Interspeech 2022
18-22 September 2022, Incheon, Korea

ema2wav: doing articulation by Praat

Philipp Buech[1], *Simon Roessig*[2], *Lena Pagel*[2], *Doris Mücke*[2], *Anne Hermes*[1]

[1]Laboratoire de Phonétique et Phonologie, UMR 7018, CNRS/Sorbonne Nouvelle, France
[2]IfL Phonetics, University of Cologne, Germany

philipp.buech@sorbonne-nouvelle.fr, {simon.roessig, lena.pagel, doris.muecke}@uni-koeln.de, anne.hermes@sorbonne-nouvelle.fr

Abstract

In this paper, we present *ema2wav*, a software conversion tool for electromagnetic articulographic data, producing multi-channel WAVE files. The data can be converted either by executing a stand-alone Python script or by using a user-friendly GUI. *ema2wav* allows the display and extraction of EMA trajectories as well as data smoothing and the computation of derivatives and Euclidean distances between sensors. A great asset of this converter is that it allows the research community to process EMA data in widespread and easy-to-use open-source programs like *Praat*. It is completely platform-independent and is thus a very promising alternative for e.g., students, teachers and researchers in experimental linguistics who have either limited access to software licenses and/or seek for an easy way to maintain open solutions for their research.

Index Terms: articulation, conversion articulatory data, Praat

1. Introduction

The investigation of the many-to-one relations between articulatory movements and acoustic speech patterns are relevant for various fields in the context of speech communication. Articulation is the source of acoustics [1] and movement patterns of the underlying articulatory dimension can be related to spectro-temporal properties of phonetic entities such as consonants and vowels and as well as prosodic modulations and intonation. Therefore, both articulation and acoustics belong to the measurable physical correlates of speech production encoding complex linguistic structure on different dimensions.

There are several invasive and non-invasive techniques to capture vocal tract movements of the speakers' different subsystems namely the phonatory, respiratory and supraglottal modulations. For oral speech organs, Electromagnetic Articulography (EMA) is widely applied in various kinds of speech production tasks. EMA is a minimally invasive technique [2] allowing the analysis of positions and movements of sensor coils placed on articulators, e.g., the jaw, tongue tip and body as well as the upper and lower lips. There is a variety of different powerful software packages for the display and annotation of EMA data, in particular, *MView* [3], *VisArtico* [4] and the speech database system *emuR* [5].

This paper presents the *ema2wav* converter, that allows converting EMA trajectories into multi-channel files. With *ema2wav* we provide the option to process kinematic signals in programs such as *Praat* [6]. At present, *Praat* is the quasi-standard for the annotation of acoustic speech signals and an integral part of the phonetician's toolkit, as it is freely available, well-maintained, available on different operating systems and can be interfaced with other environments as *R* [7] and *Python* [8]. However, so far *Praat* is restricted to acoustic data without having an option to process kinematic data.

The *ema2wav* converter aims to create an easy workflow for all *Praat* users in experimental linguistics to analyze EMA kinematics and acoustics in one place. It is easy to handle for students, teachers and researchers. Furthermore, it is designed to work platform-independent as it is written in *Python* (works on Linux, Windows and Mac), and it is open-source. Currently, it is under active development and works for data collected with the Carstens' AG500/501 models.

2. Main features

In the current version, *ema2wav* has the following capabilities:

- Supporting the models AG500/501 (will be soon extended to AG 100, 200).
- Extracting positional data for the vertical and horizontal dimension of each sensor.
- Calculating profiles for the first (velocity) and second derivative (acceleration) of each dimension, tangential velocity, its derivative, and the Euclidean distance between two sensors.
- Applying filtering to the data with a moving average/floating mean filter or a Butterworth lowpass filter to smooth the selected contours.
- Exporting the data along with or without the audio signal as WAVE and/or as a CSV file.

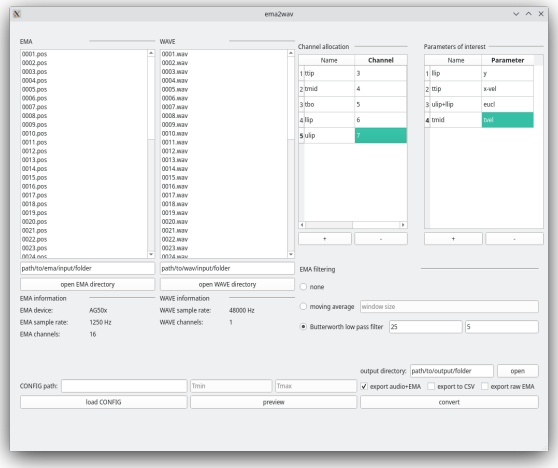

Figure 1: *GUI of ema2wav filled with user input.*

10.21437/Interspeech.2022-10813

3. Presentation

ema2wav is a lightweight and easy-to-use conversion software that allows users to convert EMA data into WAVE files. It extracts not only the position data but also allows calculations, e.g., derivations (e.g., 1st derivative as velocity, 2nd derivative as acceleration, as well as tangential velocities). *ema2wav* is written entirely in *Python* and is thus platform-independent.

The conversion software *ema2wav* is designed to satisfy the needs of users, both technical and non-technical ones, and thus consists of two modules. The first module is a core script containing all routines and functions of the conversion process. It can be imported as a *Python* module in their own script so that users can access the functions for extracting and/or converting EMA data by themselves. The second module is a fully functional graphical user interface (GUI), as shown in Figure 1, that allows users to easily enter all necessary parameters and execute the conversion process without coding by themselves. The resulting WAVE files can then be opened and annotated in *Praat* (Figure 2).

To ensure openness, accessibility and maintainability of the code, only open-source packages are used. These packages include *Librosa* [9], *SciPy* [10], *Mutagen* [11] and *PyQt* [12] for the GUI development and functionality. Although still work-in-progress, a fully functional version of *ema2wav* can already be downloaded from “ https://github.com/phbuech/ema2wav ”.

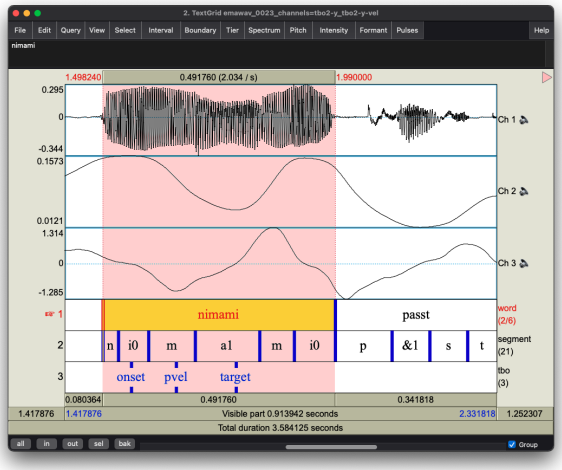

Figure 2: *Screenshot of the Praat editor window showing audio and EMA data (Ch 2 = vertical tongue body position, Ch 3 = tongue body velocity) along with annotations in a TextGrid.*

4. Conversion process

The conversion of EMA data into WAVE files in *ema2wav* consists of two parts, i.e., the user input and the main functionality of the core script. Figure 4 shows the general scheme of the conversion process.

A configuration file in the JSON format is required as the input for the core script. It can be created either directly or indirectly via the GUI. The configuration file serves not only as a device for providing the necessary information for the conversion process, but can also be used for its replication and documentation. It consists of the following information:

- Paths of the POS files from the AG500/501 data (containing all spatial data registered by the articulograph) and the corresponding WAVE files for the input as well as the output directory.
- Specification of the channel names (can be chosen by the user) and their respective channel numbers.
- Parameters that shall be extracted (e.g., x- (horizontal) or y-position (vertical), velocity, acceleration).
- Filters to apply (e.g., moving average filter).
- Export options (WAVE including/excluding the acoustic signal, CSV).

An example of this configuration file is shown in Figure 3.

```
{
        "ema_device_info": "AG50x",
        "include_audio": true,
        "export_to_csv": false,
        "export_raw_ema": false,
        "ema_input_directory": "/ema/input/path/",
        "audio_input_directory": "/wav/input/path",
        "output_directory": "/path/to/output/",
        "channel_allocation": {
                "ttip": 4,
                "llip": 7,
                "ulip": 6
        },
        "parameters_of_interest": {
                "0_ttip": "x",
                "1_ttip": "y",
                "2_llip": "x",
                "3_llip": "y"
        },
        "filter": {
                "moving_average" : 10
        }
}
```

Figure 3: *Example of a configuration file.*

Users have two options to start the conversion process: either by executing the conversion from a *Python* script or by executing the software from a console, where the latter does not require programming skills in *Python*. The first option is to write the configuration file manually. Users can write a custom *Python* script and execute the conversion function with the path to this configuration file as input. The second option is utilizing the GUI where all necessary information must be entered. By starting the conversion via the GUI, the configuration file is saved on the hard drive at the location of the output directory and the conversion function of the core script is called automatically.

After calling the conversion function manually or via the GUI, the EMA data in the POS files are extracted. The values of a sample in the AG500/501 models are stored in sequences of 56, 112 or 168 values, depending on the number of channels supported by the respective AG device [13]. Seven values are recorded for each sample and channel, including x (horizontal), y (lateral) and z (vertical) dimensions, phi and theta as azimuth and polar angles of the sensors, a root mean square value and an empty extra value. The values per sample are thus seven times the number of channels.

The data array in the POS files is split into samples, which are then reshaped into two-dimensional matrices with the rows representing the channels and the columns representing the recorded values. The EMA trajectories of all channels entered

in the channel allocation field of the configuration file are extracted by retrieving the x- and z-values of each sample according to their positions in the matrices defined by the corresponding channel row and value column (Figure 5). It should be noted that it is common practice to label the height dimension as a vertical dimension, which corresponds to the z-dimension in the POS files.

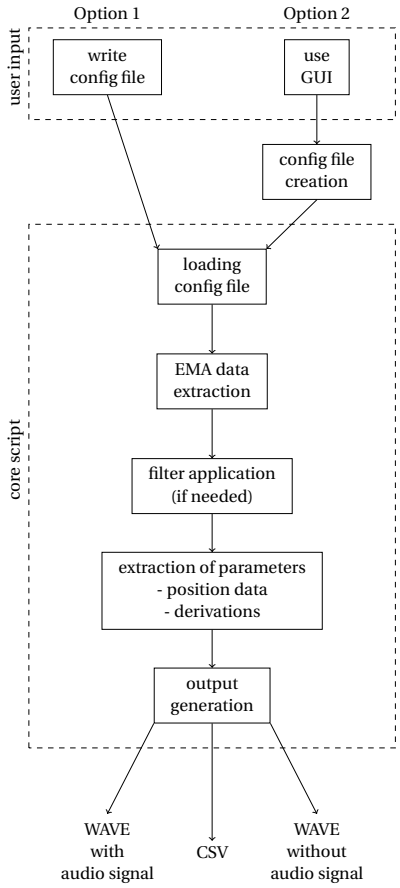

Figure 4: *Scheme of the conversion process in ema2wav.*

After the retrieval of the EMA trajectories, the data may be smoothed/filtered by a moving average or a Butterworth low-pass filter. Based on these extracted (and filtered) trajectories, the parameters that have to be saved into the output are retrieved or processed further for the calculations in the case of derivations, tangential velocity and Euclidean distance.

This data can then be exported as multi-channel WAVE files with or without the corresponding audio signal, and additionally as a CSV file. In the former case, the EMA data is up-sampled to fit the sample rate of the WAVE file as the sample rates for EMA data are significantly lower (usually 250 Hz) than the sample rate of the audio signal (e.g., 48 kHz). This is done by a cubic spline interpolation as implemented in *SciPy* [10]. Figure 6 shows an example of an /as^{S}a/ sequence with the tongue tip trajectory in the vertical dimension with values for each sample and the interpolated trajectory.

If the EMA data is exported without the corresponding audio or as a CSV file, the data is not up-sampled to the audio sample rate. The WAVE file output includes metadata consisting of the channel and parameter description as ID3 tags (generated by *mutagen* [11]).

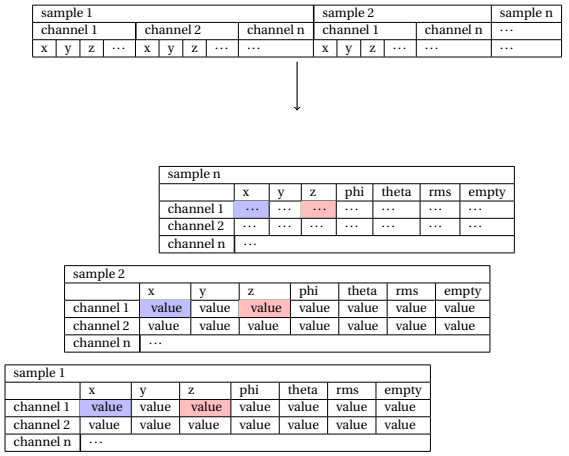

Figure 5: *Scheme of the reshaping of the data array into two-dimensional matrices per sample. Colored cells represent positions in the horizontal dimension (blue) and the vertical dimension (red) for each sample.*

5. Conclusion, limitations and future directions

In this paper, we presented *ema2wav*, a lightweight software package for the conversion of EMA data to multi-channel WAVE files. This converter allows the research community to process EMA data in widespread and easy-to-use and open-source programs, namely *Praat*.

EMA data can be converted either by executing a stand-alone *Python* script or by using a user-friendly GUI. *ema2wav* not only allows the extraction of EMA trajectories but also data smoothing. In addition, it includes options for calculations such as derivations and Euclidean distances between sensors.

The envisioned workflow using *ema2wav* for conversion and *Praat* as an annotation tool is certainly not able to provide as many elaborate features for articulatory analysis as alternative programs like *Mview* or *emuR*. Nevertheless, the solution we presented here comes with a lot of advantages. First, most researchers in phonetics and speech science have a good knowledge of *Praat*, including keyboard shortcuts and *Praat* scripts. Using *Praat* for EMA data eliminates the need to learn a new program. Since *Praat* is so widespread, annotations of the acoustic signals often already exist in the form of TextGrids, either manual annotations or as an output of a forced aligner such as the *Montreal Forced Aligner* [14]. Our solution makes it possible to combine these acoustics annotations with annotations of the EMA signals. Hence, acoustic and articulatory labels are in one place and users do not need to learn new workflows and shortcuts.

Second, *ema2wav* is built entirely with open-source tools and *Praat* itself is an open-source software package. Therefore, this solution requires no expensive software licenses. *Praat* elegantly interfaces with other open-source tools, e.g., *Parselmouth* [15] and *rPraat* [16]. Third, the converter features a user-friendly GUI that makes it accessible to users without deep technical knowledge. At the same time, the software is built with the concept of modularity in mind: The GUI and the core converter are fully separate. In consequence, the converter can be executed as a stand-alone program by users with deeper knowl-

edge on the configuration in JSON, and even be integrated into a larger workflow coded in *Python*. Fourth, with only roughly 300 lines of code, the converter script is compact. In conjunction with the fact that *Python* is a widespread programming language that many developers use, this supports maintainability and hence the longevity of the software. Finally, *ema2wav* and *Praat* are fully platform independent and work on Linux, Windows and Mac computers.

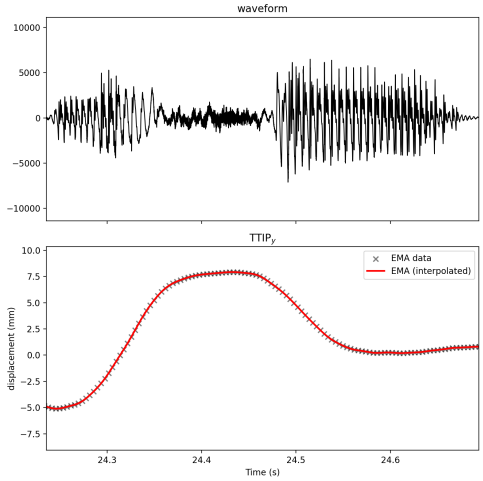

Figure 6: *Waveform (top) and vertical tongue tip trajectory (bottom) for /as§a/ spoken by a Tashlhiyt speaker. Crosses show the EMA trajectory as recorded with an AG501, the red line shows the interpolated curve.*

All this makes the presented solution a promising alternative for many different groups, including students or researchers with limited access to software licenses, and also many other groups of users that seek easy-to-maintain open solutions for their research.

Current limitations relate to the possible input sources and the WAVE files that include both EMA data and audio. Regarding the former, only data collected from the AG500/AG501 models are supported, but an extension for the models AG100 and AG200 is already planned. The latter limitation concerns the possible file size and restrictions in the analysis. WAVE files including both audio signal and EMA data may be large depending on the duration of the recordings, the sample rate of the audio and the number of parameters that have to be extracted and/or derived. For example, a recording of approx. 36 seconds with an audio sample rate of 48 kHz and an EMA sample rate of 1250 Hz consists of a 3.4 MB WAVE file and a corresponding POS file of 19.1 MB file size. If 8 parameters shall be extracted (both position tracks and velocity), a combined WAVE file of 59 MB will be created. If one of the parameters is the derivative of the tangential velocity, the combined WAVE file will have a file size of 118 MB. This is due to the up-sampling of the EMA data to store them along with the audio. In order to reduce the file size and to simplify data sharing, these files are downsampled to 16 kHz by default.

Regarding the analysis, WAVE files that include audio can be used for the display, annotation and measurements in programs like *Praat*. However, the display of the data may not be suitable when using *Praat's* default settings for waveform plotting. This is due to the different scales of the audio signal (usually normalized between -1 and 1) and the EMA data (higher scales, e.g., between -60 mm to 60 mm). This can be solved by changing the sound scaling strategy to "by window and channel" in the sound scaling option of the editor window. Furthermore, it is not recommended to play these files as they are, because the different scales of the EMA tracks lead to undefinable loud noise, when the audio signal and the EMA tracks are played simultaneously. This can be solved by muting the channels associated with the EMA tracks. We are also working on solutions to improve a smooth EMA annotation workflow in *Praat*. The forthcoming solutions will address these issues.

As *ema2wav* is an early-stage project that is for the community, we aim to receive feedback and comments from the community for further development and for fitting the community's needs.

6. Acknowledgements

This work has benefited/partially benefited from a government grant managed by the Agence Nationale de la Recherche under the "Investissements d'Avenir" programme with the reference ANR-10-LABX-0083 (contributing to the IdEx University of Paris - ANR-18-IDEX-0001, LABEX-EFL) and by the German Research Foundation (DFG) as part of the SFB1252 "Prominence in Language" (Project-ID 281511265), project A04 "Dynamic modelling of prosodic prominence" at the University of Cologne.

7. References

[1] P. Perrier, "Control and representations in speech production," in *ZAS Papers in Linguistics*, 2005, vol. 40, pp. 109–132, hal-00430387.

[2] T. Rebernik, J. Jacobi, R. Jonkers, A. Noiray, and M. Wieling, "A review of data collection practices using electromagnetic articulography," *Laboratory Phonology: Journal of the Association for Laboratory Phonology*, vol. 12, no. 1, pp. 1–42, 2021.

[3] M. Tiede, *MVIEW: Software for visualization and analysis of currently recorded movement Data*. Haskins Laboratories, 2005.

[4] S. Ouni, L. Mangeonjean, and I. Steiner, "Visartico: a visualization tool for articulatory data," in *Proc. Interspeech 2012*, 2012, pp. 1878–1881.

[5] R. Winkelmann, K. Jaensch, S. Cassidy, and J. Harrington, *emuR: Main Package of the EMU Speech Database Management System*, 2021, R package version 2.3.0.

[6] P. Boersma and D. Weenink, "Praat: doing Phonetics by Computer," v. 6.2.09, 2022, https://www.fon.hum.uva.nl/praat/.

[7] R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2022. [Online]. Available: https://www.R-project.org/

[8] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[9] B. McFee, A. Metsai, M. McVicar, S. Balke, C. Thomé, C. Raffel, F. Zalkow, A. Malek, Dana, K. Lee, O. Nieto, D. Ellis, J. Mason, E. Battenberg, S. Seyfarth, R. Yamamoto, viktorandreevichmorozov, K. Choi, J. Moore, R. Bittner, S. Hidaka, Z. Wei, nullmightybofo, A. Weiss, D. Hereñú, F.-R. Stöter, P. Friesch, M. Vollrath, T. Kim, and Thassilo, "librosa/librosa: 0.9.1 (0.9.1)," https://doi.org/10.5281/zenodo.6097378.

[10] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms

for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[11] Quodlibet, "mutagen," 2022. [Online]. Available: https://github.com/quodlibet/mutagen

[12] Riverbank Computing, "Pyqt," https://www.riverbankcomputing.com/software/pyqt/.

[13] Carstens Medizinelektronik GmbH, *AG501 Manual*, 2014.

[14] M. McAuliffe, M. Socolof, S. Mihuc, and M. Wagner, "Montreal Forced Aligner: Trainable Text-Speech Alignment Using Kaldi," in *Proceedings of INTERSPEECH, 20-24 August, Stockholm, Sweden*, 2017, pp. 498–502.

[15] Y. Jadoul, B. Thompson, and B. de Boer, "Introducing Parselmouth: A Python interface to Praat," *Journal of Phonetics*, vol. 71, pp. 1–15, Nov. 2018.

[16] T. Bořil and R. Skarnitzl, "Tools rPraat and mPraat," in *Text, Speech, and Dialogue*, ser. Lecture Notes in Computer Science, P. Sojka, A. Horák, I. Kopeček, and K. Pala, Eds. Cham: Springer International Publishing, 2016, pp. 367–374.