TargetOptimizer 2.0 User Manual

March 1, 2021

Contents

1

Introduction

This chapter will give a short introduction to getting TargetOptimizer 2.0 up and running and what to use it for. For more details please refer to chapter 2 and chapter 3.

1.1 Purpose

TargetOptimizer 2.0 (TO2) is a free and open-source PC software to estimate pitch targets parameters of the Target Approximation Model (TAM) from the f_0-contour of spoken utterances.
TO2 allows not only the reproduction of natural pitch contours with a small error but also yields more “natural” pitch targets in the sense of TAM, which can be used as an intonation model for text-to-speech synthesis. This simplifies the copy-synthesis of natural utterances using articulatory speech synthesizer.

1.2 Downloads

TO2 is distributed as free and open source software under the GNU General Public License (GPL). It is written in C++ and developed for both Windows and Linux platform. The software is free of charge and available at ☞ VocalTractLab-website or as a clone from ☞ git repository. Binaries for Windows OS and C++ sources of the software are included, together with some example files.

1.3 Installation

To install the software in Windows, simply unzip the downloaded file into a folder of your choice. The archive contains an executable (.exe) to start the Graphical User Interface (GUI), some dynamic link libraries (*.dll) and the example files.

1.4 Execution

The software can be executed as an application including a GUI or as a command line tool to support batch processing. To execute the software on Windows, simply run the application file “TargetOptimizer.exe” to start the GUI, or type “TargetOptimizer.exe -h” on the command line options for instructions. On Linux you have to build the project by yourself. For more information see section 1.5.
In addition, you can use the TargetOptimizer functions in the API within your own software. Some Matlab and Python code examples demonstrate how to use it. In chapter 3 it is shown how to use TO2 as a part of an articulatory synthesis pipeline.

1.5 Build instructions

Both the GUI and API version can be further developed and compiled by yourself. The GUI version contains a project file for Visual Studio 2019 in Windows, but it can also be compiled under Linux or Mac. The only external library required is the cross-platform GUI library ☞ wxWidgets.
The following indicates how to compile the source code of the software:

- Windows:
  Simply open the solution "TargetOptimizer.sln" and build in dependency of desired use case:
  - "Release" - for command line tool
  - "wxWidgets_Release" - for GUI version using wxWidgets
  - "wxWidgets_VCPKG_Release" - for GUI version using ☞ VCPKG and wxWidgets

  Note: "TargetOptimizer.exe" can always be found in the sub-folder "x64" in dependency on your chosen build configuration.

- Linux:
  Navigate inside the source folder and run one of the following commands:
  - For the command-line-only version:

```
g++ -std=c++14 -O3 -I.. ../dlib/all/source.cpp -fopenmp -lpthread
    -lX11 *.cpp -o TargetOptimizer
```

  - For the GUI version (requires wxWidgets):

```
g++ -std=c++17 -O3 -D USE_WXWIDGETS -I.. ../dlib/all/source.cpp
    -fopenmp -fpermissive -lpthread -lX11 *.cpp `wx-config --cxxflags
    --libs std` -o TargetOptimizer -w -lstdc++fs
```

1.6 Use and license information

As stated before, the offered software is free and open-source under the GNU General Public License.
You can refer the software by citing the most recent paper

✧ Paul Konstantin Krug, Simon Stone, Alexander Wilbrandt, Peter Birkholz. TargetOptimizer 2.0: Enhanced estimation of articulatory targets. [1]

This paper also includes more specific information on the changes between the TargetOptimizer 1.0 and 2.0. Note that the software comes with no warranty of any kind. There was no extensive test of the software, but the parts of the programs made available are considered relatively stable. The whole software is under continual development and may undergo substantial changes in future versions. Please feel free to report any bugs by using the ✉ issue tracker of the git repository.

2

Overview

In the following sections, the computational models underlying TO2 will be explained. The GUI and the usage of TO2 as commandline tool will be described in detail as well.
The main features of TO2 are:

- The f_0 contour in the voiceless parts of the original utterance is not interpolated before the target estimation compared to PENTAtrainer1 [2] and PENTAtrainer2 [3] because it is not guaranteed that the interpolated sections support the optimal reproduction of f_0 in the truly voiced sections.
- The targets for all syllables of the utterance are jointly optimized, instead of one syllable after the other.
- The regularization is used to prefer natural or plausible pitch targets, since different combinations of TAM parameters can generate almost identical f_0 contours.
- The TAM adopts 5th-order systems instead of 3rd-order systems, as they allow a more accurate modeling of pitch contours.
- The position of the target boundary positions are adjustable in the optimization process to obtain a better pitch contour in terms of minimized RMSE. The target boundaries can be as well initialized uniformly distributed across an utterance to use them as start parameters.
- The hyper-parameters of the optimization algorithm can be adjusted by the user, e.g., the iteration depth and the usage of early stopping.

2.1 Computational models

TO2 uses various models, including target approximation model, the models for estimation of target boundaries and early stopping. This section provides the most important information on those models from the papers [4] and [1], which you should consider reading if you wish to understand them in-depth.

2.1.1 Target Approximation Model

The Target Approximation Model (TAM) [2, 3] is a well-known intonation model that does not only allow the generation of very natural pitch contours, but can also be used to produce realistic trajectories of articulators, which is especially useful for articulatory speech synthesis [5]. To create highly realistic pitch or articulatory trajectories, natural acoustic or articulatory speech data can be used as a reference.
The TAM assumes one pitch target for each syllable of an utterance. According to [4], the target $x(t)$ is defined within the interval of a syllable as the linear function $x(t) = mt + b$, where m (in st/s) and b (in st) denote the slope and height of the target, respectively. The time t is defined relative to the onset of

the syllable for the interval $[0, d]$, where d is the syllable duration. The f_0 (in st) within the syllable is the response of a critically-damped low-pass filter of the order N (i.e., the concatenation of N identical first-order low-pass filters) with the time constant τ. At the beginning of the first syllable of an utterance, f_0 is given as an onset value ϕ and the derivatives of f_0 are set to zero. We used the time constant τ to characterize the linear system.
For the estimation of pitch targets based on TAM, we assume that the pitch contour is given in terms of samples $f_0(k\Delta t)$ in the voiced parts of the corresponding utterance, where Δt is the sampling interval (typically $10\,ms$) and k is the sampling index. We also assume that the syllable boundaries are given, and hence the target duration. The unknown parameters of the TAM are slope m_s, offset b_s, and time constant τ_s of every syllable (i.e., target) s. These parameters can be summarized in vectors $\mathbf{p}_s = (b_s, m_s, \tau_s)^{\mathrm{T}}$, where $s = 1, 2, \ldots, S$ and S is the number of syllables. The initial f_0 value ϕ of the TAM is set to the first f_0 sample of the utterance. In contrast to the previous approaches implemented in [3], we propose the joint estimation of all TAM parameters by regularized optimization, which has a better chance to find an optimal solution for the whole utterance. In addition, regularization helps to obtain solutions that are not only optimal in a mathematical sense but also physiologically most plausible. The proposed objective function to be minimized is

$$g(\mathbf{p}_1 \ldots \mathbf{p}_S) = \left\| f_0(k\Delta t) - \hat{f}_0(k\Delta t, \mathbf{p}_1 \ldots \mathbf{p}_S) \right\|_2^2 + \lambda \sum_{s=1}^{S} (\mathbf{p}_s - \bar{\mathbf{p}})^{\mathrm{T}} W (\mathbf{p}_s - \bar{\mathbf{p}}) \tag{2.1}$$

The first term on the right-hand side of Eq.2.1 is the squared Euclidean distance between the original f_0 samples in the voiced parts of the utterance and the corresponding values f_0 generated by the TAM. The second term is the regularization term that penalizes parameter values based on their deviation from the preferred values $\bar{\mathbf{p}} = (\bar{m}, \bar{b}, \bar{\tau})^{\mathrm{T}}$. The degrees of penalization for the different TAM parameters are adjusted by the elements of the weight matrix $W = \mathrm{diag}(w_m, w_b, w_\tau)$, and λ determines the overall degree of regularization. The weights were empirically adjusted to $w_m = 1\; s^2/st^2$, $w_b = 0.6\; st^{-2}$, and $w_\tau = 0.2\; s^2$. We furthermore set $\bar{m} = 0$ (static targets are preferred) and $\bar{\tau} = 12.5\,ms$ (this is considered as the "typical" time constant). $\bar{b}$ was set to the average pitch of the corresponding natural utterance. The bounds for τ were adopted from [3], and the bounds for b were adjusted to cover a frequency range of $76 - 767\,Hz$.
In principle, the optimization problem above can be solved using any method for bound-constrained optimization that requires only the availability of an objective function but no derivative information [6]. Here we used the algorithm Bounded Optimization by Quadratic Approximation (BOBYQA) [7], which is available as a C++ implementation in the modern open-source software toolkit dlib [8]. Like most optimization methods for non-convex problems, BOBYQA cannot guarantee to find the global minimum of the objective function. Accordingly, the method should be run multiple times with different random initial parameter values (within the respective bounds), and the best solution should be selected. Here we used $5S + 10$ random initialization per optimization, as the complexity of the problem increases with the number of syllables S.

2.1.2 Estimation of target boundaries

Within the TO2 framework, the parameter set is extended by a fourth parameter δ_B referred to as boundary delta. In contrast to the other parameters that are optimized as absolute values, the boundary delta is a relative value between -1 and 1. At each step of the BOBYQA optimization procedure, the onset of each target is changed relatively to the initial position of the corresponding boundary. Considering a number of N articulatory targets, the initial target onset positions are given by T_n (in ms), where $n = 1, \ldots N$. According to [1], the boundaries are shifted with respect to the duration of the preceding (succeeding) target if the boundary delta is negative (positive). Therefore, target boundaries can not shift beyond the initial onset boundaries of adjacent targets. However, shifted targets might move beyond a preceding shifted target. If this happens, boundaries are reordered to be monotonically increasing in time, so that the individual targets are always well defined and never overlap. Since there is no preceding target in the case of $n = 1$, the maximum negative time shift of the first boundary was defined as $-100\,ms$. Even though there should be no trajectory data before the first initial boundary, a shift of the first boundary towards an earlier time instant

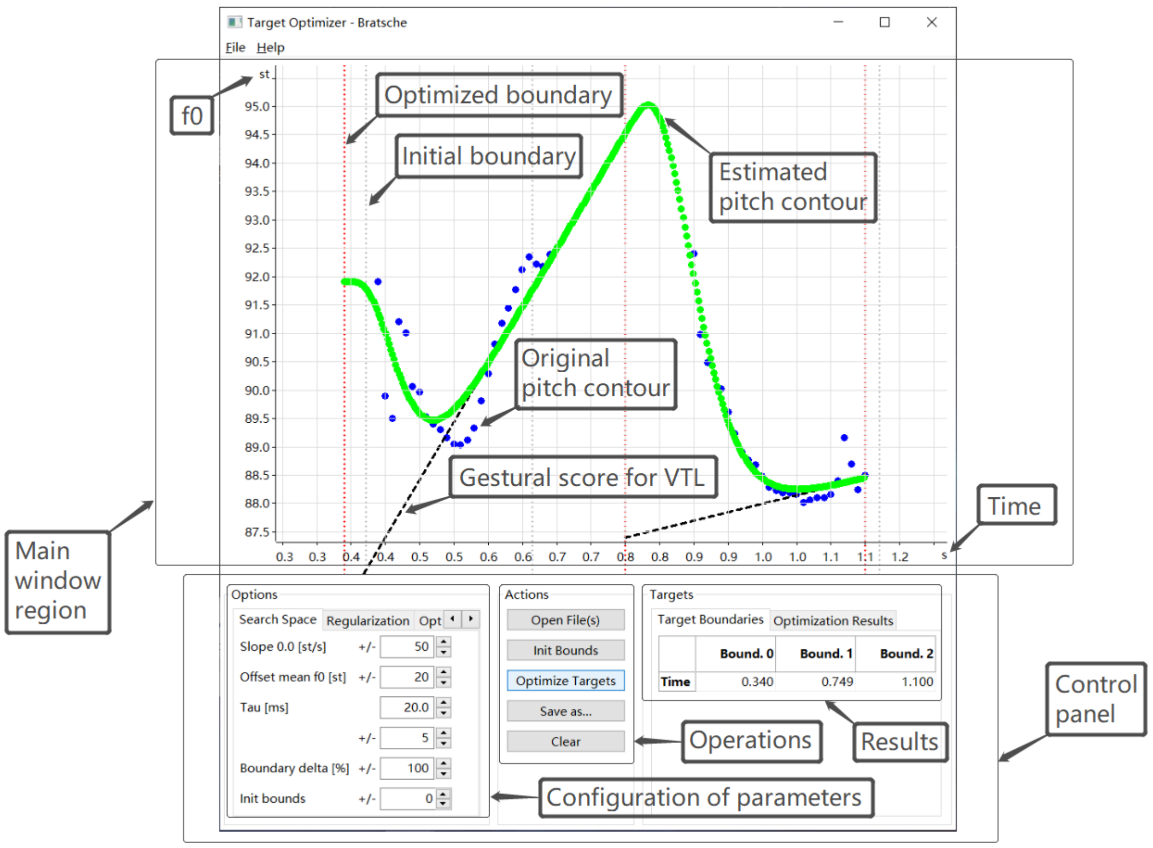
can be beneficial due to the initialization and the impulse response of the TAM filter. On the other hand, if the time shift of the first boundary is positive, it might move past the first data point. To prevent this, the first boundary is set to the time of the first data point in such a case. The offset of the last target N (boundary number $N + 1$) does not get optimized. It is always set equal to the time point of the last data point, since the region beyond the last data point has no influence on the fit.
In addition, the maximum possible boundary shift $[-\delta_{B,max}, +\delta_{B,max}]$ is controlled by user. If $\delta_{B,max} < 1$, the boundary optimization is limited to a respectively smaller region around the initial boundaries and setting it to zero restores the TO1 behavior (no boundary optimization). If boundary optimization is activated and $\delta_{B,max} = 1$, the optimal position of all target boundaries in principle can be estimated from any given initial configuration of boundaries. Therefore, an initialization by prior syllable annotation is no longer a necessary requirement as it was in TO1. Instead, the TO2 provides the possibility to initialize the boundaries at evenly spaced positions between the first and last data point.

2.1.3 Early stopping

Since the BOBYQA algorithm does not necessarily find the global minimum of the objective function in a single run, the optimization procedure must be executed multiple times, each time initialized with a different set of parameters drawn at random from the defined search space. In the following, such multiple runs are referred to as random iterations. The random iterations are independent from each other and the run that minimizes the objective function is selected as the optimization result. The number of random iterations is a tunable parameter in TO2 that can be adjusted by user. This is important because the boundary optimization added another dimension to the search space and the computational complexity of the problem increased. Therefore, runs with boundary optimization require a lot more random iterations to converge to the optimum, compared to runs without boundary optimization. On the other hand, additional random iterations become unnecessary once the optimum is found. In order to reduce the computational cost, a method referred to as early stopping has been developed to terminate the search prematurely as soon as a certain convergence criterion is met. Through the procedure of early stopping shown in [1], a temporary minimum of the TO objective function of each respective random iteration must be smaller than the current minimum (the global minimum across the multiple iterations) minus a percentage of the current minimum (determined by the hyper-parameter ϵ that can be set by user), in order to be counted as the new (temporary) global minimum. If the global minimum is updated, a convergence counter (that is increased by one after every iteration) is reset to zero. Once the convergence counter reaches a certain value referred to as patience, the search is terminated. The patience is another hyper-parameter that can be adjusted by user.
In addition, the BOBYQA optimization algorithm has two important hyper-parameters: the maximum number of cost evaluations N_f^{max} (the objective function evaluations per single random iteration), and the final trust region radius ρ_E. In TO2, both parameters can be adjusted by user. Nevertheless, since these parameters can have a huge impact on the performance and the computational costs, good default values were estimated via a grid search. Since this tuning of hyper-parameters was time-consuming from a computational point of view, it was not performed on the whole speech corpus, but only on a small subset of 8 utterances. In this subset, all syllable numbers were represented once. The parameter N_f^{max} was modified in ten equal steps from 10^5 to 10^6 and ρ_E was varied in four steps by four orders of magnitude from 10^{-6} to 10^{-3}. Each combination of utterances and parameters was evaluated, giving a total of 320 TO2 runs with boundary optimization. The number of random iterations per run was set to 1000 and early stopping was disabled.

2.2 Graphical User Interface (GUI)

As mentioned above, the TO2 can be used via a GUI. Fig.2.1 shows the layout of this GUI. From top to bottom, it consists of a title bar, a menu bar, the main window region, and the control panel.



Slope 0.0 [st/s] +/- 50
Offset mean f0 [st] +/- 20
Tau [ms] 20.0 +/- 5
Boundary delta [%] +/- 100
Init bounds +/- 0

	Bound. 0	Bound. 1	Bound. 2
Time	0.340	0.749	1.100

Figure 2.1: GUI of the TargetOptimizer 2.0

Menu bar

The menu bar has two menus “File” and “Help”.
The “File” menu includes six items: “Open File(s)” to import data from files, “Optimize Targets” to estimate the pitch contours, “Save as...” to export optimized data, “Init Bounds” to initialize boundaries, “Clear” to clean all data and “Quit” to close the software.
The “Help” menu includes two items: “Help” for instructions and “About” for the software information.
For the file formats, please refer to the Appendix A.

Main window region

The main window region shows the graphs of targets. The original and optimized pitch contours are shown in Fig.2.1 for the German word “Bratsche”. The pitch samples of the original utterance are shown as blue dots, whereas the green dots are the estimated pitch contour. The vertical green dash lines are the initial syllable boundaries, and the vertical red dash lines are the optimized boundaries.

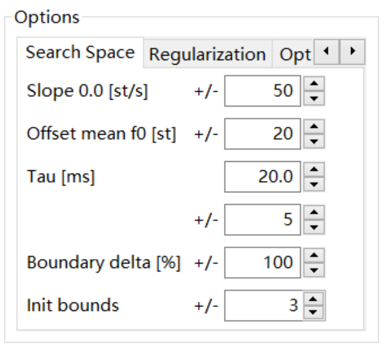
Control Panel

The control panel includes three regions: “Options” for parameters configuration, “Actions” for operations and “Targets” to show the optimized results.

① Options

The region “Options” contains the three tabs “Search Space”, “Regularization”, and “Optimizer” whose purposes will be explained subsequently:

- Search Space



Options

Search Space | Regularization | Opt

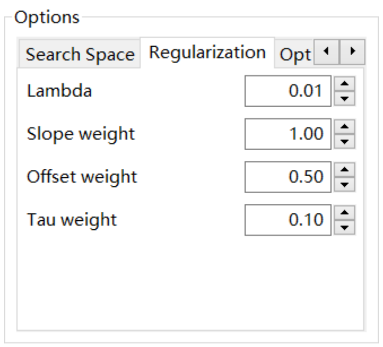
Slope 0.0 [st/s]	+/-	50
Offset mean f0 [st]	+/-	20
Tau [ms]		20.0
	+/-	5
Boundary delta [%]	+/-	100
Init bounds	+/-	3

Figure 2.2: Parameters of Search Space

This tab contains all the changeable parameters to determine the search-area of the TO2, as seen in Figure 2.2. These parameters and their properties are:

  – Slope: The maximum slope of each articulatory target, i.e., the slope parameters of the TAM,
  – Offset mean f0: The maximum offset of each articulatory target,
  – Tau: The time constant of the critically-damped low-pass filter whose response is f0,
  – Boundary delta: The boundaries shift factor to adjust the boundaries shift degree. (No boundary shift will be done if set to 0),
  – Init bounds: The amount of syllable boundaries when using the feature of the initialization of the uniformly boundary positions for the utterance.

- Regularization



Options

Search Space | Regularization | Opt

Lambda	0.01
Slope weight	1.00
Offset weight	0.50
Tau weight	0.10

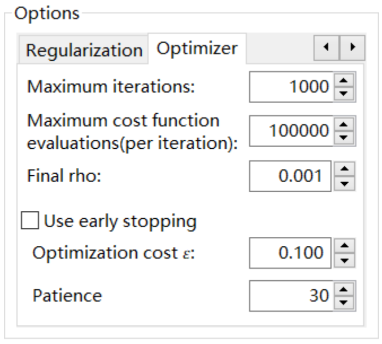
Figure 2.3: Parameters for Regularization

In this tab all controllable regularization parameters are listed, as seen in Figure 2.3. These parameters and their properties are:

  – Lambda: The regularization factor to determine the overall degree of the regularization,

  - Slope weight, Offset weight, Tau weight: Adjust the degrees of penalization for the different TAM parameters.

- Optimizer



Options

Regularization | Optimizer

Maximum iterations: 1000

Maximum cost function evaluations(per iteration): 100000

Final rho: 0.001

☐ Use early stopping

Optimization cost ε: 0.100

Patience 30

Figure 2.4: Hyper-parameters of Optimizer

  In this tab all controllable optimization parameters are listed, as seen in Figure 2.4. These parameters and their properties are:

  - Maximum iterations: The maximum times of passes of the optimization procedure,
  - Maximum cost function evaluations (per iteration): The maximal objective function evaluations of each single random iteration,
  - Final rho: The final trust region radius,
  - Use early stopping: To terminate the search prematurely as soon as a certain convergence criterion is met,
    - Optimization cost ϵ: The optimization factor in terms of the current minimum of TO objective function,
    - Patience: The threshold of the convergence counter.

② Actions

As shown in Figure 2.5, the items in Actions are equal to the items in the "File" menu, i.e.:

- Open File(s): Import the pitch samples of an utterance from a *.PitchTier file, import the target boundaries from *.TextGrid files selecting the "Position" option. *.PitchTier- and *.TextGrid-files can be imported at the same time by selecting both files in one command window,

- Init Bounds: Initialize as many equidistant boundaries as set in "Init bounds" value in the "Search Space" options according to the utterance-length,

- Optimize Targets: Starts the optimizer with the configured parameters to estimate the pitch contours and boundaries,

- Save as...: Export the optimized targets as a table (*.csv file) with pitch target parameters, as a gestural score (*.ges file) to use it in articulatory speech synthesizer like ☞ VocalTractLab (VTL), or as a PitchTier file with the f_0 samples of the synthesized pitch contour for ☞ Praat,

- Clear: Clean all data.

For more information of file formats, please refer the Appendix A.

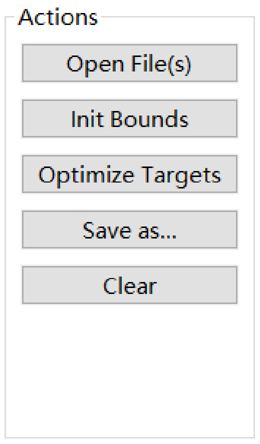
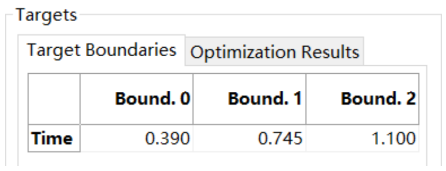


Figure 2.5: Actions

③ **Targets**

Corresponding to the graphs shown in the main window region, the optimized target results are listed in two tables:

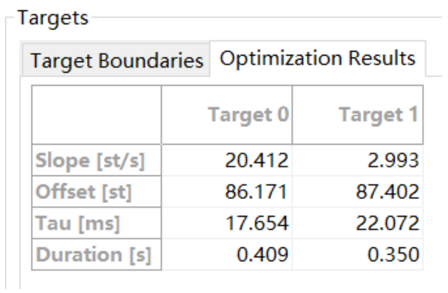
- "Target Boundaries" (see Figure 2.6) presents the values of optimized target boundaries.



	Bound. 0	Bound. 1	Bound. 2
Time	0.390	0.745	1.100

Figure 2.6: Target boundaries

- "Optimization Results" (see Figure 2.7) presents the value of optimized pitch contour parameters including slope, offset, tau and the time duration of each target.



	Target 0	Target 1
Slope [st/s]	20.412	2.993
Offset [st]	86.171	87.402
Tau [ms]	17.654	22.072
Duration [s]	0.409	0.350

Figure 2.7: Optimization results

2.3 Command line tool

To use the software as a command line tool, open the Command Line in Windows or Terminal in Linux and change the directory to the corresponding TargetOptimizer folder, then you can execute the command "TargetOptimizer -h" to see the following instructions:

```
Usage: TargetOptimizer <PitchTier-file> <LOG-name> { <options> | <arg> }
Options:
 -h                              Display this help message.
Additional Parameter Options:
 --b-range <arg>                 Specify search space for offset parameter.
 --b-weight <arg>                Specify regularization weight for offset
     parameter.
 --boundaryDelta <arg>           Specify boundary delta.
 --epsilon <arg>                 Specify the epsilon for early stopping
 --initBounds <arg>              Specify how many bounds should be initialized.
 --lambda <arg>                  Specify regularization parameter.
 --log                           Outputs f0, targets, boundaries, etc. to a
     single text file.
 --m-range <arg>                 Specify search space for slope parameter.
 --m-weight <arg>                Specify regularization weight for slope
     parameter.
 --maxCostEvaluations <arg>      Specify maximum number of costfunction
     evaluations.
 --maxIterations <arg>           Specify maximum number of optimizer iterations.
 --patience <arg>                Specify the patience for early stopping
 --rhoEnd <arg>                  Specify the rho_end parameter.
 --t-range <arg>                 Specify search space for time constant
     parameter.
 --t-weight <arg>                Specify regularization weight for time constant
     parameter.
 --tg <arg>                      Specify a TextGrid file.
 --useEarlyStopping              Stop the search early if fmin-ftmp < epsilon
     for times (x=patience).
 --utterance <arg>               Name of the utterance to optimize.
Output Options:
 -c                              Choose for csv table file.
 -g                              Choose for VTL gesture file.
 -p                              Choose for PitchTier file.
```

For a (non-advanced) pitch contour extraction for the german word "Buch" the corresponding command would be

```
TargetOptimizer Buch.PitchTier --tg Buch.TextGrid,
```

whereas an analysis with custom paramaters could look like

```
TargetOptimizer Buch.PitchTier --tg Buch.TextGrid --lambda 0.2 --patience 40
```

3

TO2 as part of natural articulatory speech synthesis

One of TO2's main purposes is to act as an interface between an audio analysis program, mainly Praat (☞ Download), and an articulatory speech synthesizer, namely VocalTractLab (VTL) (☞ Download), to obtain optimized pitch-curves for the synthesis of more natural speech. Figure 3.1 illustrates this workflow and shows which interface information is required and produced at each step. The input data for the TO2 are a TextGrid file with syllable boundaries (which is optional since v2.0) and a PitchTier file with the f_0 samples to be reproduced by TAM. Both files are created with the software Praat (☞ Download) from the audio file of the original utterance. The parameters of TO2 can be configured by using the GUI or the command line tool. The optimized results from TO2 can be saved as a gestural score (*.ges file) which can then be used inside of VTL.

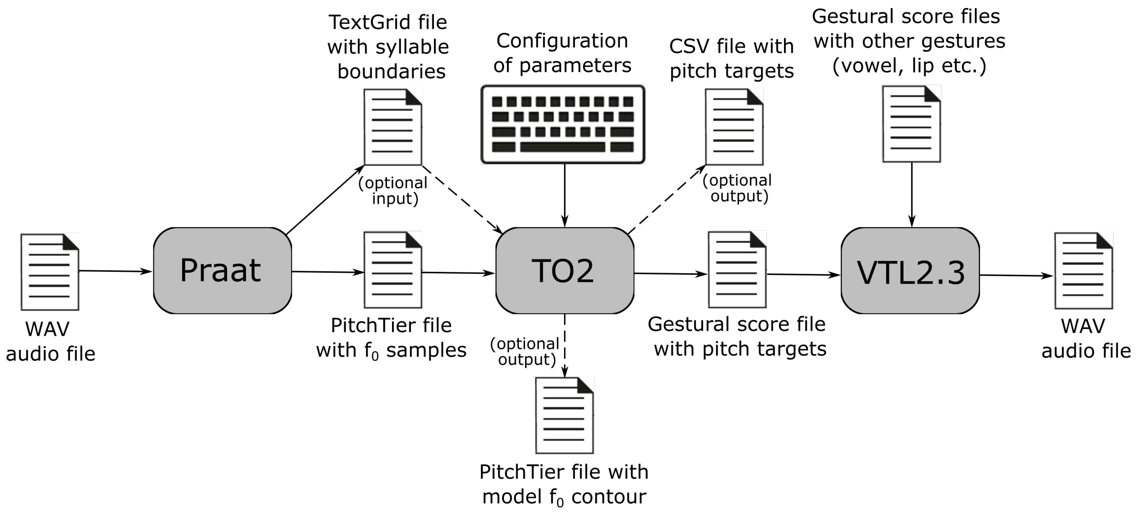

TextGrid file with syllable boundaries
Configuration of parameters
CSV file with pitch targets
Gestural score files with other gestures (vowel, lip etc.)
(optional input)
(optional output)
WAV audio file
Praat
TO2
VTL2.3
WAV audio file
PitchTier file with f_0 samples
(optional output)
Gestural score file with pitch targets
PitchTier file with model f_0 contour

Figure 3.1: Articulatory speech synthesis pipeline

In the following sections this workflow is shown step by step for the german word "Ästhetik".

3.1 Praat

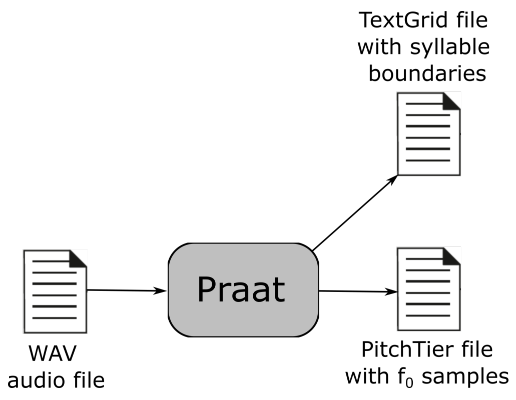


Figure 3.2: Input and outputs of Praat

Praat is used to obtain pitch frequencies and segment boundaries from a spoken utterance. Those information can be exported as a TextGrid file for the segment boundaries or as a PitchTier file for the pitch. Therefore the utterance is imported into Praat and segmented. Segmentation takes place on syllable level and not phonetically. The pitch analysed is then extracted and those information exported.

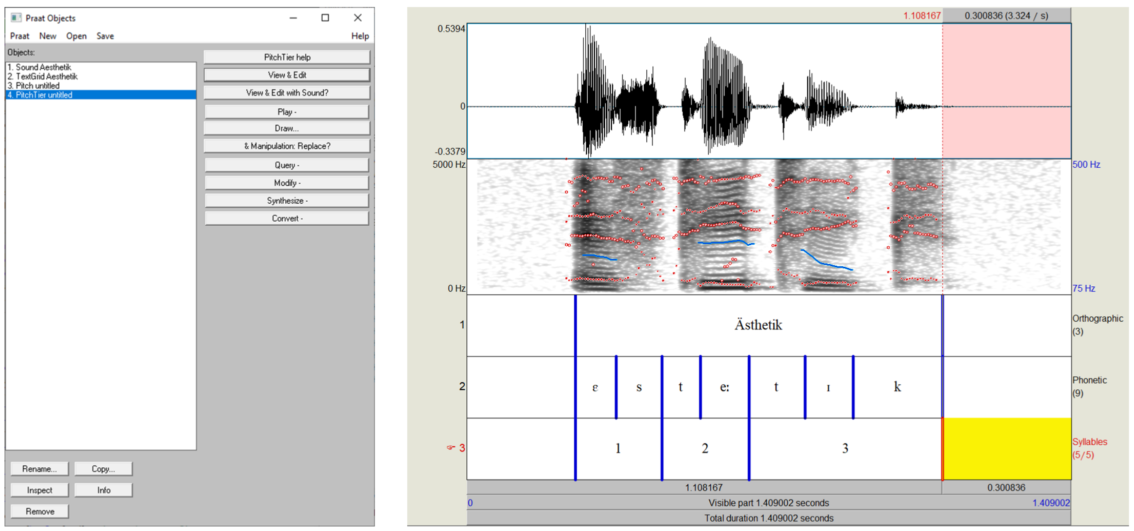


Figure 3.3: Praat GUIs: a) Objects Window, b) Editor Window

This procedure can be done with the following steps:

1. In the Objects Window (see Figure 3.3 a)) import the sound file you want to analyse via “Open” → “Read from file...” or create a new record via “New” → “Record mono Sound...”. A new Sound Object appears in the Objects Window list.
2. Select this Sound Object and use the function “Annotate” → “To TextGrid...” → <insert the tier names e.g. [Orthographic Phonetic Position]> to obtain a new empty TextGrid which appears in the Objects Window list.

3. Select both the Sound Object and the TextGrid Object and use the function View & Edit to open both in the Editor Window. In this window you can segment and label the utterance in the tiers as you need it. Every change will be saved automatically in the TextGrid Object.

4. In the Editor Window click on the menu “Pitch” → “Extract visible Pitch contour” to extract the automatically analysed pitch frequencies. A new Pitch Object appears in the Objects Window.

5. Select the Pitch Object in the Object Window and use the function “Convert” → “Down to PitchTier”. This creates a PitchTier Object.

6. To export the pitch frequencies in the PitchTier Object click on the menu ‘Save” → “Save as PitchTier spreadsheet file...” and save the PitchTier file to your desired folder.

7. To export the target boundaries in the TextGrid Object click on the menu ‘Save” → “Save as text file...” and save the TextGrid file to your desired folder. Be aware that the TO2 needs the TextGrid to be ecnoded in UTF-8 format. If you have special letters in the TextGrid file Praat is not able to convert this file to UTF-8. Please use another program to convert this file, e.g. Notepad++., before importing it in TO2.

3.2 TO2

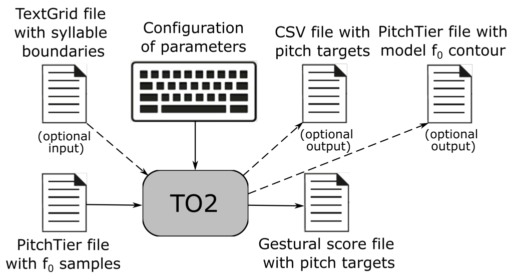
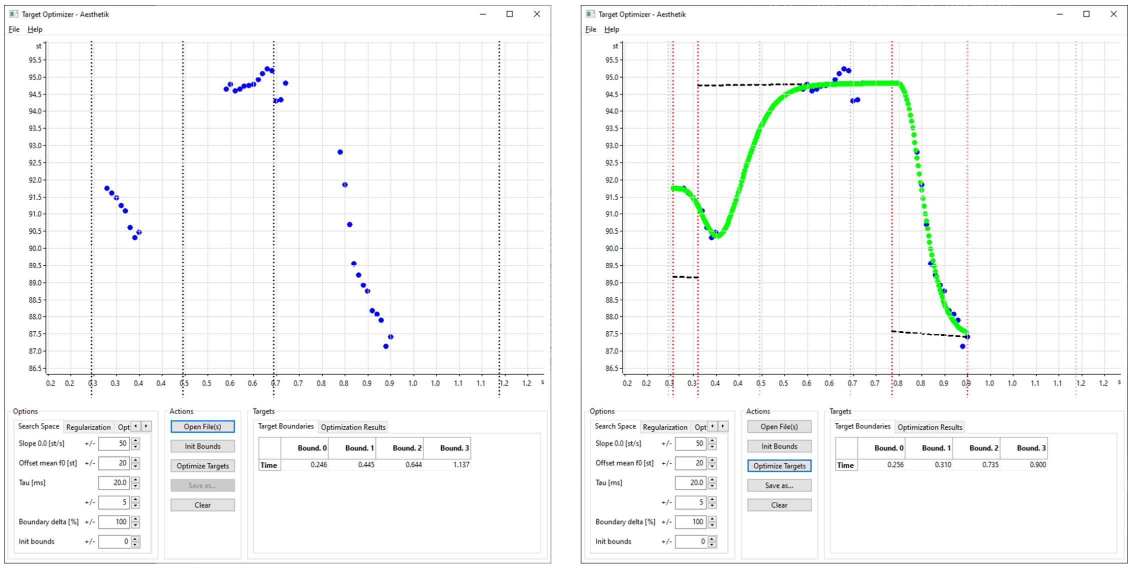

TextGrid file with syllable boundaries
Configuration of parameters
CSV file with pitch targets
PitchTier file with model f_0 contour
(optional input)
(optional output)
(optional output)
TO2
PitchTier file with f_0 samples
Gestural score file with pitch targets

Figure 3.4: Input and outputs of TO2

The TO2 uses the PitchTier file and (not necessarily) the TextGrid file to perform a target optimization and obtain the pitch targets.



a)

Slope 0.0 [st/s] +/- 50
Offset mean f0 [st] +/- 20
Tau [ms] 20.0 +/- 5
Boundary delta [%] +/- 100
Init bounds +/- 0

	Bound. 0	Bound. 1	Bound. 2	Bound. 3
Time	0.246	0.445	0.644	1.137

b)

Slope 0.0 [st/s] +/- 50
Offset mean f0 [st] +/- 20
Tau [ms] 20.0 +/- 5
Boundary delta [%] +/- 100
Init bounds +/- 0

	Bound. 0	Bound. 1	Bound. 2	Bound. 3
Time	0.256	0.310	0.735	0.900

Figure 3.5: TO2 GUI: a) with imported PitchTier and TextGrid and b) with optimized pitch contour

Following steps are necessary to successfully estimate targets for an utterance:

1. Open the pitch contour from the Praat PitchTier file.
2. Input the corresponding boundaries. This can be performed by opening a Praat TextGrid file (UTF-8 encoded) or by using the equidistant boundary function of the TO2. To set equidistant boundaries type the amount of desired boundaries into "Init bounds" option and click the "Init Bounds" button. Afterwards the graph looks similar to Fig. 3.5a).
3. Carefully think about the parameters you want to change (the default values are generally well suited).
4. To perform the Target Approximation click the"Optimize" button. After the calculation the interpolated pitch contour is visible in the graph (see Figure 3.5b)).
5. Export the results as a *.ges file for VTL.

3.3 VTL2.3

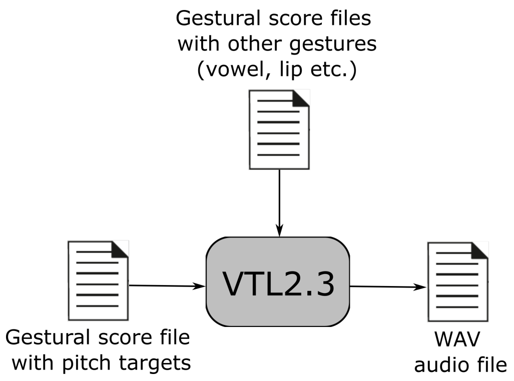


Figure 3.6: Input and outputs of VTL

The optimized pitch contour can now be used for articular speech synthesis in VTL. This can be done as follows:

1. To import the pitch-contour information press “File” → “Load gestural score” and select the desired *.ges file.
2. Click the “Gestural score” in the toolbar to go to the gestural score page. In the gestural line “F0 gestures” the information is now loaded and can be used for the synthesis.

A

File formats

- **PitchTier file (*.PitchTier)**
  PitchTier is one of the types of objects in Praat that represents a time-stamped pitch contour, i.e., it contains a number of (time, pitch) points, without voiced/unvoiced information. The PitchTier objects are used for two purposes: for manipulating the pitch curve of an existing sound and for synthesizing a new sound.
  For more information about PitchTier, please refer to ☞ PitchTier.

- **TextGrid file (*.TextGrid)**
  TextGrid is one of the types of objects in Praat, which is used for annotation (segmentation and labelling). A TextGrid object includes two kinds of tiers: an interval tier is a connected sequence of labeled intervals, with boundaries in between; A point tier is a sequence of labeled points. It is important that the TextGrid file is encoded in UTF-8.
  For more information about TextGrid, please refer to ☞ TextGrid.

- **Gestural score file (*.ges)**
  A gestural score file is an XML file that defines a gestural score. The gestural score is an organized pattern of articulatory gestures for the realization of an utterance. The root element of the file is *gestural_score*. There are eight tiers of gestures in a gestural score, each of which is represented by one *gesture_sequence* element. Each gesture sequence comprises a set of successive gestures of the same type. The start time of a gesture is implicitly given by the sum of durations of the previous gestures of the sequence.
  For more information about gestural score, please refer to ☞ VTL2.3-manual.

B

Major Changes from version 1.0 to 2.0

- I/O routines are enhanced which makes the software much more flexible and robust against undefined behavior induced by wrong file types.
- We tuned all free parameters on a test corpus to find optimal default values.
- We developed a new GUI based on the wxWidgets C++ library.
- We have extended a command line functionality.
- The position of target boundaries is now a fully optimizable parameter, which can be freely moved up to 100% of syllable long.
- The amount of bounds can be manually set according to the utterance without loading the corresponding TextGrid file.
- The time constant τ can be manually adjusted.
- Optimizer options such as: restrain the maximal iteration and cost function, use the early-stopping etc. are added to avoid over-fitting and reduce the computational overload.

C

Credits

C.1 Developers of TargetOptimizer 1.0

TargetOptimizer 1.0 was developed by Patrick Schmager and has been released under supervision of Peter Birkholz. TO1 has served as basis for the developement of TO2, which has been described in preceding chapters.

C.2 Developers of TargetOptimizer 2.0

TargetOptimizer 2.0 was deveveloped by Paul Konstantin Krug, Simon Stone (né Preuß) and Alexander Wilbrandt. It has been released under supervision of Peter Birkholz.

C.3 Documentation

This manual has been drafted by Arne-Lukas Fietkau and Yifei Li.

C.4 Supervision

The coding process and the final release have been supervised by Peter Birkholz.

Acknowledgments

Bibliography

[1] Paul Konstantin Krug, Simon Stone, Alexander Wilbrandt, and Peter Birkholz. Targetoptimizer 2.0: Enhanced estimation of articulatory targets. 2021.

[2] Yi Xu and Q Emily Wang. Pitch targets and their realization: Evidence from mandarin chinese. *Speech Communication*, 33(4):319 – 337, 2001.

[3] Santitham Prom-On, Yi Xu, and Bundit Thipakorn. Modeling tone and intonation in mandarin and english as a process of target approximation. *The Journal of the Acoustical Society of America*, 125:405–24, 02 2009.

[4] Peter Birkholz, Patrick Schmager, and Yi Xu. Estimation of pitch targets from speech signals by joint regularized optimization. In *2018 26th European Signal Processing Conference (EUSIPCO)*, pages 2075–2079. IEEE, 2018.

[5] Peter Birkholz. Modeling consonant-vowel coarticulation for articulatory speech synthesis. *PloS one*, 8(4):e60603, 2013.

[6] Luis Miguel Rios and Nikolaos V Sahinidis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293, 2013.

[7] Michael JD Powell. The bobyqa algorithm for bound constrained optimization without derivatives. *Cambridge NA Report NA2009/06, University of Cambridge, Cambridge*, pages 26–46, 2009.

[8] Davis E. King. Dlib-ml: A machine learning toolkit. *J. Mach. Learn. Res.*, 10:1755–1758, December 2009.