

PRISMA – A TIMBRE TUNING DEVICE FOR MUSICIANS AND INSTRUMENT MAKERS

Franz Bachmann, Michael Bernhard, Hans-Christof Maier

Berne University of Applied Sciences, Burgdorf (Switzerland), baf1@bfh.ch

Berne University of Applied Sciences, Burgdorf (Switzerland), bam8@bfh.ch

Zürich University of the Arts, Zürich (Switzerland), hans-christof@dplanet.ch

Abstract

Prisma is a high resolution measuring system for the analysis of musical instrument sounds that works real-time or with given wav-sounds. The system may serve as a tool for instrument makers or may support practising and teaching musicians. In this paper the *Prisma* system is described and some examples are given.

INTRODUCTION

Musicians recognise the characteristic sound of their instruments by means of scarcely audible nuances in timbre. These are determined not only by the design of the instrument but also by random irregularities due to material and manufacturing.

In the *Prisma* project, we regard the musical instrument as a black box with the properties of the instrument, the way of playing and the acoustical environment as input parameters. The output is a musical sound, which is recorded, stored digitally and analysed by means of digital signal processing methods. The computed features can give us information about the influence of the input parameters and the internal behaviour of the instrument.

Our aim was to develop a high resolution measuring system for the analysis of musical instrument sounds that works real-time or with given wav-files and runs on a standard computer. It computes selected features and displays them numerically or graphically. Instead of applying an explicit partial tracking process we use a simple intuitive classification of peaks in the local spectra. The user can choose content and scaling of the diagrams freely – either in musical or in physical terms. Finally, the results can be stored, printed or exported for further use by programs like Matlab or MS Excel. The hardware is designed portable and consists of a microphone, a sound card and a laptop computer.

The practical use of the *Prisma* system is multi-functional. It may serve as a high precision measuring tool for instrument makers. Considering not only the instrument itself but also the way of playing, the equipment can also be used as a tool for practising and teaching. The visual display of the results can contribute to improve our acoustical remembering and to remedy the lack of clear cut verbal expressions for aural perceptions.

PRISMA-REALTIME

The core of the *Prisma* system is the *Prisma-Realtime* software. It performs three basic steps in real-time: Acquisition of audio data, computation of spectral features and visualization of the spectral information. Like other sound analysis

programs, *Prisma-Realtime* uses the Short time Fourier transform, in our case with a default frame length of $N = 1025$ samples and an overlap of 256 samples. Assuming a sampling frequency of 44100 Hz, new spectral data are ready every 17.4 ms. In each time frame, the following actions are performed:

- Short time Fourier transform
- Restriction to local maxima of the magnitude spectrum
- Improvement of the spectral data
- High precision fundamental frequency computation
- Harmonicity filter

For the Fourier analysis part we use the well-known technique of zero-phase windowing [Serra, 1997]. The signal in the current time frame of length N is multiplied with a Gauss window and the result is packed in an FFT buffer of length $N_{FFT} = 2 \cdot N - 2 = 2048$, padding the middle with $N - 2 = 1023$ zeroes. The next step is data reduction: From the result of the FFT (a complex vector of length N_{FFT}) we keep only the components corresponding to local maxima (peaks) of the magnitude spectrum. This simple procedure reduces the size of the original FFT data to 10 percent in the average and is essential for real-time operation. The remaining frequency and magnitude data are improved by reassignment [Desainte-Marchand, 2000]. As a result of the above steps we obtain two vectors $F = [F_1, \dots, F_m]$ and $A = [A_1, \dots, A_m]$ containing the spectral data of the frame. Note that the length m of these vectors depends on the frame. In the present version of *Prisma-Realtime*, phases are discarded.

To estimate the fundamental frequency f_u of the current frame, we use a combination of frequency and time domain methods. Starting with the vectors F and A described above, we get raw estimates of f_u by considering rational approximations of quotients F_i/F_k . The final value of f_u is obtained by a method in the spirit of [Medan, Yair, Chazan, 1991]. Consider two adjacent data blocks u and v of equal length τ which lie symmetric with respect to the middle of the current frame (Fig. 1):

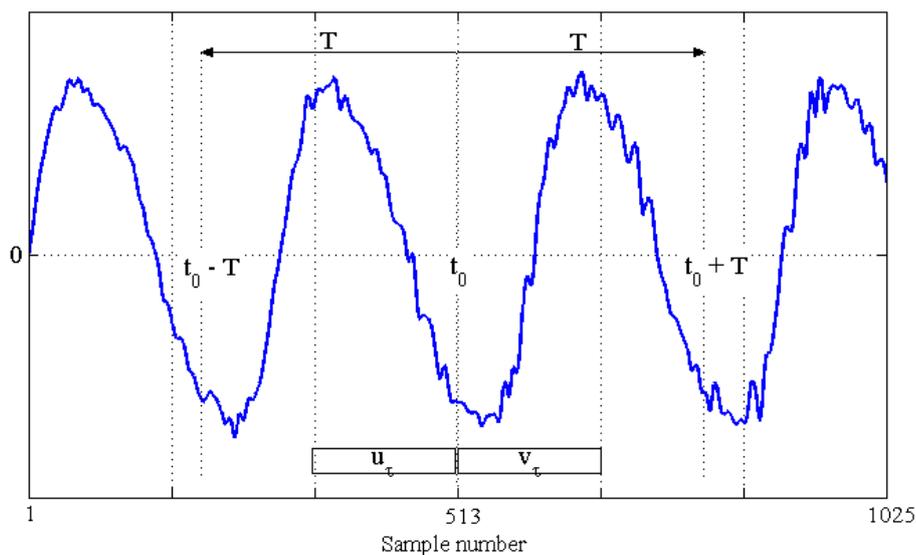


Figure 1: Two adjacent blocks in a time frame

The normalized correlation coefficient of u and v is defined by

$$\rho(u, v) = \frac{\sum_{i=1}^{\tau} u_i v_i}{\sqrt{\sum_{i=1}^{\tau} u_i^2} \sqrt{\sum_{i=1}^{\tau} v_i^2}} \quad (1)$$

and the integer period of the signal is the value of τ such that $\rho(u, v)$ is maximal. To make this optimization problem computationally feasible, we make use of the above raw estimates and of prior knowledge about reasonable frequency bounds. In addition to the estimated value of the fundamental frequency, the normalized correlation coefficient is a reliability measure for the estimation. Given this value of the fundamental frequency f_u , harmonic bands are defined around the integer multiples of f_u as soon as the user has chosen a number of cents as bandwidth (Fig. 2).

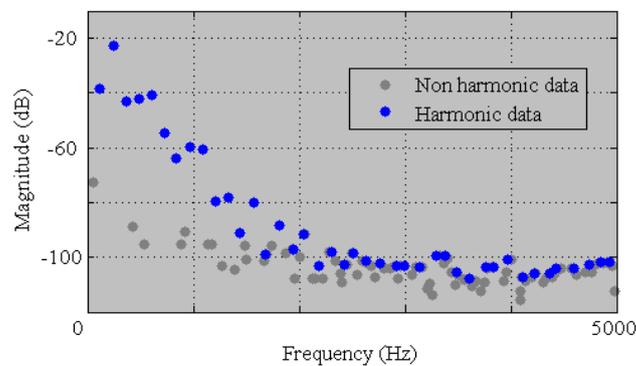


Figure 2: The spectral data of a frame

For a description of the visualization possibilities of *Prisma-Realtime* we refer to the chapter with the examples below. In order to meet the requirements of all the different users we tried to offer a maximum of flexibility in scaling. In addition to the display on the screen the computed spectral data can be stored in a file (binary or text) and the recorded sound in a wav-file.

PROGRAMMING CONSIDERATIONS

From the viewpoint of the programmer the development of *Prisma-Realtime* was a challenge. The reason is the requirement to visualize the spectral data in *real time*. Although a desktop operating system like Windows does not allow real-time applications in the strict sense; it is possible to acquire, process and visualize data in a way such that the user has the impression of continuity. There is an inevitable latency between the acquisition of the audio data and the display on the screen. In the present application, the latency is between 100 and 500 ms, mainly due to the recording buffer size for the audio capturing, the required amount of data for the calculations, the processing time and the time to produce the complex graphics. It is known that very small latency values can lead to a loss of captured data in Windows systems. Our goal was therefore not just to minimize latency, but to find a trade-off between display comfort and system reliability. However, on modern PC hardware, *Prisma-Realtime* runs stably with 60 graphics frames per second and produces a smooth graphics output.

In the development of *Prisma-Realtime* three tools were used:

- The application *Prisma-Realtime* was written in the C++ language. To implement the algorithms efficiently, the *Integrated Performance Primitives (IPP)* of Intel were used. *IPP* is a well-known library which improves the processing performance of multimedia applications dramatically. It is highly optimized for Intel processors and provides simple and advanced functions for basic operations of mathematics and signal processing.
- *OpenGL* (Open Graphics Library) is a standard specification defining a language and platform independent API for writing applications that produce 2D and 3D computer graphics. *OpenGL* contains over 250 different functions and is popular in the video game industry as well in professional graphics applications. Using this library, it is possible to show the spectral data associated to the last three seconds of the audio stream on the screen using an ordinary graphics card. Furthermore, *OpenGL* allows the programmer to implement some effects like the fading of colors in a simple way [Segal, Akeley, 2004].
- For the development of the *Prisma-Realtime* user interface the GUI library *QT* by Trolltech (<http://www.trolltech.com/products/qt>) was used. *QT* is an object oriented library which provides functionality for GUI programming, basic data handling, networking and much more. In addition *QT* has the advantage of being portable among different operating systems. At the moment *Prisma-Realtime* is designed for Windows systems only.

The basic structure of *Prisma-Realtime* is a loop containing three blocks: acquisition of audio data, computation of spectral quantities and visualization of these quantities on the screen. Between these activities, data are handled in ring buffers, so that unnecessary memory allocation and release steps can be avoided. To prevent nasty jerking of the graphics output, a synchronization with the audio data source (sound card) is required. Finally, to make use of multiprocessor computers, the data processing, the visualization as well as the GUI interaction are running in separate threads.

EXAMPLES

Clarinet Melody

The solo melody in Spohr's Clarinet Concerto op. 57 (Fig. 3) demonstrates the time-frequency resolution of the measuring system and shows some difficulties in playing the piece.



Figure 3: Louis Spohr, Clarinet Concerto op. 57: Beginning of the solo part in the Adagio

Fig. 4 shows the *Prisma* screen with this sequence from the end of the first minim at the left to the beginning of the crotchet at the right side. The diagram above shows the time-frequency view of the first partial. The ornament with a semitone upward and a whole step downward is clearly visible. Each note lasts 174 ms. The diagram below displays a frequency range from 0 to 4000 Hz. The partials are coded in rainbow colours which correspond to other views. As can be heard in the sound file the player had problems to make a direct hit to the highest note (e flat) after the ornament. This appears in form of gray-coloured dots between the tones in the diagram. The following transition to d flat, however, is a perfect legato. In the eighth notes more harmonics which are rising up towards the end of each single tone can be detected. Besides this, there are again more harmonics in the last two tones than in the two earlier ones. We guess that this may depend on the preceding fast notes of the ornament after which the player needs time to optimize the sound.

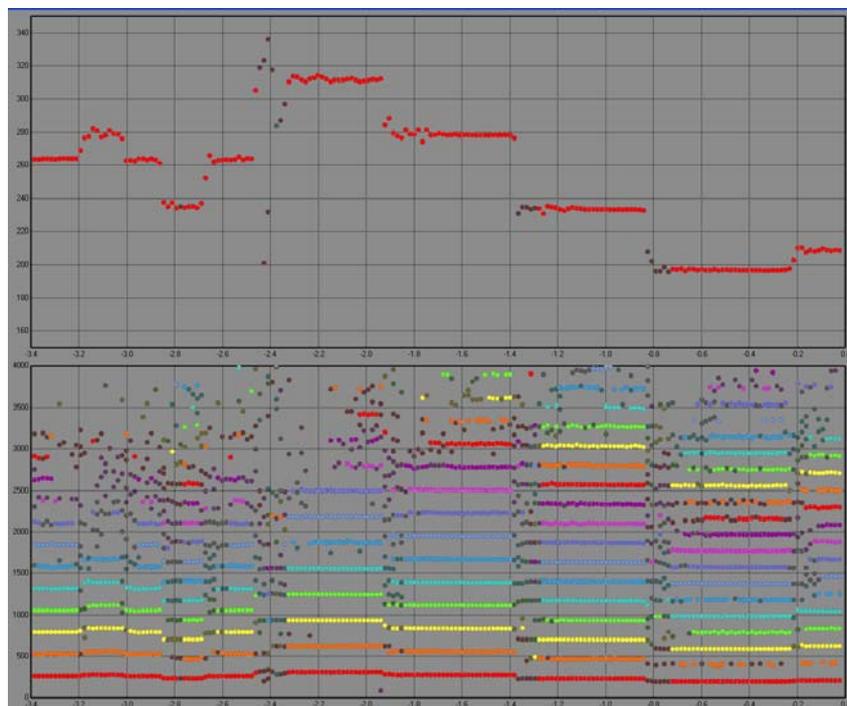


Figure 4: Time-frequency views of the music in fig. 3

Harpsichord tone

When tuning a harpsichord one often recognizes a change of the timbre during the fading sound. It seems that the sound switches between the different overtones. The diagrams in fig. 5 show the time-frequency view of the partials 1 to 20 above and the corresponding time-magnitude view below. The magnitude axis ranges from -60 to 0 dB. The time axis comprises 3.4 seconds (0.2 s per grid line). After 0.85 s the magnitude of the 2nd partial (orange) overtakes the 1st (red). After 3.2 s the magnitude of the 3rd partial (yellow) overtakes the 2nd. The 4th partial (green) is coming up and keeps on, whereas the 5th (cyan) disappears quickly. In contrast, the 6th and the 7th partials (light blue and gray-blue) are strong and fade out almost linear. The 11th (again red) rises just in the time slice where the 2nd is strongest. These measuring results are concordant with the calculations of the physical models by [Välimäki et al., 2004] or [Penttinen, 2006].

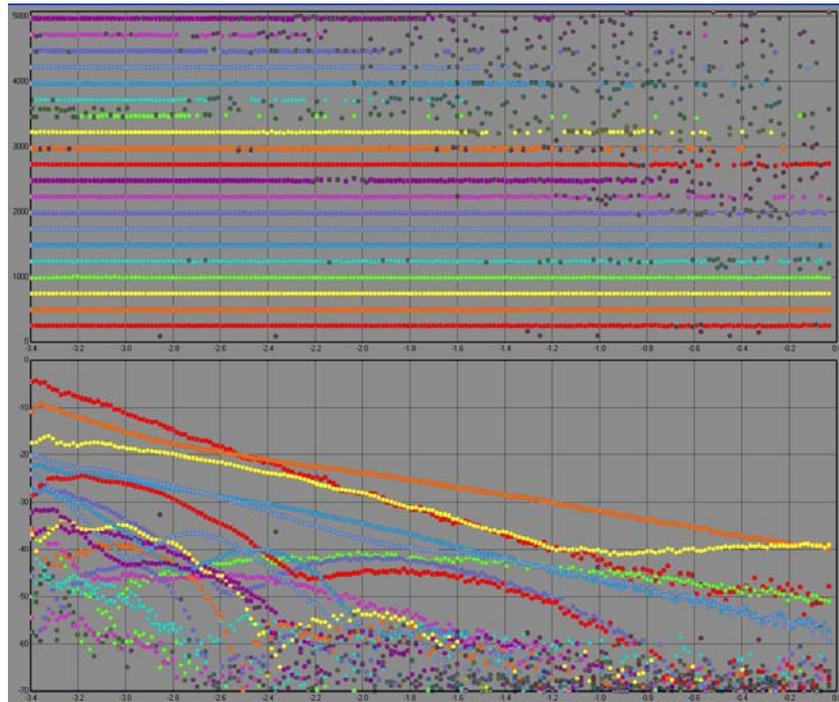


Figure 5: Time-frequency view (above) and time-magnitude view (below) of the harpsichord tone c' (keystroke exactly at the left side)

A similar effect can be observed in fig. 6 on a cello tone after the end of the bowed excitation at 0.7 s. The time scale is the same as in fig. 5. It is remarkable that the 3rd partial is out of tune in the bowed tone and thus yellow-grey coloured. During the fading period of the sound the 3rd partial becomes harmonic (bright yellow) while the 2nd partial (orange) turns slightly upward.

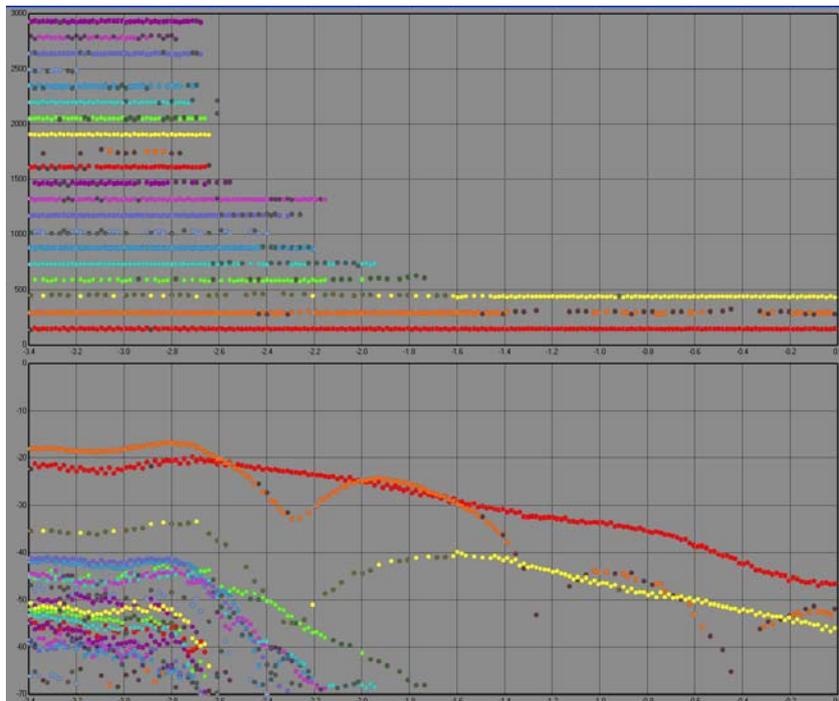


Figure 6: Time-frequency view (above) and time-magnitude view (below) of the cello tone d' (end of bowed sound between 3rd and 4th grid line)

Noisiness of voiceflutes

The noisiness in the sound of musical instruments is a never ending issue and discussion point among musicians. On the one hand hours of practising are being spent to get rid of it, but a completely noiseless sound will be perceived as sterile or “dead”. Noise belongs to the sound as well as the harmonic spectrum. Well trained players should be able to control it and use it where they like to.

Recorders have noise more or less “built in” by the maker. We measured two different voiceflutes (baroque tenor recorders in d) by different makers. Fig. 7 and fig. 8 show the frequency-magnitude view of the tone a’ on each instrument. This view is determined for sustained sounds. To give an impression of short time changes the last few time frames are displayed as a darkening trace behind the actual data. *Prisma* can show the projections of the time-frequency-magnitude space on the three coordinate planes on one screen simultaneously.

Fig. 7 shows the tone a’ on the instrument with the clear, bright timbre. The frequency scale increases by 2000 Hz per grid line. The 1st partial is the red dot in the upper left corner. The higher partials decrease up to the 22nd (3rd orange) at about -55 dB which is the highest to overtop the noise displayed as grey dots. According to the MPEG-7 features [Kim, Moreau, Sikora, 2005] we estimate the upper limit of harmonicity around 9500 Hz. The same note on the darker, noisier instrument is shown in fig. 8 with the same scaling as in fig. 7. Much more grey dots at the bottom represent the noise. The upper limit of harmonicity is here defined by the 14th partial (2nd green) at about 6000 Hz. Note that this is not an assertion about the quality of the instrument: In no way it can be said that the noisy recorder is worse than the clear one!

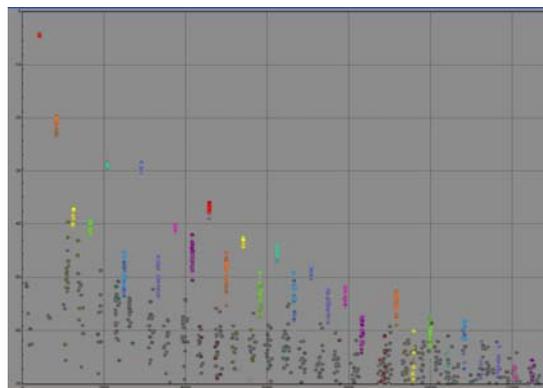


Figure 7: Frequency-magnitude view of the tone a’ on a voiceflute with little noise

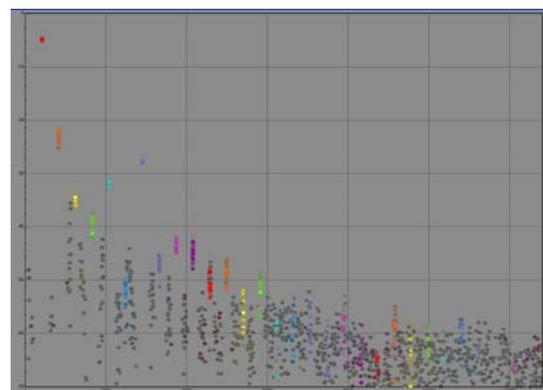


Figure 8: Frequency-magnitude view of the tone a’ on a voiceflute with more noise

CONCLUSIONS

The *Prisma* sound analysis system has been designed as a multi-functional, portable real-time measuring tool. Its practical use will be in instrument making, documentation of historical instruments in museums and teaching musicians. The real-time operation allows to observe the analysis results immediately. Further investigations can be made with stored sound and analysis data. The program is running on a standard laptop computer under Windows XP and does not require special equipment. For high precision measurements we recommend a measuring microphone and an external sound card.

The *Prisma* software can be downloaded freely from www.prisma-music.ch. This website provides further examples and information about the software and our actual work.

REFERENCES

- Desainte-Catherine, M.; Marchand, S. (2000). "High Precision Fourier Analysis of Sounds using Derivatives," *J. Audio Eng. Soc.* 48, 654-667
- Kim, H.-G.; Moreau, N.; Sikora, T. (2005). "MPEG-7 Audio and Beyond", John Wiley & Sons Ltd., Chichester GB
- Medan, Y.; Yair, E.; Chazan, D. (1991). "Super Resolution Pitch Determination of & Speech Signals", *IEEE Trans. Signal Processing*, 39, 40-48
- Penttinen, H. (2006). „On the Dynamics of the Harpsichord and Its Synthesis”, *Proc. of the 9th Int. Conference on Digital Audio Effects (DAFx-06), Montreal, 115-120*
- Segal, M.; Akeley, K. (2004). "The OpenGL Graphics System: A Specification", Silicon Graphics, Inc.
- Välämäki, V.; Penttinen, H.; Knif, J.; Laurson, M.; Erkut, C. (2004). „Sound Synthesis of the Harpsichord Using a Computationally Efficient Physical Model", *EURASIP J. on Applied Signal Processing* 2004:7, 934-948