
title('2D plot of accoustic vectors');

pause
close all

%Question 8
disp('> Question 8: Plot of the 2D trained VQ codewords')
d1 = vqlbg(c1,16);
d2 = vqlbg(c2,16);
plot(c1(5, :), c1(6, :), 'xr')
hold on
plot(d1(5, :), d1(6, :), 'vk')
plot(c2(5, :), c2(6, :), 'xb')
plot(d2(5, :), d2(6, :), '+k')
xlabel('5th Dimension');
ylabel('6th Dimension');
legend('Speaker 1', 'Codebook 1', 'Speaker 2', 'Codebook 2');
title('2D plot of accoustic vectors');
pause
close all
```