# A-MUD

**Version 1b / 2017-12-21**

# Users Manual

**ARI**

Acoustics Research Institute
Austrian Academy of Sciences

**KLIVV**

Konrad Lorenz Institute of Ethology
Department of Integrative Biology and Evolution
University of Veterinary Medicine Vienna

## Introduction

**A-MUD** is an extension to the **STx** signal processing package developed at ARI and implements the method for the automatic detection and segmentation of ultrasonic vocalizations (USVs) of mice published in [1]. This implementation provides a set of functions (scripts) that can be used to apply the segmentation algorithm to some or all sound files in a STx project. The standard functions of STx enable the user to correct the automatically detected segments, add metadata (e.g. classification) and export the segment information into files which can be processed by other programs (e.g. for statistical analysis).

[1] Zala, S. M., Reitschmidt, D., Noll, A., Balazs, P., Penn, D. (2017): Automatic mouse ultrasound detector (A-MUD): A new tool for processing rodent vocalizations, in: PLOS ONE

# Installation

## STx

### Requirements:

- Hardware: a standard desktop or laptop computer, no special hardware is required
- Operation system: Windows Vista, 7, 8 or 10

### Installation:

- Use the following link to download the most recent **STx** installer from the ARI homepage: https://www.kfs.oeaw.ac.at/pub/stx/stx-installer.exe
- Run the program **stx-installer.exe** and follow the installation instructions.
- Start STx and set your preferences in the dialog shown on startup to finish the installation process. Then select “NEW project” to create your own project or open the provided sample project “sample.stxpr”.

You can find more information about the program in our STx wiki and especially in the section User Guide. Both can also be accessed via the program’s <*Help*> menu.

## A-MUD 1

### Requirements:

- STx version 4.3.0 or higher

### Installation:

- Use the following link to download the **A-MUD** archive file from the ARI homepage: https://www.kfs.oeaw.ac.at/pub/stx/a-mud/amud1_current.zip
- Extract all files from the archive to any directory.
- Start STx and open an existing or create a new project.
- Go to the directory containing the extracted files, drag and drop the file **amud1_install.sts** anywhere onto the STx workspace window and press the button <*Run*> in the displayed dialog. This will copy all necessary files to the right places and finish the installation process.

# Using A-MUD

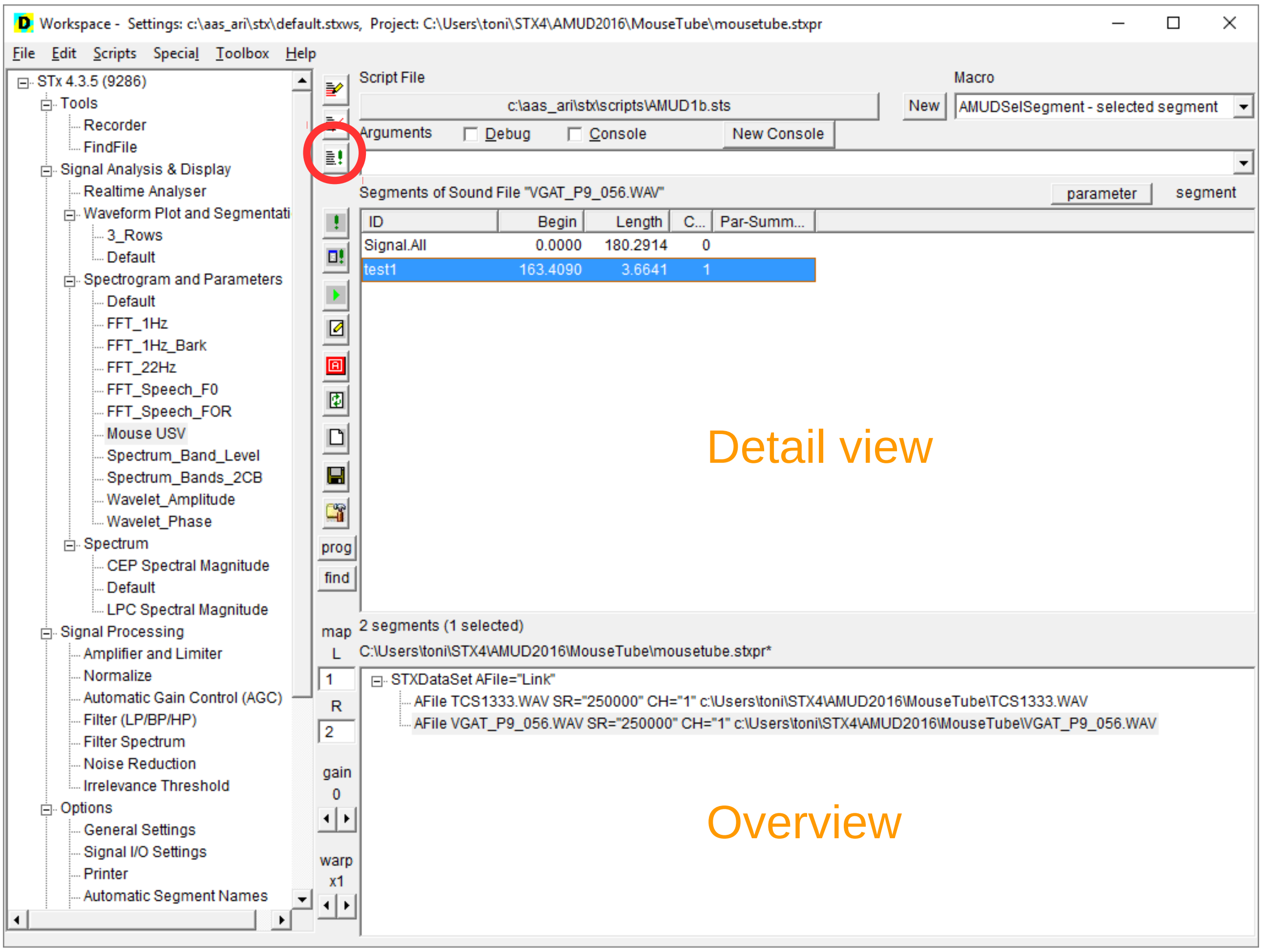
## Some necessary preparatory steps

Before you start using A-MUD, you have to create a new STx project and add your mouse recordings to this project.

- Start STx and select "NEW project". If STx is already running also the menu function *<File→Project→New>* can be used.
- Add the sound files containing the mouse recordings to the new project. The simplest way to add sound files is to select the files in the Windows Explorer and to drag and drop them onto the STx workspace window.
- Save the project (menu function *<File→Project→Save as ..>*)

Now the sound files are ready to be processed by STx.

The picture below shows the STx workspace containing two test sound files downloaded from the mouseTube archive. In this example, the short segment *test1 (~3.7s)* of the sound file *VGAT_P9_056.WAV* is selected for processing with function *AMUDSelSegment.*



The script file containing the A-MUD implementation is named **amud1b.sts** and is stored in the subdirectory **scripts** of the STx installation directory. Normally STx is installed into the directory **C:\aas_ari\stx**, but this default directory can be changed by the user during the STx installation.

To load the script file, you can drag and drop it anywhere onto the STx workspace and press the button <*Select*> in the displayed dialog, or you can use the functions of the menu <*Scripts*>. To work with scripts, the *Script Controller* section of the workspace must be visible. If it is not visible, use the menu function <*Scripts→Show Scripts*> to activate it. Look at the section The Script Controller of the STx wiki for more details on working with scripts.

Now all preparations are done and you are ready to use the A-MUD functions.

Note: the above described steps are only required once, since STx saves all settings when the program is closed.

The *Script Controller* is the part displayed on the top / right side of the STx workspace, above the detail view. The most important controls of this section are:

| | |
|---|---|
| <*Script File*> | A button to choose and load a script file |
| <*Macro*> | A list to select one function defined in the loaded script file |
| <*Arguments*> | A line to enter arguments to be passed to the selected function (the argument line should be empty for all A-MUD functions) |
| <*Run*> = | The button to start the selected function (3rd icon of the middle bar) |

The tree view on the left side of the workspace shows the standard analysis and signal processing functions of STx. This part is called the *Application & Setup view* and it provides all functionalities to work with the standard STx applications.

## Select a function and run the segmentation algorithm

To perform USV detection with A-MUD, one of three available functions must be selected in the combo box <*Macro*> of the script controller:

**AMUDSelSegment** – It applies the A-MUD algorithm to the selected segment.

Before starting this function, the user has to select a sound file in the *overview* and a segment of this sound file in the *detail view*. The segmentation will be applied only to the signal of the selected segment.

This function is mostly used for the analysis of short signals (duration <10s) to select / adjust the segmentation parameter settings (see setup parameter *flagPlot*)

**AMUDSelFiles** – It applies the A-MUD algorithm to the selected sound files.

Before starting this function, the user has to select the root element (named STXDataSet) in the *overview* and then one or more of the sound files displayed in the *detail view*. The segmentation will be applied to all selected sound files.

**AMUDAllFiles** – It applies the A-MUD algorithm to all sound files in the project.

After selecting both function and signals (depending on the function), the script can be started by pressing the button <*Run*> of the *Script Controller*. At first the parameter setup dialog is displayed (see section *Setup Dialog* in this manual), where you can select the analysis and segmentation parameters and some control options. The setup dialog will be described in detail in the next section.

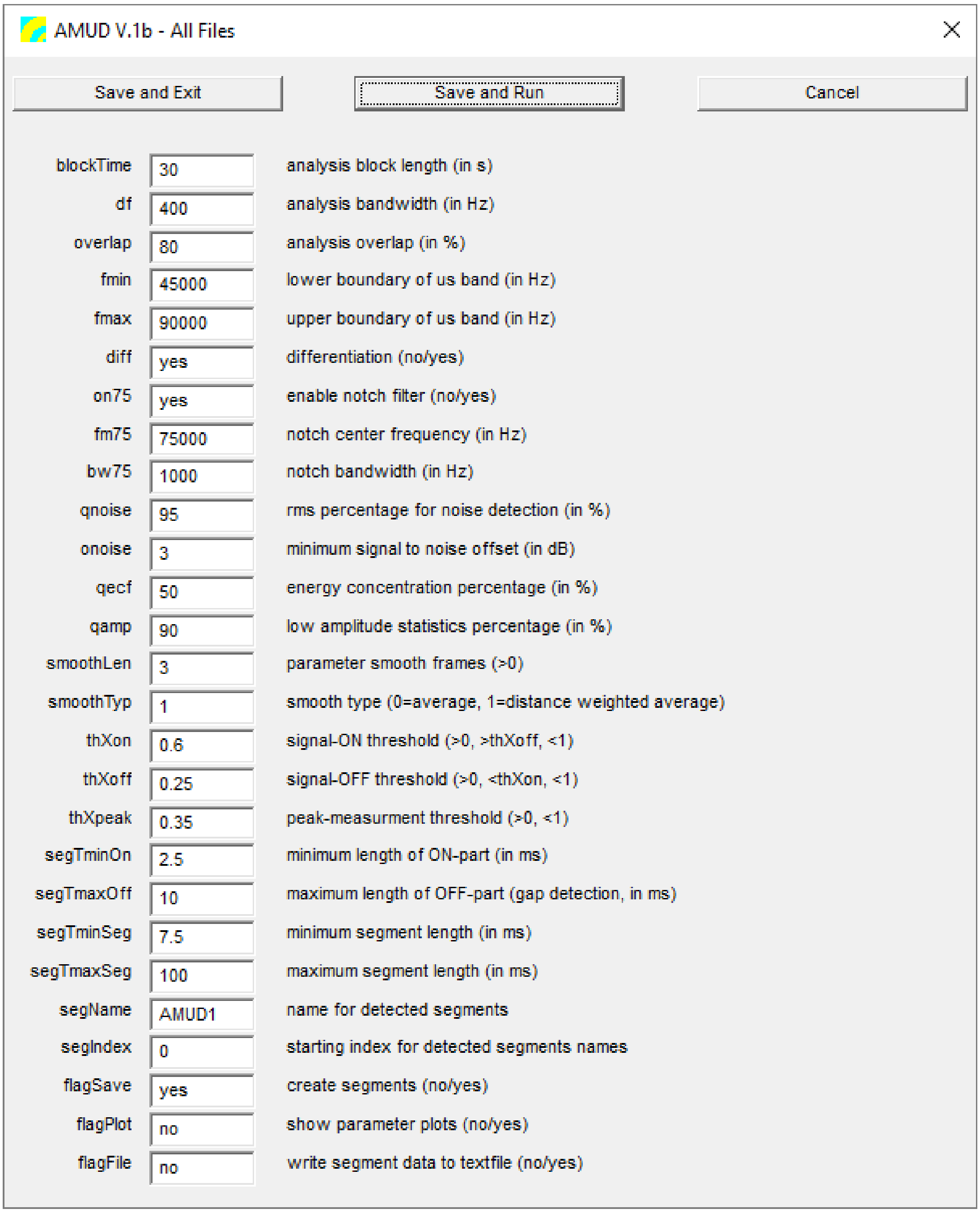
The segmentation process starts when the setup dialog is closed by pressing the button <*Save and Run*>. Now the selected function is performed, and progress and error messages are shown in a log window. At the end of the process, the log window remains open until it is closed by the user.

## A-MUD program flow and output control parameters

The settings *flagPlot*, *flagSave* and *flagFile* (at the bottom of the dialog) control the program flow and the output. They have the following meaning:

- Setting *flagPlot* to **yes**:
  - If the option *flagPlot* is ***yes*** the analysis and segmentation of each finished block is plotted and the segmentation process is suspended while the plot window is displayed. The user has to close the plot window to continue with the segmentation.
  - Therefore normally the plots are only useful in conjunction with the function AMUDSelSegment.
- Setting *flagPlot* to **no**:
  - *flagPlot* should be set **no** when using the other functions (i.e. AMUDSelFiles and AMUDAllFiles), because they are used for the automatic processing of large signals, which can last a long time (one or more hours) and will therefore run without the user interaction. After the segmentation has finished, the segments will appear in the workspace when clicking on the soundfile list inside the overview window.
- If all three output options (*flagPlot*, *flagSave* and *flagFile*) are set to **no:**
  - The results of the segmentation process will neither be displayed, nor saved.
  - To save the detected segments at least the option *flagSave* must be set to **yes**. After the segmentation is finished and the segments appear in the overview window, click on the save button (in the middle bar of the STx workspace) to save the segments in the directory of the sound file.

# Setup Dialog – some details about parameter settings




## Buttons

*<Save and Exit>* Save the settings and exit from the script

*<Save and Run>* Save the settings and start the selected segmentation function

*<Cancel>* Exit from the script without saving the settings

## Parameter settings

Spectrogram computation

*blocktime* The length of one analysis block in seconds.
Possible values: 10 ... 50

- Blocking is necessary because the algorithm has to deal with a limited amount of available memory

*df* The analysis bandwidth (frequency resolution) in Hz.
Possible values: 250 ... 400

- The analysis frame length L is the inverse of *df*: L = 1 / *df*

*overlap* The overlap of two analysis frames in %.
Possible values: 50 ... 90

- The frame distance D = L . (1 - *overlap* / 100)
- The time resolution (accuracy of segment boundaries) and the minimum segment duration depends on D

*fmin, fmax* The lower and upper boundary of the analysis frequency range in Hz.
Possible values: 20000 < *fmin* < *fmax* < 120000

- The difference *fmax*-*fmin* should be greater than 10000

*diff* Set this option to yes to apply differentiation to the signal before spectrogram computation.
Possible values: **yes** (or **1**), **no** (or **0**)

- Differentiation is simply a highpass filter to amplify the signal in the high frequency range and dampen lower frequencies.

Spectrogram preprocessing (denoising)

*on75* Set this option to **yes** to apply a notch filter to remove tonal distortions in a defined frequency band.
Possible values: **yes** (or **1**), **no** (or **0**)

*fm75, bw75* The center frequency (in Hz) and bandwidth (in Hz) of the notch filter.
Possible values: 2*df < *bw75*

- The notch filter option was implemented, because our test signals sometimes had a tonal (or narrow band) distortion at 75kHz or 100kHz. A spectrogram can be used to detect a distortion and to measure its center frequency and bandwidth. Choose the smallest possible bandwidth parameter to minimize the amount of the removed signal. The notch filter removes the signal in the frequency band *fm75* ± *bw75 / 2*.

*qnoise* Energy (rms) percentage for signal/noise threshold detection (in %)
possible values: 50 ... 95

*onoise* An offset value added to the computed signal/noise threshold (in dB)
possible values: 0 ... 6

- The parameters *qnoise* and *onoise* are used to compute a noise (or background) spectrum from the low energy signal parts. Then inverse filtering (spectral substrction) is used to get a white background noise before the segmentation parameters are computed.

<u>Parameter extraction</u>

***qecf*** Energy percentage for the computation of the spectral energy concentration (% of the overall energy). This is the <u>most important</u> parameter.
Possible values: 33 ... 80

- This parameter controls the computation of the spectral energy concentration which is used as measure to detect signals with a small bandwidth (like USVs).

*qamp* Amplitude percentage for the computation of the spectrum statistics (in %). This statistics compares the distribution of low-level and high-level amplitudes.
possible values: 75 ... 95

*smoothLen* The number of frames (+/-) for parameter smoothing.
possible values: 0 (=no smoothing) ... 5

*smoothTyp* *S*moothing weights
Possible values: 0 ... average,all weights are set to 1
1 ... distance weighted average; w = 1 / |d+1|

- Parameter smoothing is not so important, but it makes sense for weak and/or distorted signals to get more continuous parameter tracks.

<u>Segmentation thresholds and timing</u>

*thXon,*
*thXoff* *S*ignal-ON and signal-OFF threshold value.
Possible values: 0.1 ≤ *thXoff* ≤ *thXon* ≤ 0.8

- The normalized threshold is a function in the range 0 to 1 and is displayed in the plot-line **seg-th.**
- The segment detection algorithm searches for a peak which is higher than *thXon*. If a peak is found, the algorithm goes to the left/right, until the **seg-th** value is lower than *thXoff*. If the detected part is longer than *segTminOn*, it is stored, otherwise it is ignored.

*segTminOn* The minimum duration of a signal-part inside a segment (in ms).
Possible values: *segTminOn* ≥ 2.D and *segTminOn* < *segTminSeg*

*segTmaxOff* The maximum duration of a pause-part inside a segment (in ms).
Possible values: segTmaxOff ≥ 0 and
*segTmaxOff* < *segTmaxSeg*-2.*segTminOn*

*segTminSeg* The minimum duration of a segment (in ms).
Possible values: *segTminSeg* > *segTminOn* and *segTminSeg* < *segTmaxSeg*

*segTmaxSeg*
The maximum duration of a segment (in ms).
Possible values: *segTmaxSeg* > *segTminSeg*

- The following timing rules are applied:
  - Use the threshold function to detect segments with a duration equal to/longer than *segTminOn.*
  - Merge segments with a distance equal to/smaller than *segTmaxOff* into one segment.
  - Remove all segments shorter than *segTminSeg* or longer than *segTmaxSeg*.
  - Save the remaining segments.

<u>Segment naming (for the STx segment list)</u>

*segName* The prefix of the segment name.
Possible values: a combination of letters and digits starting with a letter

*segIndex* The starting value for the index appended to the segment name.
Possible values: 0 ... 9000
- A unique ID is assigned to each detected segment, which is formatted like: *segName.index*, where *index* is a 4 digit integer number starting with the value *segIndex*

<u>Program control</u>

*flagSave* If set to **yes** (or **1**), a segment in the sound files segment-list is created for each detected USV, otherwise no segments are created (see section **Output to the STx segment list** in this manual for details).

*flagPlot* If set to **yes** (or **1**), the spectrogram and the parameters for each processed block are displayed, otherwise they are not displayed.
- If *flagPlot* equals **yes**, the program waits until the user closes the plot window.

*flagFile* If set to **yes** (or **1**), the segments and segment-metadata are saved in a text file, otherwise not. If a text file should be written, the parameter *flagSave* must also be set to **yes** (or **1**).
- The text file is named *soundfilename.segName.txt* and is located in the same directory as the sound file (see section **Output to the segment data file** in this manual for details).

## Output to the STx segment list

STx manages a list of segments for each sound file. A segment is a piece of signal with defined time boundaries. In STx each a segment has a unique id with the begin time and the length. In addition, an arbitrary list of attributes can be assigned to a segment. The segment list of a sound file can be displayed in the *detail view* of the STx workspace. It is saved in xml-format in the directory of the sound file (*soundfilename*.wav.stxml).

By clicking on the save button (in the middle bar of the STx workspace) the segment lists of all sound files of the current project are saved.

If the parameter *flagSave* is set to **yes**, the following attributes for each detected segment are stored in the segment list of the sound file:

| | |
|---|---|
| *ID* | Unique ID in the format *segName.index*, where *index* is a 4 digit integer number starting with the value *segIndex* |
| *Begin* | Starting time (position) of the segment<br>(stored in samples, displayed in seconds) |
| *Length* | Length (duration) of the segment<br>(stored in samples, displayed in seconds) |
| *Type* | USV type (classification); this attribute is always set to the value "?" The current version of the A-MUD method only detect USVs but does not classify them. This attribute is currently just a place holder, but it will be used for automatic classification by future versions. |
| *Info* | A blank separated list of values derived from the frequency track (contour) of the USV (candidate). The following values are stored in this attribute: |

*Info* = "*fmean fband amean t1 f1 a1 tn fn an tfmin fmin afmin tfmax fmax afmax tamax famax amax*"

| | |
|---|---|
| *fmean* | Mean frequency |
| *fband* | Frequency range (= maximum – minimum) |
| *amean* | Mean amplitude |

Time, frequency and amplitude of 5 selected track points:

| | |
|---|---|
| *t1, f1, a1* | First point |
| *tn, fn, an* | Last point |
| *tfmin, fmin, afmin* | Point with the lowest frequency (*fmin*) |
| *tfmax, fmax, afmax* | Point with the highest frequency (*fmax*) |
| *tamax, famax, amax* | Point with the highest amplitude (*amax*) |

- In the above list the following units are used: millisecod (ms) for time values, hertz (Hz) for frequencies and decibel (dB) for amplitudes.
- All time values are offsets relative to the begin of the segment.

Notes:

- If not all of the important segment attributes are displayed in the segment list, use the <Attribute> button (red button in the middle bar of the workspace) or the function <Options→General Settings> of the Applications & Setup view to set the visibility of the attributes.
- The context menu function <Copy Attributes> of the *detail view* can be used to copy the attributes of selected / all segments to the clipboard.

## Output to the segment data file

If both the parameters *flagSave* and *flagFile* are set to **yes**, a set of values for each detected segment is written to a text file named *soundfilename.segName.txt* which is located in the same directory as the sound file. One line containing the following values (separated by blanks) is stored for each segment:

"index tbeg tdur *t*[1] *f*[1] *a*[1] *t*[2] *f*[2] *a*[2] … *t*[n] *f*[n] *a*[n]"

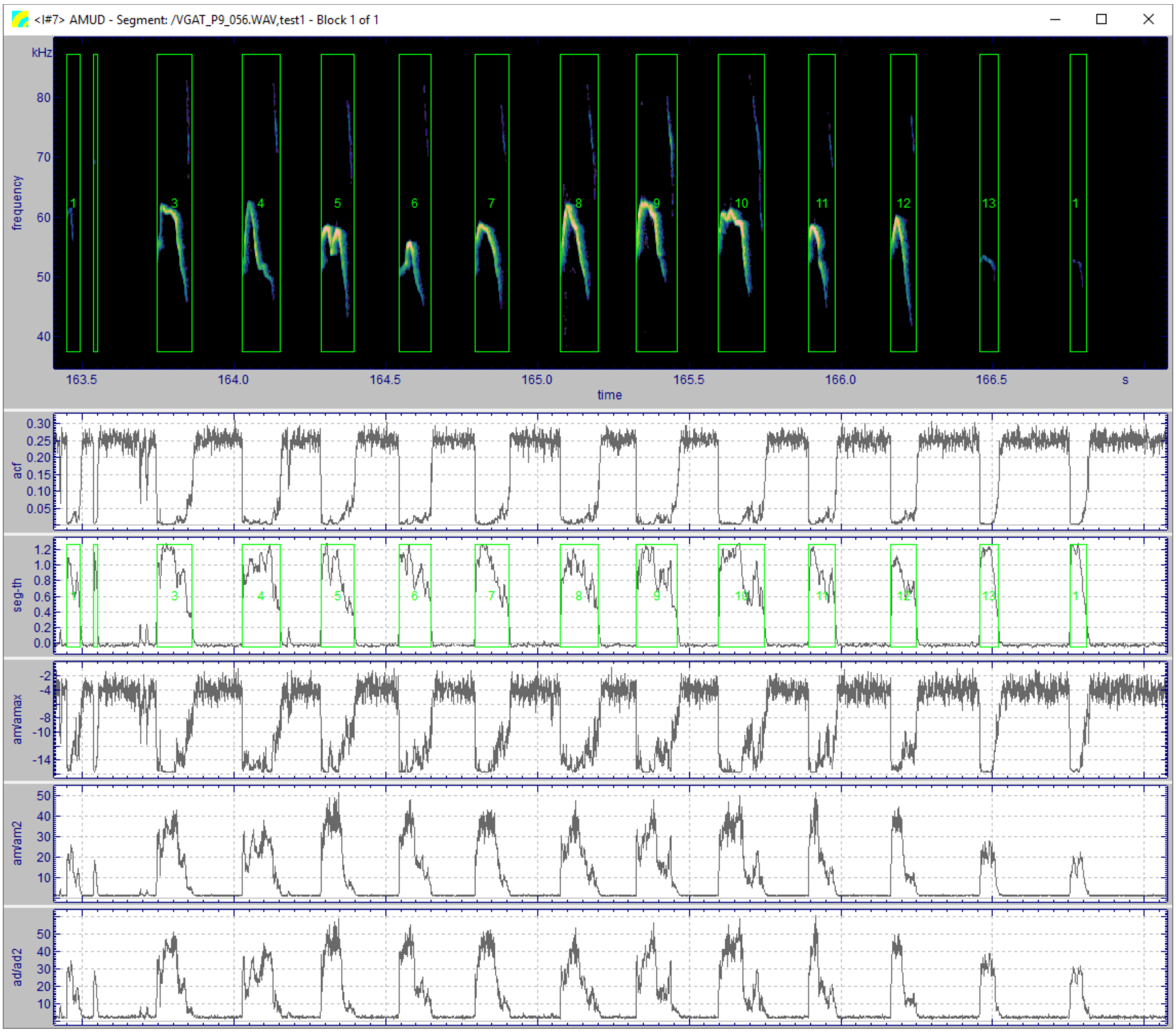
| | |
|---|---|
| *index* | Incremental segment index. This is the same value used for the second part of the ID in the segment list. |
| *tbeg* | Segment starting time |
| *tdur* | Segment duration |
| *t*[i], *f*[i], *a*[i] | Time offset, frequency and amplitude of track point i (i = 1 .. n) |

- In the above list the following units are used: millisecod (ms) for time values, hertz (Hz) for frequencies and decibel (dB) for amplitudes.
- All *t*[i] values are offsets relative to the begin of the segment

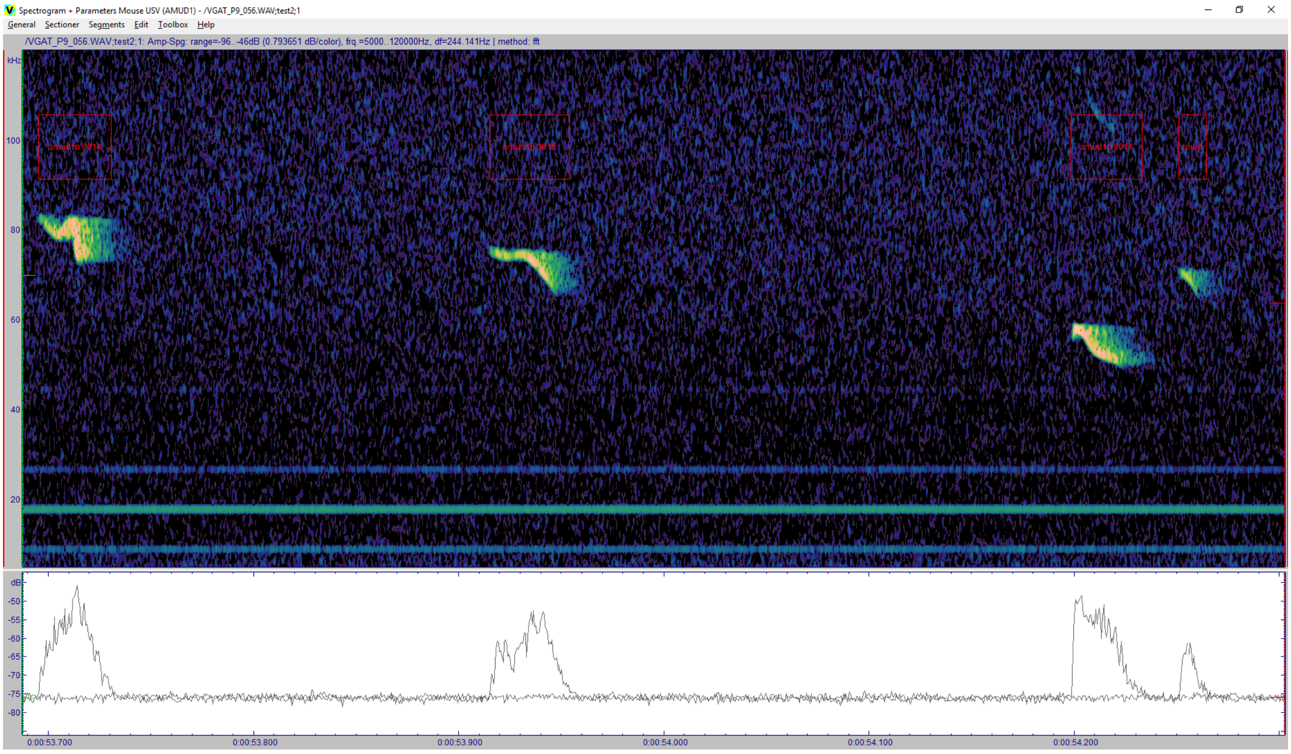
The data files contain all the information of the frequency track of the USVs detected by A-MUD. They can be used as input for other programs (like R or Mathlab) for further processing.

# Examples

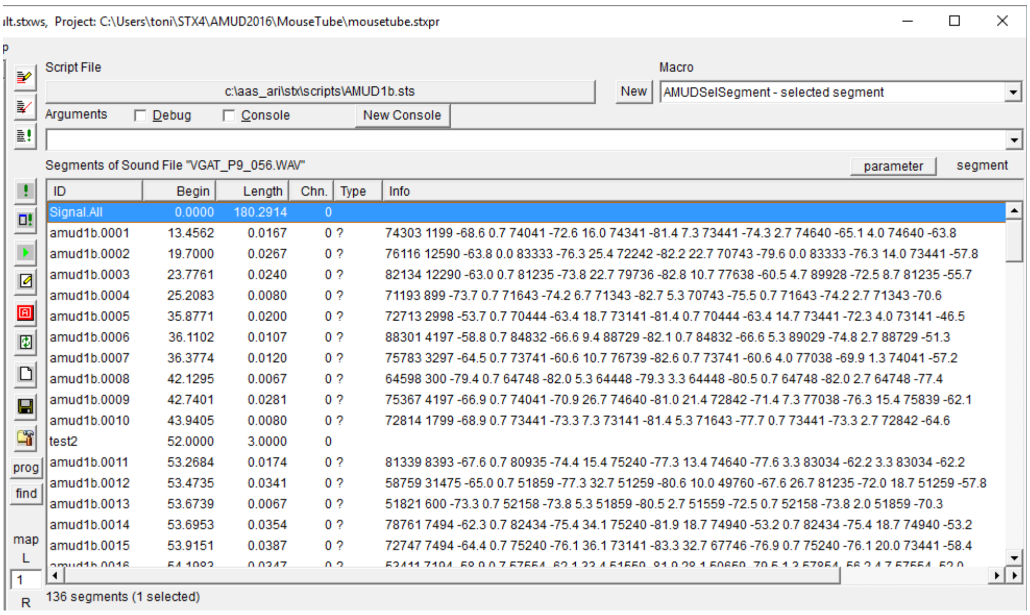
1. The picture shows the plot of the (preprocessed) spectrogram and the segmentation parameters created by AMUDSelSegment for a short test signal of a recording downloaded from the mouseTube archive. The detected segments are marked in the spectrogram and the main segmentation function (seg-th) with green rectangles. The function seg-th can be used to choose the values of the parameters *thXon* and *thXoff*.



2. This picture shows a spectrogram of a very short signal part (~500ms). The automatically detected USVs are marked with red rectangles. The lower graph shows the rms-energy for the frequency bands 5kHz-40kHz and 40kHz-120kHz. This plot was created with the standard spectrogram & parameter analysis function of STx.




3. The picture shows an example of a segment list with all important attributes. The segments named *amud1b.** (attribute ID) were created by A-MUD.




| ID | Begin | Length | Chn. | Type | Info |
|---|---|---|---|---|---|
| Signal.All | 0.0000 | 180.2914 | 0 | | |
| amud1b.0001 | 13.4562 | 0.0167 | 0 | ? | 74303 1199 -68.6 0.7 74041 -72.6 16.0 74341 -81.4 7.3 73441 -74.3 2.7 74640 -65.1 4.0 74640 -63.8 |
| amud1b.0002 | 19.7000 | 0.0267 | 0 | ? | 76116 12590 -63.8 0.0 83333 -76.3 25.4 72242 -82.2 22.7 70743 -79.6 0.0 83333 -76.3 14.0 73441 -57.8 |
| amud1b.0003 | 23.7761 | 0.0240 | 0 | ? | 82134 12290 -63.0 0.7 81235 -73.8 22.7 79736 -82.8 10.7 77638 -60.5 4.7 89928 -72.5 8.7 81235 -55.7 |
| amud1b.0004 | 25.2083 | 0.0080 | 0 | ? | 71193 899 -73.7 0.7 71643 -74.2 6.7 71343 -82.7 5.3 70743 -75.5 0.7 71643 -74.2 2.7 71343 -70.6 |
| amud1b.0005 | 35.8771 | 0.0200 | 0 | ? | 72713 2998 -53.7 0.7 70444 -63.4 18.7 73141 -81.4 0.7 70444 -63.4 14.7 73441 -72.3 4.0 73141 -46.5 |
| amud1b.0006 | 36.1102 | 0.0107 | 0 | ? | 88301 4197 -58.8 0.7 84832 -66.6 9.4 88729 -82.1 0.7 84832 -66.6 5.3 89029 -74.8 2.7 88729 -51.3 |
| amud1b.0007 | 36.3774 | 0.0120 | 0 | ? | 75783 3297 -64.5 0.7 73741 -60.6 10.7 76739 -82.6 0.7 73741 -60.6 4.0 77038 -69.9 1.3 74041 -57.2 |
| amud1b.0008 | 42.1295 | 0.0067 | 0 | ? | 64598 300 -79.4 0.7 64748 -82.0 5.3 64448 -79.3 3.3 64448 -80.5 0.7 64748 -82.0 2.7 64748 -77.4 |
| amud1b.0009 | 42.7401 | 0.0281 | 0 | ? | 75367 4197 -66.9 0.7 74041 -70.9 26.7 74640 -81.0 21.4 72842 -71.4 7.3 77038 -76.3 15.4 75839 -62.1 |
| amud1b.0010 | 43.9405 | 0.0080 | 0 | ? | 72814 1799 -68.9 0.7 73441 -73.3 7.3 73141 -81.4 5.3 71643 -77.7 0.7 73441 -73.3 2.7 72842 -64.6 |
| test2 | 52.0000 | 3.0000 | 0 | | |
| amud1b.0011 | 53.2684 | 0.0174 | 0 | ? | 81339 8393 -67.6 0.7 80935 -74.4 15.4 75240 -77.3 13.4 74640 -77.6 3.3 83034 -62.2 3.3 83034 -62.2 |
| amud1b.0012 | 53.4735 | 0.0341 | 0 | ? | 58759 31475 -65.0 0.7 51859 -77.3 32.7 51259 -80.6 10.0 49760 -67.6 26.7 81235 -72.0 18.7 51259 -57.8 |
| amud1b.0013 | 53.6739 | 0.0067 | 0 | ? | 51821 600 -73.3 0.7 52158 -73.8 5.3 51859 -80.5 2.7 51559 -72.5 0.7 52158 -73.8 2.0 51859 -70.3 |
| amud1b.0014 | 53.6953 | 0.0354 | 0 | ? | 78761 7494 -62.3 0.7 82434 -75.4 34.1 75240 -81.9 18.7 74940 -53.2 0.7 82434 -75.4 18.7 74940 -53.2 |
| amud1b.0015 | 53.9151 | 0.0387 | 0 | ? | 72747 7494 -64.4 0.7 75240 -76.1 36.1 73141 -83.3 32.7 67746 -76.9 0.7 75240 -76.1 20.0 73441 -58.4 |