A Tutorial of Using the SAFE 1.0 Toolkit for F0 Estimation

Wei Chu, Abeer Alwan
weichu@ucla.edu, alwan@ee.ucla.edu

April 2011

Contents

1 Overview

SAFE is a toolkit using a Statistical Algorithm for F0 Estimation for both clean and noisy speech.

SAFE consists of two modules: model training and F0 estimation. During training, SAFE loads wave files and their F0 ground truth files to statistically estimate models for noise robust F0 estimation. During F0 estimation, SAFE evaluates the likelihoods of F0 candidates given the trained models, then generates an F0 contour that maximizes the overall likelihoods.

2 A Tutorial Example of Using SAFE

Currently, all the code and scripts have been developed, compiled, and tested under Linux system (Fedora 14, kernel 2.6, 32-bits). The toolkit should be able to run under Mac OS X or Cygwin. But we have not tested under these environments yet. We may add support for Windows later on. The following steps should be run on a Linux machine.

The code and sample scripts of SAFE can be downloaded from:
`http://www.ee.ucla.edu/~weichu/safe/download.html` . Since we may keep updating the toolkit, please make sure that the version 1.0 mentioned in this guide is downloaded.

First extract the downloaded `proj_safe_1.0.tar.gz`:

```
$ tar xzvf proj_safe_1.0.tar.gz
```

then

```
$ cd proj_safe
```

define the current directory as `$SAFEDIR`:

```
$ export SAFEDIR=‘pwd‘
```

There are following directories:

- `bin:`
  where executable files are stored.
- `code:`
  where code is stored. SAFE is a project written in C code that can be opened by KDevelop 3 (`http://www.kdevelop.org/`).
- `config:`
  where configuration files are stored.
- `doc:`
  where documentations are stored.
- `data:`
  where F0 database, F0 ground truth files, and noise wave files are stored.
- `example:`
  where example scripts are stored.
- `list:`
  where file lists are stored.

- `model:`
  where models for F0 estimation are stored. The subdirectory `estimation` stores the trained models.

- `perl:`
  where Perl scripts are stored.

2.1 Compiling and Installing SAFE

In order to compile SAFE, General Scientific Library (GSL) (`http://www.gnu.org/software/gsl/`) and FFTW (`http://www.fftw.org`) are needed to be installed. GSL 1.14 or higher, and FFTW 3.2.2 or higher are required. They can be automatically installed as follows:

```
$ sudo yum install gsl gsl-devel fftw
```

It is possible that by the time this guide is read, the GSL or FFTW in the yum repository has been upgraded to a much higher version such that the SAFE can not be built on them. Copies of these packages are kept in `$SAFEDIR/code` directory. They can be manually installed.

To compile and install the SAFE toolkit:

```
$ cd $SAFEDIR/code
$ tar xzvf safe.1.0.tar.gz && cd safe.1.0
$ ./configure --prefix=$SAFEDIR && make && make install
```

The executable files, i.e., `PitchTrain` and `PitchExt`, are installed to `$SAFEDIR/bin`.

2.2 Data Preparation

Example scripts are created for showing the functions of the toolkit. A tar.gz file of the F0 database and noise wave files, i.e., `keele_and_noise.tar.gz`, used in the tutorial can be downloaded from: `http://www.ee.ucla.edu/~weichu/safe/download.html`. Then:

```
$ tar xzvf keele_and_noise.tar.gz -C $SAFEDIR
```

The KEELE database [1] is extracted to `$SAFEDIR/data/keele/`. The white and babble noise wave files (`w.wav` and `b.wav`) from NOISEX-92 database [2] are extracted to `$SAFEDIR/data/noisex92/`.

Before starting training F0 estimation models, noises files are needed to be added to the KEELE database. In the example, first manually install `sox` (http://sox.sourceforge.net/) (Yum is not suggested.), then:

```
$ cd $SAFEDIR/example
$ ./add_noise.sh
```

The software for adding noise used here is FaNT [3]. If the bin files `fant` and `create_list` in the `$SAFEDIR/bin` does not work, the FaNT software can be downloaded from `http://dnt.kr.hs-niederrhein.de/download.html`. In case the software is removed or updated, a copy of the software is kept in `$SAFEDIR/code/fant`. The F0 ground truth file list is also generated to `$SAFEDIR/data/keele/gth.lst`. Each line of the list is a path of an F0 ground truth file, whose format is shown in Appendix 3.3.

2.3 Training Models for F0 Estimation

The training procedure accumulates F0 statistics and estimates F0 estimation models from the wave files and F0 ground truth files. The options of `PitchTrain` are as follows:

`-c1 scf` Load signal processing configurations from file `scf`. The format of `scf` are explained in Appendix 3.1.

`-c2 pcf` Load F0 estimation configurations from file `pcf`. The format of `pcf` are explained in Appendix 3.2.

`-l w2f` Load wave files and their corresponding F0 ground truth files from file list `w2f`, which is a text file. Each line of `w2f` has two strings, which are the paths of the linear PCM WAV formatted file and its corresponding F0 ground truth files, respectively.

`-m mod` Save F0 estimation models to file `mod`. The format of `mod` is shown in function `spa_load_pch_mod` in `PitchExt.c` of the code.

In the example, the training on KEELE corpus can be started by:

```
$ cd $SAFE/example
$ ./train_models.sh
```

From each set of data, F0 training will generate two models: zero mean version `*.zm.pdf` and non-zero mean version `*.pdf`. The accumulated statistics are store in file `*.acc`.

In fact, the trained F0 estimation models are released with the project. Re-training the models is not necessary.

2.4 F0 Estimation and Error Evaluation

The testing procedure loads a wave file, trained F0 estimation models, and voicing status information, then estimates the F0 contour. The options of `PitchExt` are as follows:

`-s i` Set F0 estimation stage to i. In Stage 1, F0 candidates with likelihoods are generated; in Stage 2, the estimated F0 contour is generated.

`-c1 scf` Load signal processing configurations from file `scf`. The format of `scf` are explained in Appendix 3.1.

`-c2 pcf` Load F0 estimation configurations from file `pcf`. The format of `pcf` are explained in Appendix 3.2.

`-i inf` Load input file from file `inf`. In Stage 1, `inf` is a linear PCM WAV formatted file; in Stage 2, `inf` is a file that stores F0 candidate likelihoods.

`-f uvf` Load voicing information from file `uvf`. The format of `uvf` is shown in Appendix 3.4.

`-m mod` Load F0 estimation models from file `mod`. The format of `mod` is shown in function `spa_load_pch_mod` in `PitchExt.c` of the code.

`-o ouf` Save F0 estimation output to file `ouf`. In Stage 1, `ouf` is a file that stores F0 candidate likelihoods; in Stage 2, `ouf` is a file that stores the estimated F0 contour. The format of F0 contour file is the same as that of F0 ground truth file.

In the example, the testing on KEELE corpus can be started :

```
$ cd $SAFE/example
$ ./estimate_contours.sh
```

The estimated F0 contours are also scored by the Perl script `$SAFEDIR/perl/CalPtkErrors.pl`. Gross Pitch Error (GPE), Mean of Fine Pitch Errors (MFPE), Standard Deviation of Fine Pitch Errors (SDFPE) in percentage are generated. Note that it is not proper to train and test on the same corpus. The example scripts are just for showing the usage of the SAFE toolkit. In the paper of SAFE algorithm [4], two configurations were used: 5-fold cross-validation training and testing on the KEELE corpus, and training on KEELE corpus but testing on CSTR corpus [5].

Another script `$SAFEDIR/example/estimate_one.sh` can be used for estimating the F0 contour for one single wave file:

```
$ cd $SAFE/example
$ ./run_one.sh data/f1_w_0.wav data/f1_w_0.uvs data/f1_w_0.det w 0 0
```

The parameters are explained in the comment section of `run_one.sh`.

3 Appendix

3.1 Signal Processing Configuration

The signal processing configuration file `scf` is a text file. Each line has two strings: 'tag' and 'value'. The tags, values, suggested values, and descriptions are shown in the following:

Tag	Value	Suggested	Description
`<FFT_NUM>`	`i`	16384	Set FFT size to `i`.
`<USE_POWER>`	`b`	1	When `b=1`, use power spectrum; when `b=0`, use magnitude spectrum.
`<PRE_EMPH>`	`f`	0.97	Set preemphasis factor to `f`.
`<SAMP_RATE>`	`f`	-	Set sampling rate of the wave files to `f`.
`<FRAME_STEP>`	`f`	10	Set frame step size to `f` milliseconds.
`<FRAME_LEN>`	`f`	40	Set frame length to `f` milliseconds.
`<STRCT_FRM_TIME>`	`b`	1	When `b=1`, the first frame centers on 0 seconds; when `b=0`, the first frame starts at 0 seconds.

There are other options which are not related to SAFE. For more details, see function `spa_spec_load_cfg` in `spec_dom.c` in the code.

3.2 F0 Estimation Configuration

The F0 estimation configuration file `pcf` is a text file. Each line has two strings: 'tag' and 'value'. The tags, values, suggested values, and descriptions are shown in the following:

Tag	Value	Suggested	Description
<F0_MIN>	f	50	Set the lower bound of F0 range to f Hz.
<F0_MAX>	f	400	Set the upper bound of F0 range to f Hz.
<F0_CAND_NUM>	i	100	Set the number of F0 candidates to i.
<SPEC_CUT>	f	3000	Only the frequency components between 0 to f Hz in spectra are retained for F0 estimation.
<BAND_NUM>	i	3	Set the number of bands between 0 and the cut-off frequency to i. The bandwidths of bands are evenly divided.
<DIFF_SNR_NUM>	i f1 f2	3 -0.33 0	Set the number of normalized SNR bins to i. The bins are: (−∞ f1], (f1, f1+(f2-f1)/(i-2)], ···, (f2, −∞). Currently, only peaks with normalized SNR greater than f2 are used for F0 estimation.
<ABSO_SNR_NUM>	i f1 f2	8 5 65	Set the number of absolute SNR bins to i. The bins are: (−∞ f1], (f1, f1+(f2-f1)/(i-2)], ···, (f2, −∞).
<LR_SCA>	f	-0.33	Set the parameter α to f in logsitic regresstion function $f(x) = \frac{1}{1+\alpha e^{-\beta x}}$.
<LR_EXP>	f	1.0	Set the parameter β to f in logsitic regresstion function $f(x) = \frac{1}{1+\alpha e^{-\beta x}}$.
<OCTAVE_COST>	f	0.35	Set F0 octave jump cost in dynamic programming to f.
<DP_RATIO>	f	0.08	Set the ratio of transition cost to local cost in dynamic programming to f.

There are other options which are not needed to be changed. For more details, see function spa_spec_pch_init in pitch.c in the code.

3.3 F0 Ground Truth File

The F0 ground truth file is a text file. Each line of the file corresponds to a speech frame. The frame step size is 10 ms. Since it is assumed F0 does not change within a short section, the value in that line shows the F0 value or voicing status of that speech frame. Note the F0 ground truths of the KEELE database are obtained through running autocorrelation on laryngograph (lx) signal plus some manual corrections. There are 4 types of values in an F0 ground truth file:

- f a voiced section, the F0 value is f Hz.
- 0 an unvoiced section.
- -1 an uncertain voiced section, the speech signal is periodic, but the lx signal is not.
- -f an uncertain voiced section, the lx signal is periodic, but the speech signal is not. The F0 estimated from lx signal is f Hz.

For example, see the file $SAFEDIR/data/keele/gth/f1nw0000.gth.

3.4 Voicing Status File

The voicing status file is a text file. Each line of the file corresponds to a speech frame. The frame step size is 10 ms. The value in that line shows the voicing status of that frame. If the value is

greater than 0, it is a voiced frame; otherwise, it is not a voiced frame (an unvoiced frame or silence frame).

References

[1] F. Plante, G. Meyer, and W. A. Ainsworth, "A pitch extraction reference database," in *EUROSPEECH*, pp. 837–840, 1995.

[2] A. Varga, H. J. M. Steeneken, M. Tomlinson, and D. Jones, "The NOISEX-92 study on the effect of additive noise on automatic speech recognition," in *Technical report, DRA Speech Research Unit*, 1992.

[3] H. Hirsch and D. Pearce, "The AURORA experimental framework for the performance evaluation of speech recognition systems under noisy conditions," in *ASR2000*, pp. 181–188, 2000.

[4] W. Chu and A. Alwan, "SAFE: a statistical algorithm for F0 estimation for both clean and noisy speech," in *Interspeech*, pp. 2590–2593, 2010.

[5] P. Bagshaw, S. Hiller, and M. Jack, "Enhanced pitch tracking and the processing of F0 contours for computer aided intonation teaching," in *EUROSPEECH*, vol. 2, pp. 1003–1006, 1993.