

A Lightweight Configurable Signal Analysis Toolset for Multiple Applications

Christoph Lauer, Norbert Reithinger
German Research Center for Artificial Intelligence GmbH
Stuhlsatzenhausweg 3
D-66123 Saarbrücken, Germany
{clauer,bert}@dfki.de

ABSTRACT

In this paper we describe a lightweight and highly configurable toolset for the analysis of sound files. While there exist numerous solutions we needed a system that is modular, expandable, that runs on various platforms and operating systems, and that can be embedded in other programs. Using Java and its rich library of signal handling and data visualization components, SONOGRAM provides us with the flexibility needed for various projects in the area of data annotation, visualization, and multi-modal interface implementation. Currently, seven signal analysis and visualization algorithms are implemented. Adding new algorithms is straightforward, and advanced visualizations like three-dimensional perspectograms can be quickly implemented.

1. INTRODUCTION

For various applications and settings, it is favorable to have a visualization of the speech signal at hand. There are various free and commercial systems available that provide elaborate analyzing capabilities. In our year-long research in intelligent user-interfaces, however, we especially felt the need for a visualization tool that provides the following features to view various aspects of a speech signal

- Easy integration in various applications on various platforms
- Rich visualization options for the speech signal
- Easy extensibility to add new analysis modules

The current available speech analyze toolsets like Praat from the University of Amsterdam [1] or SpeechStation from Sensimetrics (<http://www.sens.com/>) lack the plug-in ability for environments like dialogue systems or annotations programs, or they are limited to a certain platform.

In this paper we present SONOGRAM, a Java application that fulfils our needs. Initially it was incorporated in multimedia annotation tools like ANVIL [4] or the NITE (<http://nite.nis.sdu.dk/>) NXT workbench. Further applications are in multimodal dialogues systems like SmartKom (<http://www.smartkom.org/>) or Miamm (<http://www.miamm.org/>)

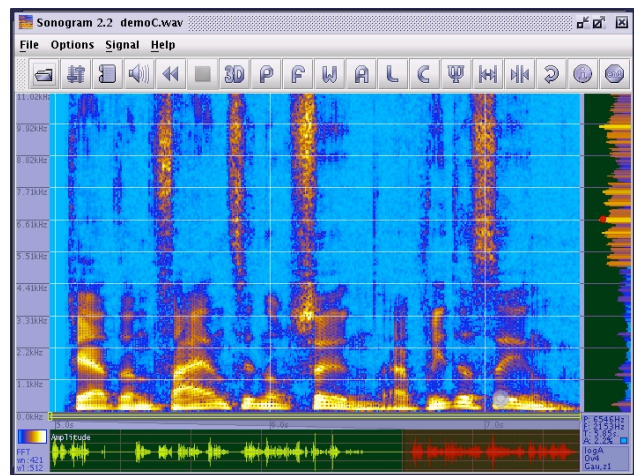


Figure 1: The SONOGRAM visible speech main window

2. THE ARCHITECTURE

In figure 2, the general architecture of SONOGRAM is shown. The central control module gets input either from a file interface or through the plug-in interface from the embedding program. Since SONOGRAM is written in Java it is able to process all media types that are accessible through the Java media framework.

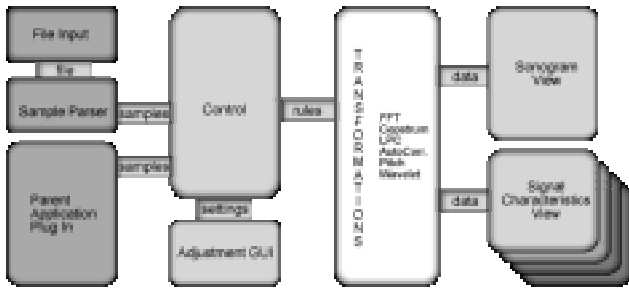


Figure 2: The Architecture

The processing and transformations take place in the transformation module. It is a collection of Java classes that provide currently seven signal transformation algorithms (see below). The results are visualized in two different types of windows. Figure 1 shows SONOGRAM's main window. It displays the frequency view of the signal, the waveform, and, to the right, the energy for the transformation window the cursor is currently pointing at. Below the frequency display you can select an interval of the signal to zoom into.

Besides the main window, for each special signal analysis algorithm, the results are displayed in separate windows; In figure 3 all currently implemented 7 analysis algorithms display the respective characteristics of the signal.

In addition to the basic controls on top like, e.g. playing the signal, the user has access to numerous parameters of the basic processing algorithms, and to control the display. Figure 4 shows the general adjustment dialogue of SONOGRAM for the LPC transformation, where basic properties like LPC coefficients can be adjusted. Of course, every algorithm parameter change immediately initiates a new determination of the signal characteristic presentation and is immediately visible.

The transformation layer uses the object-oriented features of Java. The architecture of the algorithm API allows a fast implementation of additional algorithms that are not implemented yet. All algorithms are children classes of the *SonogramAlgorithm* abstract interface class definition. They exchange the time and frequency domain based signal vectors using access functions.

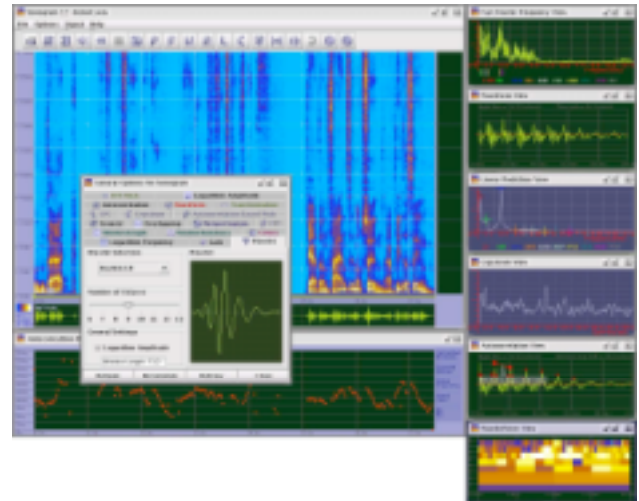


Figure 3: Sonogram with the analysis windows

The Perspectogram generation (see below) is based on the Java3D library that is fully supported on the most hardware platforms. The media file parser is based on the Java Media Framework (JMF) and supports the most common audio and video file formats.

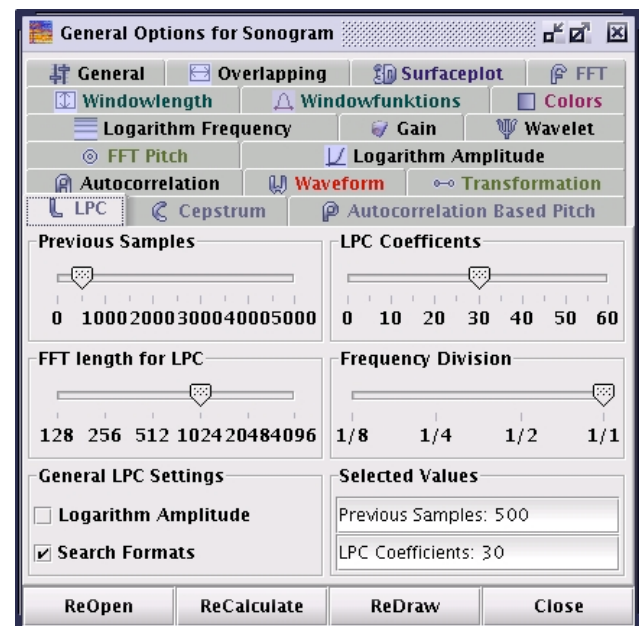


Figure 4: The General Adjustment Dialog

If SONOGRAM is used as a plug-in, the parent application talks with SONOGRAM via the Java Native Interface. The parent application sends the signal, timing information, and, if necessary, transformation parameters. The control module of SONOGRAM will then generate the spectrogram and display the selected information.

The toolset can be accessed from any platform where Java is available. The communication interface to other programming languages is realized with the Java Native Interface, which has a well-defined API to control the modules.

Java has a reputation of not being fast enough for time-critical applications. While this is partly true, the processing speed of SONOGRAM is good enough for most purposes. The signal-duration to processing-time ratio is approximately 1 on a 1 GHz PC running Linux.

3. ALGORITHM COLLECTION

Currently, we have implemented the following seven signal transformation algorithms, using the SONOGRAM transformation API:

- Fast Fourier Transformation: We use the standard Radix-2 FFT algorithm [5] [2]. The Radix-2 algorithm requires an input sample vector with the length of exactly a power of two which is the window length. The window length of the transformation can be configured in the settings dialog.
- Linear Predictive Coding: The implementation is based on a polynomial interpolation and can be scaled by four parameters [5]:
 - The number of samples for the prediction;
 - The degree of the polynomial for the interpolation;
 - The power of 2 for the final fourier transformation;
 - The frequency division representation.
- Autocorrelation: The autocorrelation algorithm is based on the autocorrelation described in [8]. The wavelength of the smoothed autocorrelation periods is used for the estimation of the first formant frequency similar to the pitch detection implementation.
- Cepstrum: For the cepstrum analysis we use the default definition for the cepstrum as the inverse of the logarithmic Fourier spectrum of the time domain based signal. The maximum of the queffrenz the cepstrum transformation computes is similar to the wavelength of the pitch. However the autocorrelation based pitch estimation delivers a steadier formant tracking as the cepstrum transformation algorithm. The window length of this algorithm part is adjustable in the settings dialog.
- Pitch Detection: We use the autocorrelation algorithm based pitch estimation which calculates the F0 progression over time for the parameterized window length and sum loop length. Apart from the autocorrelation, the pitch

can be extracted from the maximal frequency the FFT computes.

- Wavelet: The orthogonal wavelet transformation algorithm is based on the standard pyramid algorithm described in the numerical recipes [5]. The window length of the wavelet transformation correlates with the selected number of octaves adjusted for the transformation. Available filter coefficients are Daubechies (in degree 2 (Haar Wavelet), 4, 6, 8, 10, 12, 14, 16, 18, 20), Vaidynathan (in degree 24), Coifman (in degree 12, 18, 24, 30), and Beylkin (in degree 18). For more detailed information of the wavelet transformation see [6] and [7].

Technically, the algorithms realize the abstract class interface definition of the transformation API, which allows the fast additional implementation of algorithms. The visualization of the transformations results needs more programming. However using the visualization components of the already implemented algorithms this requires usually no completely new development, but only an adaptation to the requirements of the new algorithm.

4. THREE-DIMENSIONAL PERSPECTOGRAM

The power of the approach we used, namely using the component architecture and exploiting the full power of Java shows in the implementation of the three-dimensional Perspectogram. The Perspectogram shows the 2D spectrogram displayed in the main window as three-dimensional surface plot similar to the classical waterfall plot. The z-axis displays the energy distribution for the frequencies at a certain time point in the signal. The experimenter thus has a direct view of the signal's characteristics. Additionally, he is able to scale the position, zoom in and out and to change the viewpoint using the mouse. Again, the dimensions and parameters of the Perspectogram are adjustable in the options dialog.

The implementation is a straightforward extension of the sonogram shown in the main application window in the three dimensional room. The Perspectogram can be grasped with the mouse and observed from all directions

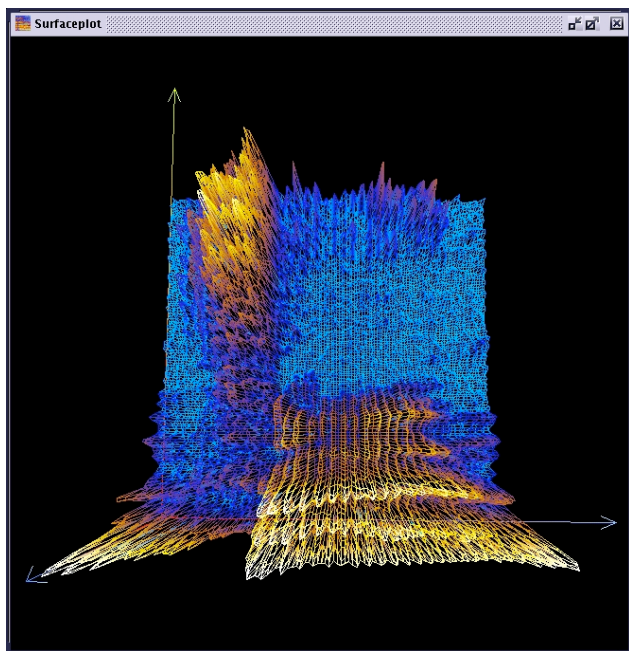


Figure 5: The Perspectogram

6. SUMMARY AND OUTLOOK

We presented SONOGRAM, a flexible toolkit to visualize the features of a signal. It is based on a modular architecture, implemented in JAVA. This enabled us to implement straightforwardly several signal transformation algorithms and to display the results user friendly. The decision to use JAVA as implementation platform was correct for our needs. It provides a rich, cross-platform collection of data import and visualization tools. Additionally, exploiting the object-oriented approach, the implementation of new algorithms is reasonably fast. The only drawback is the processing time, which is currently sufficient for most applications we use SONOGRAM for. In the future, even more efficient Java compilers and run-time systems will close the gap to, e.g., C-based programs

Future extensions to SONOGRAM will be Jitter and Shimmer algorithms for the microstructural analysis of the time and the frequency domain as well as additional pitch analysis algorithms like cross correlation. Another useful feature will be the animated real times analysis of streaming media.

7. ACKNOWLEDGEMENTS

The research described in this paper was partly funded by the European Union within the IST project NITE (Natural

Interactivity Tools Engineering, IST-2000-26095). The responsibility for the contents lies with the authors.

8. REFERENCES

- [1] Boersma Paul, Weenink David, *Praat, a System for doing Phonetics by Computer, version 3.4*. Institute of Phonetic Sciences of the University of Amsterdam. 1995
- [2] Brigham, E. Oren, *The Fast Fourier Transform and Its Applications*, Prentice-Hall PTR, Englewood Cliffs, 1988
- [3] Jurafsky Daniel, H. Martin James, *Speech and Language Processing, An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, Prentice Hall PTR, Englewood Cliffs, 2000
- [4] Kipp Michael, *Anvil - A Generic Annotation Tool for Multimodal Dialogue*, Proceedings of Eurospeech, Aalborg 2001
- [6] Lauer Christoph, *Student research project - Signalanalyse mit Wavelets*, Hochschule für Technik und Wirtschaft des Saarlandes, Saarbrücken, 2000
- [7] Lauer Christoph, Harald Wern, *Diploma Thesis - Akustische Datenkompression mit Wavelets im Vergleich zur klassischen Fouriertransformation*, Hochschule für Technik und Wirtschaft des Saarlandes, Saarbrücken, 2001
- [8] Press William H., Teulosky Saul A., Vetterling William T., Flannery Brian P., *Numerical Recipes in C, The Art of Scientific Computing, Second Edition*, Cambridge University Press, Cambridge, 1992
- [9] Roads Curtis, with John Strawn, Curtis Abbott, John Gordon, and Philip Greenspum, *The computer musics tutorial*, MIT Press, Cambridge Massachusetts, 1996.
- [10] Schukat-Talamazzini Ernst Günther, *Automatische Spracherkennung, Statistische Verfahren der Musteranalyse*, Vieweg Verlag, Braunschweig, 1995
- [8] Xuedong Huang, Alex Acero, Hsiao-Wuen Hon, *Spoken Language Processing: A Guide to Theory, Algorithm and System Development*, Prentice Hall PTR, Englewood Cliffs, 2001