A Window-based Method for Target Estimation

Paul Konstantin Krug
Institute of Acoustics and Speech Communication, Technische Universität Dresden, Germany
paul_konstantin.krug@tu-dresden.de

Abstract: The TARGET-APPROXIMATION-MODEL (TAM) describes time continuous articulatory contours through a low-pass filtered sequence of linear functions (targets). Such a representation of articulatory dynamics is well suited for computational simulations of articulatory speech production. The transfer of articulatory trajectories, e.g. measurement data, into the TAM representation is often of particular interest. Although the open source software TARGETOPTIMIZER allows to perform such a transfer successfully, its computational complexity is at least of order $\mathcal{O}(n^3)$, whereby n is the number of targets to be estimated. This work presents a sequential fit with joint local optimization based on the TARGETOPTIMIZER backend. The proposed solution reduces the computational complexity to $\mathcal{O}(n)$.

1 Introduction

During computational simulations of articulatory speech production, the dynamics of articulators may be governed by complicated (quasi-) time continuous trajectories [1]. Modelling such trajectories can be done via the TARGET-APPROXIMATION-MODEL (TAM) [2], which describes continuous articulatory contours through a low-pass filtered sequence of linear functions (targets). While TAM-contours can easily be calculated from a given set of targets, the reverse case, i.e. the determination of underlying targets from given contours is much more difficult. Nevertheless, such a fit-based process is of great interest in order to create realistic trajectories from measured articulatory data. It has been shown that the software TARGETOPTIMIZER 2.0 (TO 2.0) [3] successfully performs such fits and yields significant improvements over previous attempts [2, 4]. However, fits with TO 2.0 suffer from high computational complexity of order $\mathcal{O}(n^3)$ or $\mathcal{O}(n^4)$ in case of additional target boundary optimization, whereby n is the number of targets to be optimized. This work presents a $\mathcal{O}(n)$ solution to this problem in the form of a special sequential fit that features joint local optimization.

2 Methods

2.1 Modifications of the TargetOptimizer 2.0 Baseline

2.1.1 User Interface

The software TO 2.0 has been developed in C++ and is available as a GUI as well as a terminal application [3]. However, both interfaces are rather inconvenient when it comes to processing larger data sets, see Section 2.1.3. For this purpose, the backend was extended by a C++ API, which is bound to a PYTHON package via CYTHON. In this way, the presented algorithms for target estimation can be conveniently accessed via the popular PYTHON language. They are available as a stand-alone package[1] as well as directly integrated into the open source articulatory speech synthesis package VOCALTRACTLAB-PYTHON[2].

[1]https://github.com/paul-krug/Target-Approximation-Model
[2]https://github.com/paul-krug/VocalTractLab-Python

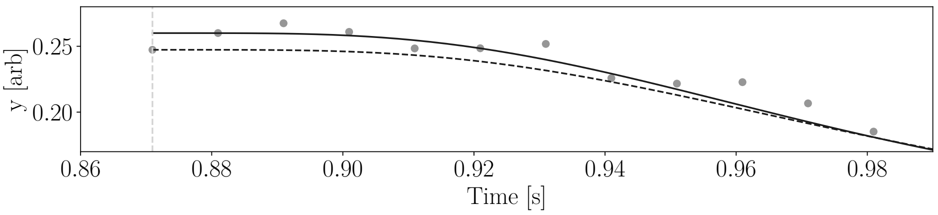

Figure 1 – Target fits with and without onset optimization are drawn as solid and dashed lines, respectively. Data points are drawn as orange dots, the first target boundary is drawn as a vertical dashed line.

2.1.2 Onset Optimization

The TO sets the onset value of the fit to be equal to the first value of the contour to be fitted. This behaviour is not optimal, as it imposes a hard constraint for the fit of the first target, as visualized in Figure 1. Therefore, fits of short contours with only single or a few targets can result in an unnecessary large error between the fit and the respective data points. This work fixed this issue by introducing the contour onset as a new parameter of the TO optimization. Since this is a single parameter, and its optimization is independent of the number of targets to be estimated, the increase in computation time due to this modification is only a small constant.

2.1.3 Data Pre and Post Processing

Although the predecessor software TO 2.0 could theoretically also process non-pitch data, in practice this required (i) a source code modification to remove a Hertz-to-semitone scale conversion that was applied to input data and (ii) non-pitch data would have to be loaded through PitchTier files which is inconvenient. Further, general non-pitch input data may also come with widely varying scales of the data. This leads to problems with the search space parameters, which are defined to operate at a certain scale only.
This work presents solutions to all three issues: First, the log-scale conversion was removed. However, fitted contours can be exported at the semitone-scale if the user wishes to do so. Second, due to the convenient PYTHON interface users can load input data in various ways, either from text files (csv, PitchTier, etc.) or directly from PYTHON-related data structures such as PANDAS data frames or NUMPY arrays. Last but not least, the scaling issue was solved by introducing an automatic normalization of the input contours to the range between $[0,1]$. After performing the fit, the resulting targets are re-scaled to the original scale.

2.2 Window-based Sequential Fits

In contrast to earlier works [2], targets are not fitted one after another. Instead, a time window with a variable length of w adjacent targets is defined, which is shifted gradually over the entire sequence. In each step of this iterative process, w connected targets are jointly optimized. This way, one can benefit from both the better performance of global optimization and the lower computational complexity of sequential optimization. This procedure has the following consequences: (i) The number of joint optimizations is $n_{\mathrm{O}} = \lceil (n-w)h^{-1} \rceil + 1$, whereby h denotes the *hop length*, i.e. the size of the window shift in each step. It applies $1 \leq h \leq w-1$. (ii) Multiple results may be obtained for the same targets and these must be somehow processed in order to obtain a final result for the whole sequence. This process is visualized in Figure 2. The computation time Γ_{s} of the window-based sequential fit is a linear function of the number

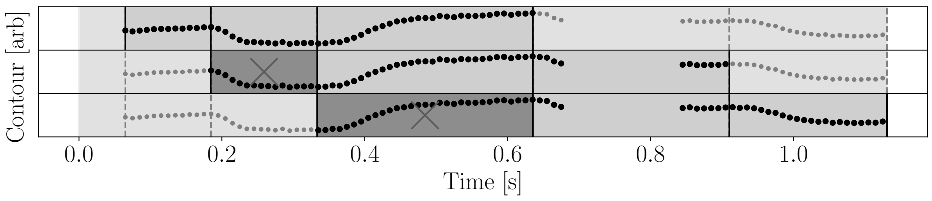

Figure 2 – A contour with five targets is optimized using a window-based sequential fit with $w = 3$ and $h = 1$. The process takes three iterations, visualized from top (first) to bottom (last). The first targets of the second and third iterations (indicated by red crosses) can not be used for the final fit contour due to the TAM onset momentum. Targets that are fitted in more than one iteration can be averaged, or as done throughout this work, the target from the last respective iteration is used. In this case that means: target 1 and 2 from iteration 1, target 3 from iteration 2 and target 4 and 5 from iteration 3.

of targets to be fitted and it is calculated via:

$$\Gamma_\text{s} = \sum_{i=1}^{n_\text{O}} \Gamma_\text{g}(\omega_i), \tag{1}$$

whereby, $\Gamma_\text{g}(\omega_i)$ denotes the computation time of the global optimization over a contour interval ω_i.

2.3 Window-based Sequential Fits with Boundary Optimization

The sequential fitting procedure becomes more complicated in case of the additional target boundary optimization. This is due to the fact that the optimized boundaries depend on the current position of the window. Optimized bounds of the same target will therefore differ for varying window positions. One solution is to use the average position of the respective target boundary. However, the values found for slope, offset and time constant will then no longer correspond to the respective target boundaries, which can lead to large errors in the fit contour. Also averaged values for slope, etc. are of little help in this case.

This work presents a solution using a multi-stage fit: First, a window-based sequential fit with boundary optimization is performed. For overlapping targets, the average optimized boundary positions are calculated. This procedure may be repeated $n_\text{P} - 1$ times (whereby n_P denotes the *number of passes*) by using the optimized boundaries from an iteration i as initial boundaries for the next iteration $i+1$. Subsequently, a normal (fixed boundary) window-based sequential fit is performed, which estimates the optimal target parameters within the determined (shifted) boundaries. This way, the computation time $\Gamma_\text{s,B}$ of the sequential fit with boundary optimization still remains a linear function of the number of fitted targets:

$$\Gamma_\text{s,B} = \left(\sum_{i=1}^{n_\text{P}-1} \sum_{j=1}^{n_{\text{O},i}} \Gamma_\text{g,B}(\omega_{ij}) \right) + \sum_{k=1}^{n_{\text{O},i}} \Gamma_\text{g}(\omega_{ik}) \Bigg|_{i=n_\text{P}} . \tag{2}$$

Thereby, $\Gamma_\text{g,B}(\omega_{ij})$ denotes the computation time of the global fit with boundary optimization over a contour interval ω_{ij}.

2.4 A Monte Carlo Generator for TAM Contours

For the evaluation of fit performances, a Monte Carlo method for generating TAM contours was developed. Thereby, TAM contours are derived from sequences of targets with random

parameter configurations. Compared to evaluations with measured data (e.g. pitch data), this method has the crucial advantage that the obtained contours are actually based on targets and are not only described by targets. Consequently, the real underlying targets are available for the evaluation of the fits. Several different data sets were created. In the case of the first set (D1), 1000 individual targets ($n = 1$) were generated. For the second data set (D2), fifteen sets ($n = 1, \ldots, 15$) with a size of 10 target sequences each were generated using the Monte Carlo method. Thereby, the target parameters were drawn from uniform distributions within the value ranges that are given in Table 1. The slope parameters were *balanced*, which means that they were either set to zero or drawn from the respective uniform distribution with a 50 % chance.
For both data sets, the contours were calculated on the basis of the individual targets or target sequences. TAM contours were derived at a sample rate of $f_{\mathrm{TAM}} = 100\,\mathrm{Hz}$. With a probability of 50 %, the contour of a target was masked on a contiguous interval with a length between 40 % and 60 % of the target. Additionally, function values were individually shifted by a random value drawn from a normal distribution ($\mu = 0$, $\sigma = 0.01$). These two operations provide contours that represent a more realistic scenario due to the similarities to pitch data. For a third data set TAM contours were derived at different sample rates of (25, 50, 100, 200, 400) Hz. In this case no masking was applied and no noise was added. The three data sets were used for different experiments, as described in the following section.

	$\mathrm{t_O}$ [s]	$\mathrm{v_O}$	d [s]	m [$\mathrm{s^{-1}}$]	b	τ [$10^{-3} \cdot \mathrm{s}$]
Range	[0.0, 1.0]	[0.0, 1.0]	[0.1, 0.4]	[-0.5, 0.5]	[0.0, 1.0]	[5, 25]

Table 1 – Value ranges of the parameters sampled by the Monte Carlo generator. The parameters $\mathrm{t_O}$ and $\mathrm{v_O}$ refer to the onset time and onset value of a target sequence, respectively. The parameter d describes the target duration. The parameters m, b and τ denote the slope, offset and time constant of a target [3].

2.5 Experiments

Different experiments were performed, which are described in more detail below. For the evaluation of the fits, the following observables were defined: $\mathrm{R_D}$ and ρ_{D} (which denote the root-mean-square error (RMSE) and the Pearson correlation coefficient between the fit and the contour data points, respectively), as well as $\mathrm{R_m}$, $\mathrm{R_b}$ and $\mathrm{R_\tau}$ (which denote the RMSE between the estimated slope, offset and time constant parameters and the corresponding true parameter values, respectively). For fits with boundary optimization, the RMSE $\mathrm{R_B}$ between the true boundaries an the optimized boundaries was calculated instead of $\mathrm{R_m}$, $\mathrm{R_b}$ and $\mathrm{R_\tau}$, since a direct comparison of true and estimated target parameters is not possible when the boundaries are different.
For all optimizations, the search space limits were set equal to the sampled target parameter ranges shown in Table 1. The other optimization parameters were set to the TO 2.0 default values, see [3]. All calculations were performed on an Intel® Core$^{\mathrm{TM}}$ i7-1065G7 CPU.

2.5.1 Experiment 1

The first experiment was conducted in order to evaluate the effect of onset optimization. For this purpose, the data set D1 was used. All targets were fitted both without and with onset optimization. Since the dataset contains only single targets, the distinction between sequential and global fit is irrelevant. However, a fit with and without boundary optimization was performed in each case, since the shift of the first target boundary is possible within certain limits [3] and

provides additional freedom for the onset optimization.
In all further experiments, the onset optimization was used without being explicitly mentioned.

2.5.2 Experiment 2

For the second experiment, the data set D2 was used. Targets were estimated (without boundary optimization) once with global optimization and once with the window-based sequential optimization method with $w = 3$ and $h = 1$. Additionally, the same kind of fits were performed on shifted boundaries and uniform distributed boundaries in order to test the performance under the condition of non-optimal boundaries. In the former case, a Gaussian distributed noise ($\mu = 0.0\,\mathrm{ms}$, $\sigma = 50\,\mathrm{ms}$) was added to the true boundary positions. To prevent a possible time overlap, the boundaries were then sorted in time ascending order. Subsequently, if the first boundary was located after the first data point, the boundary was shifted accordingly. Similarly, if the last boundary was located before the last data point, the boundary was shifted accordingly. The evenly distributed boundaries were created as in the previous work [3].

2.5.3 Experiment 3

The third experiment was designed to evaluate the boundary optimization performances by the global and window-based fits. For this experiment, the data set D2 was used with the same shifted boundaries as in the second experiment. The same fits were carried out but with activated boundary optimization. In the case of the sequential method, the parameter n_P was set to 2. In contrast to the previous experiment, only sequences with $n \leq 10$ were considered in order to reduce the high computational costs.

2.5.4 Experiment 4

The last experiment was conducted in order to investigate the target reconstruction performance for different sample rates of the input contour. Information about this performance might be helpful in situations where a specific data sample rate is required, e.g. for time series input or output vectors of neural networks or if a certain kind of data reduction is desired.
For this purpose, the targets of data set D3 were estimated (without boundary optimization) with the window-based sequential optimization method with $w = 3$ and $h = 1$.

3 Results

3.1 Experiment 1

The effect of the onset optimization is shown in Table 2. It can be seen that the onset optimization significantly reduces the median value of $\mathrm{R_D}$ by 5.2 % (without boundary optimization) and 5.6 % (with boundary optimization) compared to the respective case without onset optimization. At the same time the median computing time is significantly increased by 37.5 % and 93.3 % percent for fits without and with boundary optimization, respectively. Although this may sound like a drawback, the increase in computing time is only a constant on the order of a few milliseconds, so it does not matter much for fits of longer sequences. In case of ρ_D, $\mathrm{R_m}$, $\mathrm{R_b}$ and $\mathrm{R_\tau}$ no significant differences were found between the fits with and without onset optimization.

	Γ [$10^{-3} \cdot$ s]	ρ_D	R_D	R_m [s^{-1}]	R_b	R_τ [$10^{-3} \cdot$ s]
Baseline	**8.0***	0.995	0.01075	**0.078**	0.02	**1.13**
Onset optimization	11.0	0.995	**0.01019***	0.079	0.02	1.24
Boundary optimization	**15.0***	0.994	0.01101	**0.130**	0.286	**4.98**
Onset + boundary opt.	29.0	0.994	**0.01039***	0.139	**0.279**	5.08

Table 2 – Results of experiment 1. The table shows the median values. Better values are indicated by bold numbers. A star indicates a significant difference between the values obtained without and with onset optimization. Thereby *significant* means $p < 0.05$ based on Mood's median test.

3.2 Experiment 2

The left plot in Figure 3 shows the computation time Γ required for the fits as a function of the target sequence length n. It can be seen that the computation time in the case of the global optimization (Global) is well described by a third order polynomial, while the data for the window-based sequential method (WBS) can be described by a linear function. The middle and right plots in Figure 3 show the R_D and ρ_D distributions, respectively. It can be seen that the window-based method reduces the error between fit and contour compared to the global method while increasing the corresponding correlation coefficient. The distributions are significantly different ($p < 0.01$), based on two sided Kolmogorov–Smirnov (KS) tests.

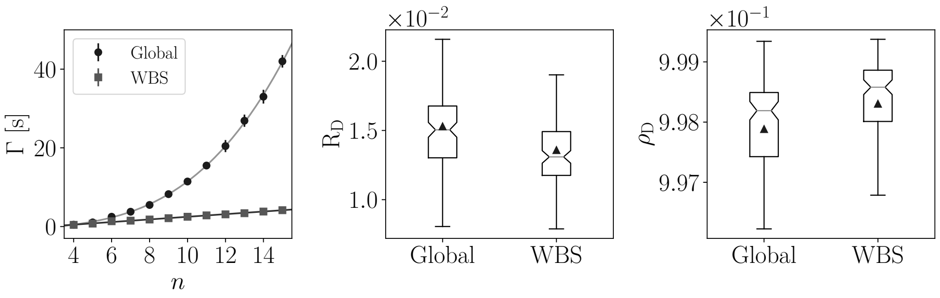

Figure 3 – **Left:** Computation time needed for fits with joint optimization (Global) and the proposed method (WBS) as a function of the number of targets within a given contour. **Middle:** RMSE between the fit and respective data points. **Right:** The corresponding distributions of the correlation coefficient.

Figure 4 shows the corresponding results for R_m, R_b and R_τ. The distributions of fits using the window-based method are consistently shifted to lower values compared to the data from the joint optimizations. The respective distributions are significantly different ($p < 0.01$), based on two sided KS tests.

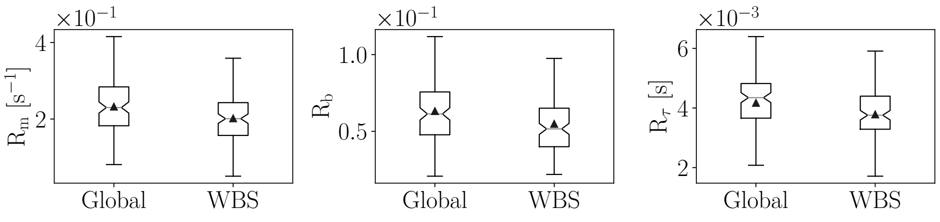

Figure 4 – Distributions related to the target parameter estimation accuracy.

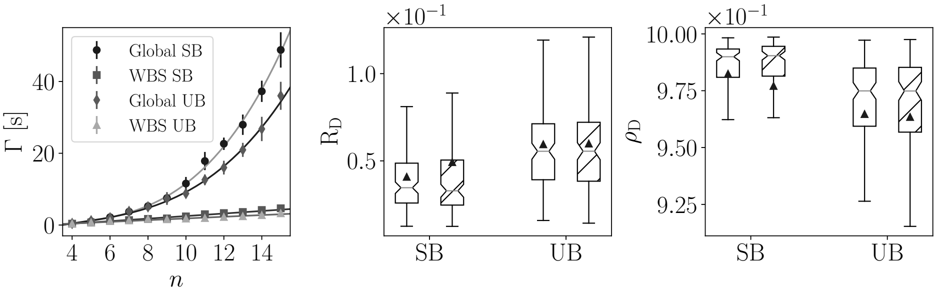

Figure 5 – Results of fits without boundary optimization on shifted (SB) and uniform boundaries (UB). In the right plot polynomials of order 3 and 1 were used to fit the Global and WBS results, respectively. Results of target estimation fits using WBS are indicated by hatched boxes in the middle and left plots.

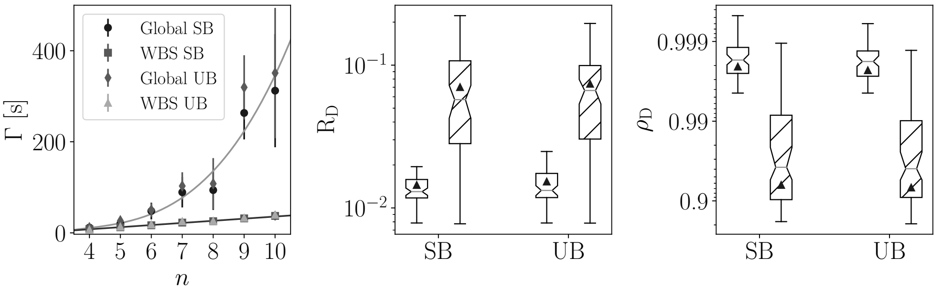

Figure 6 – Results of fits with boundary optimization. In the right plot polynomials of order 4 and 1 were used to fit the Global (SB) and WBS (SB) results, respectively. Results of target estimation fits using WBS are indicated by hatched boxes in the middle and left plots.

The results of the global and window-based sequential target estimation for data set D2 with shifted boundaries (SB) and uniformly distributed boundaries (UB) are shown in Figure 5. One can see that the optimizations with uniform boundaries require less computation time than with the shifted boundaries (left plot). However, the RMSE tends to be larger (middle plot) and the correlation coefficient values tend to be smaller (right plot). No significant differences were found between the respective distributions of the global and the window-based method.

3.3 Experiment 3

Figure 6 shows the results of the third experiment. The left, middle and right plots show the computation time, the distributions of R_D and the distributions of ρ_D, respectively. Results are shown for the fits with boundary optimization, where the boundaries were initialized with the shifted real boundaries (SB). In addition, the results for fits with boundary optimization and uniform boundary initialization are shown (UB). It can be seen that the WBS method significantly reduces the computation time, but the error R_D is significantly larger and the correlation coefficient is significantly smaller compared to the global optimization method.
The median values for R_B are 31.9 ms (SB) and 56.8 ms (UB) in case of the global optimization and 212.7 ms (SB) and 198.2 ms (UB) in case of the window-based sequential optimization, which performs significantly worse in this regard ($p < 0.01$, based on a two-sided KS test). In both cases no significant difference was observed between the R_B distributions of the SB and UB optimizations. The median RMSE between the true and shifted boundaries is 43.6 ms. This means that in the case of global boundary optimization on the shifted boundaries, the median

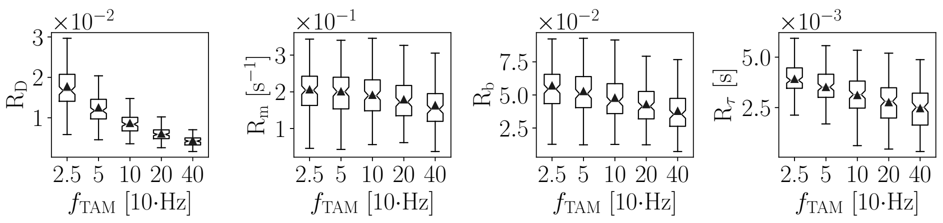

Figure 7 – Distributions related to target reconstruction accuracy, visualized as functions of the TAM sample rate f_{TAM}.

value was reduced. However, this result is not significant. In fact, the $\mathrm{R_B}$ distribution calculated from the true and shifted boundaries is significantly more concentrated around lower values compared to those of the optimized boundaries ($p < 0.01$, based on two-sided KS tests).

3.4 Experiment 4

Figure 7 shows the effect of the TAM sample rate on the reconstruction accuracy of the underlying target parameters. In a direct comparison of the sample rates 25 Hz and 400 Hz, $\mathrm{R_D}$, $\mathrm{R_m}$, $\mathrm{R_b}$ and $\mathrm{R_\tau}$ are reduced by 74.7 %, 25.6 %, 35 %, and 34.7 %, respectively.

4 Discussion

It was shown that window-based sequential target fits can achieve and even surpass the performance of global fits while reducing the computational complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(n)$. The computational complexity could be similarly reduced in the case of boundary optimization. However, it was observed that the error between the optimized boundaries and the real boundaries is greatly increased this way. At the same time the error between the fit and the respective contour is large and on average worse than in the case of the same optimization without the multi-stage method. A larger window length might help but this is again at the expense of higher computation time. In general, the question arises as to the usefulness of boundary optimization if, even in the case of global optimization, the error between the optimized and real boundaries is not significantly minimized. In fact, the additional degrees of freedom may rather allow overfitting of a given contour. In future work it should be investigated whether the true boundary positions and the other target parameters can be estimated with high precision using deep neural networks. The Monte Carlo generator for articulatory trajectories presented in this work is well suited to generate arbitrarily large data sets for the training of such networks.

References

[1] BIRKHOLZ, P.: *Modeling consonant-vowel coarticulation for articulatory speech synthesis. PloS ONE*, 8(4), p. e60603, 2013.

[2] PROM-ON, S. ET AL.: *Modeling tone and intonation in Mandarin and English as a process of target approximation. J. Acoust. Soc. Am.*, 125(1), pp. 405–424, 2009.

[3] KRUG, P. K. ET AL.: *TargetOptimizer 2.0: Enhanced estimation of articulatory targets. Studientexte zur Sprachkommunikation: Elektronische Sprachsignalverarbeitung*, pp. 145–152, 2021.

[4] BIRKHOLZ, P. ET AL.: *Estimation of pitch targets from speech signals by joint regularized optimization.* In *Proc. EUSIPCO*, pp. 2075–2079. 2018.