MACAQUE – A TOOL FOR SPECTRAL PROCESSING AND TRANSCRIPTION

Georg Hajdu
Center for Microtonal Music and Multimedia (ZM4)
Hamburg University of Music and Theater (HfMT)
Harvestehuder Weg 12
20148 Hamburg

georg.hajdu@hfmt-hamburg.de

ABSTRACT

This paper describes *Macaque*, a tool for spectral processing and transcription, in development since 1996. *Macaque* was programmed in *Max* and, in 2013, embedded into the *MaxScore* ecosystem. Its GUI offers several choices for the processing and transcription of SDIF partial-track files into standard music notation. At the core of partial-track transcription is an algorithm capable of "attracting" partial tracks (and fragments thereof) into single staves, thereby performing an important aspect of "spectral orchestration."

1. INTRODUCTION

Macaque is a component of the *MaxScore* notation software package for Max [1] allowing the transcription of analysis data in the Sound Description Interchange File Format (SDIF) into standard music notation. It has a long history dating back to March 1996 when, during a ZKM residency in Karlsruhe, Germany, I tried to recreate the workflow I used for my doctoral work at UC Berkeley's Center for New Music and Audio Technologies (CNMAT). At CNMAT, I took advantage of the analysis component of the their additive synthesis tool (CAST) running on a Silicon Graphics Indigo computer [2][3]. In contrast, the software available to me in Karlsruhe consisted of an application called *Lemur* running on the classic Mac OS for partial-tracking analysis as well as *Finale* by Coda (now MakeMusic) for music notation. Lemur implemented the McAulay and Quatieri algorithm [4] capable of modeling non-harmonic and polyphonic sounds [5] and was further developed into *Loris*, an "Open Source sound modeling and processing software package based on the Reassigned Bandwidth-Enhanced Additive Sound Model" [6][1]. I used *Lemur* for partial-tracking analysis and a *Max* patch to translate the analysis data from binary into text format and, eventually, into a MIDI file. Once imported into *Finale*, the files were exported in *Enigma* format—*Finale's* file exchange format until it was superseded by MusicXML. The *Enigma* format (despite its name) allowed me to alter the appearance of my scores by changing the cryptic code in specific locations. To this aim, I developed a number of *Max* patches. For instance, in the first scene of the second act of my opera *Der Sprung – Beschreibung einer Oper* [7] I used the MIDI velocity information of the transcribed note events to alter the size of their note heads so that the sight-reading musicians had instantaneous visual feedback pertaining to the dynamics of the music to be performed. In other instances, I have used a technique called "velocoding" to encode microtonal pitch deviation in eighth-tone resolution into the velocity part of a MIDI note-on message to modify the Enigma file in such ways that the resulting score displayed the corresponding pitch alterations. Another early example of using *Macaque* is my piece *Herzstück* for two player pianos from 1999 which premiered at the Cologne Triennale in 2000. This piece was written for two of Jürgen Hocker's instruments which he also used to tour Conlon Nancarrow's compositions for player piano [8]. These instruments had been retrofitted with a mechanism allowing them to be controlled via MIDI. In my piece, the two pianos were to "speak" the eponymous comical dialog by Heiner Müller, once dubbed the world's shortest theatre piece, with its length well below a minute. I used an audio recording by the *Berliner Ensemble* and also translated the background noises such as the frantic applause at the end.

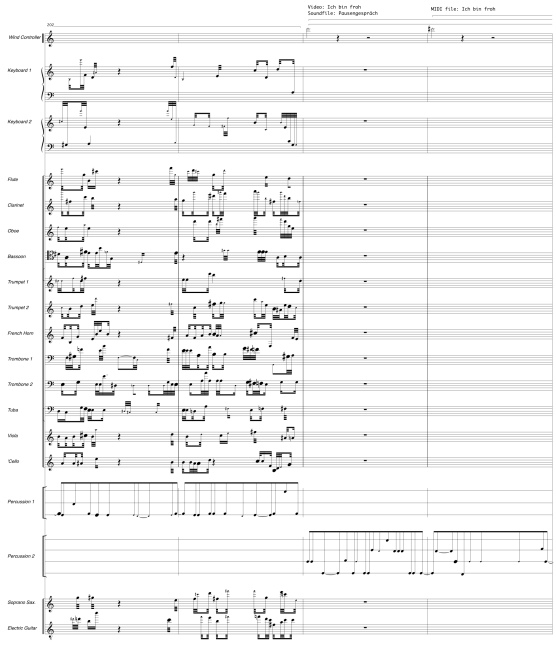

Figure 1. Excerpt from the first scene of the second act from the author's opera *Der Sprung – Beschreibung einer Oper*.

[1] *Macaque* was named after another simian whose name connotes the name of the platform it was developed on as well as the tongue-in-cheek reference to "aping" real sounds with additive or instrumental resynthesis.

In the early 2000s, *Macaque* went through several steps until it reached its current incarnation, among these were:

- Adaption of the SDIF format co-developed by IRCAM and CNMAT
- Switch to SPEAR and AudioSculpt as a source for SDIF files
- Implementation of spectral transforms and time stretching
- Development of a collapsible GUI with three separate panes
- Integration into the MaxScore ecosystem

2. SPECTRAL COMPOSITION AND ORCHESTRATION

At the core of *Macaque* is a technology intelligently assigning partial tracks to *event tracks* (see section 3.3). Partial-tracking analysis of complex sounds typically produces more tracks than an ensemble of musicians can handle. They typically either exceed the number of available musicians or the playable range of their instruments. Therefore, files generated with SPEAR should be prepared in advance. These preparations involve:

- Setting appropriate values in SPEAR's Sinusoidal Partials Analysis window (**Figure 2**) depending on the source type (instrumental sounds, music, noise, speech)
- Defining a cut-off frequency and removing partials outside the range with the Frequency Region Selection tool (typically 3000 Hz)
- Removing short partial tracks (typically <=0.2")
- Deleting soft partial tracks or "false" tracks consisting of noise
- Manually editing partial tracks where a signal has fused with noise

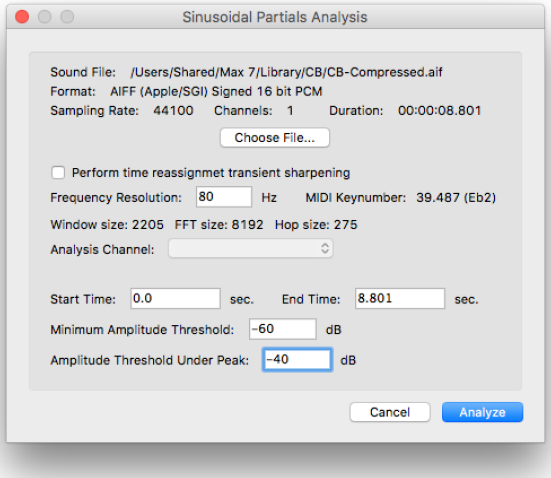

Figure 2. It is crucial to start with the right settings in the Sinusoidal Partials Analysis window before analyzing an audio file in SPEAR.

Still, this may not be enough to sufficiently reduce the number of tracks. In section 3.3, I will therefore describe an algorithm that "attracts" separate partial tracks into an instrumental staff and thereby performs, on a rudimentary level, a task which can be called *Spectral Orchestration*.

There have been a number of projects by other composers and developers tackling aspects of spectral orchestration. Those known to me include the works by the French spectralists, the software Clarence Barlow's developed for his piece *Am Januar am Nil* (1980) in which nonsense sentences are "spoken" by an ensemble, the piece *Speakings* by Jonathan Harvey for which the Matlab-based software *Orchidée* was developed at IRCAM [9]; more on this in a paper by Aurélien Antoine and Eduardo R. Miranda [10]. Other projects include the *soundalikes* by Michael Iber, the text compositions by Peter Ablinger, as well as OpenMusic [11] and the bach/cage libraries [12] capable of converting SDIF files into music notation.

3. WORK FLOW

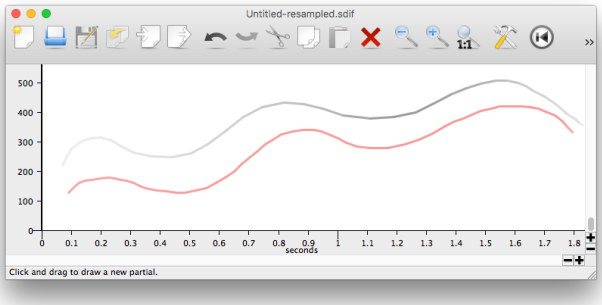

Figure 3. An example of partial tracks in SPEAR. These tracks were actually drawn by hand.

Including *Macaque* into the *MaxScore* ecosystem has simplified the workflow to a great extent and allowed me to work mainly in the Max environment. The following sections will give an overview of the crucial steps from SDIF import to score generation.

3.1 Importing from Spear

SDIF files such as the one displayed in **Figure 3** should be exported from SPEAR as "SDIF 1TRC – Exact Interpolated". *Macaque* can be conveniently accessed from within a patch called *Macaque Environment*, which is part of the *MaxScore* ecosystem and also comprises an instance of the *MaxScore* editor, a *Macaque* sound file recorder (for resynthesized SDIF files) and two modules for microtonal and multitimbral playback.

3.2 The GUI

Macaque sports three panes and four tabs for the top pane (**Figure 4**). The default view displays (i) the partial tracks in the top pane with the Transcribe button underneath (triggering the transcription of partial tracks into notation), (ii) the spectral content of a vertical time slice (spectral frame) in eighth-tone notation in the central

pane as well as (iii) curves for centroid (green) and sum of amplitudes (black) in the bottom pane.

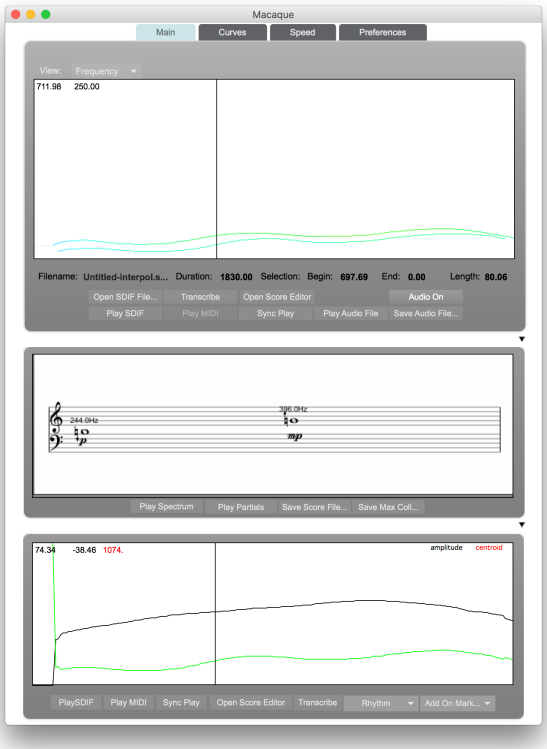

Figure 4. The *Macaque* GUI with its collapsible panes

These curves serve as the basis for event detection and markup, as we will see in section 3.5. A second Transcribe button triggers event transcription according to the markers created by the user.

The other three views of the top pane display:

- Break-point functions for spectral transforms (**Figure 5**)
- A tempo curve for time stretching/compression and
- A preferences pane with over 15 parameters affecting the outcome of the transcription (**Figure 6**)

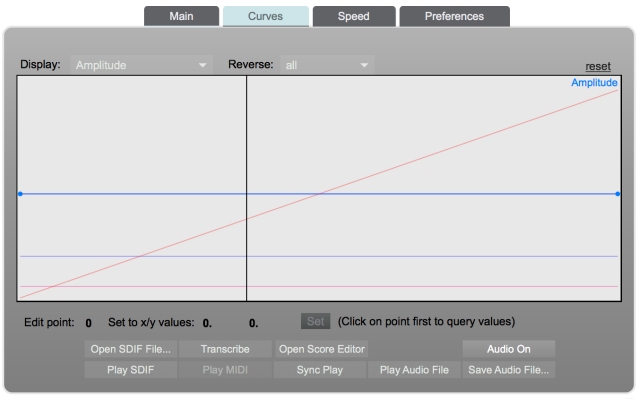

Figure 5. Break-point functions for spectral transforms

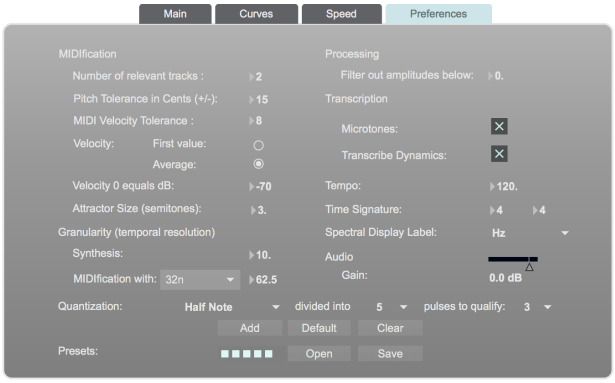

Figure 6. Preference pane with parameters affecting the outcome of the transcription

3.3 Partial-track transcription

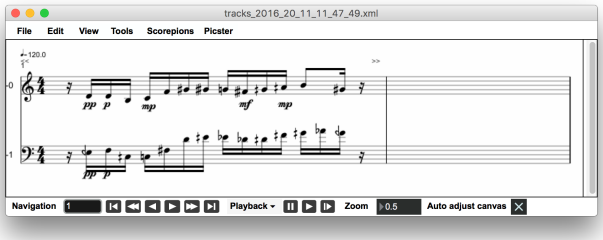

Figure 7. Transcription of the partial tracks from **Figure 3**

Macaque relies to a great extent on the CNMAT sdif objects (sdif-buffer, sdif-info, sdif-ranges, sdif-tuples) [11]. Upon opening an SDIF file in 1TRC format and loading stream number 0 (higher stream numbers are currently not supported) into the sdif-buffer, relevant information about the file is extracted and the spectral content displayed in the top pane by reading the data from the SDIF matrix contained in the buffer.

3.3.1 MIDIfication

Pressing the transcribe button will now pass the spectral data to the transcriber, at the time interval defined as MIDIfication in the Granularity preference section. This interval is calculated by taking current meter, tempo and beat subdivision settings into consideration (**Figure 7**). Note that the quantizer offers another *beat subdivision scheme* [14] which can either be aligned with the MIDIfication interval or not. When aligned the subdivision is regular, if not the subdivision is irregular and notes may be lumped together such as in **Figure 8**.

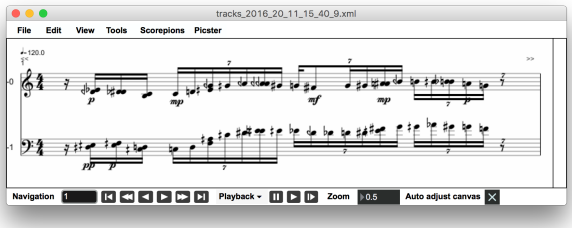

Figure 8. Transcription of the same partial tracks with misaligned MIDIfication and beat subdivision scheme settings. This may or may not be a desired effect.

After applying the spectral transforms (see section 3.3.3) the partial tracks are resampled according to their index in a 32-bit Jitter matrix. Each track is converted and analyzed according to Pitch and MIDI Velocity tolerance thresholds, i.e. the analysis looks for leaps in the resampled values exceeding a given threshold.

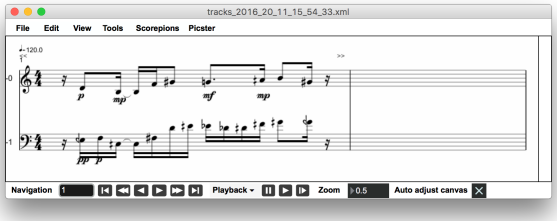

Figure 9. Transcription of the same partial tracks with vastly different Pitch and MIDI Velocity tolerance thresholds.

If a value is greater than the threshold value, a new event is assumed and the events collected in a Max coll (see **Table 1**). Each track now consists of a track velocity value (based either on the first collected amplitude value or an average of all amplitude values), an average track pitch value and four values for each event consisting of time tag (in MIDIfication intervals), event frequency, event velocity and duration (in MIDIfication intervals).

#	Events in track
0	0.14899 65.489636 1 62.14 31. 2 3 59.07 48. 2 5 64.84 53. 1 6 68.16 58. 2 8 67.11 65. 3 11 69.62 61. 1 12 71.21 51. 2 14 67.95 51. 1;
1	0.050958 59.706389 1 51.49 25. 1 2 53.14 33. 1 3 48.67 38. 2 5 53.91 38. 1 6 61.9 40. 1 7 64.54 39. 1 8 63.05 40. 1 9 61.18 42. 1 10 62.37 42. 1 11 65.33 41. 1 12 67.69 41. 2 14 66.28 41. 1;

Table 1. Event collection. Each track is represented by average velocity, average pitch and a sequence of four values denoting time tag, frequency, velocity and duration for each individual event.

3.3.2 Event Attractor

These data serve as the basis for an algorithm assigning these events to event tracks. It works as follows: The track velocity serves as a measure for its relevance; the louder the track the more relevant. All tracks are indexed according to this measure. For the first partial track (the most relevant track), all events are written to another Jitter matrix and now serve as an *attractor* to events which exist in the other tracks. If the events of the next track are close enough in pitch (defined by *Attractor Size* in the *Preference* pane) and can be inserted into empty regions of the current event track, they will be written to this track, otherwise a new event track will be created (Figures 10-12). This process is iterated until all events have either been assigned to event tracks or discarded, the maximum count being 32.

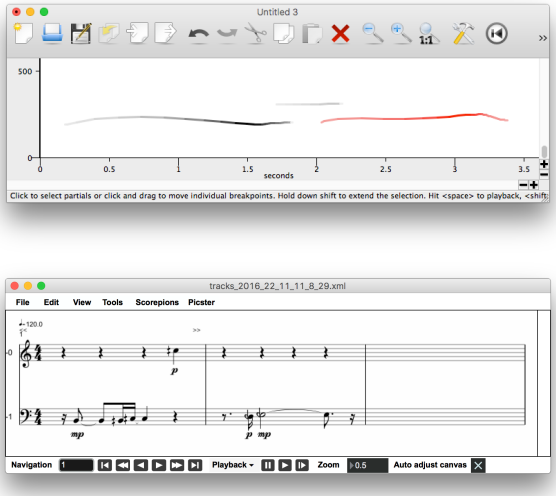

Figure 10. Transcription of overlapping partial tracks yields separate staves.

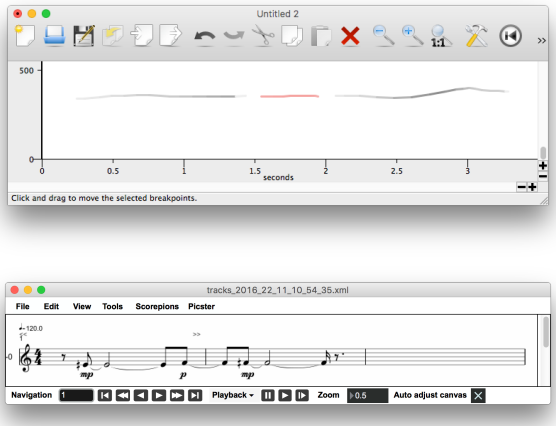

Figure 11. Transcription of consecutive partial tracks within attractor range yields one staff.

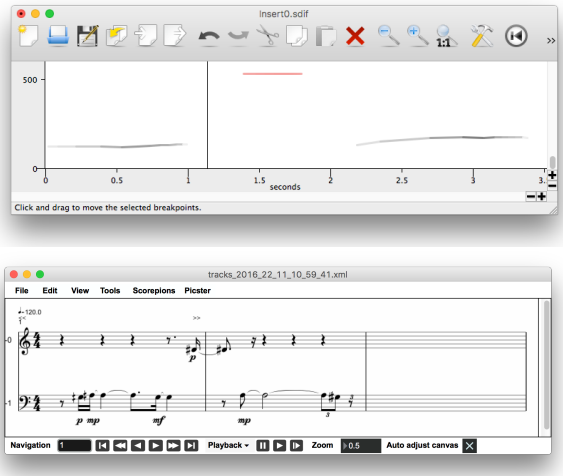

Figure 12. Transcription of consecutive partial tracks outside attractor range yields two staves.

The next steps involve sorting event tracks according to their average pitch as well as converting time tags and durations into their respective values in seconds. This is where time stretching and compression is applied (**Figure**

14). Finally, these values are fed into the *MaxScore* transcriber and displayed in standard notation.
Once transcribed the original SDIF file and its companion score file can be played back in sync by pressing the "Sync Play" button.

3.3.3 Spectral transforms
Macaque can apply time-variant transforms to spectral data. These transforms can be set by changing break-point functions (BPFs) for *amplitude*, *trajectory*, *spectral stretch*, *reference frequency* and *transposition*. They are also being applied to the playback of the SDIF file. I used Emmanuel Jourdan's *ej.function.js* JavaScript object capable of drawing multiple BPFs on top of each other and sharing its curves with an efficient Java object called *ej.fplay* for real-time processing.
While the terms *amplitude* and *transposition* don't need further elucidation, I'd like to explain the function and meaning of *trajectory* and *spectral stretching*. *Trajectory* refers to the path playback and transcriber take through the SDIF file. A straight upward line causes the sample to be played regularly, i.e. forward, a straight downward line causes the sample to be played backwards. By using any number of break points, playback and transcription can be broken up into forward and backward segments. Spectral stretching is performed according to the formula given by Mathews and Pierce [15]. It requires the partial index (defined as the ratio between partial and reference frequency), a pseudo-octave (or stretch factor; 2 = no stretch) and reference frequency (or fundamental) as inputs.

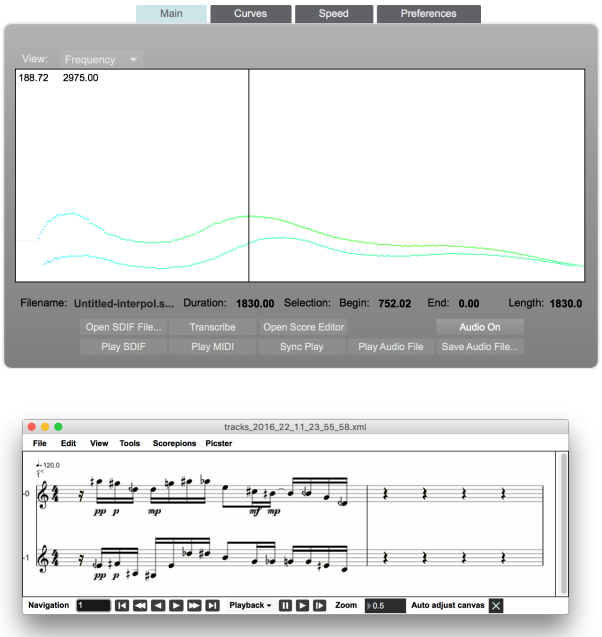

Figure 13. The transcription of the same file with spectral-stretching applied. The stretch factor is 2.95 at the beginning shrinking linearly to 1.68 over the length of the file.

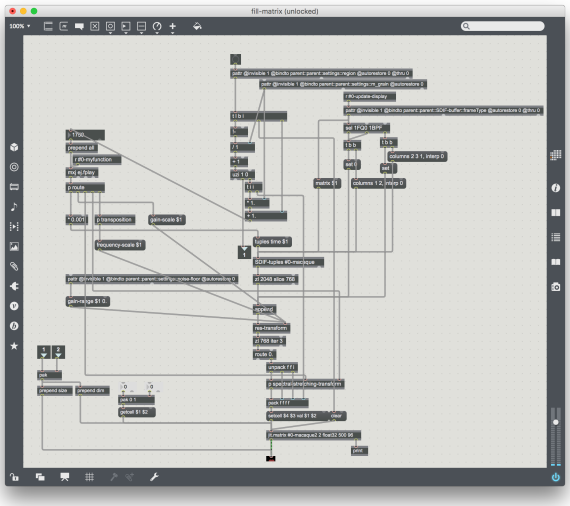

Figure 14. The subpatch in *Macaque* where spectral transforms are applied. Note the use of the *ej.fplay* object which shares the break-point functions of the *Curves* pane.

3.3.4 Tempo curve
Another editor can be used to warp time according to a time-variant tempo curve, i.e. portions of the sample can be sped up or slowed down. This tempo curve will then be applied to transcription. This is performed by calculating the integral under the tempo curve to obtain the values for duration and onset times of the note events to be used by the *MaxScore* transcriber.

3.3.5 Selection
Playback, transcription and transforms all scale to the selection made in the partial-track pane. This affords the user the possibility to specify select parts of the SDIF file.

3.3.6 Effects processing
Since all transforms are also applied to the resynthesis of the SDIF file, *Macaque* can also be used as an effects processor. Two of my compositions (see Table 2) have taken advantage of this capability.

3.4 Spectral frames

As mentioned before, the spectral slices pane displays the spectral content at the position of the play head. Its partials can be played back at once by holding the space bar or arpeggiated by pressing the "Play Partials" button. They can be saved as a *MaxScore* XML file or a Max coll. In addition, the notes can be copied and pasted into a *MaxScore* score for compositional or analytical purposes.

3.5 Event detection and transcription

The third pane is dedicated to event detection. Here, the term event is applied to the entire analysis file, in contrast to partial track transcription where "event" refers to the detection of significant amplitude and frequency changes within a single track. As automatic event detection poses some challenges [16], I have opted to (i) offer markers for manual event detection (for which the amplitude and

centroid curves lend themselves as guides) or (ii) for the import of markers generated in *AudioSculpt*, whose algorithm performs better than the one employed in an earlier version of *Macaque*. Once an SDIF file has been marked up with on-markers and off-markers, the obtained temporal structure serves as the basis for transcription and further processing. Two adjacent on-markers delineate a time interval within which the spectral frame with the highest sum of amplitudes is searched for. From this frame, the following events can be derived:

- Temporal structure with x-ed note heads.
- f0 pitch, applying the harmonic histogram technique implemented in Mikhail Malt and Emmanuel Jourdan's zsa.fund object
- Lowest partial
- Most salient partial
- Centroid
- The nearest neighbor of the centroid, as the latter is typically not contained in the spectrum
- All partials as a chord (an amplitude threshold can be set in the preference pane to skim off softer partials)

While the markup should ideally follow sharp rises or drops in the amplitude and/or centroid curves, markers can also be set to apply arbitrary rhythms to the spectrum of a sounds (**Figure 15**-**17**).

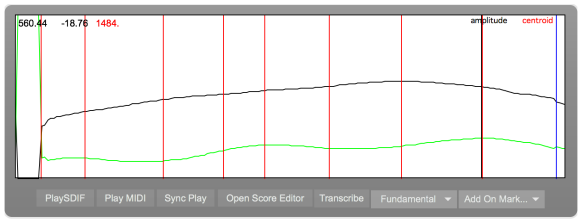

Figure 15. An (arbitrary) markup of the SDIF file from **Figure 3**

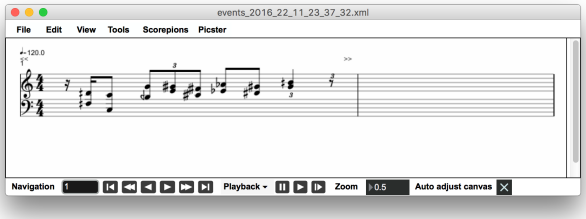

Figure 16. The transcription of the markup displaying the strongest partials within the delineated segments.

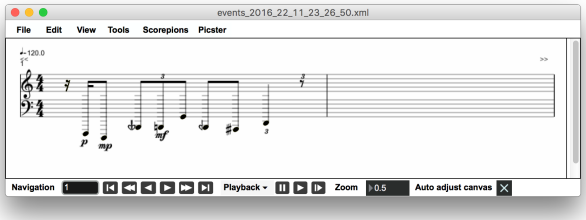

Figure 17. Same markup displaying f0 pitch.

4. COMPOSITIONS

After a hiatus of nearly 10 years during which I mainly focused on networked multimedia performance, I started to create spectral music again in 2009. Since then I have used *Macaque* in the following compositions (see http://georghajdu.de):

Composition	Year	Instrumentation	SRC	PT	SF	MU	ASUP	RSYN
Blueprint	2009	sax, egtr, db, pno, perc, elec	speech, noises	x	x	x	x	
Schwer... unheimlich schwer	2009	bcl, vla, pno, perc, elec	speech		x	x	x	
Swan Song	2011	vc, perc, elec	Beijing opera, noises	x		x	x	x
In ein anderes Blau	2012	sop, bfl, cbcl, vn, va, vc, db, perc, elec	music		x		x	x
noiwont	2014	19-tone trp, elec	speech	x		x	x	
aɪd laɪk tuː meɪk ə ʃɔːt ˈsteɪtmənt	2016	fl, cl, va, vc, pno, perc, elec	speech		x	x	x	

Table 2. Compositions by the author composed with the aid of *Macaque*. SRC = source material, PT = partial-track transcription, SF = spectral frame, MU = Markup and event transcription, ASUP = audio superimposition, RSYN = SDIF to audio resynthesis.

5. CONCLUSION AND OUTLOOK

Emerging from a situation in which I desperately needed to replace software I had relied on during my doctoral work at CNMAT, *Macaque* has become, over the years, a serious tool used by me, and others, for spectral analysis and composition. It has become fairly stable in its feature set for the past 4 years with development mainly focusing on bug fixes and support for 64-bit Max.
However, there are a few areas that are still worthwhile exploring:

- Transcription and notation of glissandi for events belonging to the same partial track
- Implementation of an efficient automatic event detection algorithm with a "rubber-band" tempo curve editor capable of taking tempo fluctuations and microtiming into consideration (see also [17])
- Implementation of a fast method for tempo curve integration
- Zooming
- Improvements of the GUI

Since the code base largely emerged before the release of Max 5, it is tempting to recreate the functionality of the partial-track transcriber and other components in the Max js object as well as improve control of the additive synthesis as a Max gen~ script.

Acknowledgments

I would like to thank the *Behörde für Forschung und Wissenschaft Hamburg* for supporting our research in the framework of its *Landesforschungsförderung.*

6. REFERENCES

[1] G. Hajdu, and N. Didkovsky, "MaxScore – Current State of the Art," in *Proceedings of the International Computer Music Conference*, Ljubljana, 2012, 156-162.

[2] A. Freed, X. Rodet and Ph. Depalle, "Synthesis and Control of Hundreds of Sinusoidal Partials on a Desktop Computer without Custom Hardware," in *ICSPAT (International Conference on Signal Processing Applications & Technology*, 1992, San José, United States. 1992

[3] http://cnmat.org/CAST/

[4] R.J. McAulay and T.F. Quatieri, "Speech analysis/synthesis based on a sinusoidal representation," in *IEEE Transactions on Acoustics, Speech, and Signal Processing,* vol. ASSP-34, no. 4, 1986, 744-754.

[5] K. Fitz and L. Haken, "Lemur: A bandwidth-enhanced sinusoidal modeling system," in *The Journal of the Acoustical Society of America,* vol. 103, 1998, 2756.

[6] http://www.hakenaudio.com/Loris/

[7] G. Hajdu, "Research and Technology in the Opera Der Sprung," in *Nova Acta Leopoldina*, vol. 92, no. 341, 2005, 63-89.

[8] J. Hocker, *Faszination Player Piano - Das Selbstspielende Klavier von den Anfängen bis zur Gegenwart. Kapitel 15 - Conlon Nancarrow und die Renaissance des Player Pianos,* Edition Bochinsky, 2009, 238-267.

[9] G. Nouno, A. Cont, G. Carpentier and J. Harvey, "Making an Orchestra Speak," in *Proceedings of the Sound and Music Computing Conference (SMC),* Porto, 2009, 277-282.

[10] A. Antoine and E. R. Miranda, "Towards Intelligent Orchestration Systems," in *Proceedings of the 11th International Symposium on Computer Music Multidisciplinary Research (CMMR): Music, Mind, and Embodiment*, Plymouth University, UK, 2005.

[11] J. Bresson and C. Agon, "SDIF sound description data representation and manipulation in computer assisted composition," in *Proceedings of the International Computer Music Conference*, Miami, USA, 2004, 520–527.

[12] A. Agostini, E. Daubresse and D. Ghisi, "cage: a high-level library for real-time computer-aided composition," in *Proceedings of the Joint ICMC and SMC conference,* Athens, 2014, 308-313.

[13] M. Wright, R. Dudas, S. Khoury, R. Wang and D. Zicarelli, "Supporting the Sound Description Interchange Format in the Max/MSP Environment," in *Proceedings of the International Computer Music Conference*, Beijing, 1999, 182-185.

[14] N. Didkovsky, "Java Music Specification Language, v103 update," in *Proceedings of the International Computer Music Conference*, Miami, 2004, 742-745.

[15] M. Mathews and J.R. Pierce, "Harmony and Non-harmonic Partials," in *The Journal of the Acoustical Society of America*, vol. 68, 1980, 1252.

[16] N. Collins, "A Comparison of Sound Onset Detection Algorithms with Emphasis on Psychoacoustically Motivated Detection Functions," in *Proceedings of AES118 Convention*, 2005.

[17] http://fab.cba.mit.edu/classes/864.05/people/lieb/lev.html

Webpages all accessed on November 23, 2016.