Fachgebiet Audiokommunikation
Prof. Dr. Stefan Weinzierl

Technische Universität Berlin

Audiotechnik 2 - Tutorium:

Grundfrequenzerkennung

10.01.2013

1 Das Host-Skript

Ziel dieser Aufgaben ist die Simulation einer digitalen Audio Workstation (DAW), zur Erzeugung von Audioeffekten. Hierbei wird die Signalverarbeitung von Funktionen ausgeführt, welchen die Daten des Eingangssignals blockweise von einem Host übergeben werden. Schreiben sie ein *Host-Skript*, welches die Audiodatei `Violin_2.wav` aus dem Downloadbereich einliest und diese in einer Schleife in Blöcken zur Verarbeitung bereitstellt.

Lösung:

Das folgende Host-Skript beinhaltet bereits sämtliche Erweiterungen für die folgenden Augaben:

```
% MAIN SCRIPT

close all
clearvars

%% Input signal
[x, fs] = wavread('Violin_2.wav');

% Block length
N      = 2^10;
nFrames = length(x)/N;

% number of process runs
endPos = length(x)-N;

% upsampling for f0 detection
upsampFactor = 10;

% fundamental frequency range
fMin      = 20;
fMax      = 5000;

%%
% allocate memory for output vectors
f0VecCorr = zeros(nFrames,1);
f0VecDiff = zeros(nFrames,1);
MIDIvec = zeros(nFrames,1);

```

```
% process audio in blocks
for idx = 1:nFrames-1

   % fill input buffer
   bufferIn = x((idx-1)*N+1:idx*N);

   % call functions
   [f0ValCorr,MIDInote] = f0_autocorr_TUT(bufferIn, fs, upsampFactor,
      fMin, fMax) ;
   [f0ValDiff] = f0_diffFunc_TUT(bufferIn, fs, upsampFactor,fMin, fMax)
       ;

   % write values to output vectors
   f0VecCorr(idx) = f0ValCorr;
   f0VecDiff(idx) = f0ValDiff;
   MIDIvec(idx) = MIDInote;

end

%% Plots

figure
plot(f0VecCorr,'b')
hold on
plot(f0VecDiff,'m')

figure,
plot(MIDIvec)
```

2 Grundfrequenzerkennung: Autokorrelation

Ein Weg, um die Grundfrequenz eines Signals im Zeitbereich zu ermitteln, bedient sich der Autokorrelationsfunktion:

$$r_{xx}(\tau) = \frac{1}{L}\sum_{j=1}^{L} x_j \cdot x_{j-\tau} \tag{1}$$

Die Grundfrequenz des Signals lässt sich aus der Autokorrelationsfunktion dann als erstes signifikantes Maximum L_{max} nach dem globalen Maximum bei $\tau = 0$ ermitteln:

a) Schreiben Sie eine Funktion, die mittels Autokorrelation die Grundfrequenz der einzelnen Blöcke ermittelt. Verwenden Sie in ihrer Funktion die Matlab-Funktion `xcorr()` zur Berechnung der Autokorrelationsfunktion und geben Sie die Grundfrequenz in Hertz, sowie als MIDI-Notenwert aus.

Lösung:

Der MIDI-Notenwert ($M = 0 \ldots 127$) läst sich nach folgender Formel berechnen:

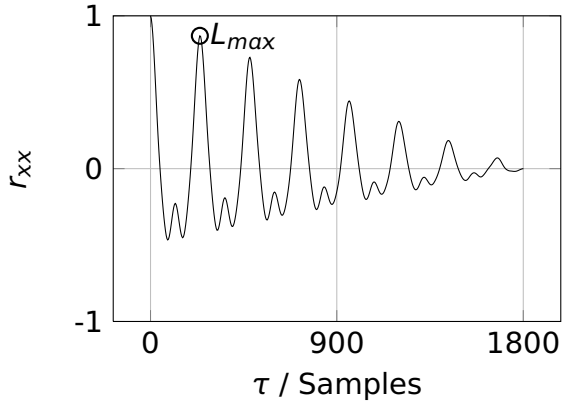

Abbildung 1: Normierte Autokorrelationsfunktion mit Maximum L_{max}

$$M = 12 \cdot \log_2(\frac{f0_{Hz}}{440.5Hz})) + 69 \tag{2}$$

Der komplette Matlab-Code befindet sich am Ende des Aufgabenteils.

b) Analysieren sie den Grundfrequenzverlauf der Datei `Violin_2.wav` mit einer sinnvollen Blockgröße. Wodurch ist letztere eingeschränkt? Erzeugen Sie Grafiken zum Grundfrequenzverlauf und deuten Sie diese.

Lösung:
Die Länge des Analysefensters für die Grundfrequenzerkennung muss mindestens das Dreifache der längsten zu erwartenden Periodendauer betragen. Es darf jedoch nur so lang sein, dass innerhalb des Fensters von einem quasistationären Signal aussgegangen werden kann. An den Notenübergängen in Abbildung 2 lassen sich sogenannte *Subharmonic Errors* erkennen. Hier wird vom Algorithmus fälschlicherweise ein vielfaches der tatsächlichen Grundfrequenz erkannt.

c) Verbessern Sie die Auflösung der Grundfrequenzerkennung, indem Sie die zu bearbeitenden Blöcke innerhalb Ihrer Funktion um einen sinnvollen Faktor upsamplen. Vergleichen Sie das Ergebnis grafisch mit dem Verlauf aus der vorherigen Aufgabe und erklären Sie, wie durch diesen Schritt die Grundfrequenzerkennung verbessert wird. Nennen sie weitere mögliche Schritte, um die Robustheit der Grundfrequenzerkennung zu erhöhen und implementieren Sie diese gegebenenfalls.

Lösung:
Das hier gewählte Verfahren zum Upsampling ist sehr rechenaufwändig. Für eine Implementierung in der Anwendung empfiehlt sich eine Methode, die den Rechenaufwand berücksichtigt. Das Signal wird hier um den Faktor 10 überabgetastet. Dies führt zu einer deutlichen Verbesserung der Genauigkeit der Grundfrequenzerkennung, durch eine feinere Quantisierung der Frequenzachse, wie in Abbildung 3 zu sehen ist.

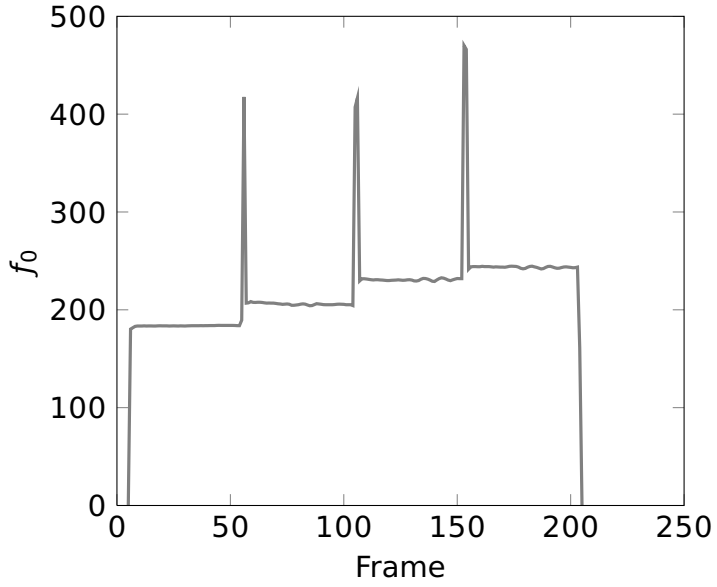

Abbildung 2: Grundfrequenzerkennung - Autokorrelation

Weiterhin sinnvoll für eine erhöhte Robustheit ist eine Begrenzung des zu erwartenden Bereichs der Grundfrequenz von etwa $30Hz \cdots 3000Hz$, sowie eine Tiefpassfiltrung for der Grundfrequenzerkennung. Außerdem kann durch das sogenannte *Center-Clipping* eine Verbesserung erzielt werden. In diesem Beispiel führen die genannten Maßnahmen jedoch nicht einer besseren Erkennung der Grundfrequenz.

```
1  function [f0_val2, MIDInote] = f0_autocorr_TUT(frame, fs, upsampFactor,
       fMin, fMax)
2
3  % pre filtering
4  [b,a] = butter(5, 5000/fs);
5  %
6  frame = filter(b,a,frame);
7
8  % upsample signal
9  frame = interp(frame,upsampFactor);
10
11 % center clipping
12 frame(abs(frame)<0.01)=0;
13
14 % get correlation vector
15 [yy, lag] = xcorr(frame);
16
17 % Use only left half of correlation vector
18 % also possible -> 'maxlag'
19 tmp = fix(length(lag)/2): length(lag);
20 yy = yy(tmp);
21 yy = yy/max(yy);
22
23 % Find first (global) minimum
```

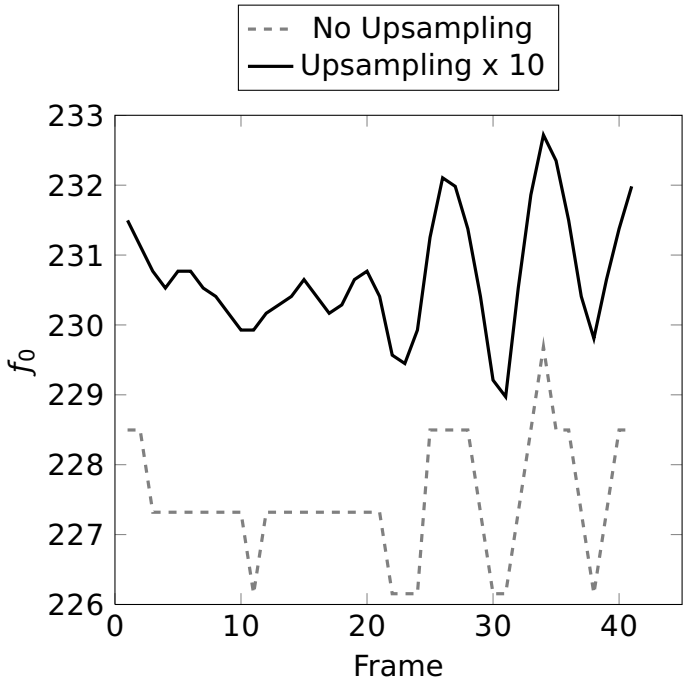

Abbildung 3: Vergleich - Upsampling

```
[~, lstart] = min(yy);

% Find largest peak after first minimum
[~, totalmax] = max(yy(lstart:end));

% get frequency position
f0_pos = (totalmax+lstart);
% get f0 value - regarding upsampling
f0_val = (fs / (f0_pos))*upsampFactor;

% check whether f0 is in range
if f0_val<fMin || f0_val>fMax
   f0_val = 0; % set "0" if not
end

f0_val2=f0_val;

% get MIDI note value
MIDInote = round(12*log2(f0_val2/440.5 )) + 69;

end
```

3 Grundfrequenzerkennung: Distance-Function

Mehr Robustheit in der Grundfrequenzerkennung, basierend auf demselben Ansatz (Ähnlichkeit eines Signals mit seiner Verzögerung), liefert die *Distance-*

Function:

$$d_{xx}(\tau) = \frac{1}{L}\sum_{j=1}^{L}(x_j - x_{j-\tau})^2 \tag{3}$$

In der *Distance-Function* wird dann nach dem ersten signifikanten Minimum nach dem globalen Minimum bei $\tau = 0$ gesucht. An dieser Stelle lässt sich die Grundfrequenz berechnen.

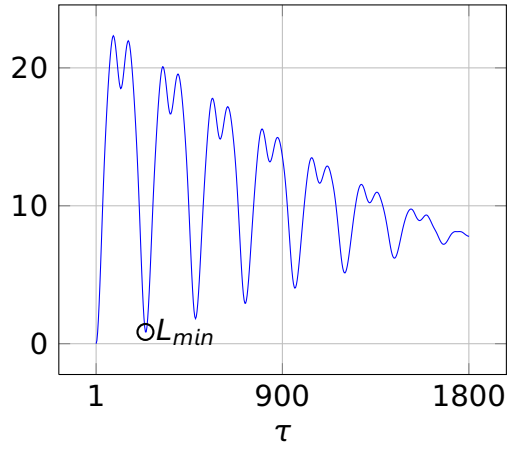

Abbildung 4: Distanzfunktion mit signifikantem Minimum

a) Schreiben Sie eine Funktion, die mittels Distance-Function die Grundfrequenz der einzelnen Blöcke ermittelt. Geben Sie die Grundfrequenz in Hertz, sowie als MIDI-Notenwert aus.

Lösung:
Der komplette Matlab-Code befindet sich am Ende des Aufgabenteils.

b) Analysieren sie den Grundfrequenzverlauf der Dateie `Violin_2.wav` mit einer sinnvollen Blockgröße. Erzeugen Sie Grafiken zum Grundfrequenzverlauf und deuten Sie diese.

Lösung:
Der mit der Distance-Function ermittelte Grundfrequenzverlauf zeigt weniger Subharmonic-Errors, zu sehen bei dem Übergang von der dritten zur vierten Note in Abbildung 5.

c) Verbessern Sie die Auflösung der Grundfrequenzerkennung, indem Sie die zu bearbeitenden Blöcke innerhalb Ihrer Funktion um einen sinnvollen Faktor upsamplen. Nennen sie weitere mögliche Schritte, um die Robustheit der Grundfrequenzerkennung zu erhöhen und implementieren Sie diese gegebenenfalls. Vergleichen Sie das Ergebnis mit dem der Autokorrelationsmethode.

Lösung:

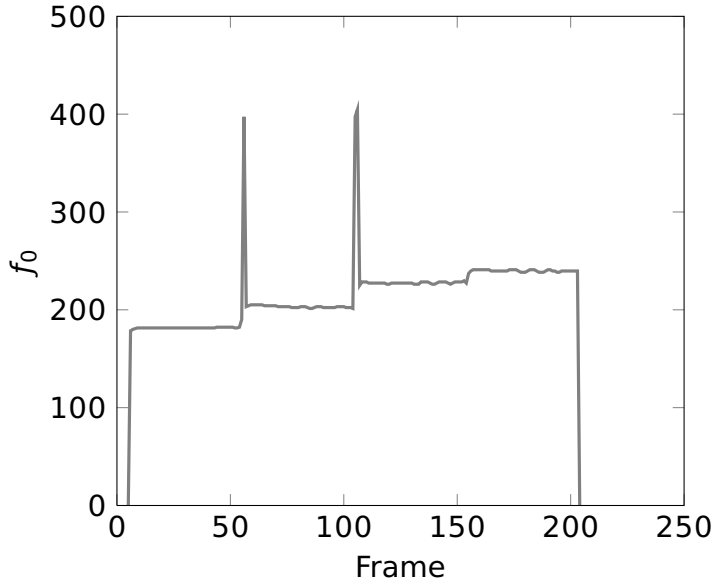

Abbildung 5: Grundfrequenzerkennung - Distance Function

Auch hier gilt: Das Upsampling sorgt zwar für eine genauere Grundfrequenzerkennung, erhöht den Rechenaufwand aber erheblich. Eine weitere Verbesserung liefert die “Cumulative mean normalized difference function”, wie sie beispielsweise der Algorithmus *YIN* verwendet.

```
1
2  function f0_val = f0_diffFunc_TUT(y, fs, upsampFactor,fMin, fMax)
3
4  % upsampling
5  y = interp(y,upsampFactor);
6
7  % allocate memory for distance vector
8  yy = zeros(size(y));
9
10 % get distance vector in loop
11 for i=1:length(y)
12
13    % simple distance function
14    yy(i)=sum((y- [y(i:end) ;zeros(i-1,1)]).^2);
15
16 end
17
18 yy=smooth(yy,20);
19 yy=yy/max(yy);
20
21 % find first max
22 [ymin, lstart] = max(yy);
23
24 % Find largest dip after first minimum
25 [totMaxVal, totalMin] = min(yy(lstart:end));
26
```

```
% get frequency
f0_pos = (totalMin+lstart);
f0_val = (fs / (totalMin+lstart+1))*upsampFactor;

% check range
if f0_val<fMin || f0_val>fMax
   f0_val= 0;
end
```