**Template optimization and evaluation procedure**

This document contains description of the procedure used to optimize and evaluate spectral templates. It is part of the supplementary material for the paper ‘Optimal spectral templates for triggered feedback experiments.’

**About Sample Data**

The description of sample data below aims to show how the input data to the procedure should be formatted. The sample data (…\analysis\_data\340) contains 50 songs from one of the birds (bird 340) used in the paper. It has been divided into two directories: …\analysis\_data\340\train\_dir and …\analysis\_data\340\test\_dir. The ‘train\_dir’ contains training data and the ‘test\_dir’ contains the test data set. Both directories contain 25 audio files (.cbin), their corresponding metadata files (.rec and .cbin.not.mat), and a batch file (txt file with the list of audio files). The two batch files are called batch\_train and batch\_test. The .rec files contain information about the sampling frequency and other metadata. The .cbin.not.mat files contain syllable-gap segmentation and label information. The only variables from cbin.not.mat that are critical to the scripts below are onsets, offsets, and labels: ‘onsets’ and ‘offsets’ contain the syllable onsets and offsets in msec. The variable ‘labels’ contains a string of characters that denote syllable labels. There is also a txt file (…\analysis\_data\340\340\_syll\_list.txt) that lists the names of all the syllables for this bird along with the identity of the target syllables among those. Syllables which are not target syllables denote sounds or noises in the audio file that occur outside the context of the song.

This procedure works with wav files just as well. In case of wav files, .rec files are not needed as wav files contain the relevant metadata about sampling frequency. You do need to create files ending in .wav.not.mat for each wav file and containing variables labels, onsets, and offsets.

**Procedure**

For the procedure outlined below:

* Ensure that the directories …\analysis\_code and …\analysis\_code\_util are at the top of your Matlab path.
* Change your Matlab working directory to …\analysis\_data\340.
* While reading this procedure, it is best to have the relevant script open in the editor.
* It is advisable to read the methods section of the paper to better understand the procedure laid out here.

The script ‘main.m’ can run the entire procedure for all syllables of a single bird. It runs several functions sequentially to achieve this. These actions/functions are described below in detail:

* **Create simdata files** (create\_simdata\_file.m, both training and test): This function segments the audio files into 256-data-point-long slices and assigns the slices to syllables and gaps. It stores that information in .simdata files. This function should be run for both training and test data.
* **Calculate amplitude threshold** (calculate\_amplitude\_threshold.m, training only): This function uses spectral amplitudes of all slices in the training data to calculate an optimal amplitude threshold. The optimization tries to minimize the misclassification of syllable-assigned and gap-assigned slices in reference to a given amplitude threshold.

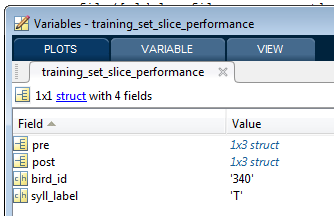
All gap slices with amplitude higher than the threshold are admitted as distractors in the optimization process. This amplitude threshold is also applied as a criterion for slice matching, in addition to spectral distance, during the evaluation process. It stores the amplitude threshold value in a file called ‘amp\_thr.mat’ in the training directory. This function should be run for training data only.

* **Create files that store syllable associated chunks** (make\_syll\_assoc\_chunks.m, both training and test): This function collates all syllable instances and stores their assigned slices in a syll\_assoc\_chunks\_syll\_SYLL\_seq\_SYLL.mat file. For example: syll\_assoc\_chunks\_syll\_A\_seq\_A.mat.
* **Create a file that stores gap associated chunks** (make\_gap\_assoc\_high\_amp\_chunks.m, training only and make\_gap\_assoc\_chunks\_keep\_together.m, test only): The first function collates all gap slices with amplitude higher than the threshold (from training data) and stores them in a single ‘gap\_assoc\_high\_amp\_chunks.mat’ file. The second function stores the gap slices from test data along with their amplitude information without collating them into a single cell array. Slices belonging to each instance of a gap are used as a unit during the evaluation of optimized templates.

* **Align syllable associated chunks for each target syllable** (align\_syll\_assoc\_chunks.m, training only): This function aligns syllable instances to their category specific modal slice length. This step will exclude instances of the syllable whose durations are more than 2 (tunable feature/parameter) standard deviations away from the mean duration for that category. This script simply replaces the previously generated syll\_assoc\_chunks file.

This alignment allows us to take an average across all instances of a syllable, given that different instances have different number of slices.

* **Write template files from aligned chunks** (make\_templates.m, training only): This function makes and writes templates (and metadata) out of the aligned syllable associated chunks. For example: template\_syll\_A\_seq\_A\_chunks\_1\_outof\_9.dat and template\_syll\_A\_seq\_A\_chunks\_1\_outof\_9\_metadata.mat.
* **Optimize templates** (optimize\_template.m, training only): This function will optimize the templates and write a file containing the optimization results. For example: optimization\_grand\_results\_syll\_A.mat. This step takes the longest as the gradient descent optimization is carried out here.
* **Write optimized templates from optimization results** (write\_optimized\_templates\_from\_optimization\_results.m, training only): This function will write the optimized template files and their metadata from the output/results files obtained during optimization. For example: template\_syll\_A\_seq\_A\_chunks\_1\_outof\_9\_optimized.dat and template\_syll\_A\_seq\_A\_chunks\_1\_outof\_9\_optimized\_metadata.mat.
* **Calculate slice-level improvement resulting from optimization** (calculate\_optimization\_performance\_training\_set.m, training only): This function calculates the slice level performance. It will write the training\_set\_slice\_performance\_\*.mat files for each target syllable. For example: training\_set\_slice\_performance\_A.mat. Since these files are the end point of slice level analysis, we describe them briefly below. The file has a variable called training\_set\_slice\_performance. It has the following fields:



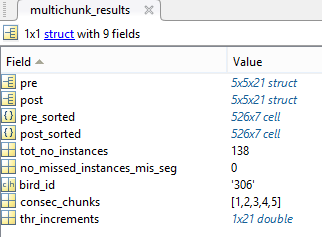
Variables ‘pre’ and ‘post’ have the same number of elements as the number of templates. They represent the performance of averaged and optimized templates respectively. Their fields are:



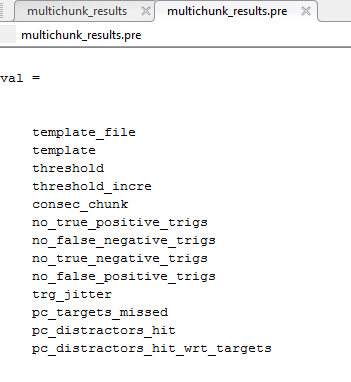
Field descriptions are given below:

* pc\_targ\_slices\_missed and pc\_ditractor\_slices\_hit – These are fractions representing false negative and false positive errors respectively.
* threshold – This value is optimal threshold for the given template.
* template – This field can be safely ignored.
* sigma – This field represents the standard deviation of the Gaussian used for smoothing the distance distributions.
* **Calculate syllable level improvement in targeting using the optimized templates against the test data** (calculate\_optimization\_performance\_train\_thrs.m, test only): This function evaluates the performance of averaged and optimized templates in detecting syllables in the test data. It writes a file for each target syllable. For example: multichunk\_results\_A.mat. Since these files are the end point of syllable level analysis, we describe them briefly below.

The above file has a variable called multichunk\_results. It has the following fields:

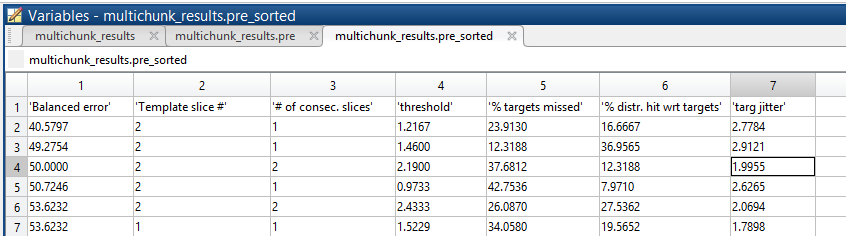


The first dimension in pre and post is the number of templates, the second one is the range of detection criteria (# of consecutive slices) and the third one is the range of threshold levels (0% to 200% in 10% steps). The fields in pre and post are:



‘consec\_chunk’ refers to the detection criterion value and ‘threshold\_incre’ refers to the threshold increment. ‘pc\_distractors\_hit\_wrt\_targets’ is the number of distractors hit as a percentage of the total number of targets. Threshold increments here are stated as the fractional added value. For example, 0% to 200% in 10% steps is stated as -1 to 1 in steps of 0.1. trg\_jitter refers to the standard deviation of latency of detection relative to syllable onset.

The fields pre\_sorted and post\_sorted contain all combinations of template slices, detection criteria, and thresholds[[1]](#footnote-1) sorted in ascending order of the balanced error. These values can be directly used for finalizing the targeting parameters for an experiment.



1. The thresholds mentioned in this data structure are for squared distances since the implementation of triggered feedback software in our lab (EvTAF) uses thresholds on squared distances. [↑](#footnote-ref-1)