1 Pitch detection

In this chapter will we describe a pitch detector or pitch extractor. The goal of a pitch detector is to find the fundamental frequency f_0 of a speech signal. There are several different techniques for finding this fundamental frequency. In this chapter a pitch detector will be chosen, based on test of some different pitch detections techniques.

1.1 Introduction

A pitch detector is used to detect the fundamental frequency (f_0), called the pitch, in a signal. In the project "Bandwidth expansion of narrowband signals" a pitch detector would be useful to find f_0 of speech signals, due to the fact that telephone signals are limited in the frequency interval 300-3400 Hz. When extending the bandwidth of a signal from a sampling frequency of 8000Hz to 16000Hz, it would also be useful to create the lower frequencies of a phone-signal, say 50-300 Hz. Because the human pitch lies in the interval 80-350 Hz(table 1.1.1 on page II), where the pitch for men is normally around 150 Hz, for women around 250 Hz and children even higher frequencies, the pitch is needed to construct this part of the speech signal.

Some of the most used detectors are; Energy based, Cepstral based, zero crossing, pitch in the difference function and autocorrelation based. In this chapter there will be tested three different pitch detectors, The "On-The-Fly", a Cepstral based and an auto correlation based.

"On-The-Fly" [1] is time based an algorithm which is developed to estimate the pitch of a signal, divided into frames. This paper is a description of this algorithm, how it's implemented and an overall test of its ability to estimate a pitch of a speech signal.

The Cepstral pitch detector is based on the log of the fourier transformation. This algorithm is only shortly described and tested in this paper.

The Auto correlation pitch detector [5] is described and tested in this chapter.

1.1.1 The test signal

The pitch detector must be able to estimate a pitch of signals in a limited frequency range. The speech signal used throughout this paper, is spoken by a man with a pitch around 150Hz $\pm$ 20Hz, tested with a pitch analyzer program[1]. If not mentioned otherwise figures illustrate a single frame (20ms) and always the same frame, so that the figures can be compared. The signal is sampled at 16000Hz and passed though a *telefilter*, which is a 10'the order band pass filter working at 300-3400Hz. All figures in this paper are illustrated with the original signal and the band passed signal. In this way it is possible to compare the difference between these.

	$f_0 min$(Hz)	$f_0 max$(Hz)
men	80	200
woman	150	350

Table 1.1: Range of human speech

Also notice that the implemented algorithms not only have been tested with the above described signal. This signal is only used to gain understanding of the algorithms.

1.2 On-The-Fly pitch estimation

1.2.1 Common time domain pitch detectors

Many time-domain pitch detectors use the autocorrelation (equation 1.1) method to estimate the pitch, which is also among the oldest methods. Another method to find the pitch is the difference function (equation 1.2), which has been shown by Yin [2] and Chowdhury [3] to be more effective than the autocorrelation method. The algorithm "On-The-Fly", is a pitch-detector is based on the difference function (d_t).

$$r_t(\tau) = \sum_{j=1}^{W-\tau} x_j x_{j+\tau} \tag{1.1}$$

$$d_t(\tau) = \sum_{j=1}^{W-\tau} (x_j - x_{j+r})^2 \tag{1.2}$$

, where W is the length of the window. t indicates this is a time-domain calculation.

1.2.2 Unifying the difference function

The difference function is an effective algorithm to detect the f_0 of a signal. Here f_0 should be at the global minimum of the difference function. This minimum is although difficult to

[1]Pitch analyzer v1.1 - written by Franz Josef Elmer

locate, because the difference function of a random signal decrease with a slope of a straight line, as shown on figure 1.1.

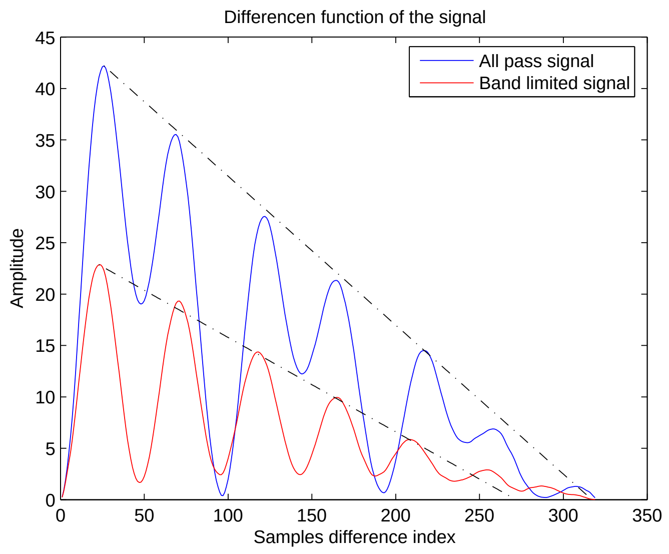

Figure 1.1: Shows the difference function taken of both the all-passed signal and the band limited signal. The figure shows that both signals approximately converge with a straight line. Also notice the difference between the local minima. The local minimas of the band limited signal are generally much smaller than the all pass signal.

To locate the actual global minima, the difference function output needs to be unified. So if the time delay τ is equal to the period of the pitch (P), the converging slope (see figure 1.1) can be seen as a straight line of a unity slope. Correlating x_{t+P} with x_t will result in a global maximum at the pitch P. Therefore by considering the covariance matrix of the random vector $X = [x_j \ x_t + \tau]^T$ we get that:

$$c_x(\tau) = E\left[(x-m)(x-m)^T\right] \tag{1.3}$$

Considering this as WSS, two orthogonal eigenvectors can be found from the covariance matrix, by decomposing it using diagonalization:

$$C_x(\tau) = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} \lambda_1(\tau) & 0 \\ 0 & \lambda_2(\tau) \end{bmatrix} \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} = \begin{bmatrix} \lambda_1(\tau)+\lambda_2(\tau) & \lambda_1(\tau)-\lambda_2(\tau) \\ \lambda_1(\tau)-\lambda_2(\tau) & \lambda_1(\tau)+\lambda_2(\tau) \end{bmatrix} \tag{1.4}$$

By assuming uniform distributions and equal means, the eigenvalues can be written like this:

$$\lambda_1(\tau) \propto \sum_{j=1}^{W-\tau} (x_j + x_{j+\tau})^2 \tag{1.5}$$

$$\lambda_2(\tau) \propto \sum_{j=1}^{W-\tau} (x_j - x_{j+\tau})^2 \tag{1.6}$$

Because the eigen-values and vectors describes new basis for the covariance matrix, a unified difference function can be derived. Thinking of eigenvector 1 and eigenvalue 1 being a 1D basis of the 2D space, the corresponding eigenvector 2 and eigenvalue 2 can be thought of as the error off this basis and vice versa. Therefore the unified difference function can be written as the following:

$$d_t'(\tau) = \frac{\lambda_2(\tau)}{\lambda_1(\tau)+\lambda_2(\tau)} = \frac{\sum_{j=1}^{W-\tau}(x_j - x_{j+r})^2}{2\cdot\sum_{j=1}^{W-\tau} x_j^2 + x_{j+r}^2} \tag{1.7}$$

The unified difference function is plotted figure 1.2.

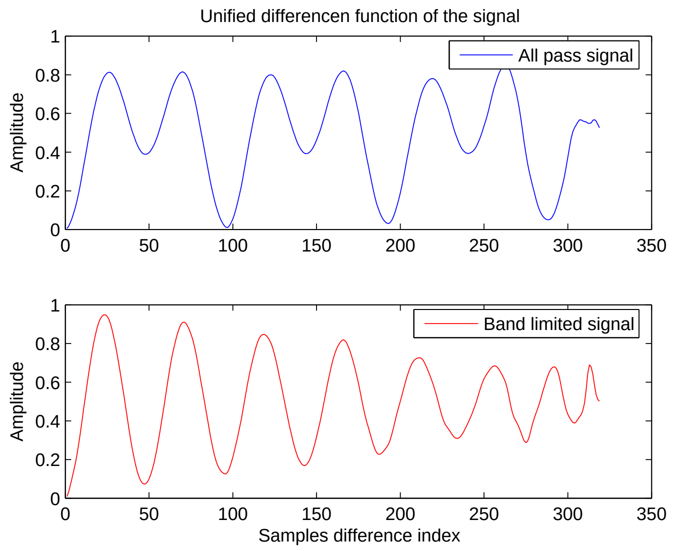

Figure 1.2: Here the two signals are unified. Now the difference min the local minima are really shown.

By Yin the the pitch is found at the global minima of the unified difference function ($d_t^{'}$). This is found as a sample index and is equal to P. f_0 is when calculated by the function:

$$f_0 = \frac{fs}{P} \tag{1.8}$$

, where fs is the sampling frequency (fs).

1.2.3 Contradiction in results

By taking further notice to the two figures in figure 1.2, we find a contradiction. The two signals doesn't seem to have the same global minimum, eventhough f_0 of the two signals should be equal. In the article [1] it does not say anything about "On-The-Fly" was developed for these types of band limited signals. But it is developed to find the actual pitch of a signal, due to the fact, that the authors' claim that the global minimum of the difference function is not always equal to the pitch period. This is actually the case for our band limited signal, where the situation is as follows:

- The all-pass signal has a global minimum around index 95, which is equal to a pitch of about 170 Hz.
- The band limited signal has a global minimum around index 47, which is equal to a pitch of about 340 Hz.

Surely one of these results must be wrong. Considering the test signal originates from a man, with a pitch around 150 Hz, the result of the band limited signal must be wrong.

1.2.4 Idea of the "On-The-Fly"

The fundamental idea of the "On-The-Fly" algorithm is to find other candidates for f_0. This is done by locating a number of the smallest minimas in $d_t^{'}$, where the overall global minimum, is still by offset selected to as f_0. Notice that at all time, the global minimum is still thought of as being some harmonic of the actual pitch. If the global minimum is not equal to f_0, the actual pitch should be found within some limits of this minimum, so it would be a candidate of a harmonic frequency to the global minima.

The surrounding local minima are therefore to be tested in 2 ways:

1. *A minimum differs only by a significant amount (threshold stage 1).*
   A small threshold (ATh1) is to be used in testing the difference in the minimas size. If the difference from the found pitch-sample is within the limit, the one with the smaller time period is considered the actual pitch. This has to be accompanied with another larger threshold (TTh1) which is used to test the cohesion between the candidate harmonic of the pitch minima and the new candidates. If the minimum does not lie with the threshold of the harmonic, this sample should be discarded.
2. *A minima differs only a relatively significant amount (threshold stage 2).*
   A larger threshold (ATh2) is to be used in testing the difference in the minimas size. If the difference from the found pitch-sample is within the limit, the one with the smaller time period is considered the actual pitch. This has to be accompanied with another smaller threshold (TTh2) which is used to test the cohesion between the candidate harmonic of the pitch minima and the new candidates. If the minimum does not lie with the threshold of the harmonic, this sample should be discarded.

Figure 1.3 is illustrating the 2 test conditions where the threshold stages are marked.

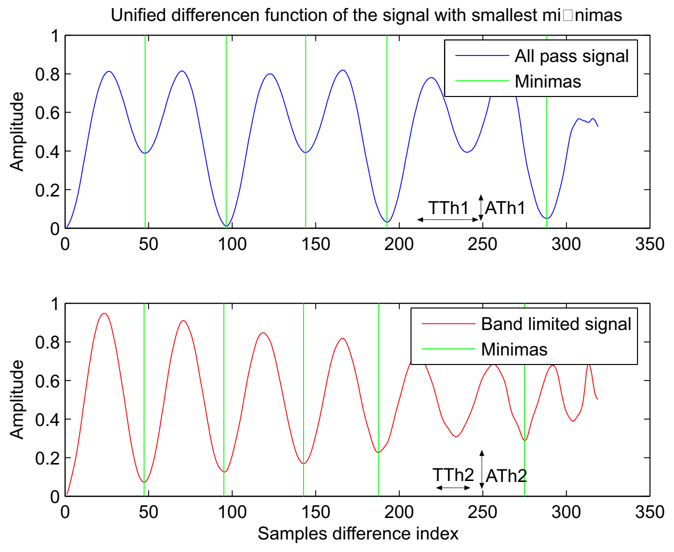

Figure 1.3: Same as figure 1.2, though here with the 5 smallest minimas marked with green lines. The threshold stage 1, ATh1 and TTh1, is shown in the upper figure, while threshold stage 2, ATh2 and TTh2, is shown in the lower figure. Of course both thresholds need to be applied to each signal when testing

1.3 The On-The-Fly algorithm and implemantation

The algorithm to On-The-Fly is relatively easy and is in figure 1.4 shown in a block diagram.

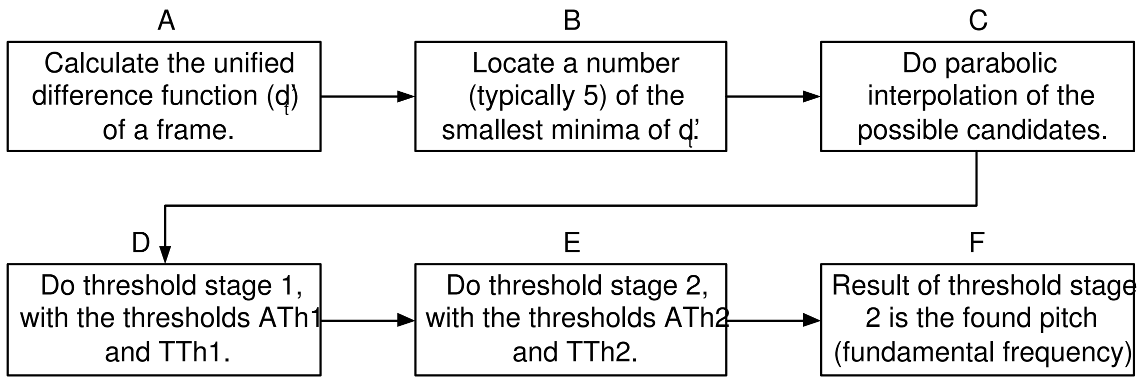
A
Calculate the unified difference function (d't) of a frame.
B
Locate a number (typically 5) of the smallest minima of d't.
C
Do parabolic interpolation of the possible candidates.
D
Do threshold stage 1, with the thresholds ATh1 and TTh1.
E
Do threshold stage 2, with the thresholds ATh2 and TTh2.
F
Result of threshold stage 2 is the found pitch (fundamental frequency)

Figure 1.4: The diagram of the On-The-Fly algorithm

A Here d'_t is calculated.

B Locating a number of the smallest local minima of $d_t^{'}$ can be a challenging task. First a minimum must be defined. But can 2 local minima be separated with 1 or only a few samples? If not, where should the limit be?

An example shows that surely minimas with only a few samples between them would not be harmonics of each other. Typical speech signals are samples at fs=16000 Hz. With the maximal pitch of 500 Hz, there would still have to be minimum (16000Hz/500Hz) 32 samples between minimas of the $d_t^{'}$. But still a local minimum might occur outside the harmonic and thereby eliminate the possibility of locating the actual harmonic to be found.

In the implementation, an algorithm to find a number of minimas of the $d_t^{'}$ was developed. It works in the following way; First it locates all minimas of $d_t^{'}$. Then any minima larger that 0.4 is deleted from data, to assure the minimas could actually be the pitch. Next the global minimum among all the minima is found and stored in t_i, where t_i is the index to the minima. If any minima exists within 10 samples of the found global minimum, these minimas are deleted from data along with the found minima, and the procedure is run again until there is no more data or the number of minimas wanted has been found.
The matlab code for this is found in section 1.8.2.

C The parabolic interpolation is needed to reduce quantization error in the candidate pitch values. An example, say fs=16000 Hz, if a minima is the pitch and located at sample 60, it equals a pitch of 267Hz. If the minima were found at sample 59, it would equal a pitch of 271Hz. No pitch is possible to get between 267-271Hz. Further this error gets higher, as the pitch rises. The parabolic interpolation solves this problem, and is implemented as a simple curve fitting problem.
The matlab code for this is found in section 1.8.3.

D Among the candidate minimas (t_i), we need to know if there are a harmonics of the global minimum is said to be the pitch. This is decided from the formula:

$$\left| N - \frac{t_g}{t_i} \right| < TTh1 \quad \text{, where N=}\{2,3,..,6\} \tag{1.9}$$

, where t_g is the index of the overall global minima. The threshold TTh1 here specifies how much a harmonic can maximal differs from the global minimum. Those t_i for which at least one of the possible N-values satisfy the threshold are then tested against threshold ATh1. ATh1 is a threshold of the difference in the minima value between t_g and t_i. If any t_i are within the 2 threshold values, t_g is changed with the one, which have the smaller timeperiod (if not t_g itself). In this threshold set, TTh1 is set to 0.2 and ATh1 is set to 0.07.
The matlab code for this is found in section 1.8.4.

E Bullet **D** and **E** are exactly the same, just with 2 different threshold sets. In this threshold set, TTh2 is set to 0.05 and ATh2 is set to 0.2.

The fully implemented "On-The-Fly" algorithm can be seen in section 1.8.2.

1.3.1 Testresults of "On-The-Fly"

Though this paper a single frame has been used to show how the algorithm works. In figure 1.5 we see the final result of the algorithm on this frame.

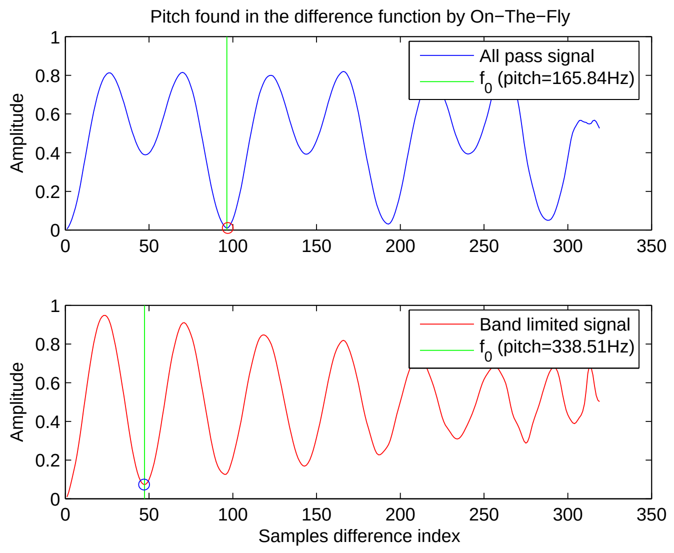

Figure 1.5: The pitch found in a single frame by the "On The Fly"-algorithm. The 'o' in each figure indicates where the global minima of $d_t^{'}$ (offset pitch) were found.

The figure clearly shows that the pitch is not the same in this frame. The pitch should be around 150Hz, which is the result from the original signal. From the band passed signal a pitch was not found anywhere near the actual pitch of the frame.
This could although be a insecurity in the algorithm. So in figure 1.6 150 frame of the signal are shown.

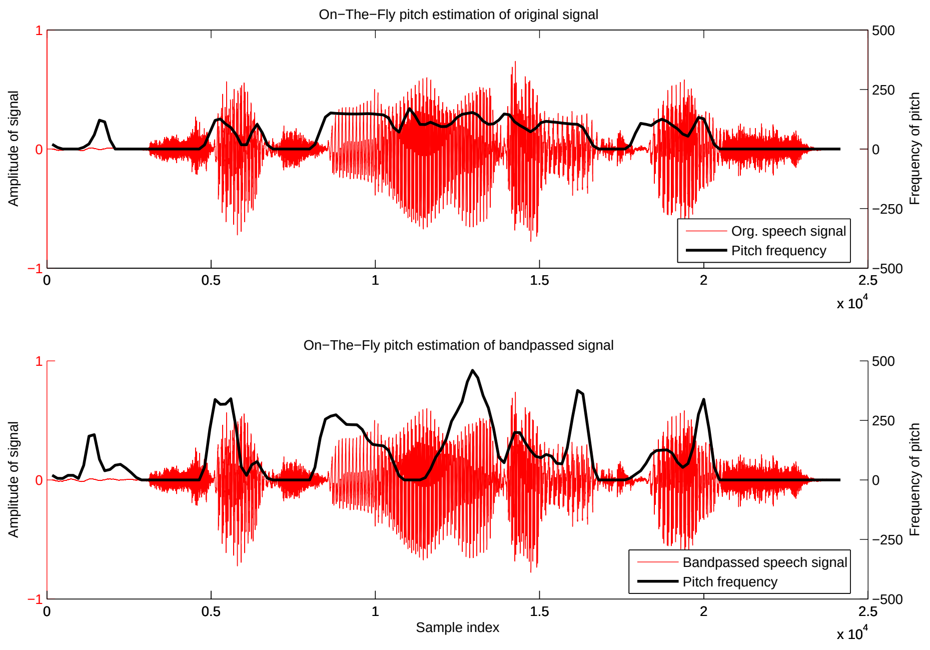

Figure 1.6: The pitch found by the "On The Fly"-algorithm.

In this figure we see that the found pitch from the original speech signal is far more consequente, than the one found in the band passed signal. From this and the fact that the signal is man with a pitch around 150Hz, we can conclude that "On The Fly" may be a good at detecting pitch in full frequencyranged signals, but not in band passed signals[2]. This algorithm is therefore not useful in the project "Bandwidth expansion of narrowband signals".

1.4 Cepstral based pitch detector

The theory behind the cepstral detector is that the fourier transform of a pitched signal usually have a number of regularly peaks, who is representing the harmonic spectrum. When log magnitude of a spectrum is taken, these peaks are reduced (their amplitude brought into a usable scale). The result is a periodic waveform in the frequency domain, where the period is related to the fundamental frequency of the original signal. This means that a fourier transformation of this waveform has a peak representing the fundamental frequency.
This method has only shortly been tested in matlab, with a non satisfying result. Due to the test, and time limitation, this pitch detector will not be explained further.

[2]A short description the algorithms flaws in detecting pitch in band limited signal can be found in section 1.8.1

1.5 Pitch detection using the autocorrelation

When calculating the autocorrelation you get a lot of information of the speech signal. One information is the pitch period (the fundamental frequency). To make the speech signal closely approximate a periodic impulse train we must use some kind of spectrum flattening. To do this we have chosen to use "Center clipping spectrum flattener". After the Center clipping the autocorrelation is calculated and the fundamental frequency is extracted. An overview of the system is shown in figure 1.7.

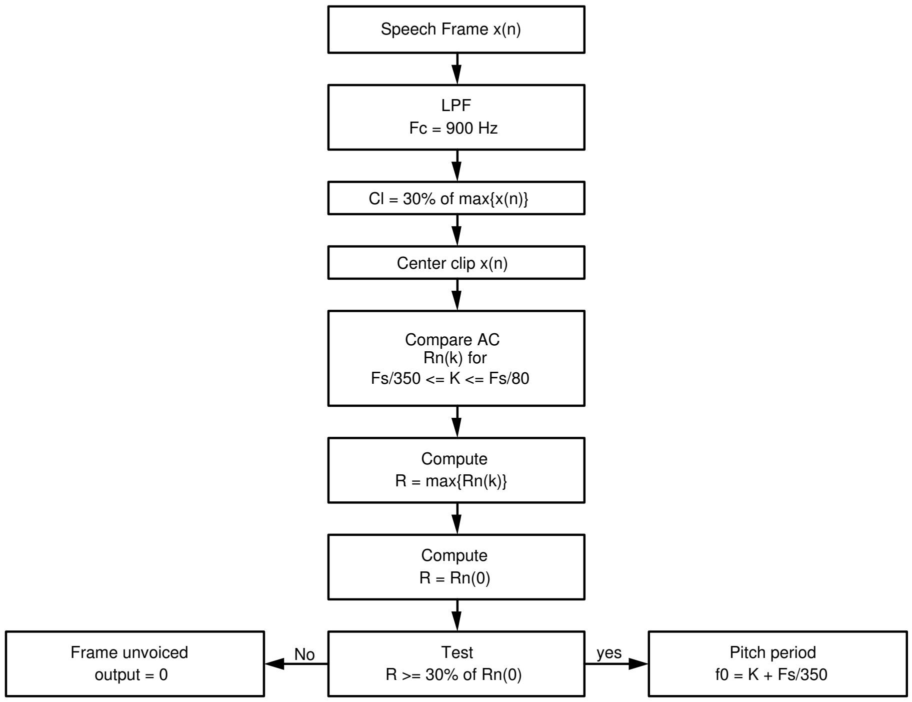

Figure 1.7: Overview of the autocorrelation pitch detector

1.5.1 Center clipping spectrum flattener

The meaning of center clipping spectrum flattener is to make a threshold (c_l) so only the high and low pitches are saved. I our case (c_l) is chosen to 30% of the max amplitude. the center clipping of a speech signal is shown in figure 1.8 on page XI

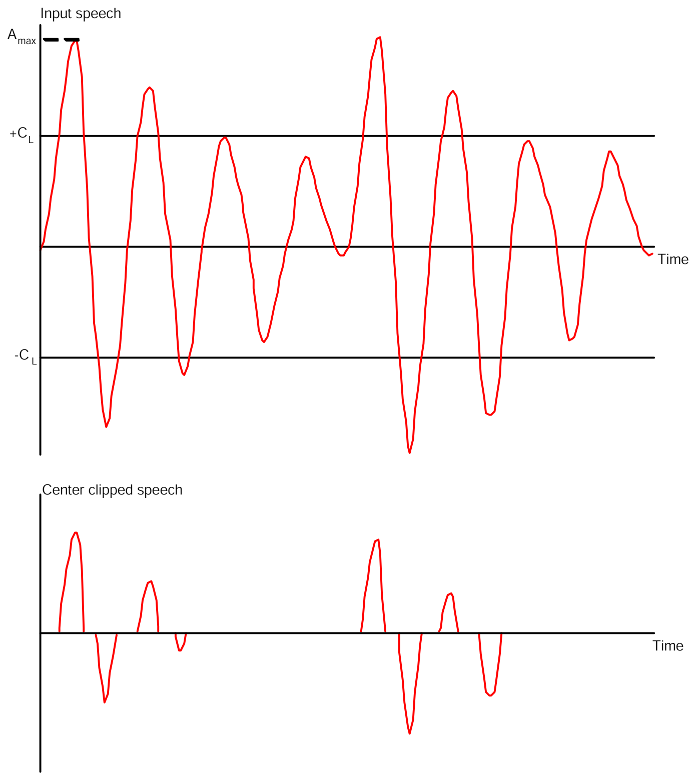

Figure 1.8: Center clipping

The center clipping is made due to the following definition:

- C_L =% of A_{max} (e.g. 30%)
- if x(n) > C_L then y(n)=x(n)- C_L
- if x(n) <= $C_L$then y(n) = 0

1.5.2 Autocorrelation

The autocorrelation is a correlation of a variable with itself over time. The autocorrelation can be calculated using the equation shown in equation 1.10

$$R(k) = \sum_{m=0}^{N.k-1} x(m)x(m+k) \tag{1.10}$$

In this project we have used the Matlab function "xcorr" to calculate the autocorrelation.

1.5.3 Median smoothing

When we have calculated the pitch detection, we can see that there are some outliers. To minimize these outliers, we have chosen to use a median smooth (Median Filter). The effect of this median filter is shown in figure 1.9. The window length (n) have been tested with n=3 and n=5. The window length have been chosen to n=5, because the pitch detector then eliminate the outliers in the noisy part. One of the problem using the Median filter with n=5 is that it smooths over five windows. I our project that means that we will have to make a delay in our system for 50 ms. The delay is calculated as:

$$number\ of\ windows\ *\ the\ length\ of\ the\ window\ *\ window\ overlap\ =\ 5*20ms*0,5=50ms \tag{1.11}$$

In our project we have chosen not to deal with this problem, cause we are not going to implement it as an real time system. If we should implement it as a real time system, we would have to predict the pitch five windows ahead.

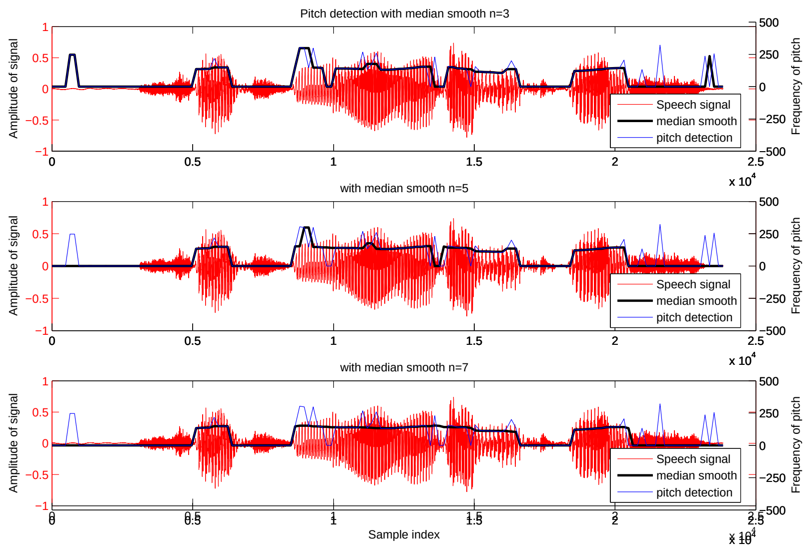

Figure 1.9: Median Smoothing

If the window length is set to n=7, shown in figure 1.9, the result seen perfect. The problem with such a long window is that the algorithm sometime finds a wrong fundamental frequency when there are "speech pause".

1.5.4 Test

The algorithm has been tested against result from the program "pitch analyzer 1.1", and has shown that the auto correlation pitch detector is very accurate.

1.6 Overall test and results

Thought this paper a speech signal has been used as a testsignal. But since pitch of a random testsignal isn't known, the algorithms has also been tested up againt signals with a known pitch. Figure 1.10 shows the 2 implemented pitch detectors, estimating the pitch of a linear swept frequency signal, which spectogram is shown in the top figure.

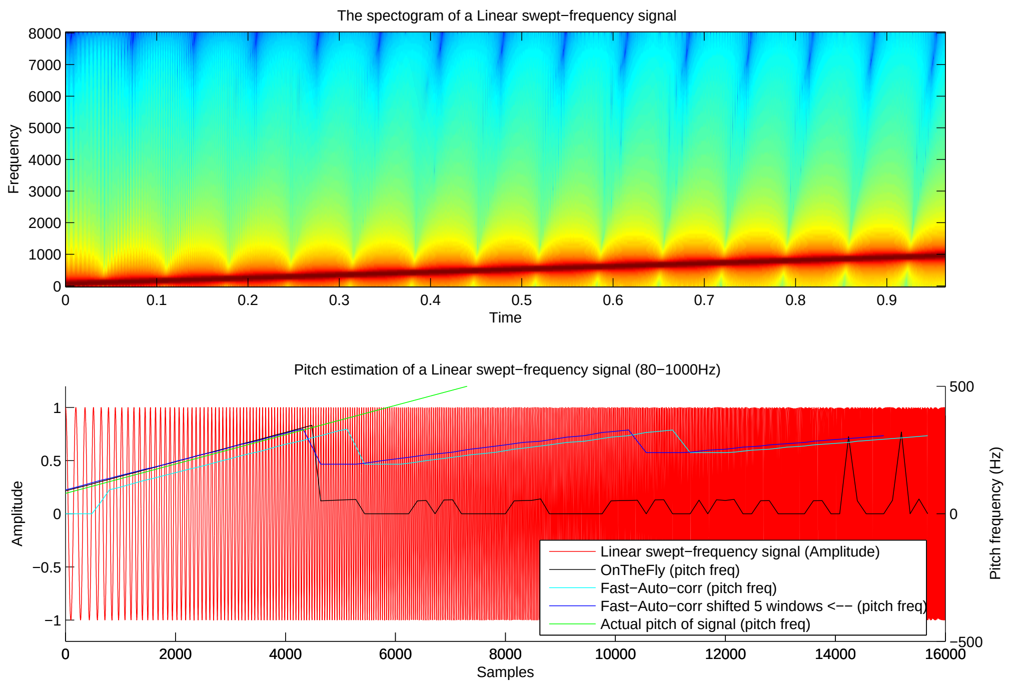

Figure 1.10: The top figure shows a spectogram of the linear swept frequency signal, starting at 80 Hz and ending in 1000 Hz. The buttom figure shows the 2 algorithms estimation of the pitch of the linear swept frequency signal. NOTICE the pitch detectors are constructed to detect the pitch in the interval 80-350 Hz.

As the figure shows, both methods are equally good to estimate the pitch within their operation limits 80-350 Hz. Outside these limits the figure shows that both methods quite poor to floor the pitch frequency, which would be the preferred outcome.
Further the 'Fast auto-correlation' method is delayed 5 frames (50ms), because it uses a median-smooth over 5 estimated pitches. This can't be compensated for and the only so-

lution to this, is therefore to remove the smoothing, which will cause more spikes (see figure 1.9 on page XII).

Another important feature are the influence of noise in signals. This is tested with a 200 Hz sinus signal, with a signal/noise ratio shifting from 100% to 0% in a timeinterval, shown in figure 1.11.

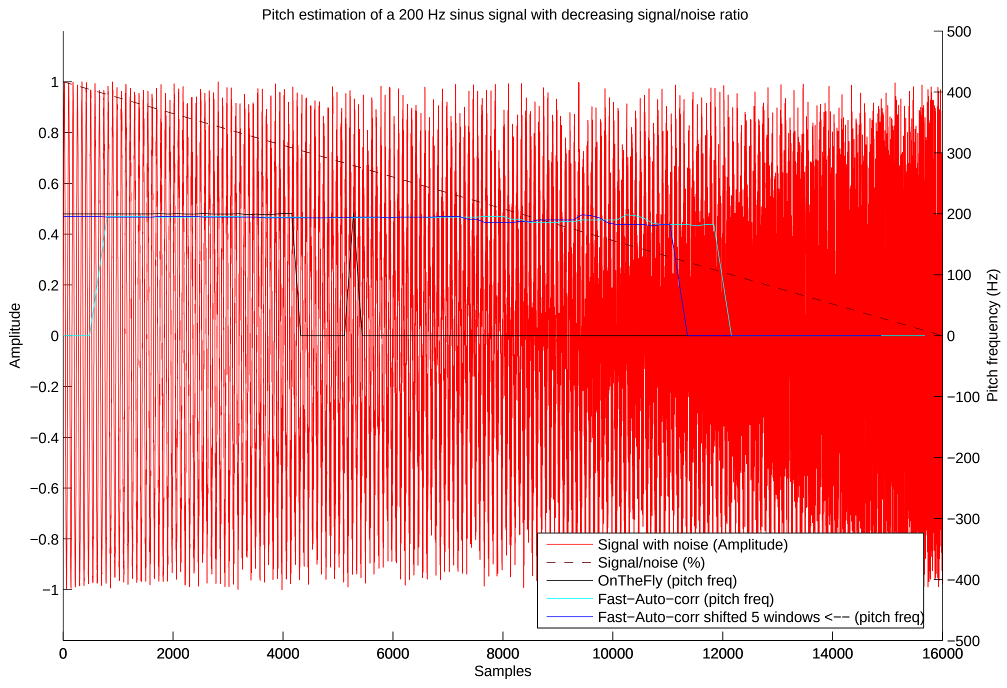

Figure 1.11: Test of the influence of noise in a signal. Here the signal/noise ratio reises with time.

The figure shows that the Fast-auto-correlation method is very robust against noisy signals, while the On-The-Fly breaks down very early in the test. Because speech signals aren't just a single sinus signal, such signals may more or less be similar to a noisy sinus. Therefore the Fast-auto-correlation would be preferred.

1.7 Conclusion

Between the 2 implemented methods of pitch detection, only the autocorrelation method are usable in the project "Bandwidth expansion of narrowband signals". It was shown (figure 1.6) that the "On-The-Fly" algorithm could not estimate a reliable pitch from band limited signals, which is a must in the project. The Fast-auto-correlation method, on the contrary, had no problem with either type of signal and is robust againt noisy signals. The preferred pitch detector is therefore the Fast-auto-correlation method.

1.8 Appendix

1.8.1 On-The-Fly's flaws when used on band limited signals

Tests of On-The-Fly algorithm showed that it was unable to estimate an accurate pitch of a band limited signal. So why is that?
The problem is with the lower cut-off frequency of the band pass filter. The On-The-Fly algorithm uses these lower frequencies harmonics to estimate the pitch. Because it is time based and only used the minimas of the difference equation, the higher order harmonics get more influence, than the lower order harmonics (see figure 1.2). Next the key idea is to be the threshold stages is to select the candidate harmonic with the smallest time period, resulting in a higher pitch. Comparing this key idea with the figure, the algorithm will most likely select a higher pitch frequency, than it should.

1.8.2 Matlab implementation of the "On-The-Fly" algorithm

```
% This function calculates the fundamental frequency of a frame of data
% with the OnTheFly algorithm.

% -- Input --
% data:      Frame of data for with pitch is to be calculated
% fs:        Sampling frequency of signal

% -- Output --
% f0:        The calculated fundamental frequency of the input data
%            0 is returned if no pitch is found

function [f0] = OnTheFly(data, fs)

frameLength = length(data);          % Framelength
curveFitSamples = 3;                 % Samples from minima to left and right

% Calculate difference function
d = zeros(1,frameLength-1);
for i=1:frameLength-1
    d(i) = sum((data(1:frameLength-i) - data(i+1:frameLength)).^2);
    d(i) = d(i) / (2*sum(data(1:frameLength-i).^2 + data(i+1:frameLength).^2));
end

% Find global minima and 4 local minima (smallest minima) with minimum 10
% samples between minimas. Minimas can have a maximum value of 0.4
[minimas minCount] = findmin(d,5,10,0.4);
%mins(end:-1:2,:) = mins(2:end,:);
nMins = size(minimas);

% If to many local minimas are located by findmin, the signal is likely to
% be noise, and therefore not to be processed. Also if no minimas has been
% found, the data is properly unvoiced, and is not to be processed.
if minCount <= 20 && nMins(1) > 0
    % Minimization of quatization error with parable curvefitting for each minima
    for i=1:nMins(1)
        % x-coodinates to make curvefitting from
        x = (minimas(i,1)-curveFitSamples):(minimas(i,1)+curveFitSamples);

        % Removes coordinates of x, which is not within the dataframe
        x(find(x < 1 | x > (frameLength-1))) = [];
        y = d(x);                                 % Corresponding values of x-coordinates
        [minimas(i,1) minimas(i,2)] = parreg(x,y); % Curvefitting and toppoint extraction
    end

    % Thresholding stage 1
    TTh1 = 0.2;     ATh1 = 0.07;              % Threshold settings
    [tg mins] = findf0(TTh1, ATh1, minimas);

    % Thresholding stage 2
    TTh2 = 0.05;    ATh2 = 0.2;               % Threshold settings
    [tg mins] = findf0(TTh2, ATh2, [tg; minimas]);

    f0 = fs/tg(1);      % Calculation of the fundamental frequency of the frame
else
    f0 = 0;             % Zero if frame is unvoiced or noise!
```

```
end
```

Locating minimas

Main function in locating minimas of data:

```
% Function locates nMins of the smallest minimas of data, where minimas
% have to be separated with at least spred samples. Also a minima can't
% excide maxMinValue!

% -- Input --
% data:         Data to locate minimas in
% nMins:        # of minimas to find in data (if possible)
% spred:        Minimum # of samples between minimas located
% maxMinValue:  Maximum value of a minima

% -- Output --
% ti:           A n-by-2 matrix where column 1 is indexes of minima and
%               column 2 is values of minima.
% minCount:     Total number of found minima in data

function [ti minCount] = findmin(data, nMins, spred, maxMinValue)

[n m] = size(data);     % Flips vector correct!
if m == 1
    data = data';
end

minimas = localminima(data)';                % Locates ALL minimas in data as indexes
minimas(:,2) = data(minimas(:,1))';          % Extends mins to hold datavalues to indexes of minimas
minCount = length(minimas);                  % Total number of local mins found

indexes = find(minimas(:,2) >= maxMinValue);% Find indexes of minima that excides maxMinValue
minimas(indexes,:) = [];                     % Deletes minimas that excide maxMinValue
[m n] = size(minimas);                       % New size of mins matrix

if m >= 1                                    % Only runs if data are available
    ti = zeros(nMins,2);                     % Allocates number of smallest minimas wanted
    for i=1:nMins
        [minima mIndex] = min(minimas(:,2));% Finds global min among minimas
        ti(i,:) = minimas(mIndex,:);         % Copies found global minima intro new matrice
        indexes = find(abs(minimas(mIndex,1)-minimas(:,1)) < spred);    % Finds indexes of minimas within 'spred' samples
            of found minima
        minimas([mIndex indexes'],:) = [];   % Deletes minima within 'spred' samples of found minima
        if length(minimas) <= 0              % Break loop if no more data
            ti(i+1:end,:) = [];              % Deletes used allocated space
            break;
        end
    end
else
    ti = zeros(0,2);                         % Allocates a 0-by-2 matrix (to avoid runtime error)
end
```

Sub function that locates all minimas of data:

```
% mins = LocalMinima(x)
%
% finds positions of all strict local minima in input array

function mins = LocalMinima(x)

nPoints = length(x);
Middle = x(2:(nPoints-1));
Left = x(1:(nPoints-2));
Right = x(3:nPoints);
mins = 1+find(Middle < Left & Middle < Right);

% Written by Kenneth D. Harris
% This software is released under the GNU GPL
% www.gnu.org/copyleft/gpl.html
% any comments, or if you make any extensions
% let me know at harris@axon.rutgers.edu
```

1.8.3 Curvefitting function

```
% This function calculates the toppoint of a curvefitted parable.
function [xx yy] = parreg(x, y)
%x:      x-koordinates to do curvefitting around
%y:      y-koordinates to do curvefitting around

[m n] = size(y);    % Vector x has to stand (a m-by-1 vector)
if n ~= 1
    y = y';
end
[m n] = size(x);    % Vector y has to stand (a m-by-1 vector)
if n ~= 1
    x = x';
end

% Curve fitting in the form Ax=b
X = [ones(size(x))  x  x.^2];  % X values
a = X\y;                       % Least sqaures solution of inclination
xx = -a(2)/(2 * a(3));         % Calculates x of toppoint of parable
yy = [1  xx  xx.^2] * a;       % Calculates t of toppoint of parable
```

1.8.4 Threshold stage function

```
% Function that locates the fundamental frequency of ti according to input
% thresholds TTh and ATh.

% -- Input --
% TTh:  Harmonic threshold
% ATh:  Amplitude threshold
% ti:   A n-by-2 matrix [index_of_minima,value_of_minima]
%       Row 1 MUST be the current picked fundamental frequency minima

% -- Output --
% tg:   New minima of the fundamental frequency
% ti:   Remaining minimas

function [tg, ti] = findf0(TTh, ATh, ti)
N = [2 3 4 5 6];         % Threshold constants to test for harmonics

tg = ti(1,:);            % Extract current fundamental frequency of minimas
ti(1,:) = [];            % Deletes the current fundamental frequency of minimas from t
ti = sortrows(ti,1);     % Sorts rows according to indexes (ascending)
ti = ti(end:-1:1,:);     % Reverses ti... sort must be desending
index = [];              % Must to allocated to avoid runtime warnings

% Test for candidates of harmonic minimas to the fundamental frequency minima
for i=1:length(N)                              % Run elements in n - times
    x = tg(1)./ti(:,1);                        % Calculates tg/ti
    newIndexes = find(abs(N(i)-x) < TTh);      % Finds indexes of ti, that satifies the threshold TTh
    if length(newIndexes) > 0
        index = [index; newIndexes];           % Adds the found index to a general index variable
    end
end

% Test among the candidates of a minimas to the fundamental frequency minima
% Test the candidates if the thredshold is within the threshold ATh
newTgIndex = 0;                            % Index among index of new fundamental frequency minima
for i=1:length(index)                      % Runs only among the harmonic candidates
    if ti(index(i),2) - tg(2) < ATh        % Test if ATh threshold if satified
        newTgIndex = i;                    % Save index of 'index' of new fundamental frequency minima
    end
end
if newTgIndex > 0                          % If a new fundamental frequency was found, set new tg
    tgTemp = ti(index(newTgIndex),:);      % Makes copy of new fundamental frequency minima
    ti(index(i),:) = tg;                   % Copies old fundamental frequency minima into ti
    tg = tgTemp;                           % Copies new fundamental frequency minima into tg
    ti = sortrows(ti,1);                   % Sorts rows according to indexes (ascending)
    ti = ti(end:-1:1,:);                   % Reverses ti... sort must be desending
end
```

1.9 Litterature list

[1] Saurabh Sood and Ashok Krishnamurthy: A Robust On-The Fly Pitch (OTFP) Estimation Algorithm. The Ohio State University, Columbus OH.

[2] A. D. Cheveigne and H. Kawahara. Yin: A fundamental frequency estimator for speech and music. Journal of the Acoustical Society of America, 111(4), 2002.

[3] S. Chowdhury, A. K. Datta, and B. B. Chaudhuri. Pitch detection algorithm using state phase analysis. Journal of the Acoustical Society of India, 28(1), Jan. 2000.

[4] http://www.cnmat.berkeley.edu/~tristan/Report/node4.html

[5] http://www.ee.ucla.edu/~ingrid/ee213a/speech/vlad_present.pdf