PROSAX_001.maxpat

Omax/Somax2 suite

(picture,Thanks to Gemini.ai)

Mikhail Malt,
Representation Musical Team
Ircam
March 2025

Table of Contents

PROSAX_001.MAXPAT

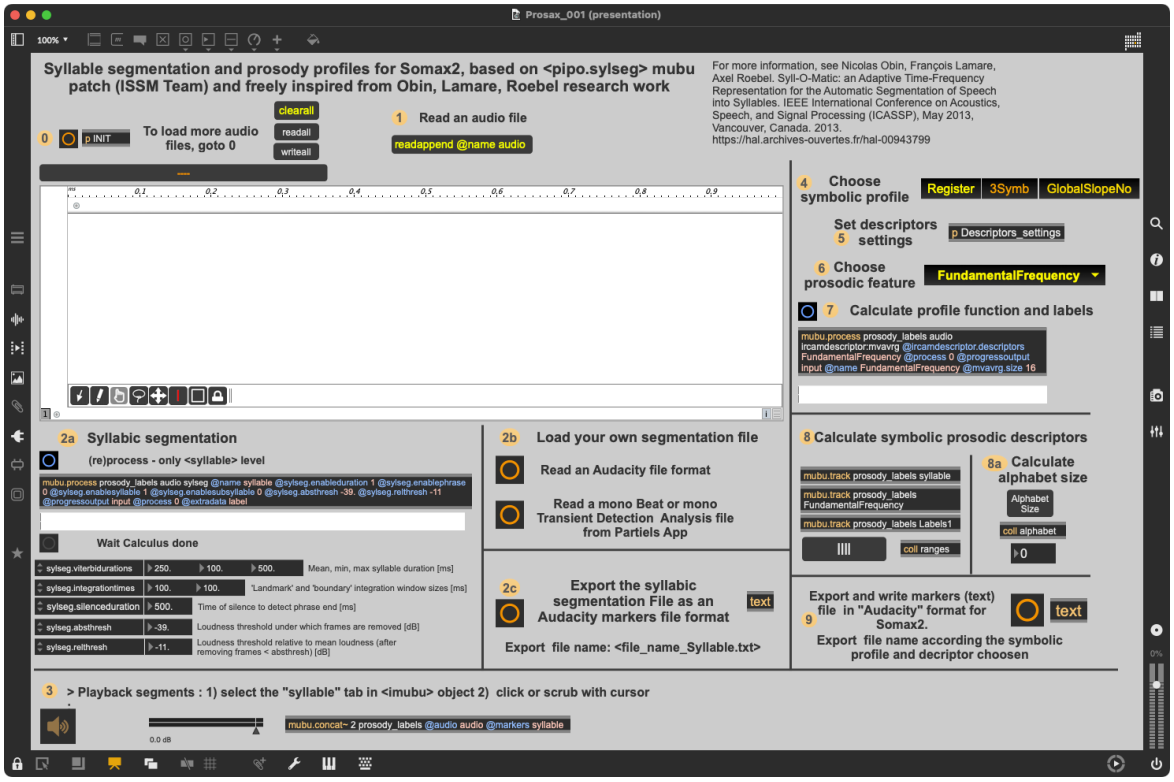

Figure 1: The Main patch

Prosax_001.maxpat is a research patch, a proof-of-concept one, intended to explore prosodic profiles on segmented audio to generate event labels for OMAX and SOMAX2. It is not a finished, a completed one. It is a work in progress. A process mainly intended for speech segmentation, but you can use it for other audio cases, with more and less success. Just explore.

The main ideas in this work emerged from Artistic Research carried out with Valérie Philipin (to whom we are indebted for her musical and literary expertise), from October 2023 to June 2024 (In the Reach project context), on the use of Somax2 in a spoken and sung voice context.

This patch is an adaptation of <pipo. sylseg> help patch (from Mubu for Max Max Package) and based on the Nicolas Obin, François Lamare and Axel Roebel research [Obin, Lamare, Roebel 2013]:

Prosax_001.maxpat is part of the Omax/Somax2 suite and part of the project REACH supported by the European Research Council under Horizon 2020 program (Grant ERC-2019- ADG #883313) and project MERCI supported by Agence nationale de la Recherche (Grant ANR-19-CE33-0010).

This documentation is a hand on one, so, let's dive in it.

1 DEPENDENCIES

This patch needs the previous installation of the last “MuBu For Max” package. The MUBU for Max Package is “A Toolbox for Multimodal Analysis of Sound and Motion, Interactive Sound Synthesis and Machine Learning.” Developed by the ISMM Team at Ircam (https://ismm.ircam.fr/mubu/). You can install it by the Max “Package Manager.”

2 INITIALIZATION (0)

After opening the patch, the first step is to initialize all processes and buffers. To do this, press the button next to the number “0” (Figure 2).

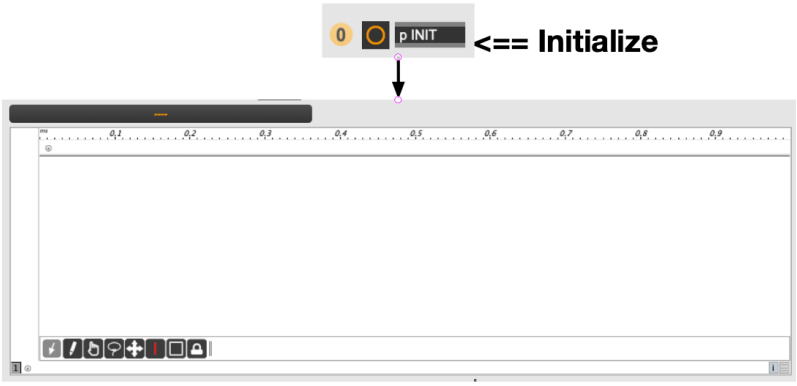

Figure 2: <imubu> empty and initialized

It cleans all buffers and the main one, the <imubu>.

3 LOADING AN AUDIO FILE (1)

Second step, load an audio file. You can use.aiff or.wav formats, in mono or stereo. Just be aware of the sampling rate of your file and Max. It’s more convenient if you work with the same sampling rate for both (the audio file and Max).

Figure 3: <imubu> with a sound file loaded

Click on the message box (written in yellow) [readappend @name audio]. This will load the audio file into the main <imubu>. You can verify this, by viewing the waveform in the <imubu> (Figure 3). You'll also see the name of the audio file (written in orange color) appear in a message box. We strongly recommend that you choose to file names with "no spaces," and using only alphanumeric characters without accents. Hyphens and underscores are permitted. This is only a recommendation; you are free to name your files as you see fit, at your own risk.

4 SEGMENTATION (2)

The next step is segmentation.
For both OMAX and SOMAX2, segmentation, and the way it's done, is a musical point of view of the utmost importance. But we won't discuss that in this context, which is more a manual for <Prosax> the cure for all your segmentation woes.
Returning to our patch. We propose two ways of integrating segmentation, for the calculation of prosodic profiles. One way is to use "syllabic segmentation," and the second is to load your own segmentation. This second segmentation can originate from other algorithms, such as "Transient detection" or "Beat Detection," both available through the Partiels application.

4.1 Syllabic segmentation (2a)

The "syllabic segmentation" is 6alculated by <pipo.sylseg> (based on the work of Obin, Lamare and Roebel). This process has 5 parameters to set (Figure 4):
1) The average, minimum and maximum syllable durations, in milliseconds.
2) the size of the "landmark" and "boundary" integration windows, in milliseconds.
3) The silence time to detect the end of the sentence, in milliseconds.
4) The loudness threshold under which frames are removed [in dB], <absthresh>.
5) The Loudness threshold relative to mean loudness (after removing frames smaller than <absthresh>) [in dB].

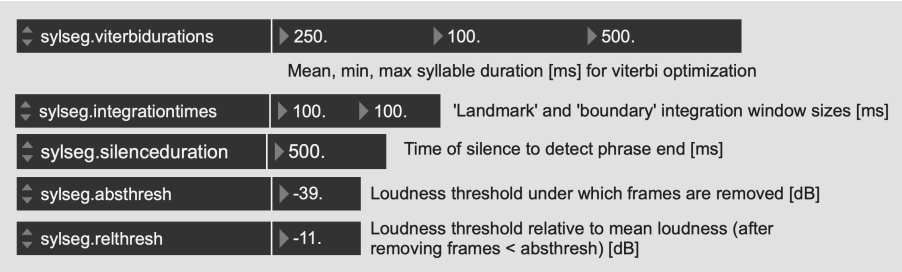

Figure 4: Parameters for the <pipo.sylseg> object

Remember that setting the starting parameters is fundamental to the result of the analysis. We offer you a set of parameters that work for a medium-speed, articulate speaking style. Depending on the speed of your speech, differences in intensity, articulation, intonation, etc., the parameters will have to be adjusted, both for the syllable segmentation part and for the fundamental frequency analysis part.
This applies to both segmentation and prosodic function analysis parameters.
Okay, if you're going to calculate a syllabic segmentation, all you have to do is set the necessary parameters, depending on your speech rate and/or loudness levels.
If you've opted for syllabic segmentation, simply skip to this step: ***5 Checking*** the segmentation**.**

4.2 Loading your own segmentation (2b)

In some cases, it may be that segmenting by hand or using other software may be better suited to your work. Again, we have two possible choices, for now (Figure 5):

Reading an Audacity file format or

Reading a “mono” Beat or Transient Detection Analysis file, from Partiels.

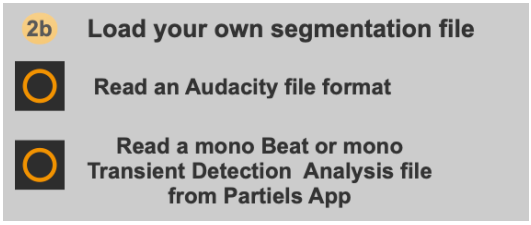

Figure 5: Loading your own segmentation

4.2.1 Reading an Audacity file format

Uploading a file in Audacity file format can happen in a number of cases:
- You have made your own segmentation, by hand, in Audacity. In our opinion, the best way to segment is by hand. We're using Audacity here as a reference, since it's a royalty-free tool and quite easy and efficient for this task.
- In another case, you've made your own segmentation using other tools.

In all these cases, the file you upload should have the “Audacity” (simplified) format, i.e.:

$$segment\_start_0\ SPACE\ segment\_end_0$$
$$segment\_start_1\ SPACE\ segment\_end_1$$
$$segment\_start_2\ SPACE\ segment\_end_2$$
$$\dots$$
$$segment\_start_n\ TAB\ segment\_end_n TAB\ CR$$

knowing that $\boldsymbol{segment\_start_n}$ **must always be equal to** $\boldsymbol{segment\_end_{n-1}}$.
Here you just need to click on the button to the left of ***"Read an Audacity file format"***, navigate and choose the corresponding text file (Figure 6).

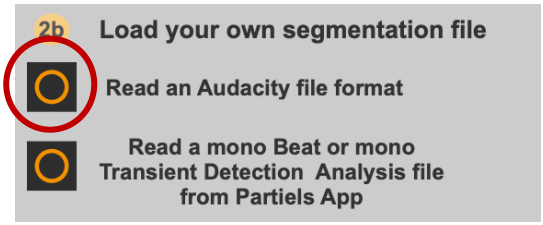

Figure 6: Read an Audacity file format

4.2.2 Reading a “mono” Beat or Transient detections Analysis file from Partials app

Reading a “mono” Beat or Transient Detection Analysis file, from Partiels (designed and developed by Pierre Guillot @Ircam, in the IMR department).

“Transient Detection” and “Beat Detection” algorithms (designed by Geoffroy Peeters and developed by Geoffroy Peeters and Frédéric Cornu of the Analysis-Synthesis team of the STMS Lab hosted at IRCAM.) are very important algorithms allowing a huge control on the segmentation. In addition, the "Beat Detection" detection algorithm allows you to work with pulsed music, even metric, within the framework of SOMAX2.
However, one important point for error-free file exchange is that the analysis in partials must be performed in "mono" mode. These two algorithms, for stereo files, perform a separate analysis for each channel. This doesn't seem very useful in the vast majority of cases. A segmentation analysis on a mono reduction of a stereo file is often sufficient. We won't go into the more general case of segmentation for multichannel files.
Returning to our case, if you have a stereo audio file to analyze, it's important that the "Audio Files Layout" in the Partials application displays only one voice, so that the analysis is performed on the mono reduction of the stereo file (Figure 7).

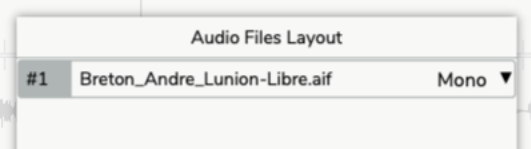

Figure 7: “Audio Files Layout" in the Partials app

Another point in using these segmentation algorithms is to be careful to export the data in .CSV format with "Space" as the column separator (Figure 8).

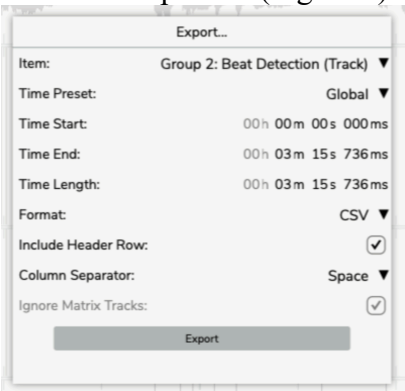

Figure 8: Partials export parameters

Here you just need to click on the button to the left of ***"Read a mono Beat or mono Transient Detection Analysis file from Partiels App"***, navigate and choose the corresponding text file (Figure 9).

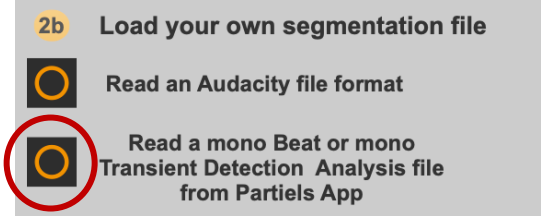

Figure 9: Read a mono Beat or mono Transient Detection Analysis file from Partiels App

4.2.3 A side effect of this patch

A side effect of this patch is the ability to use it to convert Beat or Transient Detection (mono) analysis files into Audacity format, which can be quickly used for corpus construction via the "Manual Annotations" feature. You just need to load a “mono” Beat or Transient Detection Analysis file, from Partiels (see 2b, paragraph 4.2.2) and export it (see 2c, paragraph 4.2.3).

5 CHECKING THE SEGMENTATION

Perfect, you've either calculated a syllabic segmentation or you've imported a segmentation. In both cases, you should see the audio file displayed on the <imubu> with orange vertical lines, proof that you have your segmentations correctly (Figure 10).

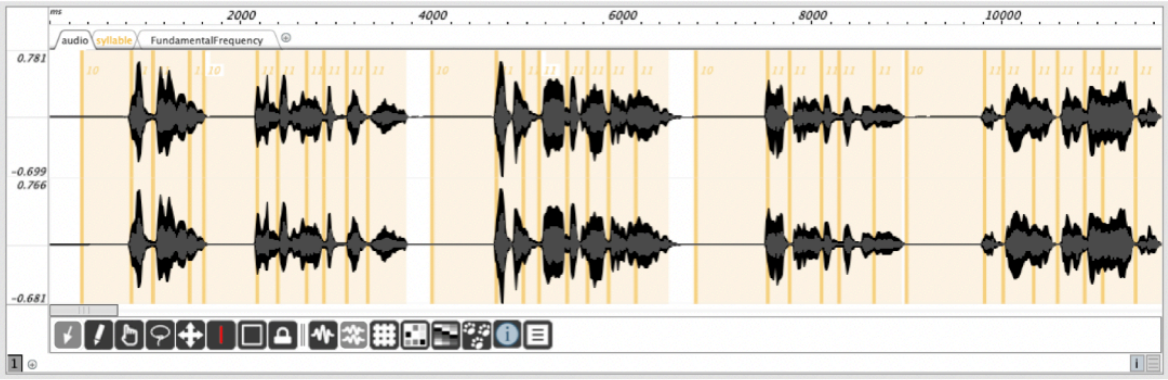

Figure 10:<imubu> displaying the segmentation

However, it's important to check the segmentation. Check if all (or most) of the segments are correctly identified, and if everything sounds good when you listen.
To do this, you can listen to all the segments, one by one, to check if the beginnings and ends are consistent. First, choose the "select" selection mode from the pictographic menu at the bottom left of the <imubu> (Figure 11).

Figure 11: <imubu> bottom menu

Then, click on the "syllable" tab of the <imubu>, to bring it back to the front (Figure 12).

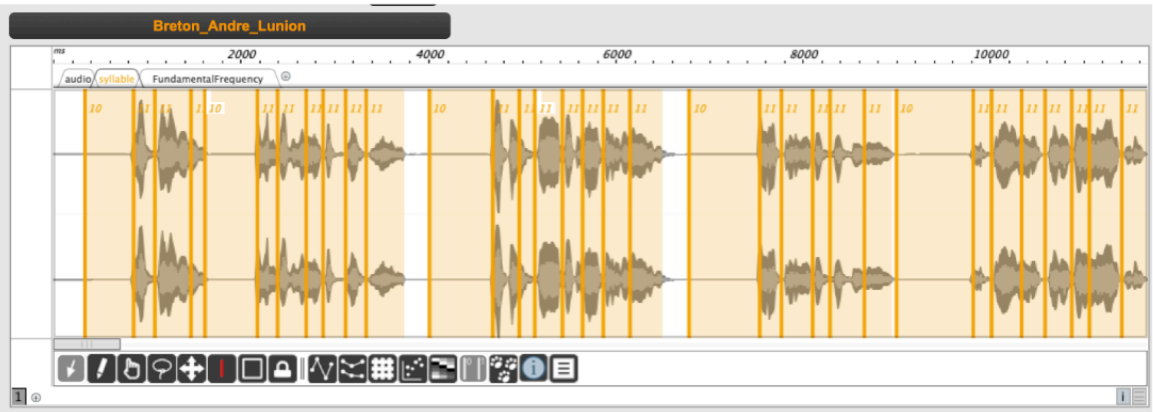

Figure 12: Putting “syllable” tab in frontend

The segmentation layer is clearly visible in the foreground.

Be careful not to forget to turn on the DSP in max. You have an <ezdac~> (number 3), at the very bottom of the patch (Figure 13).

Figure 13: Playback segments control

Now you just click (while in the "select" mode of the <imubu>) to listen to one segment at a time.

6 HAPPY OR NOT?

We reach a point, are we happy or not with the segmentation obtained?

6.1 Not Happy

We are not happy with the segmentation (regardless of its origin). It is possible to edit the segmentation in the <imubu>, but this requires some experience and learning certain manipulations. This is not an indictment of the <imbu> object. It's simply the fact that Audacity is an audio editor, made for this kind of manipulation. This may be a solution. If so, correct the segmentations manually and jump to 7 Prosodic profile.
Another possibility, which we consider is handier and allows for much more precision, is to export the segmentation we have to process and correct it in Audacity (Figure 14).

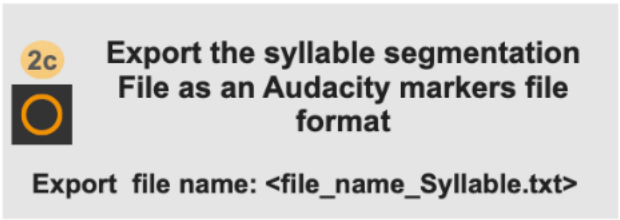

Figure 14: Exporting segmentation data to correct/change in Audacity

In this case, simply press the "2c" button, navigate to export your text file with the segmentation to be imported into Audacity and corrected. The exported file will have the name < File_Name_Syllable.txt>. Import the file into Audacity, make the desired corrections and return to paragraph ***4.2.1 Reading*** an Audacity file format.

6.2 Happy

Great, you're happy, the segmentation suits you. In this case, go directly to ***7 Prosodic profile.***

7 PROSODIC PROFILE (4)

As mentioned earlier, this patch takes the main ideas of prosodic profiling and tests them in different ways and with different audio descriptors.

7.1 How we generate the prosodic profiles?

Having the segmentations, now we will see how each segment "sings", how each segment moves internally. For this we can calculate the evolution of the fundamental frequency or other descriptor that we consider important for our project. This descriptor, we will call it, a "prosody indicator". If in speech, normally the fundamental frequency is used, we can imagine that in the case of noisy or non-tonal sounds, we could either use other descriptors, such as the Spectral Centroid, for example.
Let's say for the moment that we have a segmentation and a function (a curve) that we call "prosody indicator".
As we have pointed out, we are basing ourselves on the work of Obin, Lamarre and Roebel, but by making a simplified codification. It would be simple, if you want, to modify the patch to implement exactly the method of their work.
The problem now is how to codify, express the prosodic behavior in each segment (Figure 15).

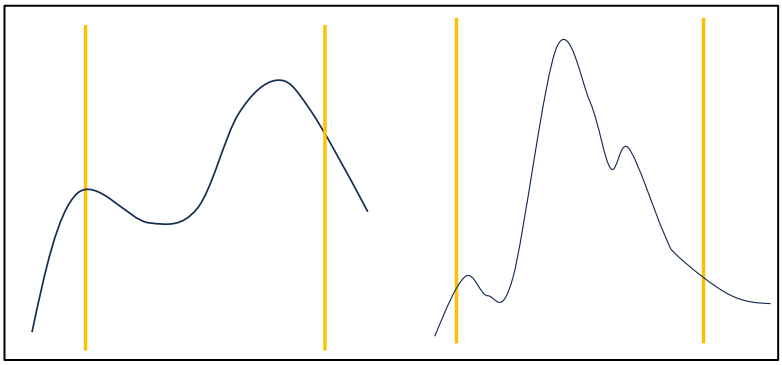

Figure 15: Examples of "prosody indicator curve in a segment

One way could be to divide each segment (syllable) into "n" parts. "n" can be from 1 to what you want. As you'll see, 5 is already too many. It is easy to understand that the greater the number of parts into which the segment is divided, the more labels we will have. As a practical value, we recommend having a basic alphabet of about ten to twenty symbols.

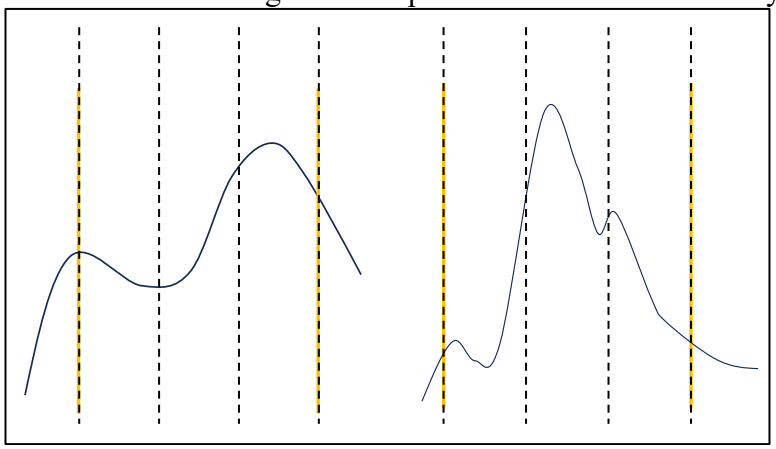

Figure 16: segment divided in "n" parts (in this case, n=3)

For the sake of the example, let's say "n" = 3 (Figure 16).
The next step will be to assign a symbol to each sub-part of the segment, so as to have a symbolic "word" designating the internal dynamics of the segment.
In this patch we used two ways to assign a symbol. The first considers the frequency register, divided into 5 parts as done in [Obin, Lamarre, Roebel 2013] (Table 1): Very Low, low, middle, high and very High.

H	Very High
h	high
m	middle
l	low
L	Very Low

Table 1: Five registers

These registers are calculated dynamically for each prosodic indicator curve. Figure 17 presents an example of this process.

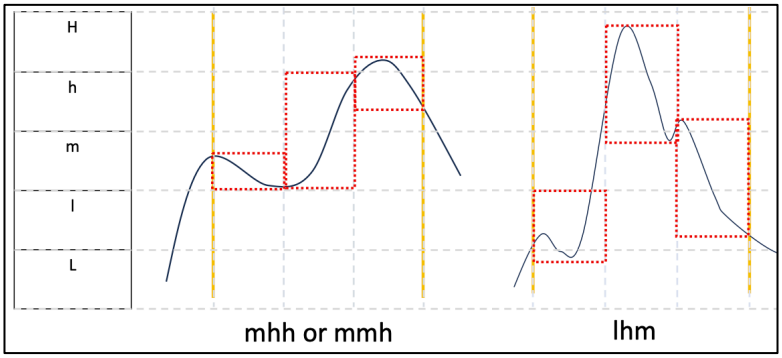

Figure 17: Examples of word labels from frequency registers

Another way to construct the labels is to consider the direction, the slope of the curve within the segment's sub-section. We considered three levels, "u" for rising, "d" for falling, and "s" for stable (Table 2).

u	up	/
d	down	\
s	stable	–

Table 2: Three slope symbols

Figure 18 presents an example of this way of labeling.

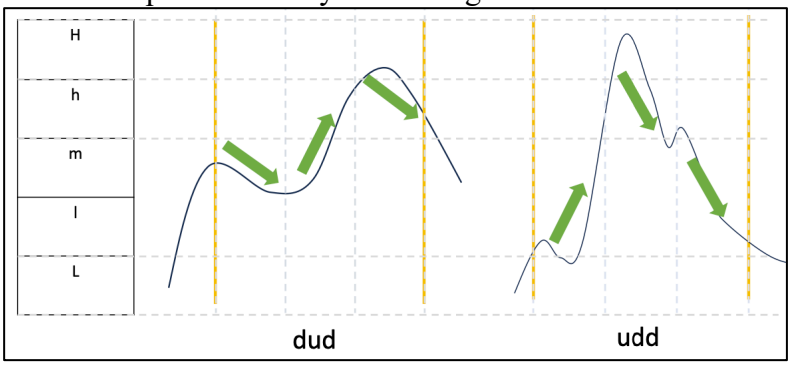

Figure 18: Examples of word labels from slopes

8 CHOOSING A SYMBOLIC PROFILE

The next step will then be to choose how to construct the prosodic label (Figure 19).

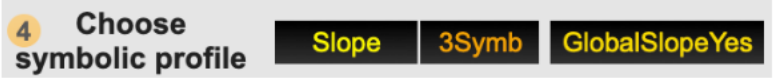

Figure 19: Menus to choose prosodic label format

We then have three options:
1) Use the slope or register to choose the sub-segment symbol (Figure 20),
2) How many sub-segments each segment will be divided into (Figure 21), and
3) a third parameter, which will be the "global" slope—the average direction of the prosody indicator function's evolution within the segment (Figure 22). This third parameter allowed us to differentiate between segments with a repetition of a symbol, but which had different behaviors.

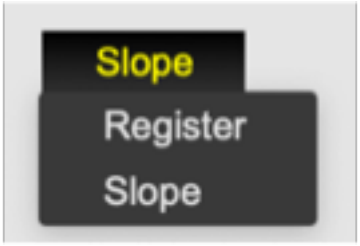

Figure 20: Register or Slope

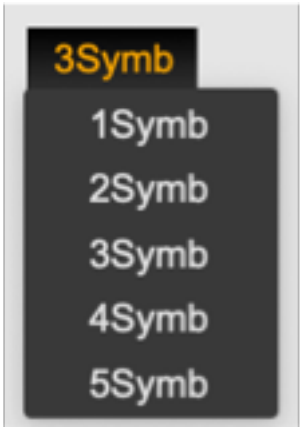

Figure 21: How many symbols

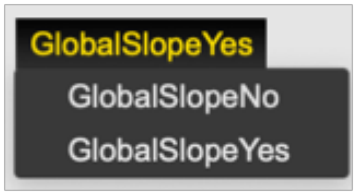

Figure 22: Global Slope, yes or no

9 SETTING DESCRIPTORS PARAMETERS (5)

As mentioned before, "setting the parameters is fundamental to the result of the analysis." Especially for the prosodic indicator function parameters, in the following example being the fundamental frequency, take your time to check if the parameters are suitable for the audio file you are using. For example, think about resampling, you must avoid making the analysis algorithm work on registers where there would only be noise. The same goes for the windowing parameters, and for the fundamental frequency analysis parameters. The parameters will necessarily be different for a woman's voice or a man's voice, and so on.
At number (5) you have a sub-patch with the parameters to adjust (See Figure 23 and Figure 24).

Figure 23: Sub patch with Descriptor settings

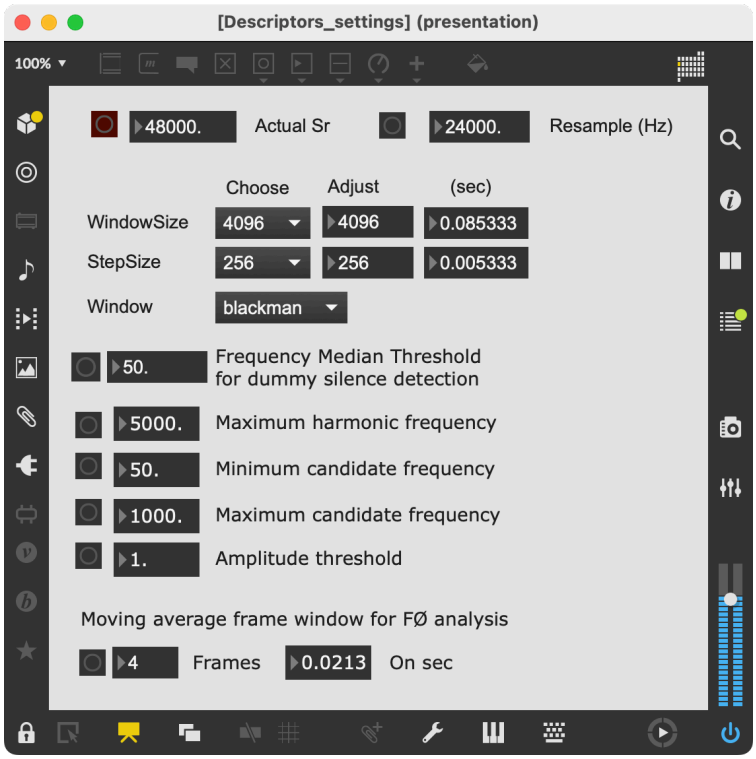

Figure 24: Main parameters for Audio descriptor analysis

10 CHOOSE PROSODIC AUDIO FEATURE (6)

At this point, we just need to choose the audio descriptor that we will use as a prosody indicator (Figure 25).

Figure 25: Choosing the prosodic audio feature

At the moment, we have 3 possibilities: Fundamental Frequency, Spectral Centroid and Perceptual Spectral Centroid (Figure 26). As we're using <ircamdescriptors> encapsulated in a <pipo>, it's very easy to add other descriptors. Simply add the descriptor name, as understood by <ircamdescriptors> encapsulated in a <pipo>.

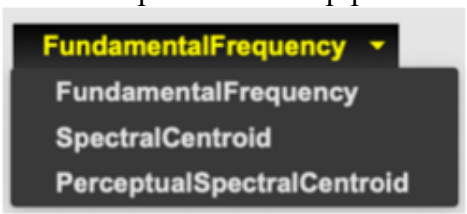

Figure 26: The three audio descriptors currently available.

11 CALCULATE PROFILE FUNCTIO N AND PROSODIC LABELS (8 AND 8A)

At this point, we don't have much to do; the labels are calculated automatically after the prosodic function is calculated. However, we can click on the message box with the text "Alphabet size" (see 8a in Figure 27) to know the number of "letters" in our symbolic alphabet. This number is a good indicator of how interesting the generation will be in Somax or Omax. As we mentioned earlier in this text, we should be wary of alphabets that are too small, i.e., less than ten symbols, or alphabets that are too large. Obviously, this will also depend on the size of our audio file and the number of segments.

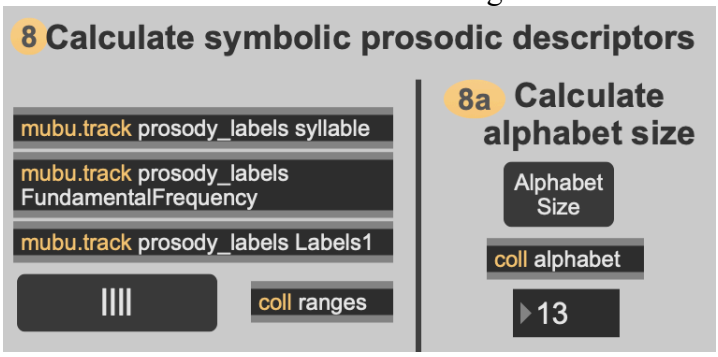

Figure 27: Prosodic descriptors

12 CHECK PROSODIC LABELS

After the labels have been calculated, we still need to check whether it works well, whether the labels match, whether we need to adjust any specific parameters, or whether the choice of descriptor or symbolic form suits us. In the Figure 28 example we calculated a symbolic prosody descriptor with 3 symbols, the slope and the overall slope.

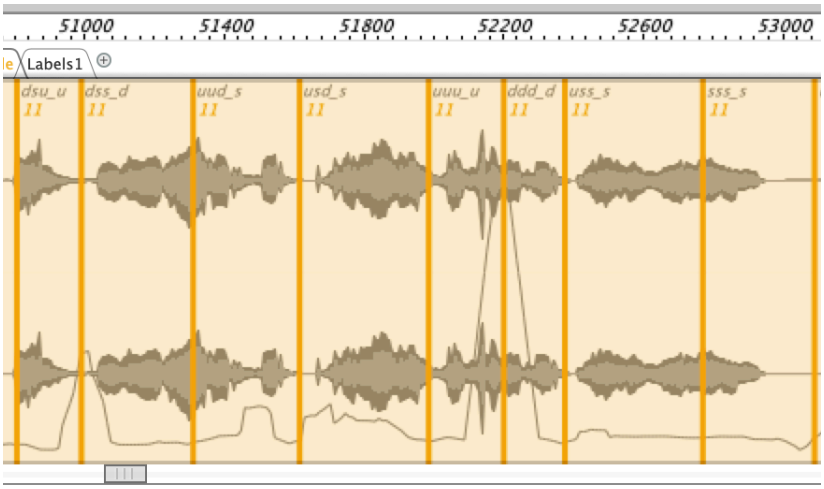

Figure 28: Checking prosodic labels

13 HAPPY OR NOT?

Again we have reached a point where we are either happy with the result or not.

13.1 Not Happy

Okay, you're not happy with the result. Depending on the problem you detected, you may need to consider a different segmentation (go back to 2, ***Segmentation (2)***), use a different

symbolic arrangement for prosodic labels (go back to 4, ***Prosodic profile (4)***), or change the audio descriptor or one of its parameters (go back to 5, ***Setting descriptors parameters (5)***).

13.2 Happy

If you are happy with the result now, just move on to the next step and export the text file.

14 EXPORT FILE MARKERS WITH PROSODIC LABELS (9)

Just click on the orange button to export the file text.

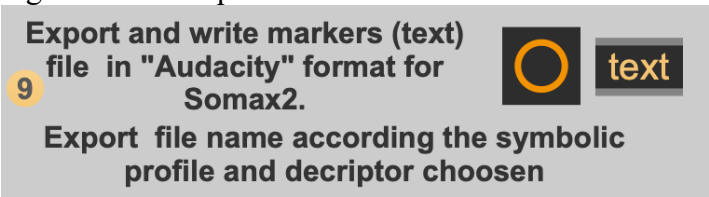

Figure 29: Export of markers and labels to build a corpus in Somax or an oracle in Omax

The exported text file will have an Audacity format (with tabs) with the third column being the prosodic labels.

0.341 0.861 dss_d
0.861 1.086 udu_u
1.086 1.471 dss_d
1.471 1.621 ssd_s
1.621 2.181 ssu_s
2.181 2.396 duu_u
2.396 2.701 dss_d

Again, this would allow you to import this data into Audacity to verify and possibly correct it. As a reminder, we believe that the best segmentation is done by hand, or at least verified and re-verified. The exported text file will be in Audacity format (with tabs) with prosodic labels in the third column. The name of the exported file will be based on the audio file name, the symbolic combination for the label, and the chosen audio descriptor. For example: “Breton_Andre_Lunion_Syllable_FundamentalFrequency_Slope_3Symb_GlobalSlopeYes”

15 ANNEXES

15.1 Project

This research is part of the project REACH supported by the European Research Council under Horizon 2020 program (Grant ERC-2019- ADG #883313) and project MERCI supported by Agence nationale de la Recherche (Grant ANR-19-CE33-0010).

15.2 Audacity file format

$$segment\_start_0\ TAB\ segment\_end_0\ TAB\ CR$$
$$segment\_start_1\ TAB\ segment\_end_1\ TAB\ CR$$
$$segment\_start_2\ TAB\ segment\_end_2\ TAB\ CR$$
$$\dots$$
$$segment\_start_n\ TAB\ segment\_end_n TAB\ CR$$

15.1 Final export file names

Text file names are composed as follows:
File_Name_Syllable_**Audio_Descriptor_Name_Slope_or_Register_How_Many_Symbols_Global_Slope (yes/no)**.txt

16 REFERENCES

Obin, Nicolas, François Lamare, and Axel Roebel. “Syll-o-matic: An adaptive time-frequency representation for the automatic segmentation of speech into syllables. ” *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013.