Project 0: Part 1
A first hands-on lab on Speech Processing
Time-domain processing

February 12, 2020

In this lab, you will get acquainted with speech signals and their short-time processing. You will explore the time domain structure of the most basic speech elements, such as vowels and consonants. You will learn about time domain properties of speech signals and you will apply basic metrics on speech, such as energy and zero-crossings.

The final goal of this lab is to implement a simple, fully-automated *voiced/unvoiced/silence (VUS) discriminator*. Such an algorithm is very practical in real-life systems such as mobile communications systems. A typical Voice Activity Detector (VAD), which is a subset of a VUS discriminator, is used in the Global System for Mobile Communications (GSM), the European system for cellular communications. The algorithm you will implement is based on energy and zero-crossing measures of the speech signal - such computations are fast enough for real-time implementations. However, in this lab, we will consider an off-line approach. This means that we have the whole signal available from the first place.

1 Theoretical Background

The algorithm is based on energy and zero-crossings measures of the speech waveform. Let's take a short introduction on these subjects.

1.1 Short-Time Energy

We have observed that the amplitude of speech signals varies appreciably with time. In particular, the amplitude of unvoiced segments is generally much lower than the one of voiced segments. The short-time energy of the speech signal provides a convenient representation that reflects these amplitude variations. In general, we can define the short-time energy as

$$E_n = \sum_{k=-\infty}^{\infty} x^2[k] w^2[n-k] \tag{1}$$

This expression can be written as

$$E_n = \sum_{k=-\infty}^{\infty} x^2[k] h[n-k] \tag{2}$$

where $h[n] = w^2[n]$ is the squared analysis window applied on a speech segment. We can safely assume that the analysis window is supported in $[-N, N]$. The choice of the impulse response, $h[n]$, or equivalently the analysis window, determines the nature of the short-time energy representation. To

see how the choice of window affects the short-time energy, let us observe that if $h[n]$ in the equation above was very long, and of constant amplitude, E_n would change very little with time. Such a window would be the equivalent of a very narrowband lowpass filter. Clearly what is desired is some lowpass filtering but not so much that the output is constant; i.e., we want the short-time energy to reflect the amplitude variations of the speech signal. Thus, we encounter for the first time a conflict that will repeatedly arise in the study of short-time representations of speech signals. That is, we wish to have a short duration window (impulse response) to be responsive to rapid amplitude changes, but a window that is too short will not provide sufficient averaging to produce a smooth energy function.

The effect of the window on the time-dependent energy representation can be illustrated by discussing the properties of two representative windows, i.e., the rectangular window

$$h[n] = \begin{cases} 1, & \text{for } 0 \leq n \leq N-1 \\ 0, & \text{otherwise} \end{cases} \tag{3}$$

and the Hamming window

$$h[n] = \begin{cases} 0.54 - 0.46\cos\left(\frac{2\pi n}{N-1}\right), & \text{for } 0 \leq n \leq N-1 \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where N is the window length in samples. The rectangular window corresponds to applying equal weight to all the samples in the interval $(n - N + 1)$ to n, whereas the Hamming window gives more weight to the center of the window, which is preferable in many applications. If the window size, N. is too small, i.e., on the order of a pitch period or less, E_n will fluctuate very rapidly depending on exact details of the waveform. If N is too large, i.e., on the order of several pitch periods $(3 - 4)$, E_n will change very slowly and thus will not adequately reflect the time-varying properties of the speech signal. Unfortunately, this implies that no single value of N is entirely satisfactory. With these shortcomings in mind, a suitable practical choice for N is on the order of $300 - 500$ samples for a 16 kHz sampling rate (i.e., $20 - 30$ ms duration).

1.2 Short-Time Zero Crossings

In the context of discrete-time signals, a zero-crossing is said to occur if a sample has different algebraic sign from the previous (or the following) one. The rate at which zero crossings occur is a simple measure of the frequent content of a signal. This is particularly true for narrowband signals. For example, a sinusoidal signal of frequency f_0 Hz, sampled at a rate of F_s, has F_s/f_0 samples per cycle of the sine wave. Each cycle has two zero crossings so that the long-time average rate of zero-crossings is

$$Z = \frac{2f_0}{F_s} \text{ crossings per sample} \tag{5}$$

Thus, the average zero-crossings rate gives a reasonable and simple way to estimate the frequency of a sine wave.

Speech signals are broadband signals and the interpretation of average zero-crossing rate is therefore much less precise. However, rough estimates of spectral properties can be obtained using a representation based on the short-time average zero-crossing rate. Before discussing the interpretation of zero-crossing rate for speech, let us first define and discuss the theory behind. An appropriate definition is

$$Z_n = \sum_{m=-\infty}^{\infty} |sgn(x[m]) - sgn(x[m-1])|w[n-m] \tag{6}$$

where

$$sgn(x[n]) = \begin{cases} 1, & \text{for } x[n] \geq 0 \\ -1, & \text{otherwise} \end{cases} \tag{7}$$

and

$$w[n] = \begin{cases} \frac{1}{2N}, & \text{for } 0 \leq n \leq N-1 \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

This representation shows that the short-time average zero-crossing rate has the same general properties as the short-time energy. However, the definition of zero crossings equation make the computation of Z_n appear more complex than it really is. All that is required is to check samples in pairs to determine where the zero-crossings occur and then the average is computed over N consecutive samples (the division by N is obviously unnecessary as well).

Now let us see how the short-time average zero-crossing rate applies to speech signals. The model for speech production suggests that the energy of voiced speech is concentrated below 4 kHz, whereas for unvoiced speech, most of the energy is found at higher frequencies. Since high frequencies imply high zero-crossing rates, and low frequencies imply low zero-crossing rate, there is a strong correlation between zero-crossing rate and energy distribution over frequency. A reasonable generalization is that if the zero-crossing rate is high, the speech signal is unvoiced, while if the zero-crossing rate is low, the speech signal is voiced. This, however, is a very imprecise statement because we have not said what is high and what is low, and, of course, it really is not possible to be precise. Despite this imprecision, zero-crossing rate is definitely a simple and convenient measure for speech discrimination.

2 MATLAB point of view

2.1 Short-Time Energy

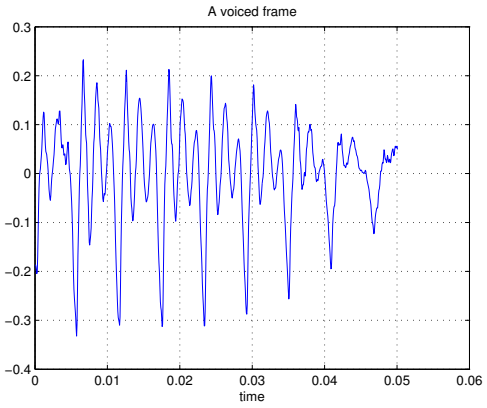
As you know, in discrete time the energy of a N-length signal is the sum of all squared values, divided by the number of samples. For example, let's extract a frame from the signal. The extracted frame is shown in Figure 1a.

```
% Let's load the speech signal first.
[s,fs] = wavread('H.22.16k.wav');
frame1 = s(3600:4400);
N1 = length(frame1);
figure; plot(0:1/fs:N1/fs-1/fs, frame1); grid;
title('A voiced frame'); xlabel('time');
% Listen if you want
% soundsc(frame1, fs);
```

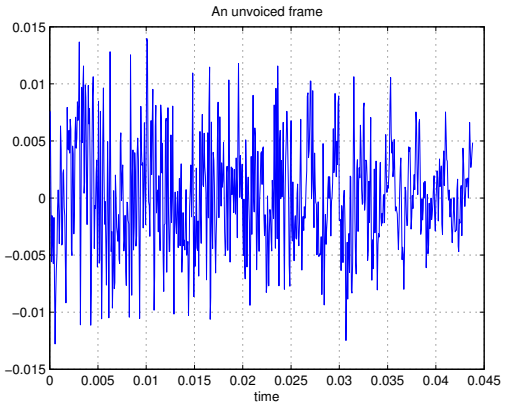
You can see that the signal has a periodicity of some form. It's not periodic, it's quasi-periodic. It has been observed that voiced speech can be represented using periodic or quasi-periodic waveforms. So, speech signals like this one, which have a strong periodicity of some kind, can be thought of as voiced speech signals, like /a/, /e/, /o/, etc. You can also see that it is a low frequency signal. It doesn't change very quickly over time. Let's see what happens with the energy of a signal like that one.

```
E1 = (1/N1)*sum(frame1.^2);
```

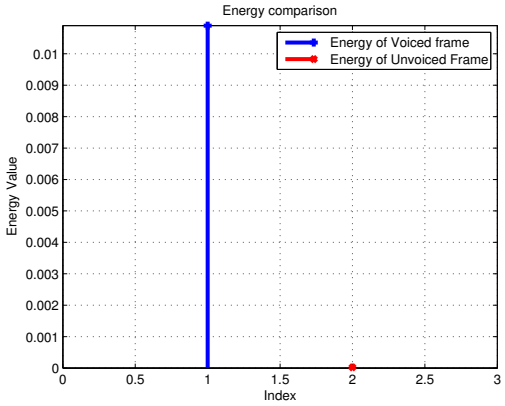
We'll hold this result, we'll need it later. Let's now cut another piece of speech from our waveform. Let's try this one.



(a) *A voiced part of speech.*



(b) *An unvoiced part of speech.*



(c) *Energy comparison between different parts of speech waveform.*

Figure 1: *Voiced and Unvoiced speech sounds, along with their energies.*

```
frame2 = s(4800:5500);
N2 = length(frame2);
figure; plot(0:1/fs:N2/fs-1/fs, frame2); grid;
title('An unvoiced frame'); xlabel('time');
% Listen if you want
%soundsc(frame2, fs);
```

You can see in Figure 1b that this one is very peaky! It has no periodicity at all, and it changes very quickly in time. The last one means that it is a high-frequency signal. Speech signals like that can be thought as unvoiced speech signals, like /s/, /f/, /sh/, etc. Let's take a look at the energy of this kind of speech signal, and compare it with the first one.

```
E2 = (1/N2)*sum(frame2.^2);
```

Let's compare them.

```
figure; stem(1, E1, '+', 'LineWidth', 3); grid; hold on;
stem(2, E2, 'rx', 'LineWidth', 3); hold off;
axis([0 3 0 max(E1,E2)]); title('Energy comparison');
xlabel('Index'); ylabel('Energy Value');
legend('Energy of Voiced frame', 'Energy of Unvoiced Frame');
```

You can see in Figure 1c that the energy of the voiced speech signal is really higher than the unvoiced one. If you do this in several other voiced/unvoiced examples, you will see that our observation holds in general. If we do the same for a full speech waveform, what we will get is depicted in Figure 2.

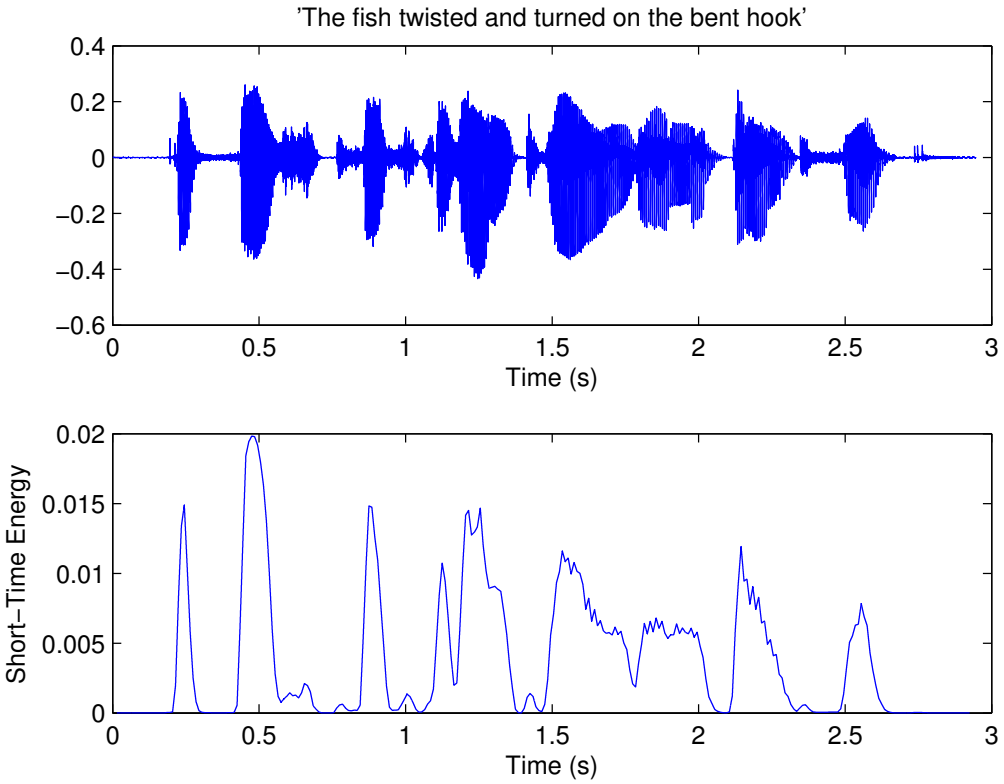


Figure 2: *Energy distribution for full speech.*

You can clearly see that voiced parts of speech have higher energy than unvoiced or silent ones. So, the energy of a speech frame is a good indicator of whether a frame is a voiced or unvoiced one.

2.2 Zero crossings

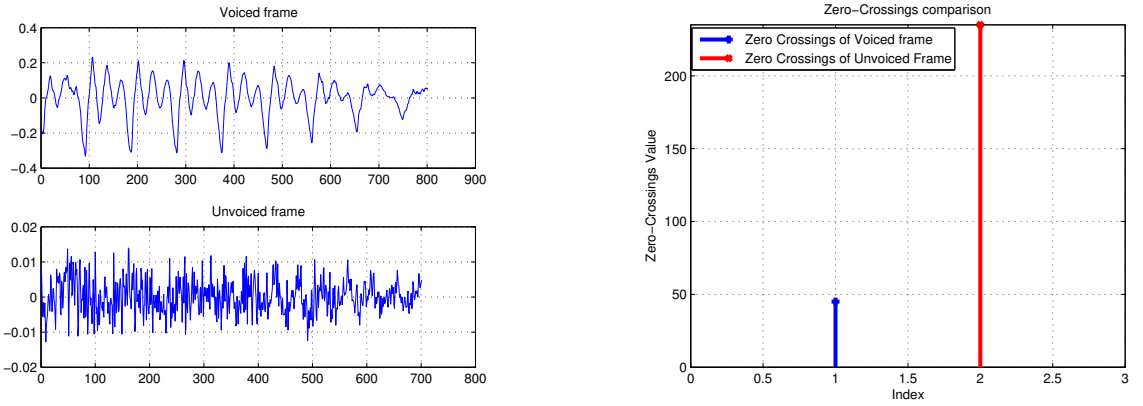
Again, what is a zero crossing? A zero-crossing is a point where the sign of a function changes (e.g. from positive to negative), represented by a crossing of the axis (zero value) in the graph of the function.

How can we exploit the number of zero crossings in a speech frame to make the discrimination between voiced and unvoiced speech? Let's take a look again into our two extracted speech frames:

```
figure; subplot(2,1,1); plot(frame1); grid; title('Voiced frame');
subplot(2,1,2); plot(frame2); grid; title('Unvoiced frame');
```

Take a look at Figure 3a. Can you guess what is going on with the zero-crossings? :-) You can see that the number of zero-crossings are really high in the unvoiced speech frame, much higher than the voiced speech frame! So, this can be our second indicator of whether a speech frame is voiced or unvoiced. There is a straightforward way to find out how many zero crossing are there in a speech frame. Let's see it:

```
ZCr1 = 0.5*sum(abs(sign(frame1(2:end))-sign(frame1(1:end-1))));
ZCr2 = 0.5*sum(abs(sign(frame2(2:end))-sign(frame2(1:end-1))));
```



(a) *Our two parts of speech.* (b) *Zero crossings comparison.*

Figure 3: *Voiced and Unvoiced speech sounds, along with their zero-crossings.*

The above equation counts the number of zero-crossings in a speech frame. The first term returns the sign of each sample value, starting from sample 2, and the second term returns the sign of each sample value, starting from sample 1 and ending a sample from the last one. The subtraction of these two gives us a value different than zero (+1 or -1) when we have a change of the function's sign. So, the addition of all these non-zero values will give us the number of zero-crosssings, but doubled. So, we need to keep half of them. That's why we multiply the number by 0.5. Try this in paper to see why it is valid. A **for** loop and an **if** statement would do the same thing, but we prefer not using loops in MATLAB, especially when the loops are too many, because all this procedure is not so time-efficient.

Let's take a look at the zero-crossings results:

```
figure; stem(1, ZCr1, '+', 'LineWidth', 3); grid; hold on;
stem(2, ZCr2, 'rx', 'LineWidth', 3); hold off;
axis([0 3 0 max(ZCr1, ZCr2)]); title('Zero-Crossings comparison');
xlabel('Index'); ylabel('Zero-Crossings Value');
legend('Zero Crossings of Voiced frame', 'Zero Crossings of Unvoiced Frame');
```

You can see the result in Figure 3b. We can generalize for a full waveform, and what we get is shown in Figure 4. You can see that the number of zero-crossings is quite high in unvoiced parts of speech, whereas is low in voiced parts. So, zero-crossings is a quite convenient way to discriminate voiced from unvoiced speech.

2.3 Limitations

A combination of the methods we have described seems reasonable and powerful enough for our purpose. And indeed, it is, up to a certain level. :-) However, there are some problems in our simple approach. Some frames are transient frames (something between voiced and unvoiced), which cannot be easily detected and categorized. Some others are neither purely voiced, nor purely unvoiced, such as fricatives, plosives, or nasals (/p/, /g/, /d/, /b/, /m/, /n/, etc). Also, and more importantly, there might be some voiced frames with low energy and some unvoiced with high energy. This depends on the speaker and the speaking style. Even the zero crossing rate may differ among different speakers or

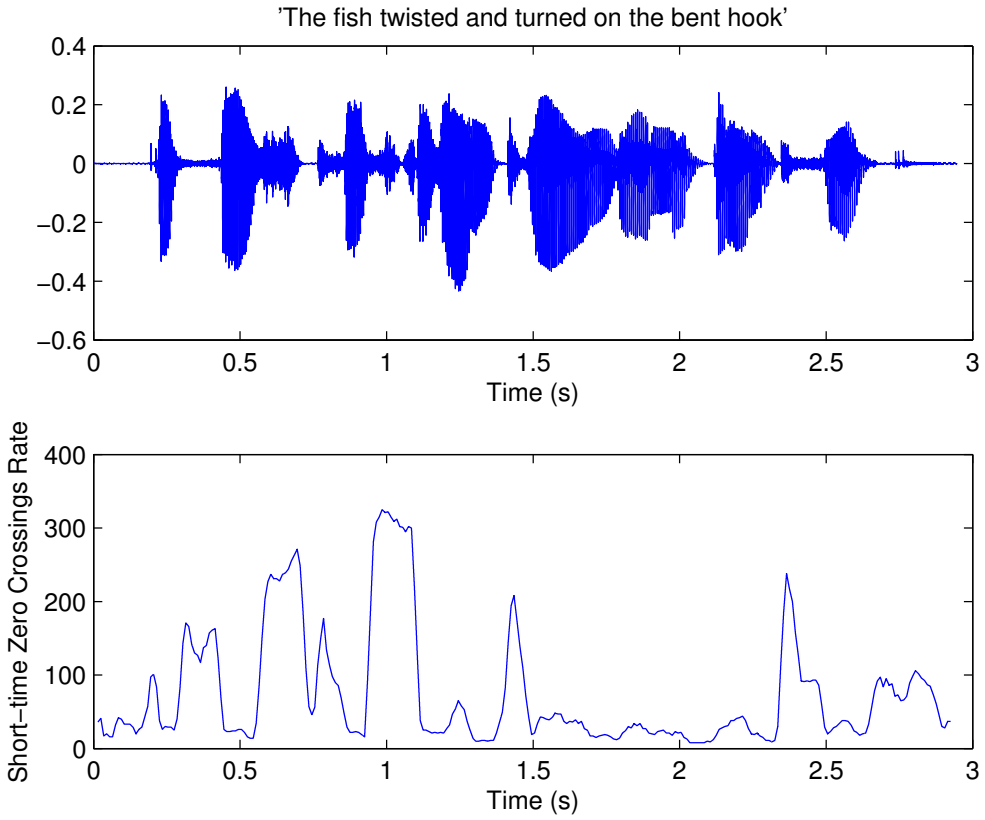


Figure 4: *Zero-crossings distribution in full speech.*

speaking style. So, there are limitations to our apporach, as you may have guessed. However, it is not quite far from the one used in real systems, like the GSM standard of the European Union Cellular Communications Systems, and the purpose of this lab is not to build a robust VUS discriminator, but to use simple tools and familiarize yourselves with time-domain properties of speech.

3 VUS discriminator implementation issues

Now that you are familiar to the basic analysis tools, you can start building your VUS discriminator. Most of the code is given to you at the end of this PDF. Please consider the following for your implementation:

- What about silence? We haven't mentioned anything about it in our analysis. Well, ideally, a silence frame would be a frame with all samples equal to zero, right? This means energy equal to zero and no zero-crossings at all. But in practice, because of microphone noise or noise from the environment (breath, room reflections etc.), silence frames have some kind of small sample variation. This variation is so small that the energy should be less than both voiced and unvoiced frames, and the zero-crossings should be also less than the other two cases. Usually, this is what happens in the real world. :-)

- As we have mentioned at the beginning, our algorithm has an adaptivity of some form, which means that its results depend on some statistics of the input waveform. This adaptivity is presented in the thresholds we introduce, in order to make our discrimination into voiced/unvoiced/silence

frames. You will see that there are two thresholds, one for the energy and one for zero-crossings. These thresholds depend on the input waveform, so they are not fixed numbers.

- Your algorithm should work on a frame-by-frame basis. That means you have to estimate your energy and zero-crossings in a set of *analysis time instants* of your choice, and with a step size of your choice. However, you should pay attention that your analysis window should be long enough for your statistics to be meaningful, but also short enough to have good time localization. :-) It is suggested that your analysis window and your frame rate should be $20 - 30$ ms and $5 - 10$ ms, respectively. Your energy and zero-crossings estimates are considered to be localized in the **center** of the analysis window.

- So, for each frame, you will have to calculate the above measures (energy and zero-crossings) and find out if the frame is voiced/unvoiced/silence.

- Because of its simplicity, this VUS discriminator should perform adequately but not perfectly (actually, a very accurate VUS is still a subject of research in speech community, although several robust and highly accurate VUSs have been proposed over the years). However, it should at least correctly detect the voiced parts of speech.

- Also, since your VUS results in estimates every $5 - 10$ ms, you will have to interpolate your results over the whole speech waveform, in order to have a continuous estimate (that means, for every time sample). To do this, you can use the **interp1** function that is already available for you in MATLAB. Try different interpolation methods, like *splines, cubic*, and *linear* interpolation schemes. You can visually inspect the results and check which method performs better. Justify your results and make a short comment.

- Moreover, you can try different analysis window sizes, different analysis window types, or different frame rates to see how the results change.

- Finally, in the speech examples that we provide, there are two speech files ending in *-sin.wav* and *-swn.wav*. These are two utterances that have the same context (the speakers says the same thing) but in a different speaking style. In the *-sin.wav* file, the speaking style is more "stressed", more "intense" in a way. This is called *Lombard* speech, because the speaker changes his/her speaking style in order to produce speech that is more intelligible in a noisy environment. It is like speaking in a very quiet place (*-swn.wav* file) and in a cafeteria, an airport, or in any other crowded place (*-sin.wav*). Can you see any significant energy or zero-crossing rate changes between these two waveforms? If yes, comment on your results.

A sample code is given below:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%% SAMPLE CODE %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Reading the speech signal
[s, fs] = wavread('sample.wav');

% Remove mean value (DC component)
s = s - mean(s);

% Signal length
D = length(s);
```

```
% Frame length (30 ms, how many samples? )
L = %INSERT CODE HERE

% Frame shift (10 ms, how many samples? )
U = %INSERT CODE HERE

% Window type (Hamming)
win = hamming(L);

% Number of frames
Nfr = %INSERT CODE HERE

% Memory allocation (for speed)
energy = zeros(1, Nfr+1);
ZCr = zeros(1, Nfr+1);

% Loop which calculates the speech features
for i = 1:1:(Nfr+1)
    frame = %INSERT CODE HERE      % a frame of speech windowed by the Hamming window
    energy(i) = %INSERT CODE HERE  % calculate energy
    ZCr(i) = %INSERT CODE HERE     % calculate zero crossings
    T(i) = L/2 + (i-1)*U;         % Next analysis time instant
end

% THRESHOLDS (you can play with it!)
Ethres = mean(energy)/2;
ZCRthres = (3/2)*mean(ZCr) - 0.3*std(ZCr);

% Clssification
for i = 1:1:Nfr
    if % INSERT CONDITION HERE
        % VOICED
        VUS(i) = 1.0;
    elseif % INSERT CONDITION HERE
        % SILENCE
        VUS(i) = 0.0;
    elseif % INSERT CONDITION HERE
        % UNVOICED
        VUS(i) = 0.5;
    end
end

% Interpolation with interp1
VUSi = INSERT CODE HERE

% Visualize
figure;
t = 0:1/fs:length(s)/fs-1/fs;
plot(t, VUSi);
```

```
hold on; plot(t, s/max(s), 'r'); hold off;
xlabel('Time (s)');
title('Energy & Zero-Crossings Rate-based VUS discrimination');
grid;
```

If everything went well, you should see something like Figure 5 below. You can see that it is not perfect but it works. :-)

You can try your own .wav file or you can use the ones we provide.

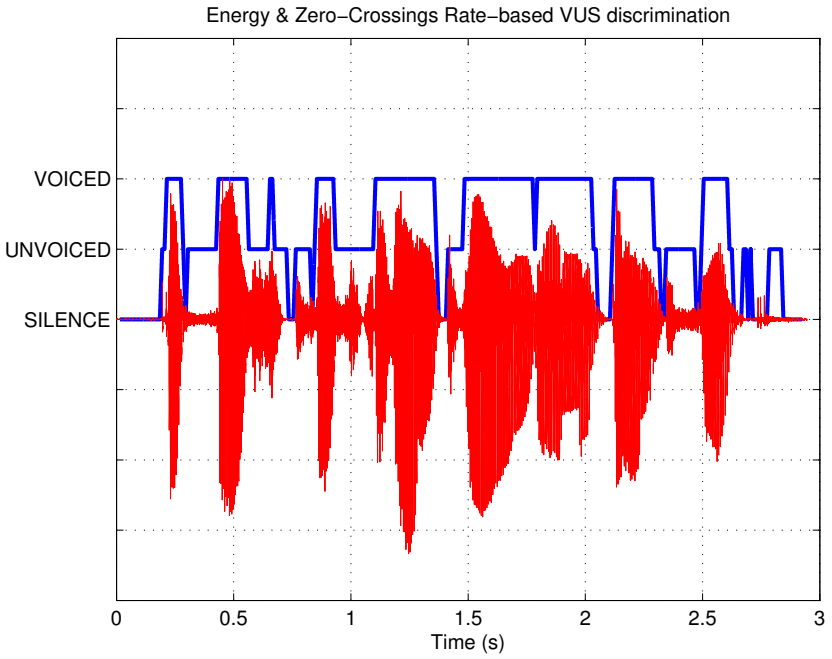


Figure 5: *A VUS discrimination using linear interpolation.*

4 Minimum deliverables in your report

- Complete (where indicated by %INSERT CODE) and deliver the MATLAB code provided. Standard parameters are 30 ms for each (Hamming) analysis window, and 10 ms for the frame rate. For the interpolation of your results, select the *linear* interpolation as a parameter to *interp1.m* MATLAB function. Include the figure that is generated by the code, along with your code in your report.
- Comment on the results of your VUSD, in all provided speech waveforms. Does it work well? If not, where does it have problems? Comment and include figures for all waveforms in your report.
- What happens if you increase the frame rate to 20 or 30 ms? What happens if you decrease the frame rate down to 2.5 or 5 ms? What happens if you change the window size (make it shorter or larger)? Comment (figures are NOT necessary).

- Record your voice with a microphone and save it into a .wav file. Use a sampling frequency of $F_s = 16$ kHz for your recording and a 16-bit precision. Load it into MATLAB (use audioread.m) and apply your algorithm on the waveform. Include a figure of your voice and the corresponding result of the algorithm.
- **Delivery deadline: 24 February 2020**

If you have ANY questions on this lab, please send an e-mail to : hy578-list@csd.uoc.gr