A Manual of TransShiftMex

Shanqing Cai
Speech Communication Group, Research Laboratory of Electronics, MIT
E-mail: cais@mit.edu

December 2008

Section 0. Getting started - Running the demos

There are two demo routines in this package. They are mcode/TransShiftDemo_Monophthong.m and mcode/TransShiftDemo_Triphthong.m. The former demonstrates fixed perturbation (F1-up) on a steady-state vowel (/a/ in Mandarin); the latter shows time-varying perturbation (F1-inflate) on a triphthong (/iau/ in Mandarin).

Running either file will generate two windows. The first window shows the spectrograms of the original and shifted speech sounds with the F1 and F2 tracks overlaid. The second window plots the original and shifted F1-F2 trajectories versus time and in a formant plane.

Note that in each of the two m-files, you can modify Line 3 to switch from a sound sample produced by a male speaker to that produced by a female speaker, or vice versa. In Line 4, you can specify whether the original and shifted sound will be played for you to hear.

Section 1.1. Usage of TransShiftMex

TransShiftMex(0)

Enumerate all recognized audio input/output devices.

TransShiftMex(1)

Start a trial.

TransShiftMex(2)

End a trial.

TransShiftMex(3, paramName, paramValue, toPrompt)

Set a parameter. Table 1 contains a complete list of the parameters of TransShiftMex.
paramName is a char string. It is the name of the parameter to be set.
paramValue is an int, Boolean (0/1) or double type numerical scalar or vector. The appropriate type and size are listed in Table 1.
toPrompt is a Boolean number. It specifies whether TransShiftMex should generate a text prompt in MATLAB upon setting the parameter.

[signalMat, dataMat] = TransShiftMex(4)

Get data from TransShiftMex. This is usually done after a trial.
signalMat is a $N_s \times 2$ matrix. N_s is the number of samples. The first column contains the input acoustic signals, whose sampling frequency is specified in the parameter *srate*[1]. The second column contains the output acoustic signal. When shifting is done (i.e., *bshift* = 1), this is the shifted sound. It has the same sampling frequency as the input signal.

[1] In this manual, Italic fonts indicate parameters of TransShiftMex.

dataMat is a $N_f \times k$ matrix, N_f being the number of frames. A frame corresponds to *framelen* time samples. The number of columns, k, depends on the order of LPC and the number of tracked formants.
The meanings of the columns of dataMat are listed below.

Column 1: sample number at the beginning of each frame.
Column 2: unsmoothed frame-by-frame RMS amplitude of the input signal.
Column 3: smoothed frame-by-frame RMS amplitude of the input signal.
Column 4: smoothed frame-by-frame RMS amplitude of the pre-emphasized (high-pass filtered input signal.
Columns 5 - 8: formant frequency estimates of the first *ntracks* formants (Hz). Assume *ntracks* = 4.
Columns 9 - 12: radii in the z-plane of the first *ntracks* formants. Asume *ntracks* = 4;.
Columns 13 - 14: time derivatives of F1 and F2.
Columns 15 - 16: F1 and F2 in the output signal. When shifting is done (i.e., *bshift* = 1), these are the shifted F1 and F2.
Columns 17 - k: the frame-by-frame LPC coefficients.

TransShiftMex(5, frameData)

Offline calling of TransShiftMex. This is usually used in offline processing of data or debugging. frameData is a 1×(*framelen*×*downfact*) vector.

TransShiftMex(6)

Reset the status of TransShiftMex.

TransShiftMex(11)

Sine wave (pure tone) generator. Plays a continuous pure tone of frequency *wgfreq* (Hz), amplitude *wgamp* and initial time *wgtime*, that is, *wgamp*×sin(*wgfreq*×(t+*wgtime*)). No ramp is imposed.

TransShiftMex(12)

Waveform playback. The waveform is specified in the 1×120000 array *datapb*.

Table 1.Input parameters

Parameter Name	**Type**	**Description**	**Default value**[2]
srate	int	Sampling rate in Hz	12000[3]
framelen	int	Frame length in number of samples	16
ndelay	int	Processing delay in number of frames	7
nwin	int	Number of windows per frame. Each incoming frame is divided into nwin windows	1
nlpc	int	Order of linear predictive coding (LPC)	13 for male speakers and 11 for female speakers
nfmts	int	Number of formants to be shifted.	2
ntracks	int	Number of tracked formants. The 1st to the nfmts-th formants will be tracked.	4
avglen	int	Length of the formant-frequency smoothing window (in number of frames)	8
cepswinwidth	int	Low-pass cepstral liftering window size	Depends on the F0 of the speaker . See Section 1.3.
fb	int	Feedback mode. 0: mute (play no sound) 1: normal (speech only) 2: noise only	1

[2] These default values are contained in mcode/getDefaultParams.m.
[3] 48000 Hz downsampled by a factor of 4.

		3: speech + noise. Note: these options work only under TransShiftMex(1).	
minvowellen	int	Minimum allowed vowel duration (in number of frames)	60 (60 * 16 / 12000 = 80 ms)
scale	double	Scaling factor imposed on the output	1
preemp	double	Pre-emphasis factor	0.98
rmsthr	double	Short-time RMS threshold	Varies. Depends on many factors such as microphone gain, speaker volume, identity of the vowel, etc.
rmsratio	double	Threshold for short-time ratio between original energy and high-pass energy. Used in vowel detection. See Section 1.2.	1.3
rmsff	double	RMS calculation forget factor	0.95
wgfreq	double	Sine-wave generator frequency in Hz	1000
wgamp	double	Sine-wave generator frequency (wav amp)	0.1
wgtime	double	Sine-wave generator initial time, used to set the initial phase.	0
datapb	double, 1×120000 array	Arbitrary sound waveform for playback. The sampling rate of the playback is 48000 Hz. Therefore TransShiftMex can playback 2.5 seconds of sound.	zeros(1,120000)
f2min	double	Lower boundary of the perturbation field (mel)	N/A
f2max	double	Upper boundary of the perturbation field (mel)	N/A
f1min	double	Left boundary of the perturbation field (mel)	N/A
f1max	double	Right boundary of the perturbation field (mel)	N/A
lbk	double	Slope of the tilted left boundary of the perturbation field (mel/mel)	N/A
lbb	double	Intercept of the tilted right boundary of the perturbation field (mel)	N/A
pertf2	double 1×257 array	The independent variable of the perturbation vector field (mel). See Section 1.2.	N/A
tripertamp	double 1×257 array	The 1st dependent variable of the perturbation field: amplitude of the vectors (mel). See Section 1.2.	N/A
tripertphi	double 1×257 array	The 2nd dependent variable of the perturbation field: orientation angle of the vectors (radians). See Section 1.2.	N/A
triallen	double	Length of the trial in sec. “triallen” seconds past the onset of the trial, the playback gain is set to zero.	2.5
ramplen	double	Length of the onset and offset linear ramps in sec.	0.05
afact	double	α factor of the penalty function used in formant tracking. It is the weight on the bandwidth criterion (see Section 1.4).	1
bfact	double	β factor of the penalty function used in formant tracking. It is the weight on the a priori knowledge of the formant frequencies (see Section 1.4)..	0.8

gfact	double	γ factor of the penalty function used in formant tracking. It is the weight on the temporal smoothness criterion (see Section 1.4)..	1
fn1	double	A priori expectation of F1 (Hz)	591 for male speakers; 675 for female speakers. (Note these values were selected for the Mandarin triphthong /iau/.)
fn2	double	A priori expectation of F2 (Hz)	1314 for male speakers; 1392 for female speakers. (Note these values were selected for the Mandarin triphthong /iau/.)
bgainadapt	Boolean	A flag indicating whether gain adaptation is to be used (See Section 1.6)	0
bshift	Boolean	A flag indicating whether formant frequency shifting is to be used. Note: the following parameters must be properly set beforehand in order for the shifting to work: rmsthresh, rmsratio, f1min, f1max, f2min, f2max, lbk, lbb, pertf2, pertamp, pertphi, bdetect.	1
btrack	Boolean	A flag indicating whether the formant frequencies are tracked. It should almost always be set to 1.	1
bdetect	Boolean	A flag indicating whether TransShiftMex is to detect the time interval of a vowel. It should be set to 1 whenver bshift is set to 1.	1
bweight	Boolean	A flag indicating whether TransShiftMex will smooth the formant frequencies with an RMS-based weighted averaging.	1
bcepslift	Boolean	A flag indicating whether TransShiftMex will do the low-pass cepstral liftering. Note: cepswinwidth needs to be set properly in order for the cepstral liftering to work.	1

Section 1.2. The perturbation field

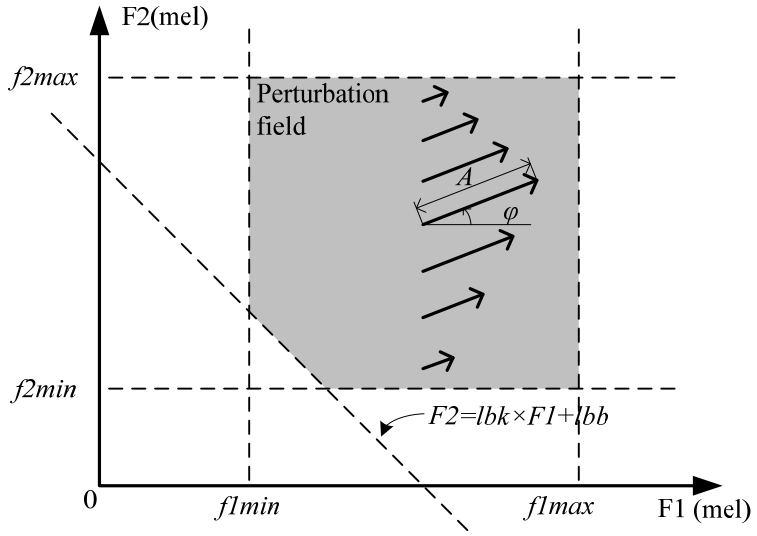

Figure 1. A schematic drawing of the perturbation field. The dashed lines shows the boundaries of the perturbation field. The arrows show the perturbation vectors. The shaded region is the perturbation field. A and θ are the magnitude and angle of the vector, which are both functions of F2.

The perturbation field is a region in the F1-F2 plane in which F1 and F2 will be shifted in a F2-dependent way. As shown schematically shown in Fig. 1, the location of the field is defined by five boundaries, which are respectively

$$F1 \geq f1Min; \quad (1)$$

$$F1 \leq f1Max; \quad (2)$$

$$F2 \geq f2min; \quad (3)$$

$$F2 \leq f2max; \quad (4)$$

$$F2 \geq lbk \times F1 + lbb, \text{ if } lbk \leq 0; \text{ or } F2 \leq lbk \times F1 + lbb, \text{ if } lbk > 0; \quad (5)$$

Meanwhile, a set of criteria about the short-time RMS need to be simultaneously met in order for the formant shifting to happen. These are,

$$(RMS_S > 2 \times rmsthr \text{ and } RMS_S / RMS_P > rmsratio / 1.3),$$

$$\text{or, } (RMS_S > rmsthr \text{ and } RMS_S / RMS_P > rmsratio), \quad (6).$$

In Equation (6), RMS_S is the smoothed frame-by-frame RMS amplitude of the input signal, and RMS_P is the smoothed frame-by-frame RMS amplitude of pre-emphasize (i.e., high-pass filtered) version of the input signal. The ratio between RMS_S and RMS_P is an indicator of how much the acoustic energy in the frame is dominated by the low-frequency bands. This ratio should be high during a vowel sound, and relatively low during a consonant sound. The criterion on this ratio reduces the possibility that an intense consonant is recognized as a vowel.

In summary, detection of a vowel and shifting its formant frequencies is contingent upon simultaneous satisfaction of Equations (1) – (6). The boundary defined by Equation (5) is in general a tilted line (see Fig. 1), and may seem a little bit peculiar. It was added because it was found to improve triphthong detection reliability in the Mandarin triphthong perturbation study. If you find it not necessary, the most convenient way to disable it is to set *lbb* and *lbk* both to zero.

Similarly, if your project is concerned with only a fixed amount perturbation to a steady-state vowel, you may wish not to use the boundaries *f1min*, *f1max*, *f2min*, and *f2max*, and rely only on the RMS criteria in Eqn. (6). You can achieve this by simply setting *f1min* and *f2min* to 0 and *f1max* and *f2max* to sufficiently large values (e.g., 5000).

The perturbation field is a vector field (arrows in Fig. 1). Each vector specifies how much F1 and F2 will be perturbed, respectively. Each vector is defined by a magnitude A (mel) and an angle φ (mel), which corresponds to *pertamp* and *pertphi* in the parameter list. Both A and φ are functions of F2. The mappings from F2 to A and φ are specified in the form of look-up tables (LUT) by the three parameters *pertf2*, *pertamp* and *pertphi*, which are all 1×257 vectors. During the trial, the amount of formant frequency shifting are determined by linear interpolation in this LUT.

This design should be general enough to allow flexible F2-dependent perturbations. However, your project may concern with only fixed perturbation to a steady-state vowel, and hence not require this flexible setup. If that's the case, you can simply set both *pertamp* and *pertphi* as constant. For example, if you want to introduce a 300-mel downward shift to the F1 of a steady-state vowel (e.g., /ɛ/), you can simply let *pertamp* be a 1×257 vector of all 300's and let *pertphi* be a 1×257 vector of all π's. Here, *pertf2* should be a 1×257 linear ramp from *f2min* to *f2max*.

You should also keep in mind that the parameters *f1min*, *f1max*, *f2min, f2max, lbk, lbb, pertf2,* and *pertamp* are all supposed to have the unit of mel, despite the fact that the formant frequency outputs in dataMat (See Section 1.1) and other parameters of TransShiftMex (e.g, *srate*, *fn1*, *fn2*, *wgfreq*, see Table 1) have the unit of Hz.

Section 1.3. Cepstral liftering

To improve the quality of formant estimation for high-pitch speakers, low-pass liftering was performed on the cepstrum, which consisted of the following steps.

1) Log magnitude spectrum of the signal was computed using fast Fourier transform.
2) The log magnitude spectrum was Fourier transformed to give the cepstrum.
3) The cepstrum was low-pass liftered by applying a rectangular window. The cut-off quefrency of the window q_c (in s) can be selected to be,

$$q_c = 0.54\,\mathrm{sec}/\overline{F}_0\,, \qquad (7).$$

where $\overline{F}_0$ is the average fundamental frequency (F0) of the speaker. For example, if the average F0 of the speaker is 200 Hz, then $q_c = 0.54 / 200 = 0.0027$ *(s)*. Since the sampling rate of the signal is 12000 Hz by default, *cepswinwidth* should be 0.0027 s × 12000 Hz ≈ 32.

4) The liftered cepstrum was transformed back into the frequency domain, and then back into the time domain. LPC analysis was performed on the resultant time-domain signal.

The effect of the cepstral liftering procedure is quantitatively evaluated in Section 2.1. It is our recommendation that cepstral liftering should almost always be used, on both female and male speakers. However, if you wish to disable it, you can achieve this by setting *bcepslift* to 0.

Section 1.4. Formant tracking based on a dynamic programming algorithm (Xia and Espy-Wilson 2000)

To improve the estimation of moving formants of the time-varying vowels, the LPC coefficients were subject to a dynamic programming formant tracker (Xia and Espy-Wilson, 2000), which was based on a cost function involving the following three criteria: (1) the bandwidth of the formants, (2) deviation from *a priori* template frequencies, and (3) non-smoothness of the frequencies. This algorithm uses Viterbi search to find the best path through the lattice of candidate formants. Further details of this algorithm can be found in ref/ Xia&Espy-Wilson2000.pdf.

Criterion (1) posits that a pole with relatively smaller bandwidths is more likely to be a true formant. Criteria (2) compares different formant candidates to a priori (expected) values of F1 and F2, which can be set in parameters *fn1* and *fn2* (in Hz). For example, if you know in advance that the speaker will produce a vowel /ɛ/, you should set fn1 and fn2 to values appropriate for this vowel. Criterion (3) prevents sudden jumps in the tracked formant values, which is based on the assumption that changes in the resonance property of the vocal tract should be relatively smooth.

The relative weights of criteria (1), (2) and (3) can be set in parameters *afact*, *bfact*, and *gfact*, respectively. For example, if you wish to put strong emphasis on the temporal smoothness of the formant frequencies, you should set *gfact* to a value greater than the default 1.

This formant tracking algorithm can be disable by setting *btrack* to 0 if you wish. However, it is strongly recommended not to do so.

Section 1.5. Smoothing of formant frequencies

To further improve the smoothness of the formant tracks, the estimated formant tracks were smoothed online with a window whose width is *avglen* frames. This smoothing involves a weighted averaging with the weights being the instantaneous root-mean-square (RMS) amplitude of the signal. This effectively emphasizes the closed phase of the glottal cycles, which was aimed at reducing the impact of the coupling of the sub-glottal resonances on the formant estimates.

The default value of *avglen* is 8 frames (10.33 ms). Larger *avglen* results in smoother formant frequency estimates, however, it also introduces larger lags into the tracked formant frequencies. Lags may not matter too much for steady-state vowels, but may pose a problem for time-varying vowels (diphthongs and triphthongs).

Section 1.6. Gain adaptation

In Mark Boucek's original design, he offered an option to adjust the gain of the shifted formant to make it sound more natural. Details of this gain adaptation algorithm can be found in pp. 52 – 53 of his thesis (Boucek 2007, see ref/Boucek-MSThesis-2007.pdf).

We found that this algorithm didn't significantly improve the naturalness of the shifted sound (the shifted sound already sounds reasonably natural). Therefore we decided not to use this algorithm by default. If you wish to use it, you can set *bgainadapt* to 1.

Section 1.7. The onset and offset ramps

During each trial, TransShiftMex introduces ramps to the output sound in order to prevent the unpleasant discontinuities at the beginning and end. You can set the duration of the ramps in parameter *ramplen* (in seconds). The duration between the two ramps, *triallen*, is equal to the

duration of the trial. *triallen* has the default value of 2.5 sec. Be careful if you wish to set *triallen* to a value greater than 2.5 sec. There is no guarantee that it will work.

The onset and offset ramps are effective not only under the speech-only mode (fb=1), but also under the noise-only (*fb* = 2) and speech+noise (*fb* = 3) modes (See Section 1.10). However, it doesn't work under the pure tone generater (TransShiftMex(11)) or the waveform playback (TransShiftMex(12)) modes.

Section 1.8. Using the pure tone generator

In our experiment, it is often desirable to have a pure tone generator, which can be used in audiometric procedures and calibrations. TransShiftMex offers such a capability. Running it under mode 11, i.e., TransShiftMex(11) will generate a continuous tonal output. The frequency of the sound is set in parameter wgfreq (in Hz); the amplitude is set in parameter wgamp (peak amplitude); the initial time is set in wgtime (in sec). For example, if you want to generate a continuous tone of 1 kHz with peak amplitude 0.1, of duration 1 sec and starting at phase 0, you can use the following MATLAB commands.

```
TransShiftMex(3,'wgfreq',1000,0);
TransShiftMex(3,'wgamp',0.1,0);
TransShiftMex(3,'wgtime',0,0);
TransShiftMex(11);
pause(1);
TransShiftMex(2);
```

However, this pure tone has no onset and offset ramps. If you wish to generate a tone burst with onset and offset ramps, you will have to use the waveform playback function of TransShiftMex (Section 1.9).

Section 1.9. Using the waveform playback function

To use the waveform playback function, you need to first set the waveform buffer in parameter *datapb*. datapb has a sampling rate of 48000 Hz, and a buffer size of 120000 samples, that is, 2.5 seconds. For example, running the following MATLAB commands will let TransShiftMex playback the sound represented in the "snd", a 1×120000 vector.

```
TransShiftMex(3,'datapb',snd,0);
TransShiftMex(12);
pause(2.5);
TransShiftMex(2);
```

You should note that no onset and offset ramps are imposed during the playback.

Section 1.10. Blending noise with speech feedback during the trials

In speech feedback perturbation experiments, it is often desirable to entirely mask all auditory feedback of speech by playing a relatively intense noise through the earphones, or to mix speech feedback through the earphones with a masking noise of a certain level to mask bone conducted feedback. You can achieve either of these by using *fb* = 2 or *fb* = 3 options under TransShiftMex(1). The noise waveform can be set in *datapb*. It should be a 1×120000 vector. When you use these options, onset and offset ramps will be imposed (See Section 1.7).

Section 2.1. Evaluating the accuracy of formant tracking

The accuracy of oformant tracking function of TransShiftMex was evaluated by running TransShiftMex on a set of synthesized vowel sounds[4]. These 14 vowels sounds are IY, IH, EH, AE, AH, AA, AO, UW, UH, ER, AY, EY, AW and OW [5] in American English. Three different profiles of F0s are generated: 1) constant, 2) falling and 3) rising. In constant-F0 profile, the F0 stays at one of the 8 F0 values throughout the course of the vowel. For the falling and rising profiles, the F0 falls or rises linearly with time by 20% during the course of the vowel. A set of 8 onset F0 values were used for each gender: 90, 100, 110, … 150 for male; and 160, 180, 200, …, 300 for female.

Hence, the set of test vowels consisted of full combination of 2 genders, 8 onset F0 values, 3 temporal profiles of F0, and 14 vowel identities, which amounted to 672 vowels in total (336 for each gender). Further details regarding the synthesis of these test vowels can be found in speechsyn/genTestVowels.m. These results can be reproduced by running mcode/evalTransShiftMex.m.

The error of the formant tracking was quantified as the relative error between the formant frequencies used in synthesizing the vowels ($F1_S$) and the formant frequencies estimated by TransShiftMex ($F1_T$):

$$Err_1 = \sqrt{\frac{1}{N_t}\sum_{i=1}^{N_t}\left(\frac{F1_T(i) - F1_S(i)}{F1_S(i)}\right)^2}, \quad (8)$$

$$Err_2 = \sqrt{\frac{1}{N_t}\sum_{i=1}^{N_t}\left(\frac{F2_T(i) - F2_S(i)}{F2_S(i)}\right)^2}, \quad (9)$$

In Equations (8) and (9), i denotes the temporal frame number, and N_t is the number of frames in the vowel. Err_1 and Err_2 are the RMS fraction error for F1 and F2, respectively. In this evaluation, the set of default parameters as listed in Table 1 is used. It should be noted that different orders of LPC (*nlpc*) were used for the male and female voices. For male ones, *nlpc* = 13; for female ones, *nlpc* = 11. The window size of the low-pass cepstral liftering (if used) is determined by Equation (7), according to the average F0 during each vowel.

Figures 2 and 3 show the results of evaluation for the male and female voices, respectively. In these figures, each data point comes from averaging the results for 14 different English vowels (see above). It can be seen that in both voices, there were trends for the error of formant tracking to increase with increasing F0, as expected. These trends were more pronounced for the female voice, which had higher F0s than the male one. Another noticeable general trend is that the accuracy of formant tracking was poorer if the F0 is changing (falling or rising) during the course of the vowel, which was of course due to the interference to LPC by F0. Comparison between the blue and red curves in these two figures clearly shows that the cepstral liftering improves the accuracy of formant tracking for both F1 and F2, in both constant-F0 and changing-F0 vowels. In general, the effect of cepstral liftering is more salient at higher onset F0s. These observations lead to our recommendation that the cepstral liftering should be almost always used (*bcepslift* set to 1). This is especially important for high-pitch speakers and for utterances with changing F0s.

[4] A MATLAB version of HLSyn (mlsyn) was used to synthesize these vowels.

[5] The phonetic notation here obeys ARPABET.

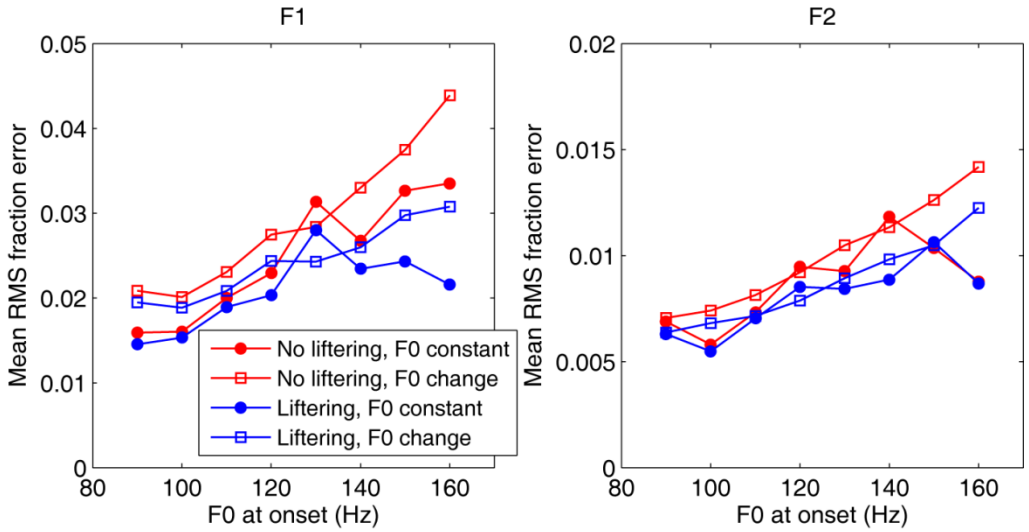

Figure 2. Results of evaluation of the accuracy of formant tracking by TransShiftMex on a male voice (See text for details). RMS fraction errors are plotted against onset F0. Different colors of the curves indicate whether cepstral liftering was used. Different symbols correspond to different temporal profiles of F0 during the vowel (filled circles: F0 constant; unfilled squares: F0 changing, that is, falling or rising). Left panel is for F1 and right for F2.

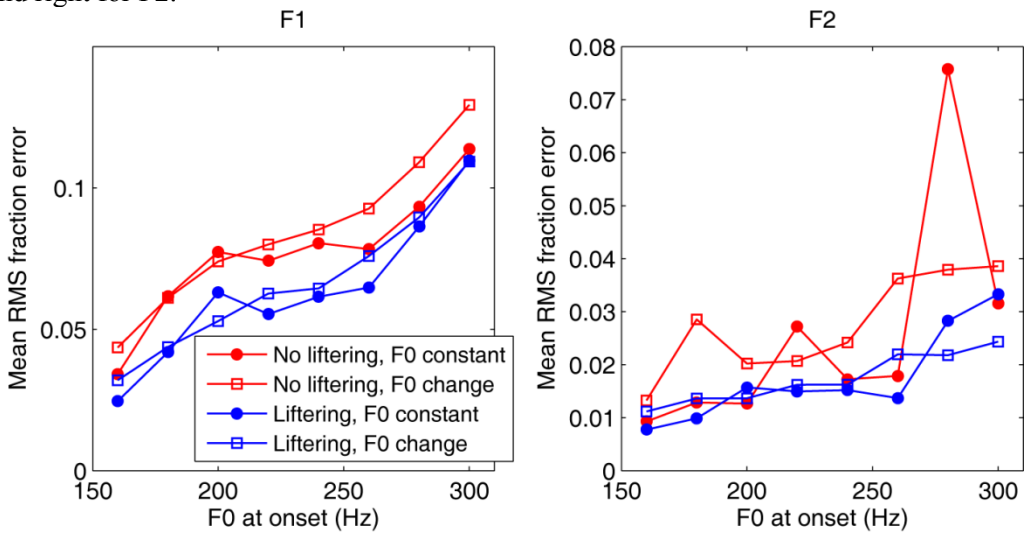

Figure 3. Results of evaluation of the accuracy of formant tracking by TransShiftMex on a female voice (See text for details). The format of this figure is the same as that of Fig. 2.

References

Boucek M. (2007). The nature of planned acoustic trajectories. Unpublished M.S. thesis. Universität Karlsruhe.

Xia K, Espy-Wilson C. (2000). A new strategy of formant tracking based on dynamic programming. In ICSLP2000, Beijing, China, October 2000.