Technische Universität Kaiserslautern

Institute of
Integrated Sensor Systems
Dept. of Electrical Engineering and Information Technology

Speech-Music+ Advertisement and Non-advertisement Recognition

February, 2010

Bartosz Niedźwiedź, Maciej Marczyński, Paweł Błażejowski

Overview

1. Introduction
   - Topic introduction
2. Parts of Project
   - Sound data acquisition
   - Feature selection and extraction
   - Classificiation
   - Results
3. Conclusion

Topic introduction

Main task

Developement of a system that is able to distinguish human speech from music in advertisement and non-advertisement sound tracks.

Sound data acquisition
Recording, preprocessing

Sound data was collected with dedicated sound processing software (CoolEdit Pro 2.0) by microphone from a TV. All of them were recorded with sampling frequency 44.1kHz, only one channel (mono), and saved to *.wav format (PCM).

TV channels: CNN, Super RTL, RTL II, ProSieben, RTL, Phoenix, 3Sat, N24.

To obtain samples from collected data, we have cut it randomly into 4-second long files (4s = 176400 samples).

Sound data acquisition
Sounds classification

Sound data was divided into three classes:

Train data:

- human speech (TV news) – class1 (15 sounds)
- advertisement (TV) – class2 (15 sounds)
- music – class3 (15 sounds)

Test data:

- 30 sounds (10 per class)

Sound data acquisition
Normalization

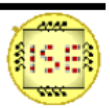
Sound samples were loaded to Matlab and normalized to standardize data by function shown below:

```
[n_of_samples, n_of_signals] = size(data);          %take sizes of input data
input_data_mean = zeros(1,n_of_signals);            %create empty table for results

for i=1:n_of_signals                                %calculate mean value
    input_data_mean(1,i) = mean(data(:,i));
end
for i=1:n_of_signals                                %normalize
    data(:,i) = data(:,i) - input_data_mean(1,i);
end
```

Feature selection and extraction: FFT

Code for FFT calculation:

```
[n_of_samples, n_of_signals] = size(data);          %take sizes of input data
NFFT = 2^nextpow2(n_of_samples);                    %calculate size for FFT
FFT = zeros(NFFT, n_of_signals);                    %create empty table for results

for i=1:n_of_signals                                %calculate FFT for each signal
     FFT(:,i)= fft(data(:,i),NFFT)/n_of_samples;
     Mag = abs(FFT(:,i));
     Pha = unwrap(angle(FFT(:,i)));
     Re = real(FFT(:,i));
     Im = imag(FFT(:,i));
end
```

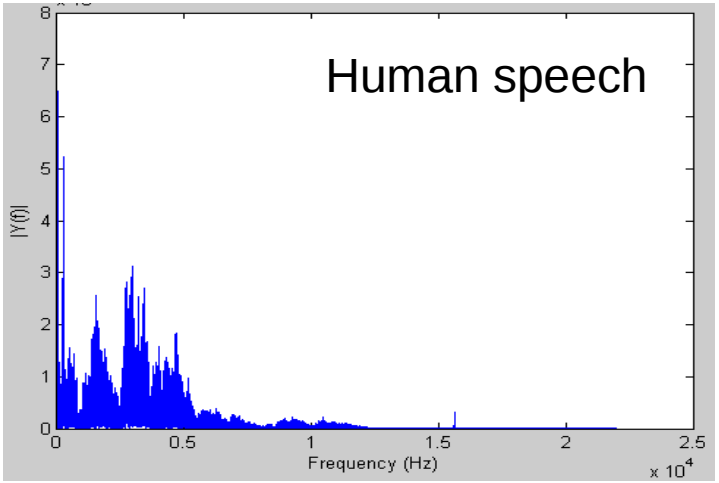



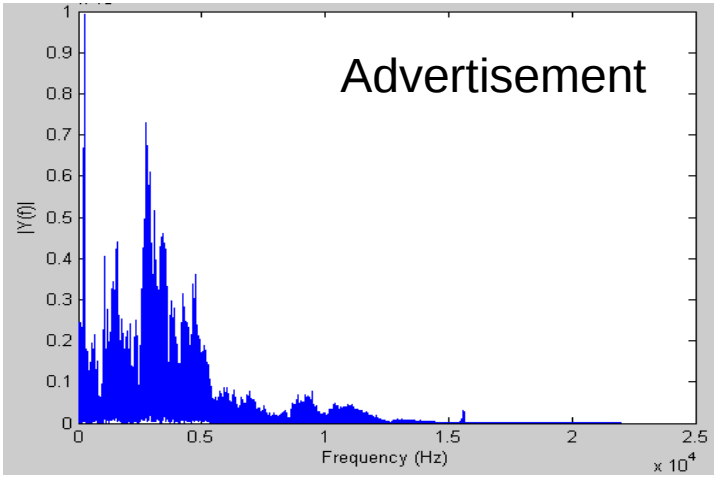
Feature selection and extraction: Mean FFT

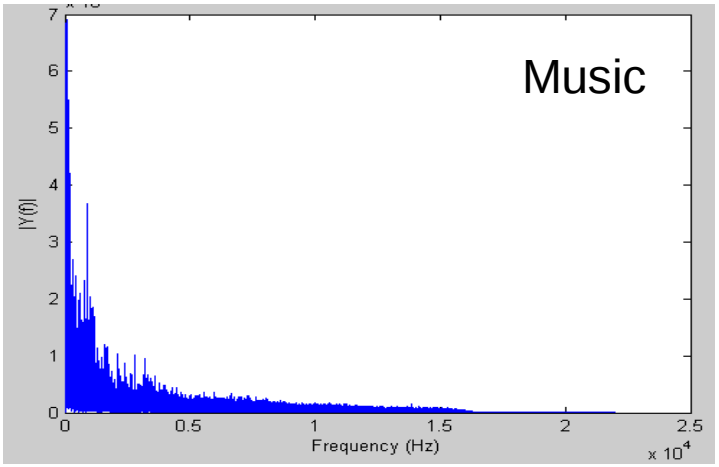
Mean FFT for each class was calculated and plotted. It was necessary to check whether any filtering is required.

```
[n_of_samples, n_of_signals] = size(FFT);              %get all required sizes
size_of_class = n_of_signals / number_of_classes;
meanFFT = zeros(n_of_samples, number_of_classes);      %prepare outoput matrix

for j = 1 : number_of_classes                               %for each class
    for i = (j-1)*size_of_class + 1 : j*size_of_class       %compute mean FFT
       meanFFT(:,j) = meanFFT(:,j) + FFT(:,i);
    end
    meanFFT(:,j) = meanFFT(:,j) / size_of_class;
end
```

Feature selection and extraction: Mean FFT



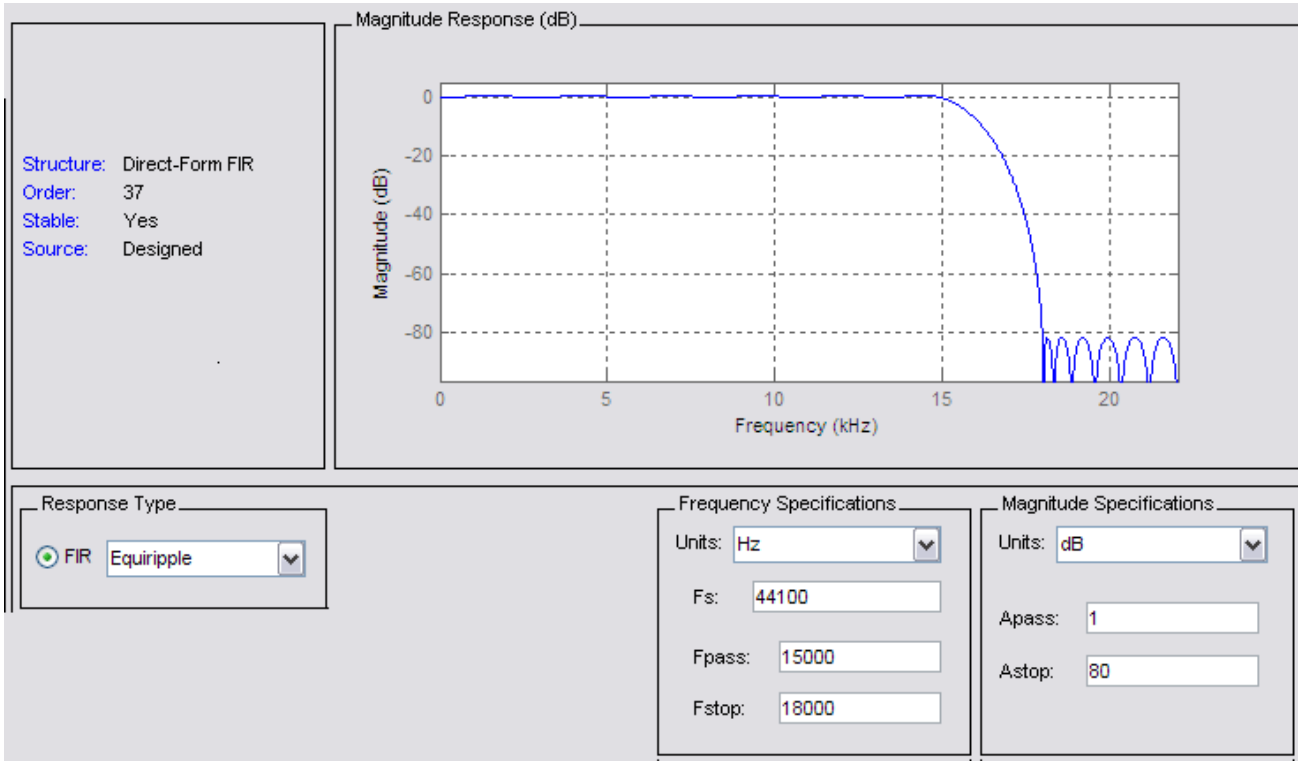




Charts present mean values of FFT for each class.

Feature selection and extraction: Filtering

Each sound sample was filtered to remove the highest frequencies, to reduce amount of data to process. In sound data, most important information is included in lower part of the spectrum.



Feature selection and extraction: Filtering

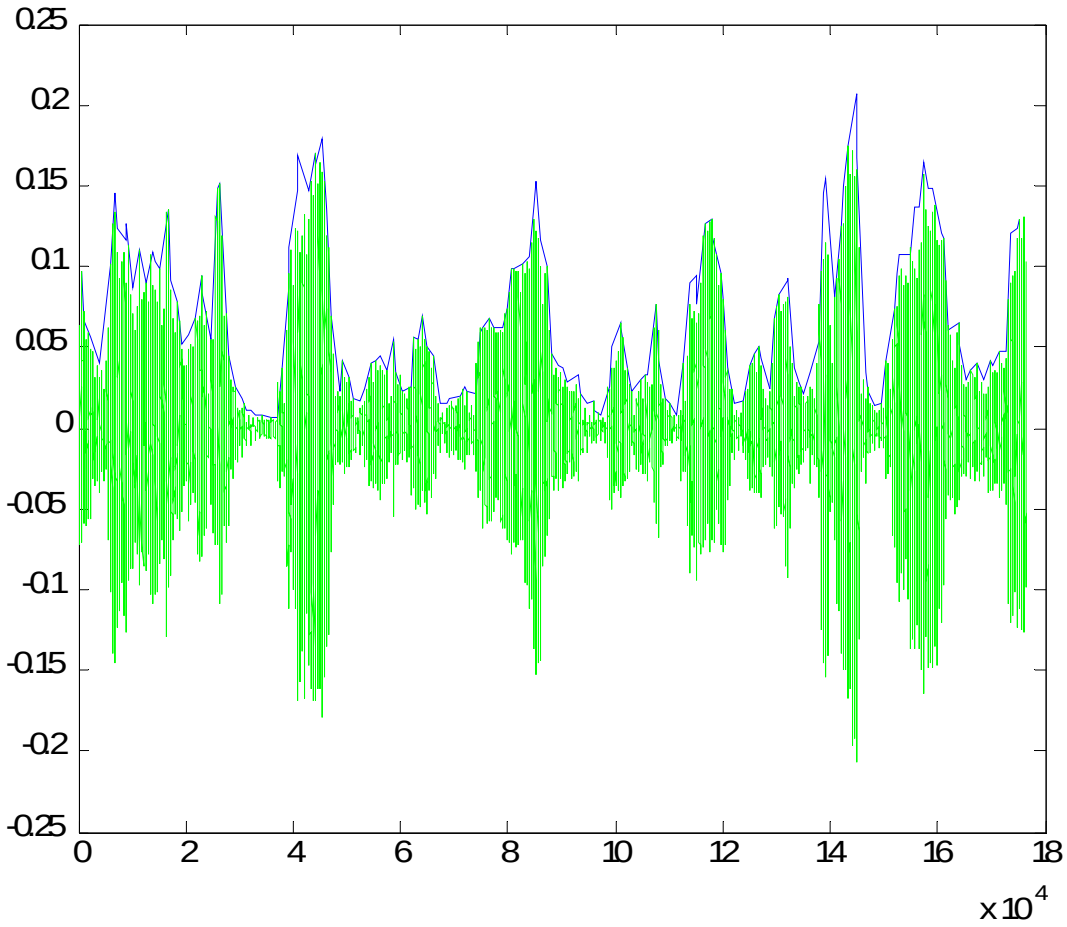
Low pass filter parameters:

- Type: FIR
- Order: 27
- f_pass: 1,5 kHz
- f_stop: 1,8 kHz
- a_pass: 1 dB
- a_stop: 80 dB

Feature selection and extraction: Envelopes (in time domain)

Example of human speech envelope:

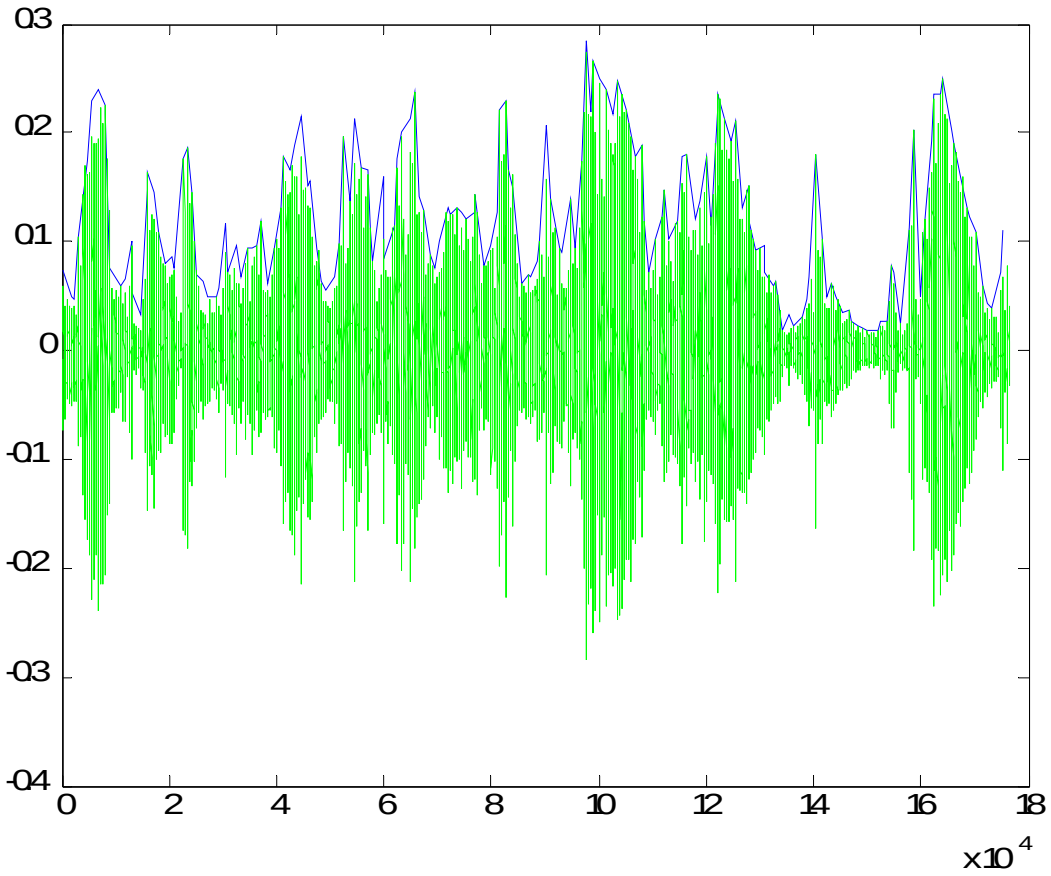
High fluctuations of envelope.



Feature selection and extraction: Envelopes (in time domain)

Example of advertisement envelope:

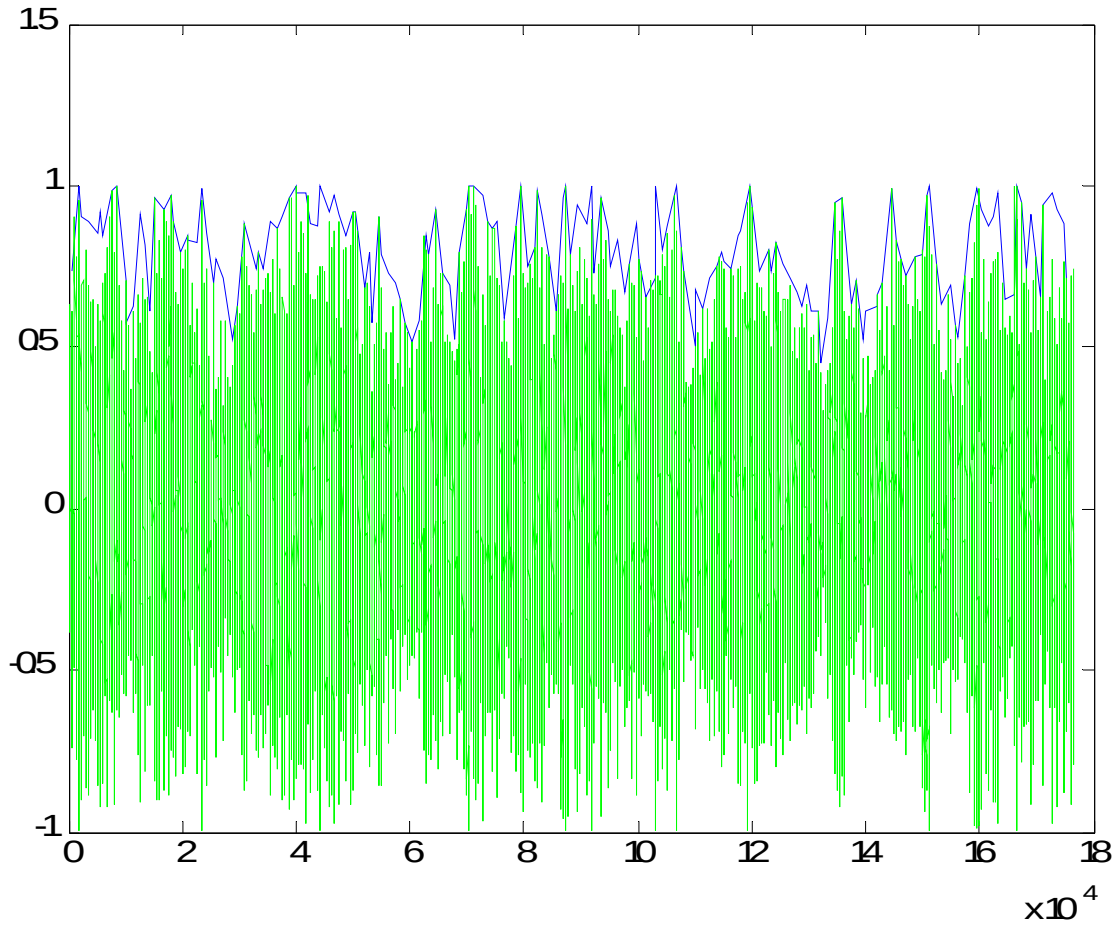
Weaker fluctuations than for human speech



Feature selection and extraction: Envelopes (in time domain)

Example of music envelope:

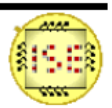
The weakest fluctuations



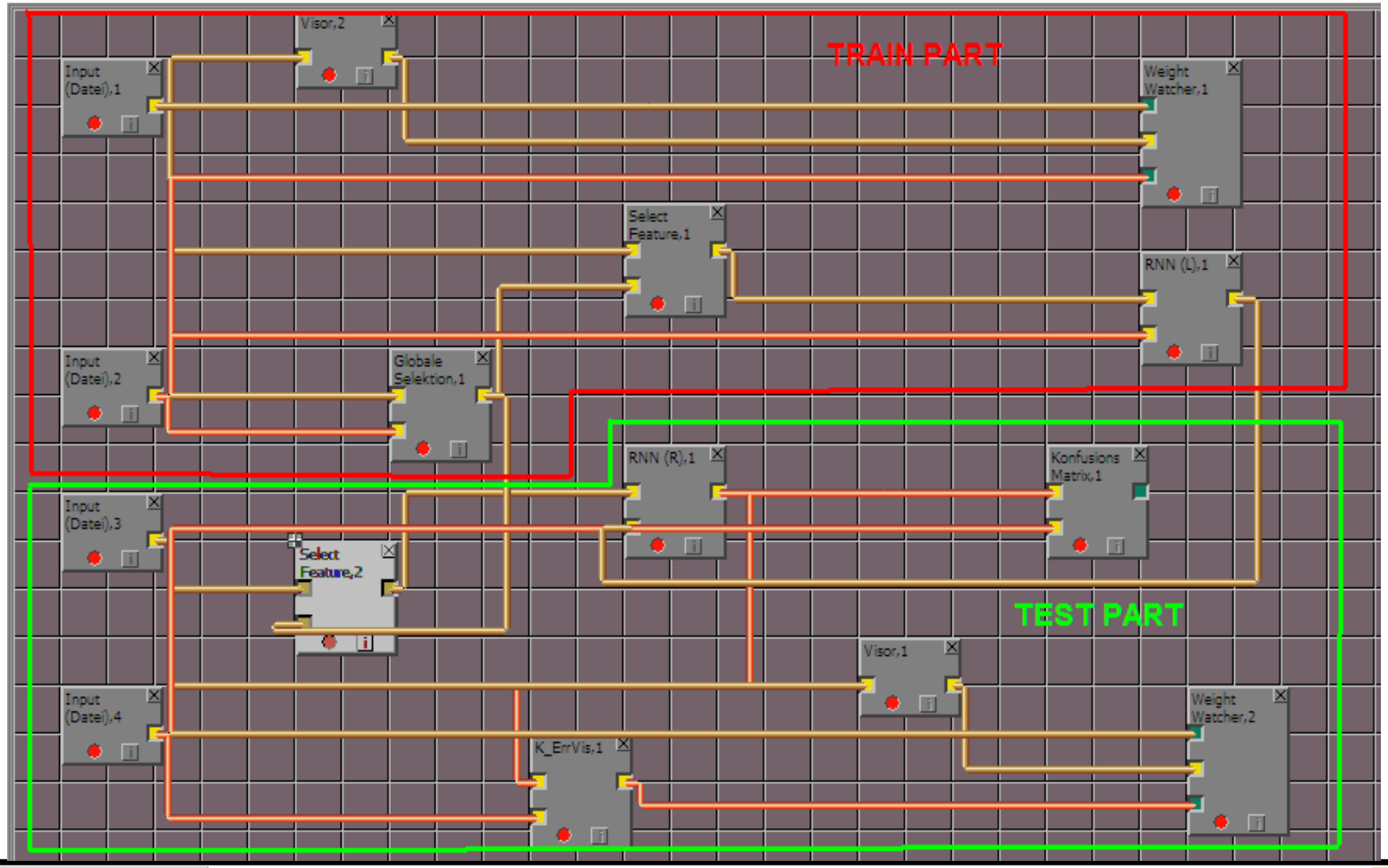
Feature selection and extraction: Envelopes (in time domain)

Envelopes function code:

```
for i=1:n_of_signals
    Xnew = abs(data(:,i));                          %absolute value of data
    lenXnew = length(Xnew);                         %get dimension

                                                    %compute envelopes
    for j = 1:fix(lenXnew/EnvWindow)
        [envelop(j,i),id] = max(Xnew((1+(j-1)*EnvWindow):j*EnvWindow));
        envTimeAxis(j,i) = id + (j-1)*EnvWindow;
    end
end
```

Classificiation
QuickCog structure - RNN

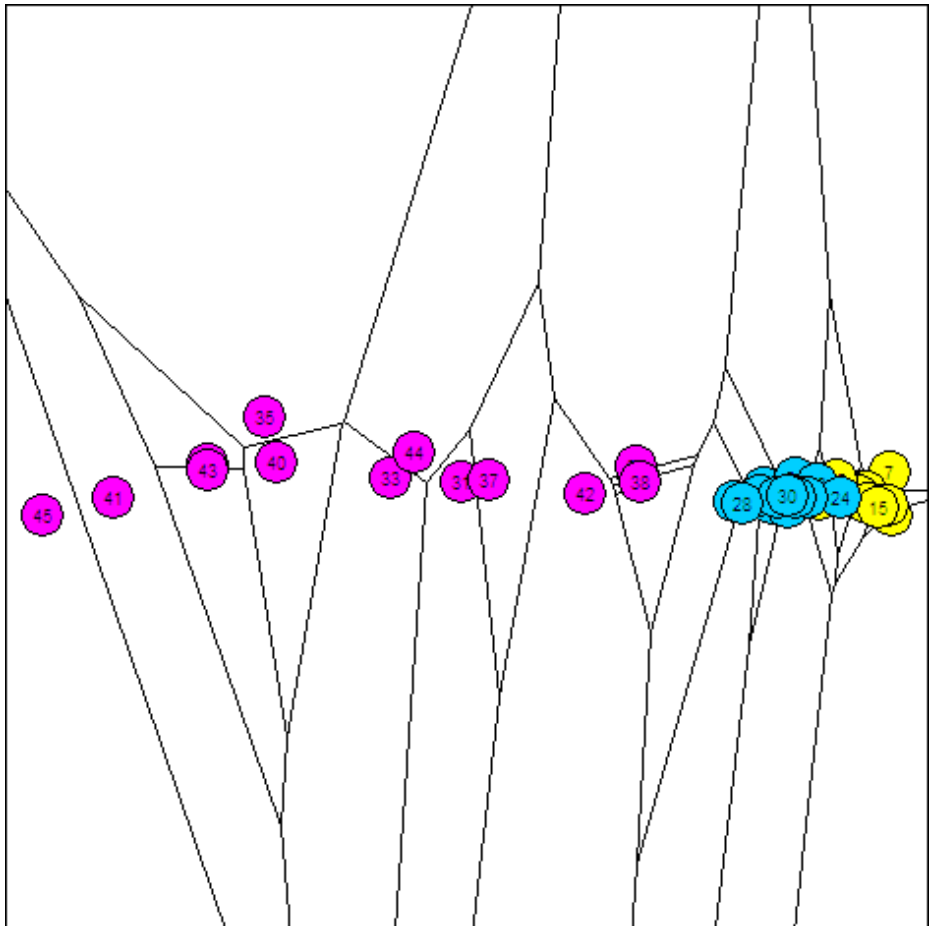


Classificiation
QuickCog results

- Overlaping – 0.94
- Feature set quality – 0.933

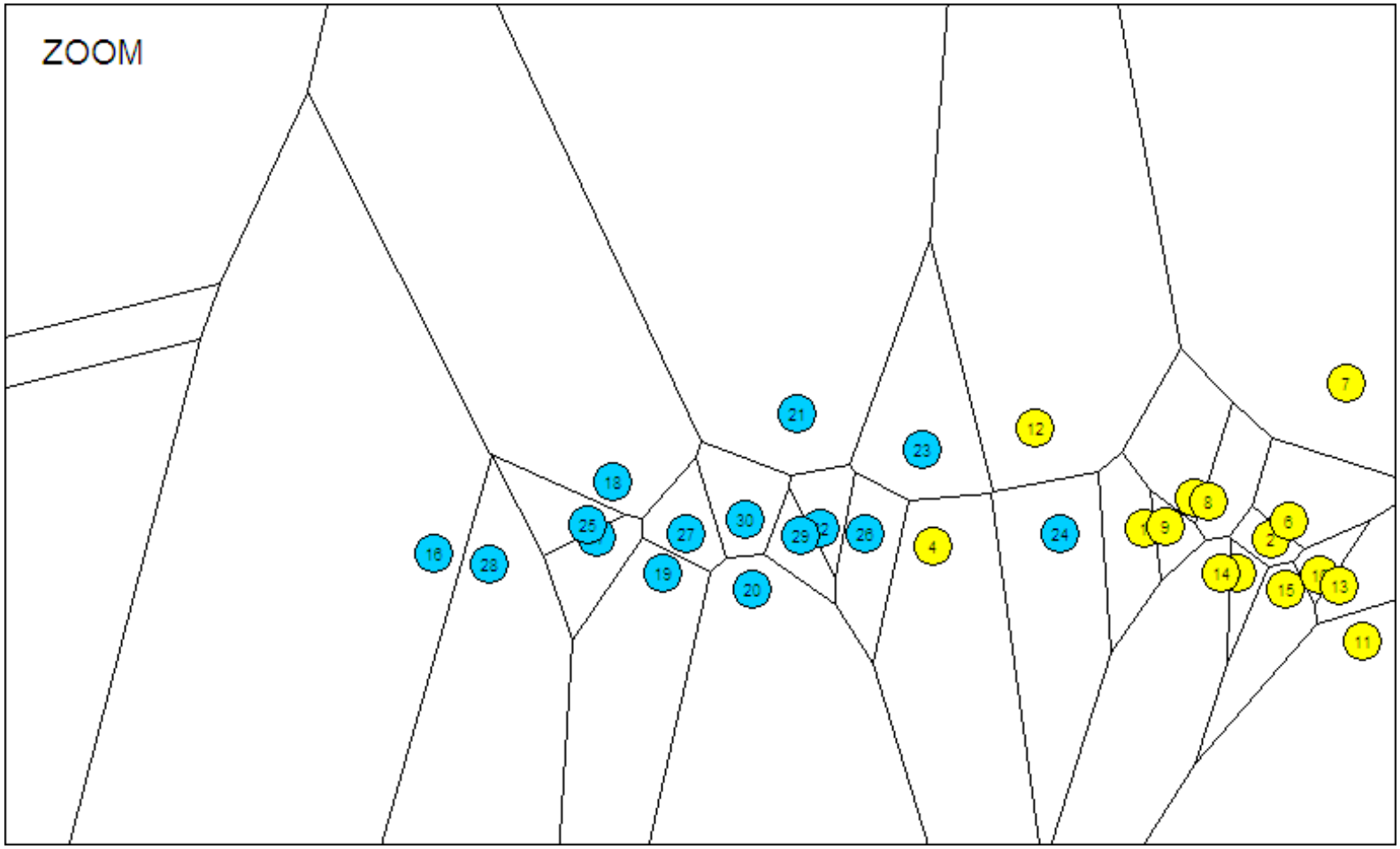
- Very short computation time (decreased amount of data)
- Very good quality of recognition

Classificiation
QuickCog results



Yellow – human speech
Blue – advertisement
Pink – music

Classificiation
QuickCog results



Results

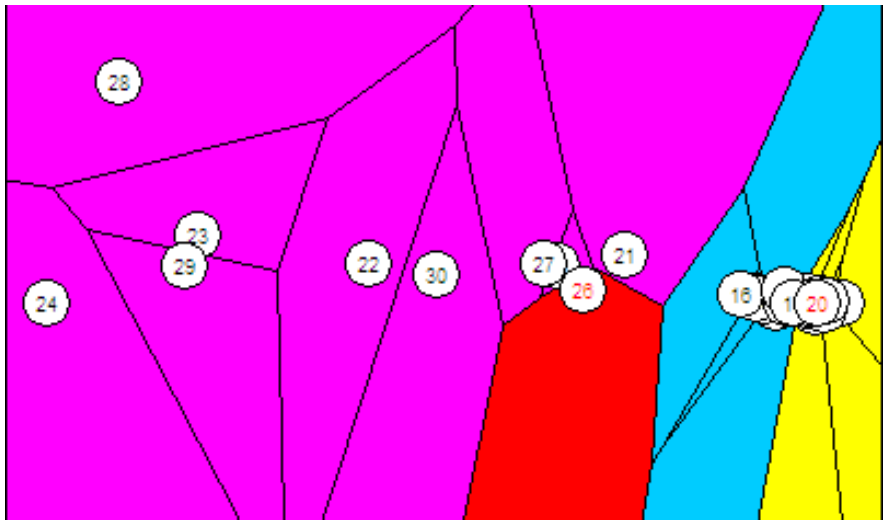
```
Search Strategy: Sequential Backward Selection
Quality Measure: qsi
```

Red area contains sounds which were not correctly recognized.

Yellow – human speech;
Blue – advertisement;
Pink – music.

Klasse1 – human speech;
Klasse2 – advertisement;
Klasse3 – music.

Test vector contained set of 30 sounds – 10 per class.




```
Best Quality: 1.00000
Best Features: 97  107  122
Konfusions Matrix,1
Klassifikationsresultate:
Klasse1 (10): 0      (R)    100    0      0
Klasse2 (10): 0      (R)    20     80     0
Klasse3 (10): 0      (R)    0      10     90
Erkennungsrate: 90.000 %
```

Conclusion

Envelope methode is the most efficient according to the computation time, data quantity and quality of solution.

For given test vector, our system recognized 90% of the cases. Results are satisfying.

Thank you for your attention!