Menura:
Realtime pitch detection in C++

Valentina Visintini

Practical Course "Advanced Software Development with Modern C++"
Summer Term 2018
Institute for Computer Science
Ludwig-Maximilians-Universität München

1 Introduction

A typical task of digital signal processing is to identify the pitch of an audio signal. In this project, a whistled audio signal is analyzed to identify the played note in realtime. To achieve this, the programming language C++ was chosen, as it is very efficient and allows for an implementation based on formal concepts.

2 A Brief Outline of Acoustics

In music, a *pitch* is the position of a single sound within the complete range of sound, which is perceived according to the frequency of vibration of the sound waves. Human listeners perceive higher frequencies as higher pitches, hence a lower pitch is indicative of a lower frequency. A pitch detector aims to find the fundamental frequency of an audio signal, i. e. the lowest frequency of a periodic waveform, to determine what note is being played. Since the fundamental is also perceived as the loudest, the ear identifies it as the specific pitch of the musical tone

Sound waves cause a local pressure deviation from the ambient atmospheric pressure, which is known as sound pressure and perceived as *loudness.* Even though it can be ordered on a scale extending from quiet to loud, loudness cannot be measured objectively, as its perception changes individually, whereas the actual sound pressure can be measured using a microphone and expressed in decibel (dB). The sound pressure level L_p is defined by a logarithmic measure of the effective pressure of a sound p relative to the standard reference sound pressure p_0 in air or water.

$$L_p = 20 * \log_{10}(\frac{p}{p_0})\text{dB} \tag{1}$$

3 Frequency Spectrum Analysis

The fundamental frequency of an audio signal can be obtained by a fast Fourier transform (FFT). This algorithm samples a signal over a period of time and divides it into its frequency components, converting it from the time domain to a representation in the frequency domain.

$$\sum_{n=0}^{N-1} x_n e^{-i2\pi k\sqrt{-1}/N} \tag{2}$$

In order to detect the pitches of an audio signal, it is split up in equal sample windows and each window is analyzed separatedly. The size of the windows, i. e. the FFT size, affects the resolution of the resulting spectra: the larger the size, the higher it is. The frequency resolution f_{res} is equal to the reciprocal of the time window duration T_{win}. So for M samples going into the FFT and a sampling period of T seconds, then $T_{win} = M * T$, thus $f_{res} = 1/(M * T)$. F_s is the sampling frequency, with $F_s = 1/T$. The resulting output contains the magnitude information for the respective frequency at each bin (index) k and can be converted to the frequency this way:

$$f = \frac{k}{M \cdot T} = \frac{k}{M} \cdot F_s \tag{3}$$

The magnitude of each FFT bin i can be expressed like this:

$$mag(k) = \sqrt{(re(k)^2 + im(k)^2)} \tag{4}$$

To obtain the loudness of each frequency, the magnitude must be transformed to decibel:

$$A(k) = 20 * \log_{10}(mag(k))\text{dB} \tag{5}$$

Starting from this, the frequency with the largest magnitude can be identified as the fundamental frequency of the analyzed window.

4 Matching Frequency to Note

A note can be matched in many ways but one would be via the corresponding key on a piano. It is calculated by the relation of the frequency to the concert pitch (A4) and the result is the keyboard index of the detected note.

$$d_f = 49 + 12 \cdot \log_2\left(\frac{f}{440 \text{ Hz}}\right) \tag{6}$$

Every note has their ideal frequency but, of course, it is not very likely to be met when the audio input is whistling. So there is a range in which a pitch will be accepted as note that can be mathematically expressed by cents.

$$n = 1200 \cdot \log_2\left(\frac{b}{a}\right) \tag{7}$$

5 Modeling

To enable an analysis of the frequency spectrum, an audio signal must be converted to a sequence of samples first (see Fig. 1).

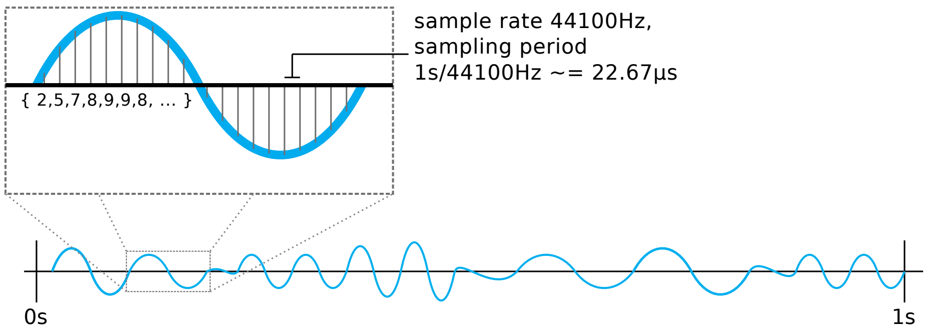

Fig. 1: Audio sampling

Now the signal can be split up into sample windows and transformed from time domain to frequency domain, as described in section 3 (Fig. 2.

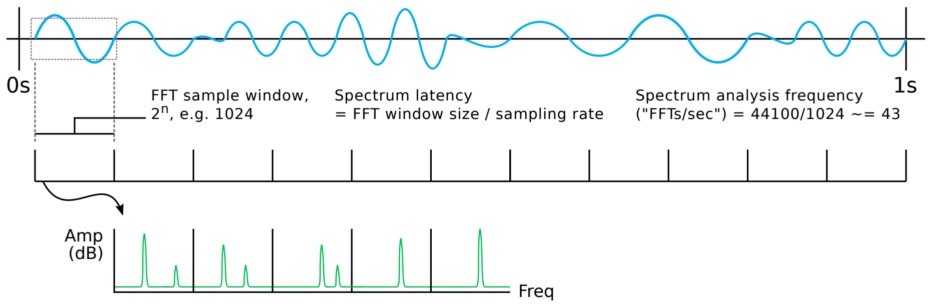
0s
1s
FFT sample window, 2^n, e.g. 1024
Spectrum latency = FFT window size / sampling rate
Spectrum analysis frequency ("FFTs/sec") = 44100/1024 ~= 43
Amp (dB)
Freq

Fig. 2: Frequency Spectrum Analysis & Signal Transformation: Time Domain to Frequency Domain

In order to find the fundamental frequency, the maximum must be determined by finding the largest magnitude in the spectrum (Fig. 3).

The last step is to match the frequency to the corresponding note (Fig. 4), so it can be processed further, e. g. to be written to MIDI.

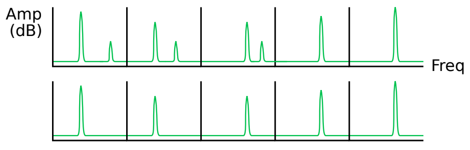

Fig. 3: Pruning

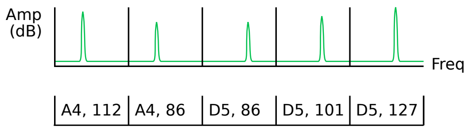

Fig. 4: Labeling

6 Implementation

Menura contains several separate modules as shown in Fig. 5. The input module handles the audio input stream which is passed to the next module, performing the frequency spectrum analysis. The resulting frequencies are further processed by pruning and the fundamental frequency is selected in order to be labeled a note by the next module. The detected note is then written to the the command line.

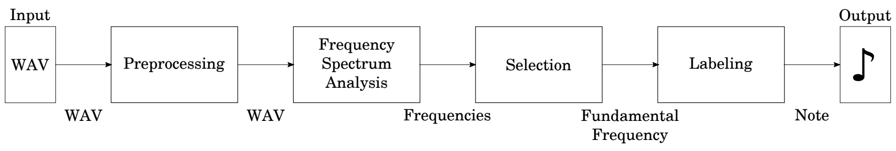

Fig. 5: System diagram

The audio is recorded by means of an asynchronous callback of the PortAudio [2] library, which enables cross-platform realtime audio in- and output. The data written to the input buffer and then passed to the frequency spectrum analysis. For calculating the FFT the FFTW [1] library, a C subroutine library for computing the discrete Fourier transform, was used.

The labeling was realized as described in section 4:

Listing 1.1: Note detection

```
note note_of(double frequency, double pitch_hz = 440.0) {

  int note_idx = 12 * std::log2(frequency / pitch_hz) + a4_idx;

  double ideal_freq = pitch_hz * std::pow(2.0, (note_idx - a4_idx) /
      12.0);

  int cent_idx = 1200 * std::log2(frequency / ideal_freq);

  if(frequency >= ideal_freq) {
    if(cent_idx > 50) {
      note_idx++;
      cent_idx = 100 - cent_idx;
      if(cent_idx != 0)
        cent_idx = (-1) * cent_idx;
    }
  }
  // else analogously
}
```

The performance of the system depends on the FFT, which is more or less efficient depending on the specified FFT size that can be set at program start. As for the rest, every step of the process is deterministic in both in time and memory, independently of the input.

References

1. FFTW. `http://fftw.org/`. Accessed: 2018-10-12.
2. PortAudio: Portable Cross-platform Audio I/O. `http://portaudio.com/`. Accessed: 2018-10-12.