Mar 2024

PENTAtrainer USER MANUAL

Dr. Albert Lee

The Education University of Hong Kong

Table of Contents

1. Introduction

PENTAtrainer is a semi-automatic software package for modelling speech prosody, integrating a Praat script and Java programs to achieve computational modelling. It implements the workings of the **PENTA (Parallel Encoding and Target Approximation) Model** and quantitative Target Approximation (qTA) model, and is developed by Yi Xu (University College London) and Santitham Prom-on (King Mongkut University of Technology Thonburi).

In PENTA, communicative functions representing intended communicative meanings such as lexical, focal, sentential, phrasal, emotional are transmitted in parallel with different encoding schemes and then uniquely contribute to target specification. As a result, it is transformed into the dynamic changes in surface f_0 values and timings of various f_0 events (Prom-on & Xu, 2016).

With PENTAtrainer, users can specify the functions to be encoded by creating tiers and labels flexibly according to research topics and goals.

1.1. Core concepts: PENTA Model (Xu, 2005)

- F_o-to-meaning mapping by parallel encoding
- F_o contour generated through target approximation (Xu & Wang, 2001)

1.2. Features of PENTAtrainer (quoted from Xu & Prom-on, 2014)

- High synthetic accuracy of prosody
- Many-to-one mapping from variable surface F_o to invariant underlying targets
- Effectively handles both contextual and non-contextual variability
- Combines deterministic synthesis and data-driven parameter learning
- Large-scale and full-detailed prosody synthesis as tool for theory testing

2. Data Preparation and Annotation

2.1. System requirements

- Operating Systems: Mac OS X, Windows, or Linux
- Praat version: 5.3 or newer
- Java Runtime Environment: Version 1.6 or newer
- *(Optional)* Audio Editing Program (e.g., Audacity, Adobe Audition etc.)

2.2. Required materials

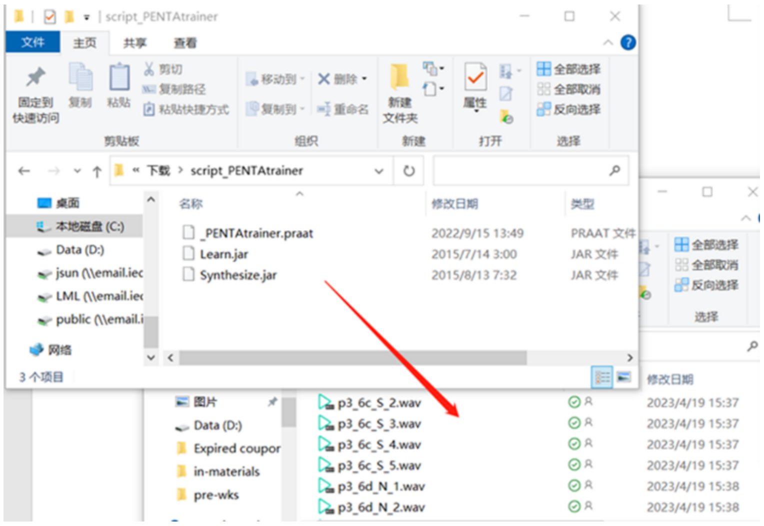
All materials stated below should be placed into the same working folder.

1) Speech data

   Split long audio files into individual utterances and save them as .wav files, with Praat or other audio editing software. It is recommended because this simplifies annotation and reduces memory usage of the learning program.

2) Praat script & Java programmes

   Copy the Praat script **(PENTAtrainer.praat)** and the Java programs **(Learn.jar & Synthesize.jar)** into the same working folder.



名称	修改日期	类型
_PENTAtrainer.praat	2022/9/15 13:49	PRAAT 文件
Learn.jar	2015/7/14 3:00	JAR 文件
Synthesize.jar	2015/8/13 7:32	JAR 文件

	修改日期
p3_6c_S_2.wav	2023/4/19 15:37
p3_6c_S_3.wav	2023/4/19 15:37
p3_6c_S_4.wav	2023/4/19 15:37
p3_6c_S_5.wav	2023/4/19 15:37
p3_6d_N_1.wav	2023/4/19 15:38
p3_6d_N_2.wav	2023/4/19 15:38

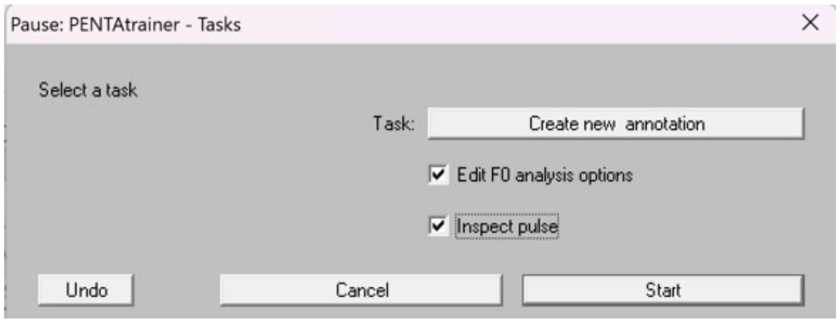
2.3. Annotation

If annotation files (*.annotation) and pulse marking files (*.pulse) are prepared with other Praat scripts (e.g. ProsodyPro), they should also be placed in the same working folder before learning parameters. Please jump to section 2.3.3.

2.3.1. Create new annotation

Apart from using PENTAtrainer built-in annotation tool, users can use other means to generate functional annotations from existing TextGrid files. **LabelEditor.exe** is one tool that automates this process. Please refer to the 'Downloads' section on the website for a tutorial of **LabelEditor.exe**.

1) Open PENTAtrainer.praat and choose the task 'Create new annotation'. Check both 'Edit F0 analysis options' and 'Inspect pulse' boxes. Click 'Start' to continue.


Pause: PENTAtrainer - Tasks

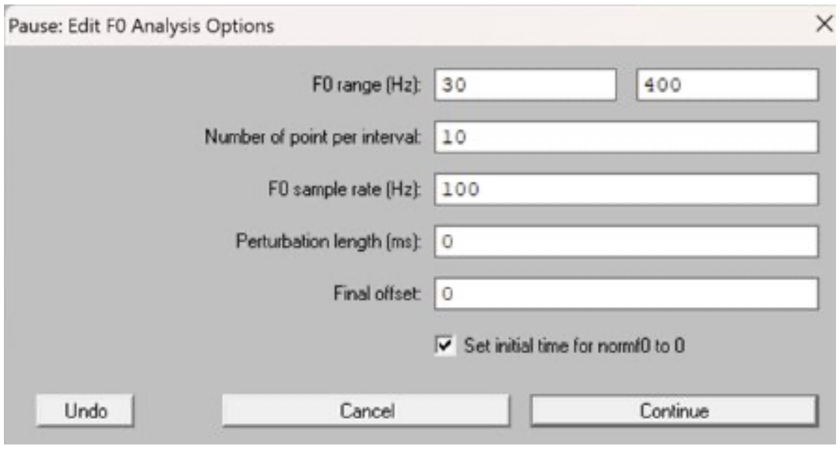
Select a task

Task: Create new annotation

[✓] Edit F0 analysis options

[✓] Inspect pulse

2) Edit the F_0 analysis parameters used in the F_0 estimation process in the 'Pause: Edit F0 Analysis Options' window if necessary.


Pause: Edit F0 Analysis Options

F0 range (Hz): 30 400

Number of point per interval: 10

F0 sample rate (Hz): 100

Perturbation length (ms): 0

Final offset: 0

[✓] Set initial time for normf0 to 0

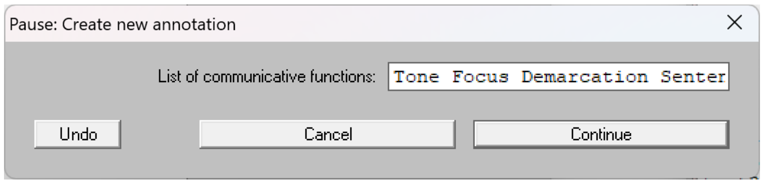
a) **F_0 range (Hz):** the range of F_0 values to be searched.

b) **Number of points per interval:** the number of F_0 points in each annotated interval in .timenormf0.

c) **F_0 sample rate:** the sampling rate used for F_0 estimation.

d) **Perturbation length, Final offset, Set initial time for normf0 to 0:** no changes are needed.

3) Specify the functional tiers in ‘List of communicative functions’. Each function should be separated by a space. The choice of communicative functions should reflect the design of the production experiment (e.g. Focus for a data set of focus prosody). Users can consider this a hypothesis-testing process – compare the synthesis accuracy of a model with a given communicative function and one without.

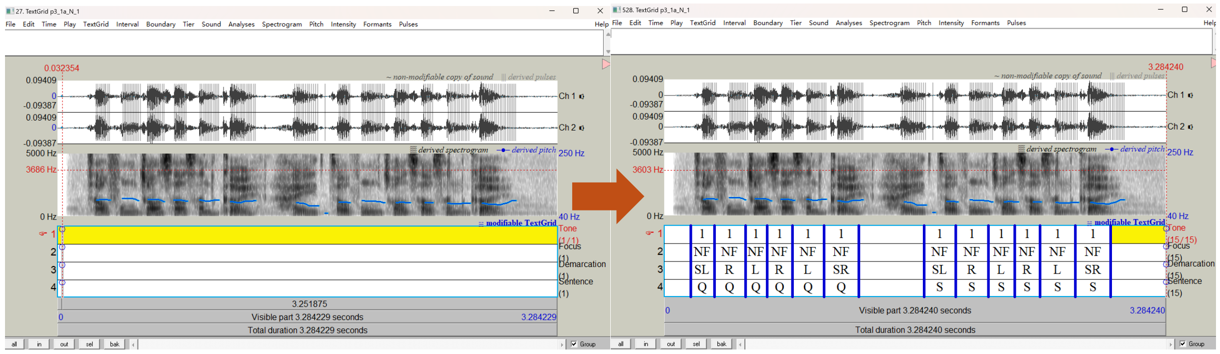


List of communicative functions: Tone Focus Demarcation Senter

4) Two windows will prompt for each .wav file. Users can annotate the files and rectify vocal pulses in respective window.

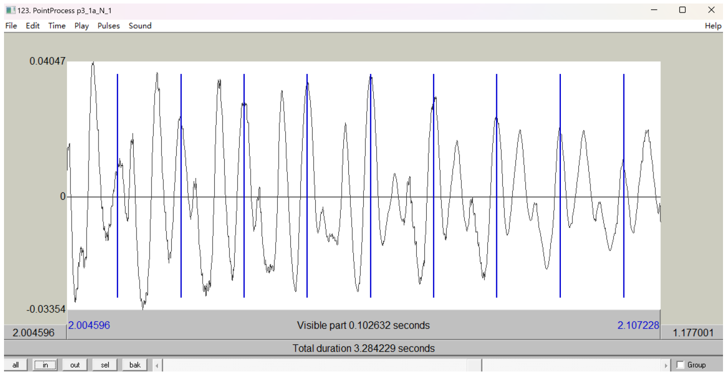
a) Annotation window

Users need to insert the boundaries for each prosodic event in the annotation window below. However, there is a limitation. There must be boundaries in each layer, and the leftmost and rightmost boundaries in all other layers should align with those in the layer with the most intervals. Note, again, that this process can be automated – users can generate functional annotations from pre-existing TextGrid files.


Tone
Focus
Demarcation
Sentence
Visible part 3.284229 seconds
Total duration 3.284229 seconds
Visible part 3.284240 seconds
Total duration 3.284240 seconds

b) Pulse marking window

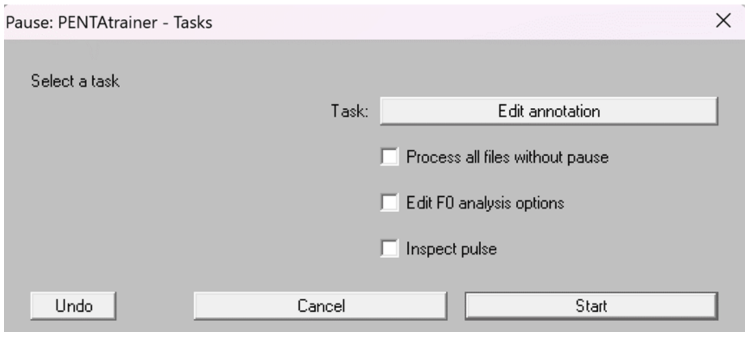
Initial markings could be incorrect. To guarantee accurate F_0 calculation, users should manually examine and modify the locations of pulse marks in the window below. Note that the pulse marking window will only show when the ‘**Inspect pulse**’ box is checked.


123. PointProcess p3_1a_N_1
Visible part 0.102632 seconds
Total duration 3.284229 seconds

5) Repeat step 4) until annotations of all sound files are done.

2.3.2. Edit existing annotation

1) To view or edit existing annotation, select ‘Edit annotation’ and click ‘Start’.



Pause: PENTAtrainer - Tasks

Select a task

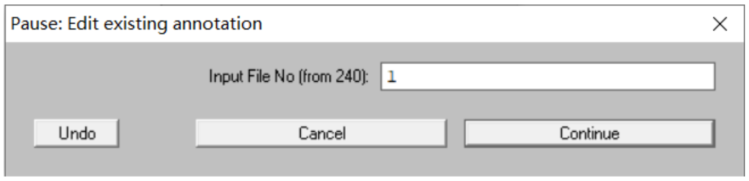
Task: Edit annotation

☐ Process all files without pause

☐ Edit F0 analysis options

☐ Inspect pulse

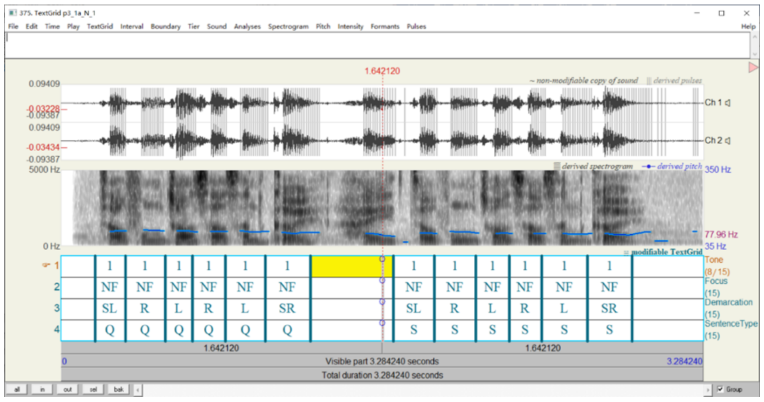
2) Users can enter the starting file number in the ‘Pause: Edit existing annotation’ window. File number can be identified from the row number of the ‘FileList.txt’ by opening the text file with a spreadsheet program. If users enter ‘1’, the program will go through all annotation files. Click ‘Continue’ to proceed.



Pause: Edit existing annotation

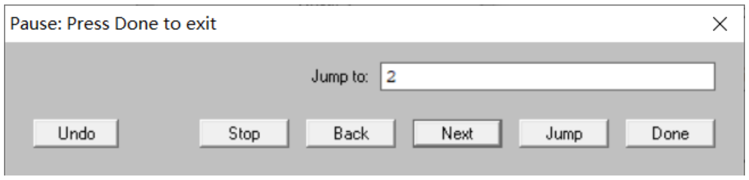
Input File No (from 240): 1

3) The program will then show the annotation window with data from the existing annotation files.



1		1	1	1	1	1	1		1	1	1	1	1	1
2		NF	NF	NF	NF	NF	NF		NF	NF	NF	NF	NF	NF
3		SL	R	L	R	L	SR		SL	R	L	R	L	SR
4		Q	Q	Q	Q	Q	Q		S	S	S	S	S	S

4) Once completing editing or reviewing, click ‘Next’ to proceed to the next file.



Pause: Press Done to exit

Jump to: 2

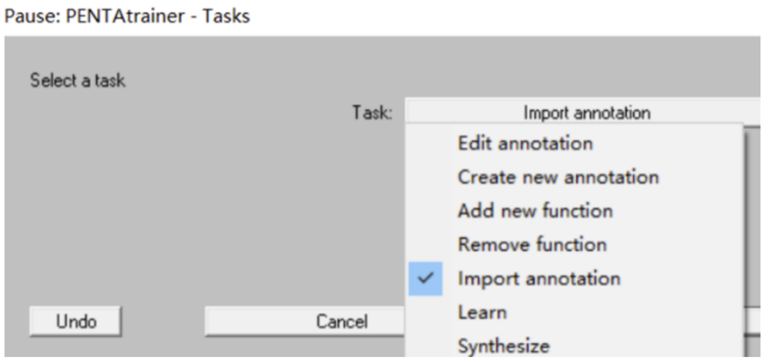
5) Repeat steps 3) and 4) until all annotation files are reviewed or edited. If users wish to stop in the middle of reviewing or editing, users can close all PENTAtrainer windows.

✧ **Tip**

If the annotation process is terminated before all annotation files are processed, some Praat objects may remain in the 'Praat Objects' window. Users should remove these objects before running other scripts to ensure correct object referencing.

2.3.3. Import existing annotation

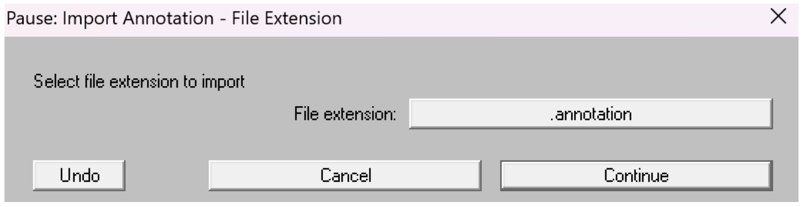
1) If users have existing '.annotation' and '.pulse' files before running PENTAtrainer, they should be placed in the same working folder together with the .wav files. Users can then open the PENTAtrainer script, and select the 'Import annotation' task.


Pause: PENTAtrainer - Tasks

Select a task

Task: Import annotation

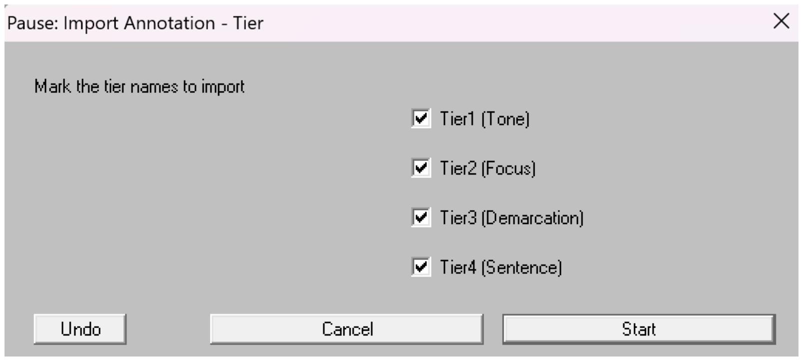
2) Choose the file extension of the existing annotation files. Then, click 'Continue'.


Pause: Import Annotation - File Extension

Select file extension to import

File extension: .annotation

3) Select the tiers to be imported. Click 'Start' and all the annotation files will be converted.


Pause: Import Annotation - Tier

Mark the tier names to import

☑ Tier1 (Tone)

☑ Tier2 (Focus)

☑ Tier3 (Demarcation)

☑ Tier4 (Sentence)

2.4. Introduction on Output Auxiliary Files

1. **FileList.txt:** A headerless, single-column table generated at the beginning of the annotation process. Each row represents a sound filename used as input in the process.
2. **config.txt** (required for parameter learning): A list of parameters to be passed along the the annotation process, formatted as a two-column table with the headers of 'parameter' and 'value'. It Is generated upon completing the annotation process.
3. **function.txt** (required for parameter learning)**:** A list of communicative functions presented as a single-column table. Each row represents a function. This file is generated at the end of the annotation process.
4. ***.annotation** (required for parameter learning): Annotation data in the Praat TextGrid format.
5. ***.annotation_short:** More user-friendly annotation file in Praat TextGrid format.
6. ***.pulse:** A Praat PointProcess object containing pulse markings for F_0 estimation.
7. ***.rawf0:** F_0 data converted from corresponding '.pulse' files.
8. ***.PitchTier:** Trimmed F_0 data in Praat PitchTier format.
9. ***.f0:** Trimmed F_0 data, directly converted from the corresponding '.PitchTier' files.
10. ***.samplef0:** F_0 data, sampled only for non-empty intervals.
11. ***.f0velocity:** F_0 velocity data, sampled only for non-empty intervals.
12. ***.timenormf0:** Timeless F_0 data normalized to have an equal number of samples per interval.
13. ***.actutimenormf0** (required for parameter learning): F_0 and actual time values sampled and normalized to a specified number of samples per interval. It is a three-column table, with each row corresponds to a F_0 data point. The first column corresponds to the annotation intervals of the first layer, while the second and third columns represent the time and F_0 data respectively. In the learning process, PENTATrainer uses data in this file to estimate parameters.

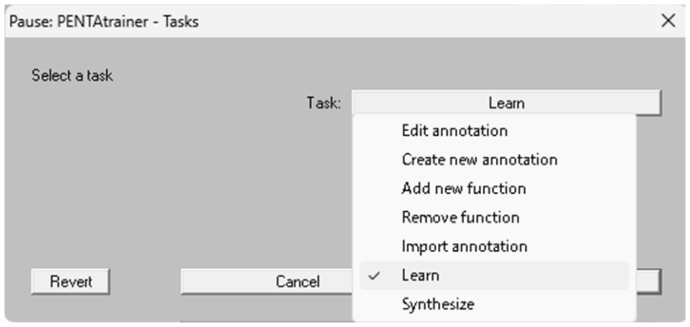
Note: File #4 to #13 is regenerated in each annotation process.

3. Learning parameters

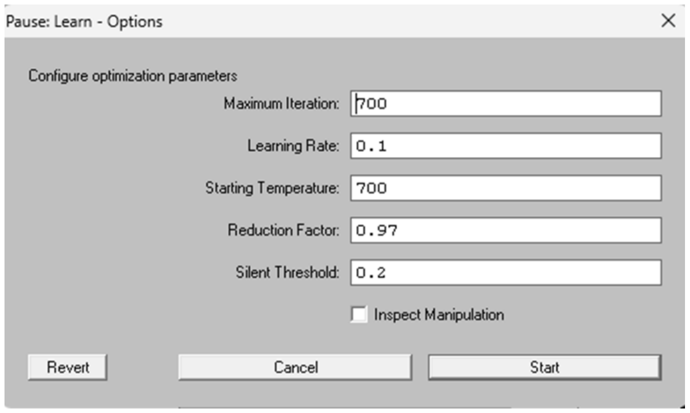
This chapter serves as a comprehensive guide on how to use the PENTATrainer learning tool for estimating model parameters of annotated communicative functions. The roles of model parameters in the target approximation process and the parameter learning algorithm used in PENTATrainer are then discussed.

3.1. Learning Parameters

1) ''Open PENTAtrainer.praat and select 'Learn' in the starting window.



2) Users can configure optimization parameters. Depending on sample size, this step can take anywhere from a few hours.



Pause: Learn - Options

Configure optimization parameters

Maximum Iteration:	700
Learning Rate:	0.1
Starting Temperature:	700
Reduction Factor:	0.97
Silent Threshold:	0.2

☐ Inspect Manipulation

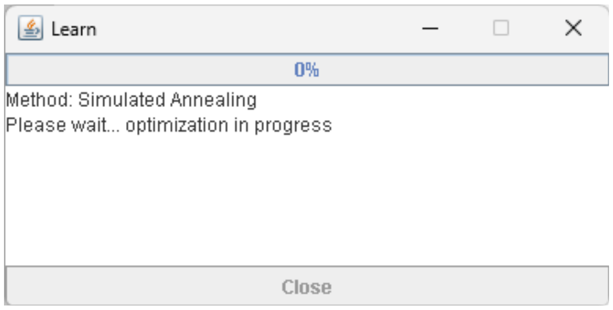
a) **Maximum Iteration:** Optimal value needs to be empirically determined. It should be large enough to allow parameter convergence.

b) **Learning Rate:** Optimal value needs to be empirically determined. It is the scaling factor for parameter adjustment and the value should be small enough not to miss the right solution.

c) **Reducing Factor:** scaling factor of the annealing temperature for each iteration

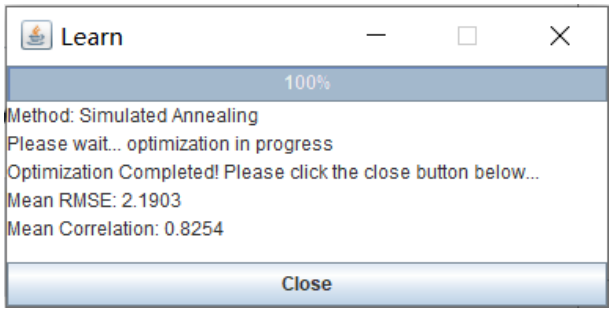
d) **Inspect Manipulation:** This allows users to visually inspect and listen to the synthesized sounds. To activate this feature, users should check the 'Inspect Manipulation' box. After the learning process is complete, Praat manipulation objects of the sound files will be generated. By clicking 'View & Edit', users are able to inspect the synthesized F_0 contours.

3) Press ‘Start’ to begin the learning process. Do not close the progress window until the progress bar reaches **100%**. If the progress window does not display properly, users should check whether Java is properly installed.



Method: Simulated Annealing
Please wait... optimization in progress

4) After the learning process is complete, the progress window will then display the resulting average per-utterance Root Mean Square Error (RMSE) and Pearson’s correlation between the original and the synthesized F_0 contours, providing users with a measure of how closely the synthesized contour matches the original.



```
Method: Simulated Annealing
Please wait... optimization in progress
Optimization Completed! Please click the close button below...
Mean RMSE: 2.1903
Mean Correlation: 0.8254
```

3.2. Output Files

1. **parameters.txt:** target approximation parameters of each functional combination. This file is produced once the optimization process concludes.

2. ***.synf0:** both original and synthesized F_0 data along with corresponding time data. It allows users to visually inspect the F_0 contours. It is regenerated during every learning process.

3. ***.intervals:** a Praat Table derived from the ‘.annotation’ files. It is utilised by the Java program during the optimization process, and it is regenerated in each learning cycle.

4. **accuracy_learning.txt:** the synthesis accuracy of each utterance. Accuracy is measured using two metrics: Root Mean Square Error (RMSE) and Pearson’s correlation coefficient.

5. **total_error.txt:** total RMSE from all sound files in each iteration. It is used for fine-tuning the optimization process.

6. **log.txt:** A file that logs the initial occurrence of each functional combination. It allows users to check for errors. If an error occurred, it will be displayed as a separate category.

4. Synthesis

This chapter explains how to use PENTATrainer to synthesize speech prosody with the learned parameters.

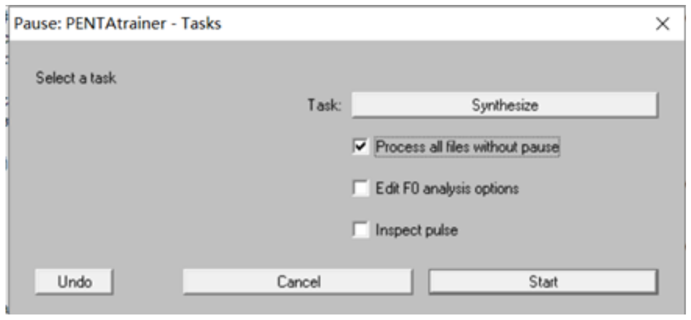
4.1. Required Input Files

1) **parameters.txt:** target approximation parameters for all function combinations that have been trained. This data, along with the annotation data, is used to generate the F0 contour.
2) ***.wav:** original speech utterances. The PENTATrainer synthesis tool utilizes the Praat manipulation object from this original sound file as a basis and incorporates the synthesized F0 contour into it.
3) ***.annotation:** annotation data in Praat TextGrid format.
4) ***.intervals:** annotation data in Praat Table format.
5) **config.txt:** the dataset configuration parameters.
6) **function.txt:** a list of communicative functions.

4.2. Speaker-dependent Synthesis with Learned Parameters

The Synthesize task employs parameters from the parameters.txt file and associated annotation files to generate the synthesized F_0 contour and incorporates it into the Praat manipulation object.

1) Open PENTAtrainer.praat. Select 'Synthesize'. Check 'Process all files without pause' and click 'Start'.


Pause: PENTAtrainer - Tasks

Select a task

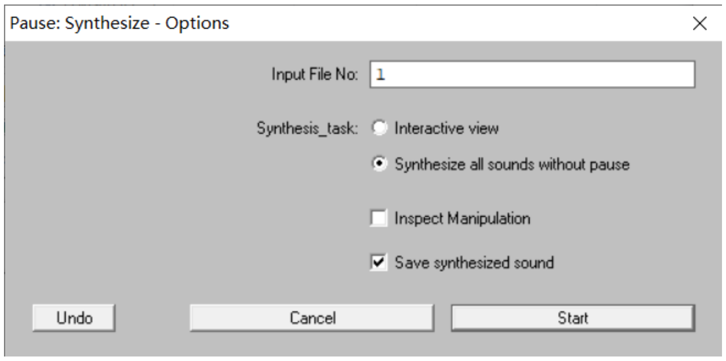
Task: Synthesize

[✓] Process all files without pause

[ ] Edit F0 analysis options

[ ] Inspect pulse

2) Specify where to save the synthesized sounds (in .wav format) by checking ‘Save synthesized sound’. If users want to inspect each synthesized F_0 contour visually and audibly while synthesizing, ‘Interactive view’ should be selected.


Pause: Synthesize - Options

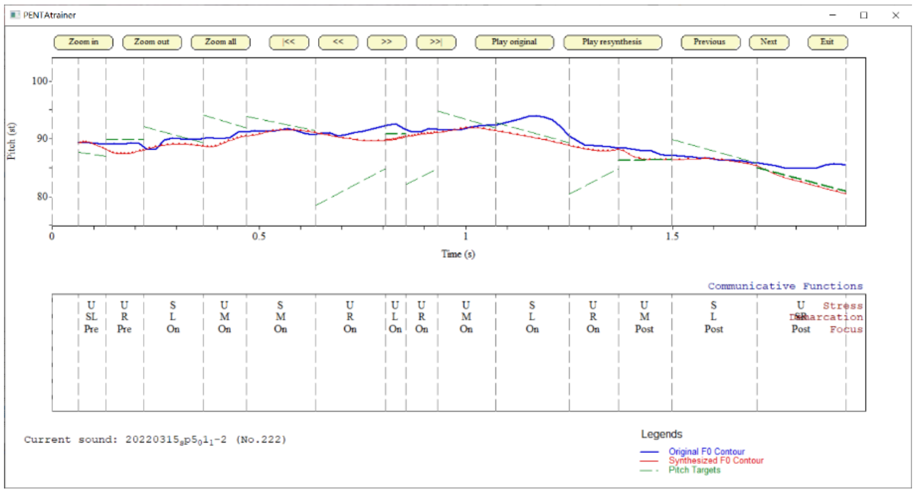
Input File No: 1

Synthesis_task: ( ) Interactive view [✓ Synthesize all sounds without pause]

☐ Inspect Manipulation

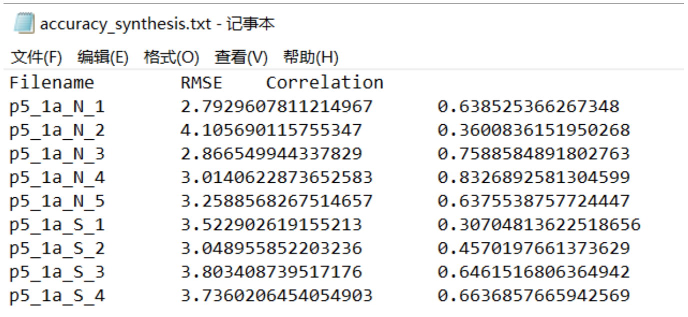
☑ Save synthesized sound

3) By checking ‘Interactive view’, an interactive Synthesis window will open. Communicative functions are listed below the F_0 contours. Users can navigate through the speech corpus with ’‘Previous’ and ‘Next' buttons in the top control panel. Using the ‘Play original’ and ‘Play synthesis’ buttons, users can play original and resynthesized sounds. Zoom and arrow buttons allow close-ups of contours.



Current sound: $20220315_8p5_01_1$-2 (No.222)

4) After synthesizing all sounds, the ‘accuracy_synthesis.txt’ file will be generated, storing the utterance-specific RMSE and correlation of synthesized contours.


accuracy_synthesis.txt - 记事本

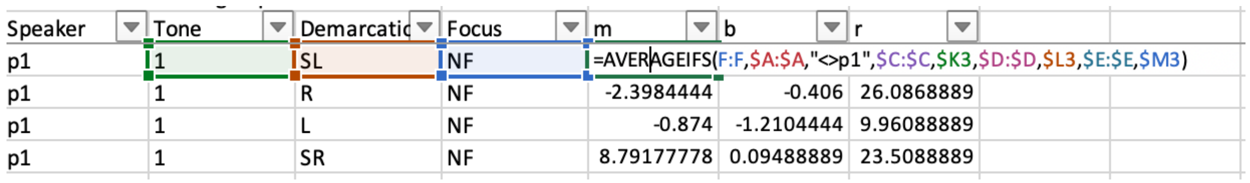
```
Filename        RMSE     Correlation
p5_1a_N_1       2.7929607811214967      0.638525366267348
p5_1a_N_2       4.105690115755347       0.3600836151950268
p5_1a_N_3       2.866549944337829       0.7588584891802763
p5_1a_N_4       3.0140622873652583      0.8326892581304599
p5_1a_N_5       3.2588568267514657      0.6375538757724447
p5_1a_S_1       3.522902619155213       0.30704813622518656
p5_1a_S_2       3.048955852203236       0.4570197661373629
p5_1a_S_3       3.803408739517176       0.6461516806364942
p5_1a_S_4       3.7360206454054903      0.6636857665942569
```

5) Synthesized F_0 data of the sound files is stored in the .synthesizedf0 file in the working folder.

4.3. Speaker-independent Synthesis with Learned Parameters

After performing speaker-dependent synthesis, it is possible to run speaker-independent synthesis (Jackknife procedure) as follows:

1) To prepare the parameters.txt file for a given speaker, parameters learned from all other speakers are needed. For example, there are 10 speakers and the 'Learn' task is performed 5 times for each of them. To generate parameter values for speaker 1, 45 sets of parameter values from the **other** 9 speakers should be averaged.



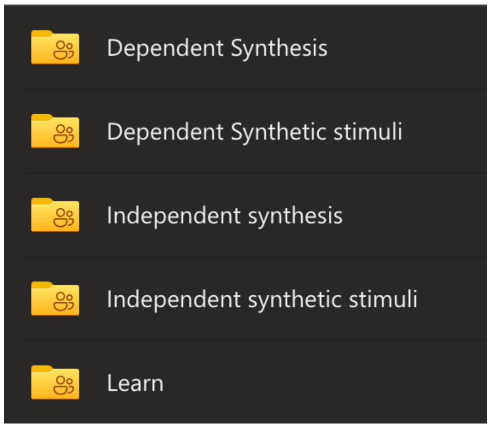
Speaker	Tone	Demarcatio	Focus	m	b	r
p1	1	SL	NF	=AVERAGEIFS(F:F,$A:$A,"<>p1",$C:$C,$K3,$D:$D,$L3,$E:$E,$M3)		
p1	1	R	NF	-2.3984444	-0.406	26.0868889
p1	1	L	NF	-0.874	-1.2104444	9.96088889
p1	1	SR	NF	8.79177778	0.09488889	23.5088889

2) Make a copy of the required input files as described in section 4.1.

3) Copy and paste the averaged parameter values to the corresponding the parameters.txt file.

4) Run PENTAtrainer.praat and repeat the steps in section 4.2.

5) By running all tasks mentioned above, there will be five types of output files.



Dependent Synthesis
Dependent Synthetic stimuli
Independent synthesis
Independent synthetic stimuli
Learn

5. References

Xu, Y. and Prom-on, S. (2014) “Toward invariant functional representations of variable surface fundamental frequency contours: Synthesizing speech melody via model-based stochastic learning”, *Speech Communication*, 57:181-208.

Prom-on, S. and Xu, Y. (2016) *PENTAtrainer User Manual* (version 1.1).

Please contact Albert Lee (albertlee@eduhk.hk) for enquiries / comments.