Automatic landmark detection

1 Articulatory landmarks

In 0_ema_cheat_sheet.pdf we saw the typical landmarks that are usually used in EMA studies. We saw also that there are multiple ways of detecting these landmarks automatically. A common approach is is the usage of 20% peak velocity. While such a description is not very precise, the approach is clear: the maximum and minimum peak velocities are found in a given window and based on these values the landmarks for the target, release, onset and offset are detected. Some researches use also the tangential velocity for detecting landmarks. This is the case in the program MVIEW for example, if multiple dimensions for a certain sensor are selected.

2 Script for landmark detection

The script "landmark_detection.py" allows you to test these approaches on real data. You can find the script on the GitHub repository. The data you can use to test can be found in the Session 3 folder in the google drive. This data is experimental data and consist of the following: 1) [aCa] sequences with several coronal target consonants 2) sentence productions with the carrier sentence "Innayam _ bahra" with differently structured target words in the middle

You will find .pos files, .wav files and annotations in the TextGrid format, in which only the segment boundaries for the target words are corrected.

In order to use the script, you need to have Python. It is best if you create a new environment:

[ ]:
```
conda create --name lm python=3.10
conda activate lm
```

It is also necessary that you install then the required packages:

[ ]:
```
pip tqdm numpy scipy pandas chardet scikit-learn matplotlib
```

This scripts detects landmarks automatically, based on phone alignments. This means that the script goes through a tier of the TextGrid, and if the label fits to your selected targets, the respective landmarks will be detected. The output are TextGrid files, that have tiers and landmark boundaries for each target sound in the annotation.

To use this script, you have to create a JSON files first. Find here information about a typical JSON file structure. For this specific script, the landmark_config.json can be found on GitHub. The structure is as follows:

```
{
"channels":{
            "LE":1,
            "RE":2,
            "TDOR":3,
            "TMID":4,
            "TTIP":5,
            "ULIP":6,
            "LLIP":7
            },
"LA": ["ULIP","LLIP"],
"tier":2,
"targets": {
            "k": {"channel":"TDOR_y",
                  "method": "vel20"
                },
            "t" : {"channel":"TTIP",
                    "method": "tvel20_windowed"
                },
            "b": {"channel": "LA",
                    "method": "vel20"
                }
            },
"padding":0.15,
"input_path": "/home/philipp/Workspace/data/",
"output_path": "/home/philipp/Workspace/TextGrid_output/",
"signal_filter": {
    "butter" : [25,5]
    }
}
```

This config file consists of the following parts: 1) channels enter here the channels names in quotation marks and the respective channel numbers after : For the examples here, there is no need to change this part 2) LA This part is for the cases in which lip aperture is important. Its value are the two channel that are used to compute the euclidean distance, combined in square brackets. If lip aperture is not needed, enter "None" as value. 3) tier In "tier" you have to enter the tier number of the TextGrids that contain the phone alignments. In the case of the sentece productions, this is tier 2. For the logatomes, this is tier 1. 4) targets In this part you have to enter each speech sound label, for which corresponding trajectory you want to detect the landmarks. Within `"targets": {}` the labels have to be structured as follows:

```
"phone label" : { "channel": "the channel name",
                  "method" : either "vel20", "vel20_windowed", "tvel20", "tvel20_windowed"
                }
```

It is important that for the channel name is exactly the one in the channels part. An explanation of the different methods is below. If you use "tvel20" and "tvel20_windowed" you have to enter only the name of the channel, e.g., "TTIP" for a coronal consonant. If you use "vel20" or "vel20_windowed" you have to specifiy the channel and the dimension, e.g., "TTIP_y". This is not the case if you use the lip aperture "LA" and "vel20"/"vel20_windowed".

5) padding This parameter determines the additional size of the window in seconds. This is constructed by the acoustic boundaries of the phone, adding the values, e.g., 0.15 s, to the left and the right

6) input_path This is the path to the folder that contains .pos, .wav and TextGrid files

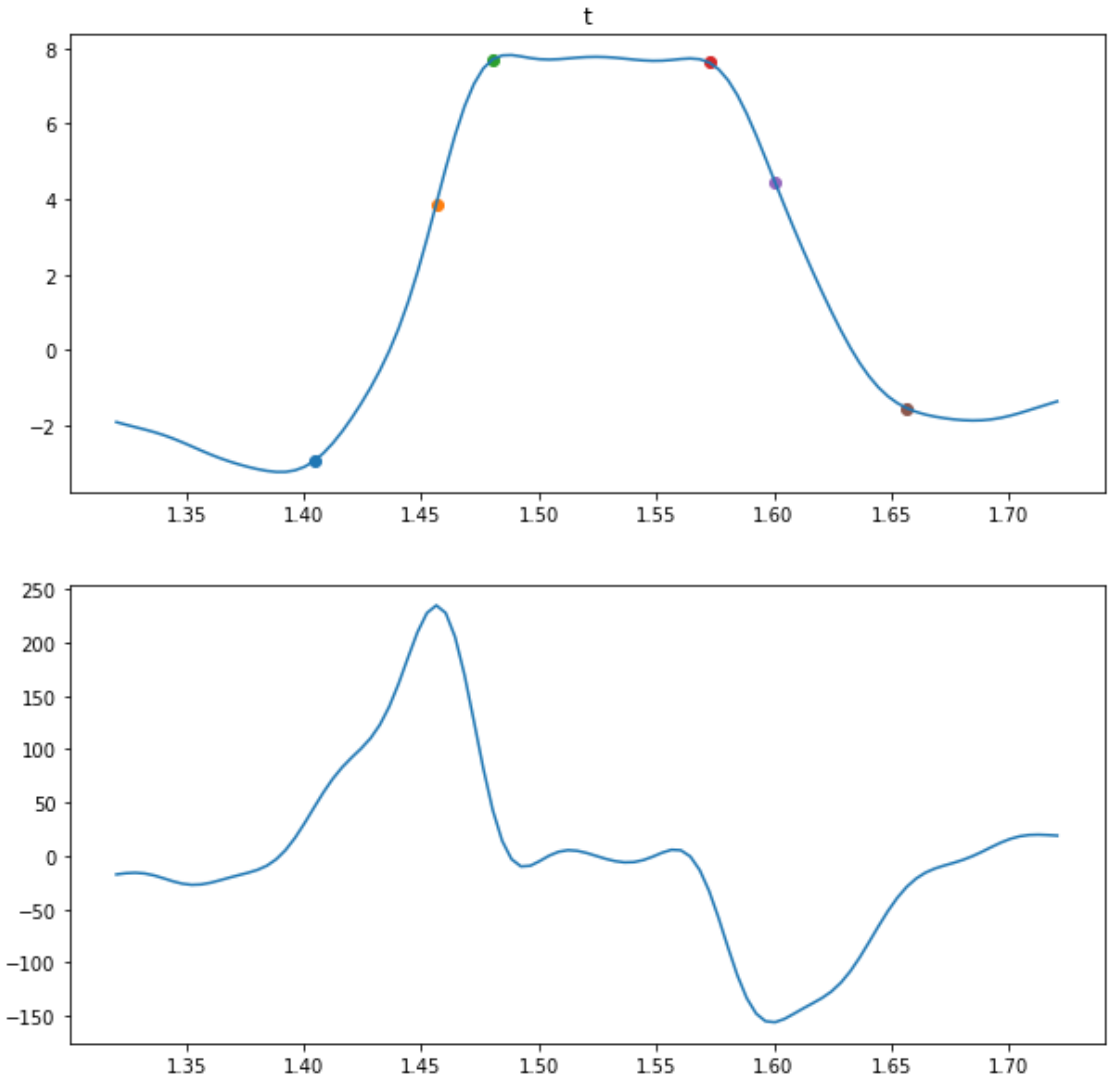
7) output_path This is the path to the folder in which the resulting TextGrids are saved. This folder has to be created first manually

8) signal_filter This parameter filters the data. You can leave this parameter as is

3 Detection methods

3.1 vel20

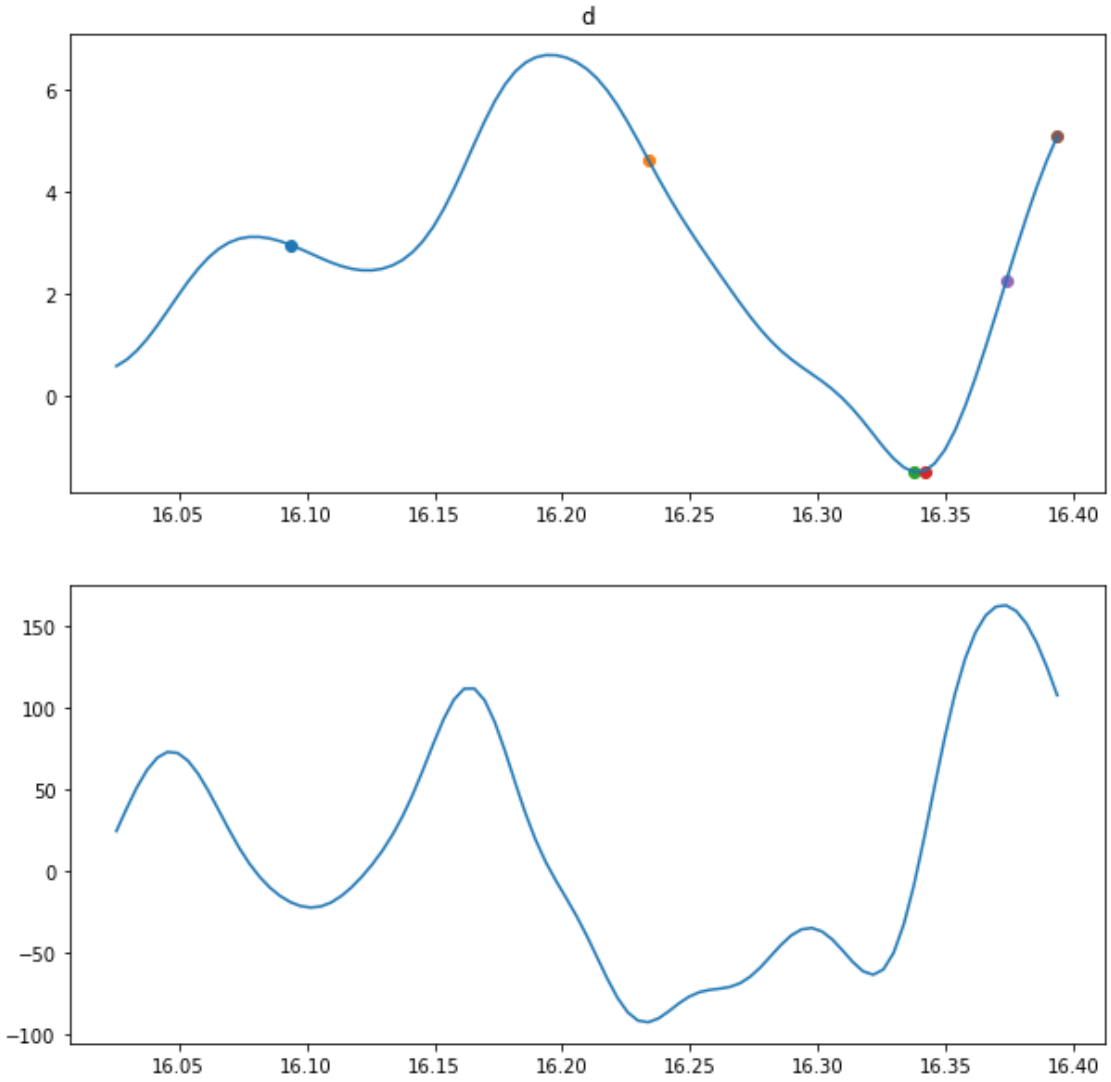
This method detects the landmarks of the onset, offset, target and release based on the 20% of the maximum and minimum peak velocities. This is a very common approach for landmark detection. An example can be seen here:



First the maximum and minimum peak velocity are detected. Then the target and the release landmark are found based on the 20% peak velocities in the range between pVel_to and pVel_fro. In the last step, the onset is detected from the beginning of the window to pVel_to and the offset in the range from pVel_fro to the end of the window. This is based again on 20% of the respective peak velocities.

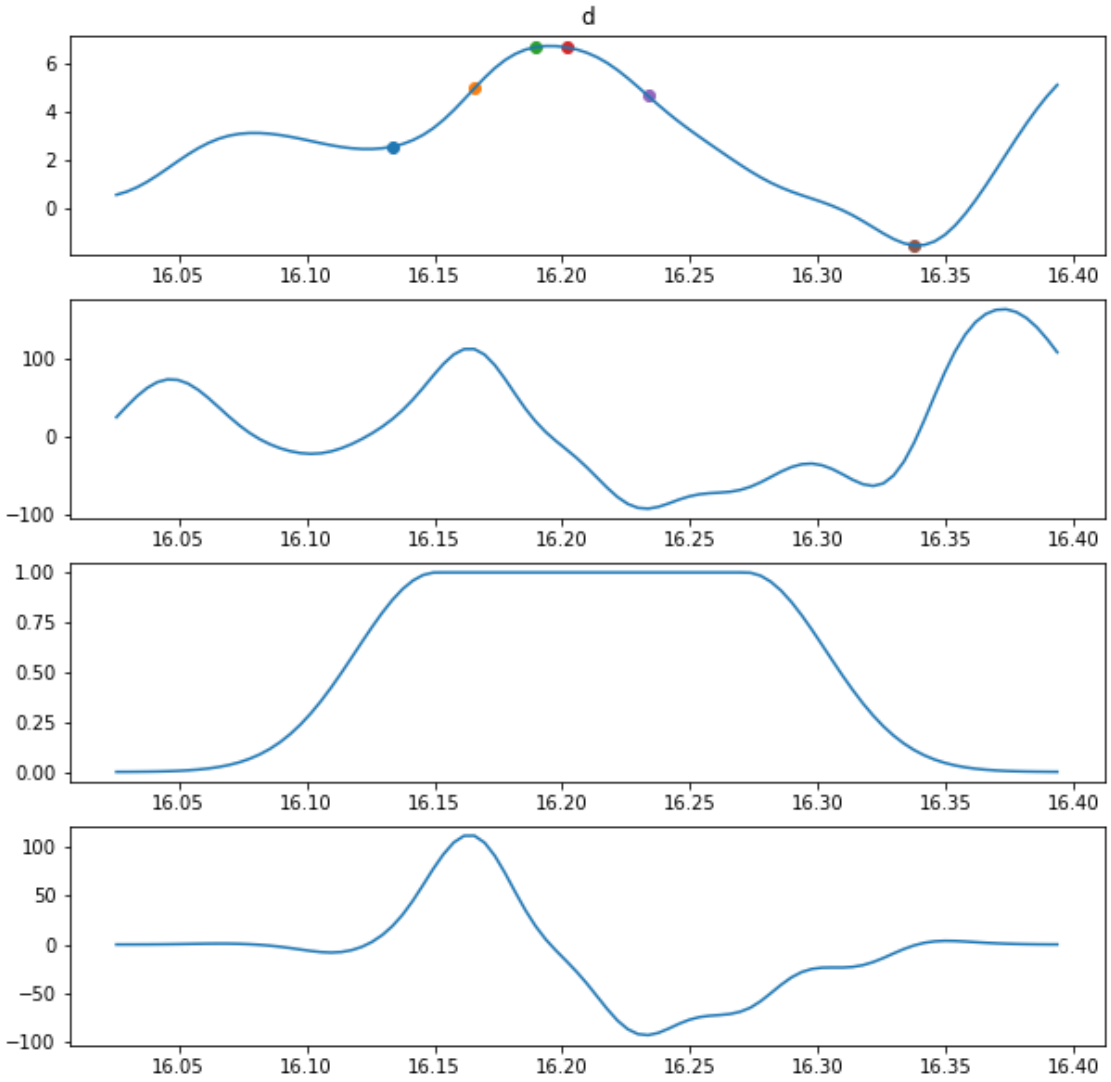
3.2 vel20_windowed

This 20% peak velocity method has a very important characteristic: It is prone to too large window sizes. If the selected window is too large, the maximum or minimum peak velocity may be found not for the gesture you are interested in, but for a another gesture before or after the segment. An example is this trajectory for [d]:



The tongue tip gesture for [d] is in the middle. The minimum peak velocity is detected correctly, but the maximum peak velocity is near the end to the right, which results in a mis-alignment.

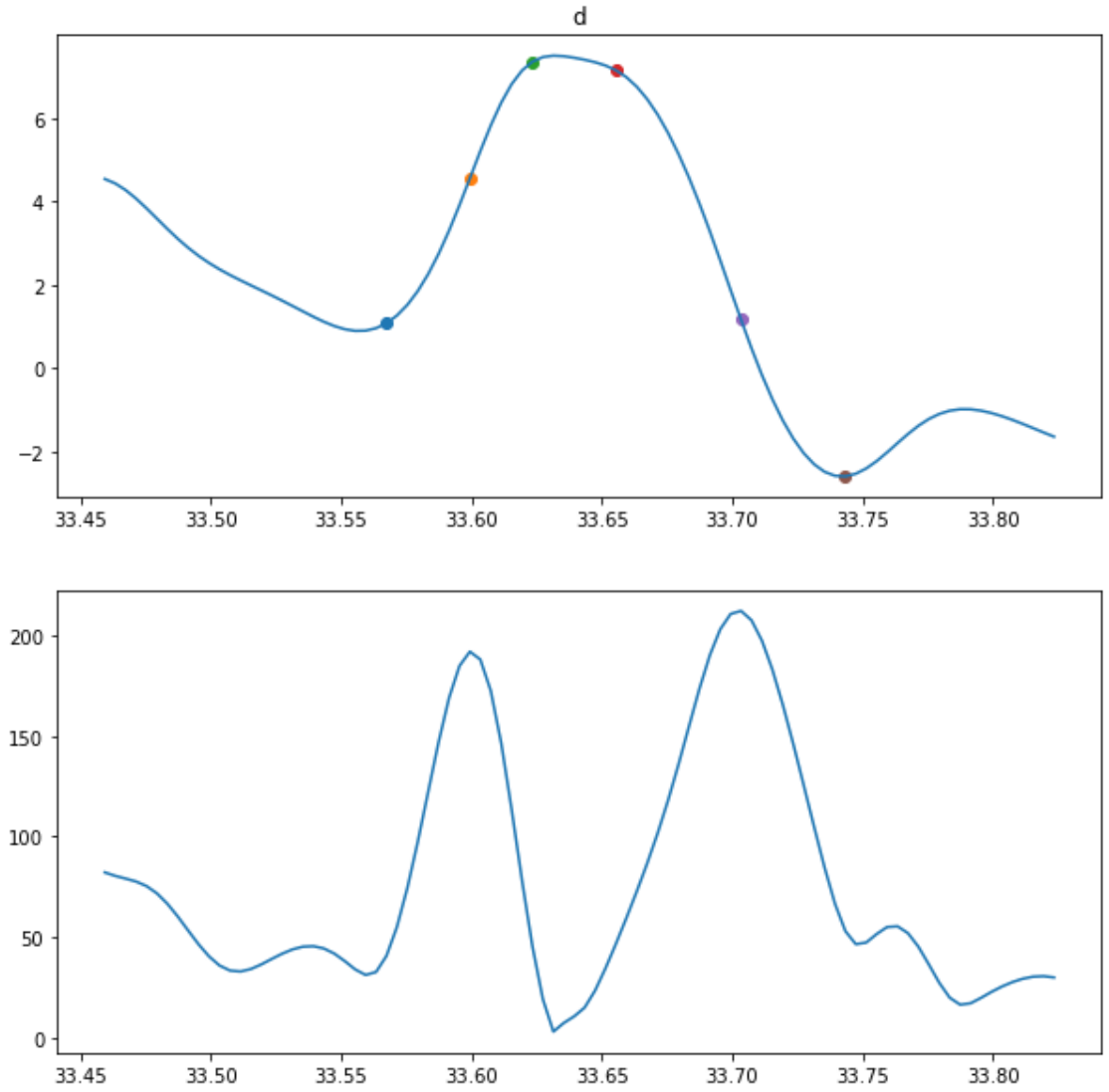
The vel20_windowed method solves this problem. The landmarks are detected based on 20% of the peak velocities, but not from the raw data. Instead, a window function is constructed that is stable in the range of the acoustic boundary with an additional range of 25 ms to the left and the right, which is roughly the range of the target and relase location. The further the velocities are away from this range, the stronger they are reduced. The resulting velocity trajectory maintaines the peak velocities of the gesture of interest, but the influence of other local minima and maxima is reduced. See the third row for this window function and the last row for the resulting velocity.



This approach makes the landmark detection more robust to large window sizes and this should solve problems in the overwhelming majority of the cases. However, if the window for this method is way too large (e.g., segment boundaries and 1 second before and after), this advantage of this method is canceled, but no one would use windows of this size.

3.3 tvel20

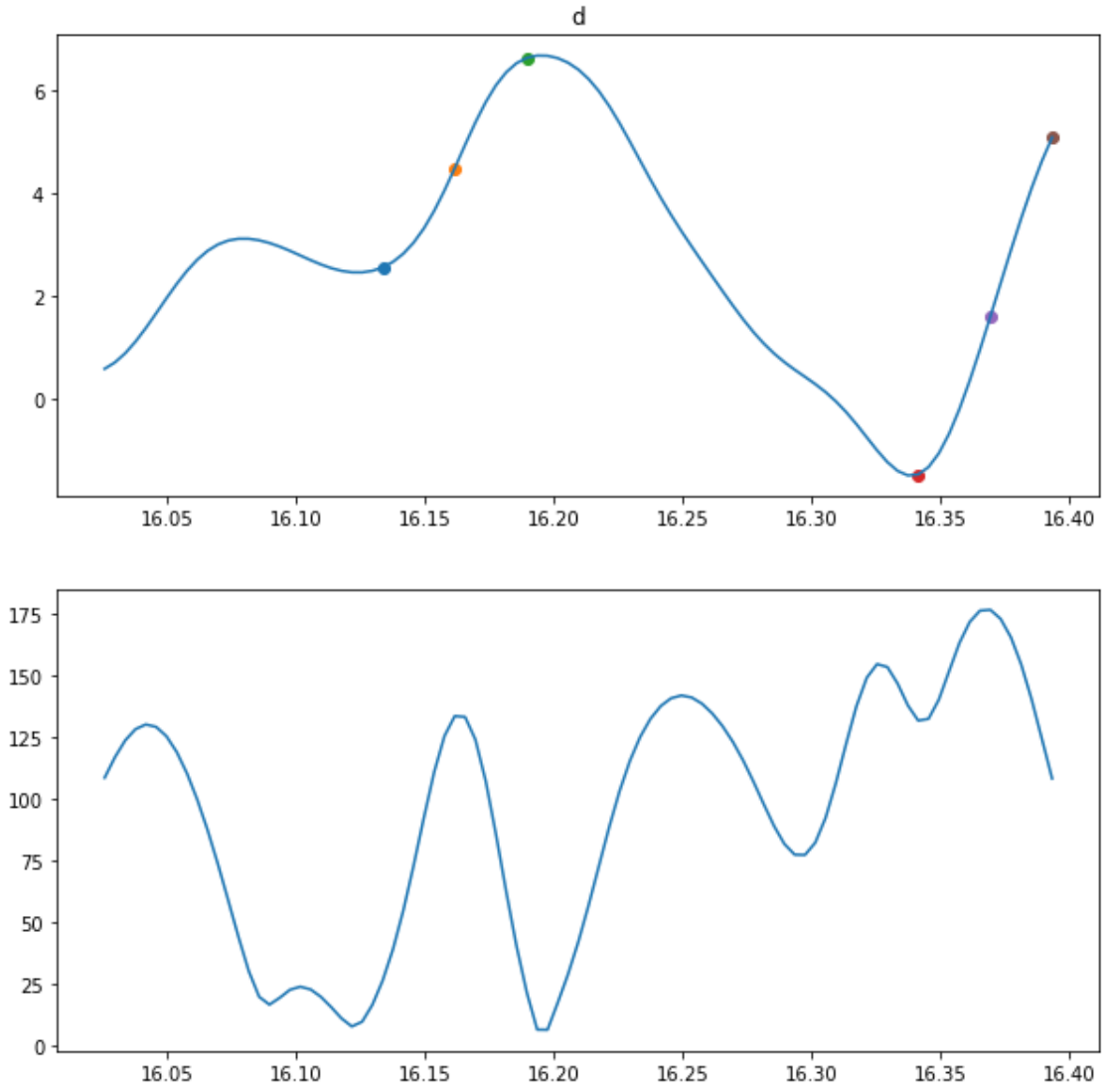
This method uses also 20% peak velocities, but this time from the tangential velocity as a basis. This algorithm looks for the maximum peak velocities in the left and the right half of the window. Afterwards, the respective 20% peak velocities for the target and the release landmark are detected between pVel_to and pVel_fro. Then, the onset is parsed from the beginning of the window to pVel_to, and the offset from pVel_fro to the end of the window.



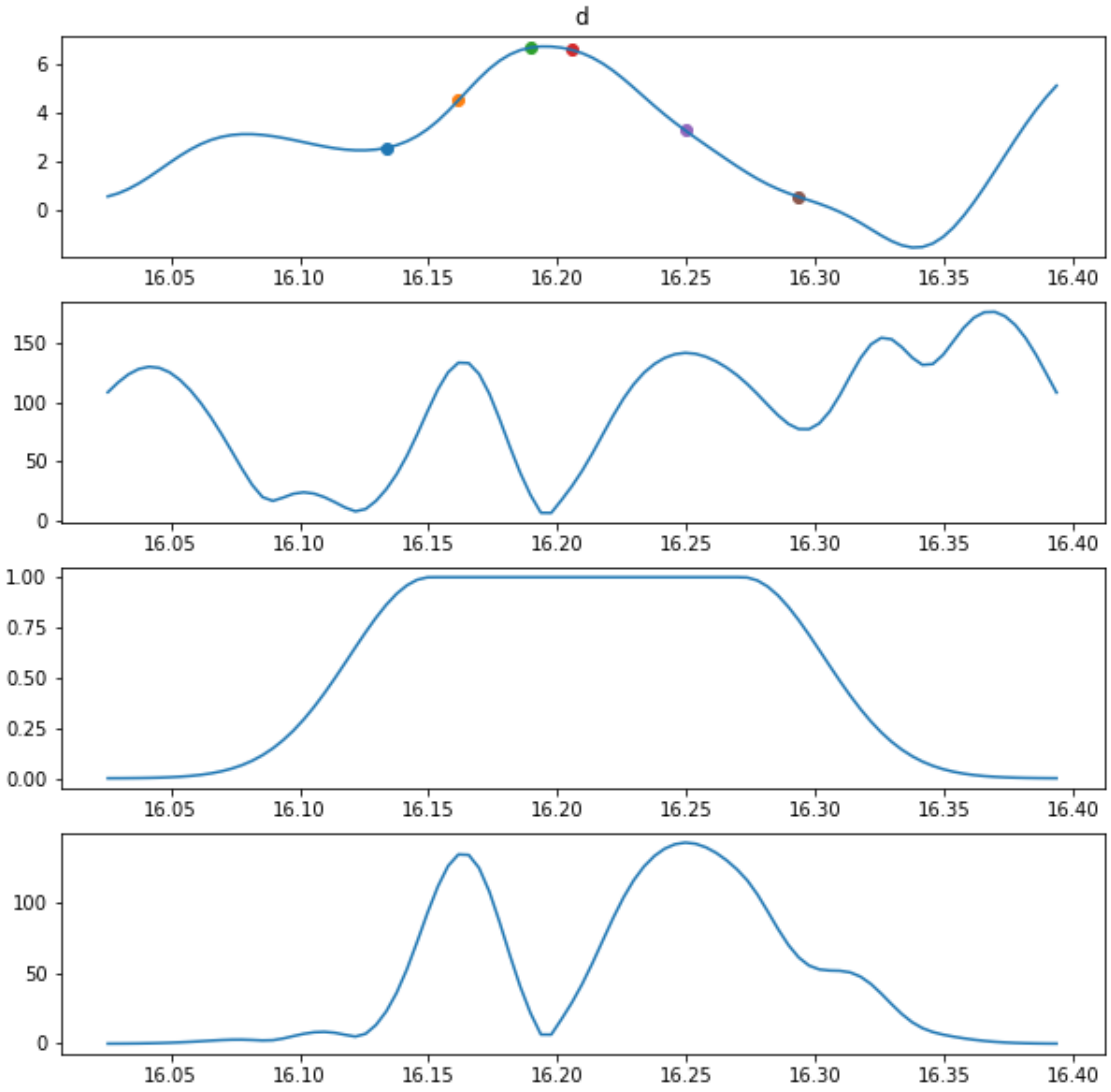
It should be noted that the algorithm as implemented in the script makes use of local minima. This means that the target, release, onset and offset are not detected within a large range such as the beginning of the window to pVel_to in the case of the offset. Rather local minima are used to restrict the range. The target landmark is detected between the pVel_to and the first local minimum after pVel_to. For the release, 20% of the peak velocity of the right half of the window is used and the location is computed between the minimum before pVel_fro and pVel_fro. The onset is found between the first minimum before pVel_to and pVel_to and the offset between pVel_fro and the first minimum after pVel_fro. This is necessary because otherwise the location of 20% peak velocity is relatively uncontrained and leads regularily to mis-alignments.

3.4 tvel20_windowed

The method of using 20% peak velocities in the tangential velocity has the same problems as the usage of the velocity: too large window sizes will result in errors.

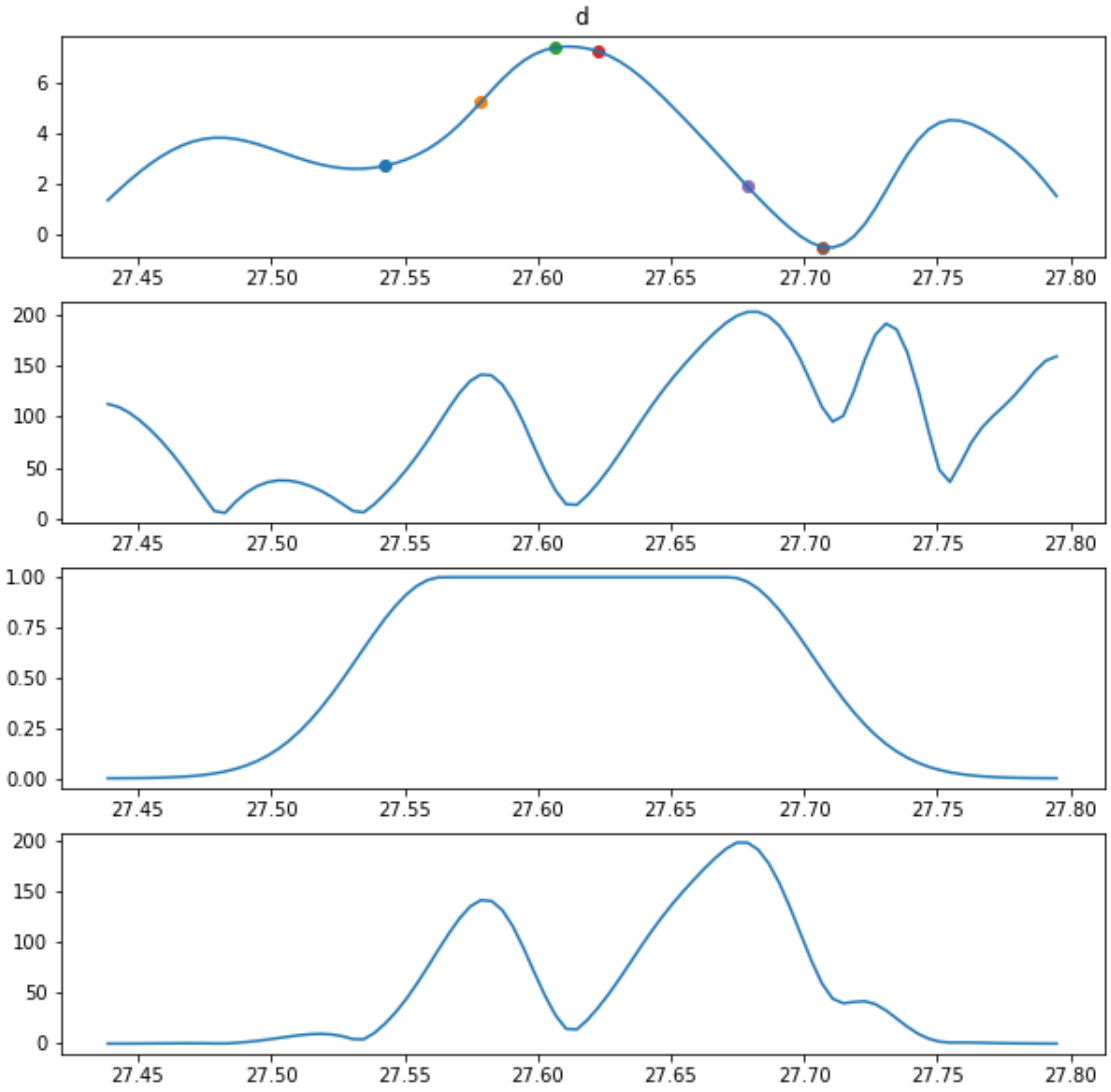


In order to improve this method, "tvel20_windowed" can be used an works in the same way as "vel20_windowed": a window function is constructed that reduces the influence of the tangential velocities outside of a stable range in the mid. The resulting alignments will be more accurate.



You may notice that the offset in this example is detected before a human annotator would set the landmark. Such cases are rare and may happen due to the tangential velocity profile. If only the original tangential velocity is used, the offset landmark would be at exactly the same location as it is depicted. On the other hand, one would say that using the 20% peak velocity of the modified tangential velocity in the entire right half from pVel_fro to the end of the window would be best, and indeed in this example this would be the better decision. However, applying this technique in general would result in more mis-alignments in other cases. Regarding the offset in this example, most people do not use the gestural offset in their research, and if such a mis-alignment happens, it can be easily corrected manually.

Other examples work well with this approach:



4 Running the script

After introducing the installation and the mulitple algorithms implemented in the script, the script can be run via the following command:

[ ]:
```
python landmark_detection.py -i path/to/landmark_config.json
```

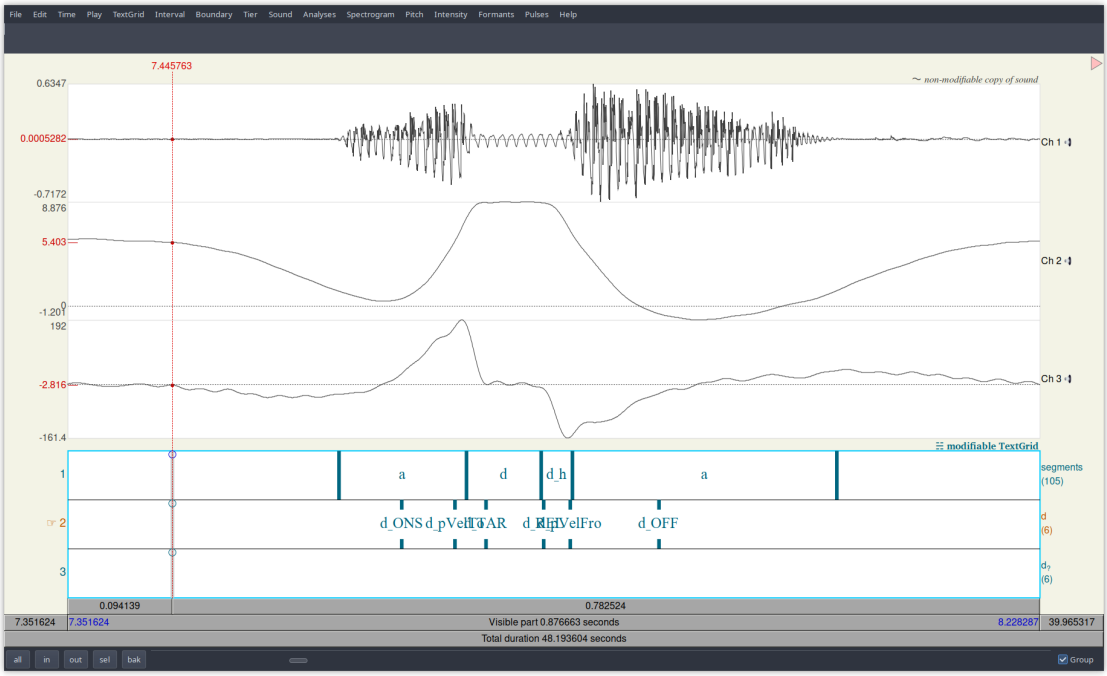
The latter part `-i path/to/landmark_config.json` is important, because the landmarkd_config.json contains all necessary information, especially the path to the input directory and the output folder.

You can test the script on the sentence and the logatome data and try different methods for the alignment.

If you run the script, you will see a progress bar in the command line and at the end you will see how many gestures were parsed and how many gestures are missing.

The output are TextGrids that are enriched with tiers that contain Point tiers with the landmarks. If there are gestures not parsed, you will see also an error_log.txt, that contains the symbols of the phones that were not parsed as well as their location (filename and start time of the segment).

The TextGrids can then be used together with ema2wav and its wav file output to check the data and correct them, if necessary:



Feel free to try out the script on the logatome and sentence data. Note: For the logatome data, phone alignments are in tier 1 and in tier 2 for the sentence data. You can test also the different methods for the alignment. In general, vel20 and tvel20 should perform worse and may lead to many unparsed gestures, while vel20_windowed and tvel20_windowed are more robust. However, it should be noted that there is no perfect alignment method up to now, and as it holds for automatic annotations in general, the alignments have to be manually checked and corrected, if necessary.

[ ]: