TR-IT-0170

PSOLA Module for CHATR

Gregor Möhler

1996.5

ABSTRACT

In this work a signal processing module is presented that can deal with mismatches between the target specifications and the features of a selected unit in concatenative speech synthesis. Due to finite database length a selected unit will not necessarily meet its target specification in F_0 and duration. However, in prosodically important parts of speech such a mismatch cannot be accepted. The described module modifies the synthesized sentence in F_0 and duration based on the Pitch-Synchronous-OverLap&Add (PSOLA) algorithm. Different approaches of how to apply the algorithm to the selected units are discussed and details of the implementation are described. Finally, a proposal is given of how to apply the algorithm with varying prosodic relevance over the utterance.

Contents

1 Introduction

The work presented here was carried out during a three months stay by the author at ATR. It was done within the framework of CHATR, the ATR-ITL generic speech synthesis system.

The main interest followed by developers of CHATR is the non-uniform unit selecting, i. e. units are selected from large speech databases according to a set of parameters and then concatenated afterwards. Generally speaking, the unit that comes closest to a specific target and fits best into the synthesis waveform will be selected. Targets are mainly defined by the pitch, power and duration of the unit to select and its neighbors [2]. So the selection process could be seen as a search through the space of database phones [1].

We are dealing with databases containing typically 20 to 40 minutes of rough speech. Due to the finite size of the database an appropriate unit can not always be selected. A mismatch between the target values in pitch, power and duration and those of the unit selected is the result. For large parts of the sentence this will not cause any perceptional difficulties because in most cases the synthesized utterance carries a smooth F_0, even though it is not exactly the desired one. But there are parts within a phrase where a correct F_0 and duration is crucial. These are the parts where stress or accents are expressed. And also the F_0 on phrase boundaries carries important prosodical information.

Phrase accents are usually correlated to specific movements (tunes) in the pitch contour, whereas the most important acoustic cue to word stress is the duration of the vowel in the stressed syllable, i. e. if a syllable is accented, it will necessarily be much longer than the syllable in an unstressed position [3]. Sophisticated models have been developed to describe the prominent relevant parts in the pitch contour (e. g. ToBi, Fujisaki-model). These models are based on the observation that accents are in most cases expressed by rises and falls in the pitch contour. Additionally the last part of a phrase, the so-called phrase boundary, also carries important prosodic information. Hence, if we don't achieve to synthesize utterances with specific F_0 contours and duration values (at least at the significant parts) we cannot be sure that this utterance will carry the meaning we were aiming at.

The problem of insufficient database length could only be solved by collecting even bigger databases (with therefore a wider distribution of units) or signal processing after the selection process. In this work we are applying the signal processing algorithm PSOLA to the selected units to modify their fundamental frequency and duration.

The algorithm was most intensively tested on the German database that was also set up during the author's stay at ATR-ITL.

2 The PSOLA Algorithm

The Pitch Synchronous Overlap & Add (PSOLA) algorithm has been designed to independently modify a speech signal in F_0 and duration [4][5]. It works the following way:

The speech signal is first windowed pitch-synchronously by Hanning windows. The result are pitch synchronous short time (ST-) signals of 2 pitch periods length overlapping by one pitch period with the neighbors. Pitch synchronous labeling of the speech is required for this purpose, which could be achieved by the ESPS program *epochs*. This labeling is very crucial for the quality of the modified speech. The mentioned ESPS program, however, does not always locate the correct epochs. Especially in creaky periods the program tend to locate too many pitch-marks. This problem will be addressed later in this report (see section 7). For unvoiced regions pitch-marks are created with a fixed distance (10ms).

To achieve *pitch variations* the ST signals are shifted against one another in time and then added again (overlap & add - OLA). Pushing them together will result in higher pitch, tearing them apart lowers the frequency.

Duration on the other hand is changed by doubling or leaving out certain ST segments before the final process of addition. Inserting additional ST signals into the speech will slow down the speech, taking segments out will lead to shorter durations.

One crucial part of the algorithm is the mapping between the ST signals in the input stream and the ST signals in the output stream which is controlled by the flow of modification factors. The careful study of the time synchronization leads to a mapping represented by the following equation[5]:

$$t_s(u+1) - t_s(u) = \frac{1}{t'_s(u+1) - t'_s(u)} \int_{t'_s}^{t'_s(u)} \frac{P(t)}{\beta(t)} dt \qquad (1)$$

where t_s denotes the pitch-marks (and correspondent ST signals) of the input signal; t'_s are the pitch-marks (and ST signals) of the output signal; $P(t)$ is the pitch period of the input signal and $\beta(t)$ is the pitch scaling factor.

This integral equation is relatively easy to solve since the factors are piecewise linear. In the implementation described here the procedure is mainly the following (assuming constant factors over a pitch period for explanation purpose only):

1. At a specific time instant the output pitch period $P'(t)$ is determined by dividing the input pitch period at that time $P(t)$ by the modification factor $\beta(t)$.

2. Adding $P'(t)$ to the last pitch mark $t'_s(u)$ in the output stream gives us the next pitch mark $t'_s(u+1)$ in the output stream.

3. Considering the duration modification factors up to this time point (by integrating them) the time point $t_s(u+1)$ in the input stream corresponding to $t'_s(u+1)$ could be found.

4. The ST signal (i.e. pitch-mark) lying closest to this time point $t_s(u+1)$ is mapped next. This is actually the mechanism described above: ST signals are doubled or skipped depending on the distance between $t_s(u)$ and $t_s(u+1)$.

For factors changing within one pitch period the algorithm becomes only slightly more complicated. In this case $P'(t)$ is also the result of an integration.

The PSOLA algorithm produces very natural speech for smaller modification factors. For very good quality F_0 modification factors should generally range between 0.7 and 1.3. The range for high quality duration modifications depends largely on the phoneme the modification is applied to. For longer vowels stretching up to factor 2 still leads to good results, for shorter vowels (like schwas) even a stretch of 1.2 might cause considerable disturbances. This would suggest a duration dependent duration variation which is, however, not subject of this work.

3 The PSOLA Module within CHATR

3.1 General Description

In this chapter the architecture of the PSOLA Module is described. Mainly two questions are addressed: First, on which parts of the signal should the modification be carried out and how are these parts concatenated afterwards; and second, in which way are the modification ratios calculated. As we will see these question do not only affect technical aspects of the implementation but also some underlying phonetic issues.

3.2 Scope of Manipulations

When the PSOLA module is called within CHATR the unit selection process has already been carried out. This selection is mainly based on targets for F_0 and the phoneme duration. Search space is the entity of all the phonemes of the database with their feature vectors. Additionally a smooth F_0 and spectrum within the synthesized signal has been a goal of the search. Both, meeting the targets and signal continuity are weighted during the search.

The structures handed over to the PSOLA module thus contain a stream of units, each consisting of one or more subunits. In this case subunits are equivalent to phones, i. e. a unit is built up by a number of phones. Each of the phones within a unit had been neighbors in the speech signal where they come from. Thus a unit is a part of continuous speech without any artificial joints.

Two different scopes could be considered for manipulation: Either, the whole utterance is cut into ST signals, these ST signals mapped according to the modification factors, and then added by the overlap-add (OLA) algorithm. In this case the units are automatically joint through the OLA process. The other possibility is to consider one unit as scope for the PSOLA variations. Then after the OLA process the units are still un-joint. In this

case one of the previously implemented concatenation modules could be taken to carry out the unit splicing.

The first method has the advantage of doing everything in one process which would therefore lead to faster synthesis. But on the other hand spectra out of different contexts are overlapped over a whole pitch period, which might lead to perceivable joints. In fact, in an earlier version of PSOLA (module *PS_PSOLA*) this philosophy has been followed. And it turned out that the quality was reduced compared to pure concatenation by the concatmodule *DUMB+*. But it is unclear which eventually was the cause for this loss of quality.

The second method is somewhat slower. But it has the advantage that no overlay of units that come from different speech segments has to be carried out. Beside that we can effectively use the concatenation module *DUMB+* that has proved useful in non-signal processing concatenation. This slight preference and the will to try out a new approach has led to the decision to follow the second path.

3.3 Some Ways to Get the Modification Factors

In this subsection we will will discuss the question how the speech variation ratios should be calculated for the utterance. This is crucial especially for the F_0 contour of the utterance, the carrier of the prosodic information.

First, *duration modification factors* are calculated very simply for each phone by dividing the target phone duration by the duration of the phone in the selected unit. These factors are calculated for all phones although only unvoiced pitch period are finally manipulated by PSOLA. This is because the voicing information is not part of the `Sub_Unit` C-structure and therefore not available without additional computational cost.

In the case of the F_0 *factors* a first idea would be to calculate the F_0 average over the whole unit and compare this one to the mean target of that unit (cf. Figure 1). This has the advantage of a very natural F_0 contour because the details (like micro-prosody) are preserved from the original speech signal and F_0 is only shifted as a whole. But since units are of unpredictable length (1 to sometimes over 10 phones) it is an unrealistic approach.

The next possibility is to base the calculation on the *subunits* (i. e. phones) by dividing the mean target F_0 by the mean F_0 of the selected phone (cf. Figure 2) and keeping this factor constant over the length of the subunit. The average is only taken from the voiced parts of the phone. Expressed in an equation this gives:

$$f0\_factor(t) = \frac{\int_{sub\_unit} target\_f0(t)dt}{\int_{sub\_unit} unit\_f0(t)dt} \qquad (2)$$

This would also preserve the fine details of the F_0 since the subunits are only shifted in F_0 as a whole. However, at the phone boundaries jumps in F_0 will occur because neighbors may have different shifting factors. But we could assume that the original F_0 as well as the target F_0 has only slowly changed from one phone to its neighbor. Hence, the jump in F_0 may not be

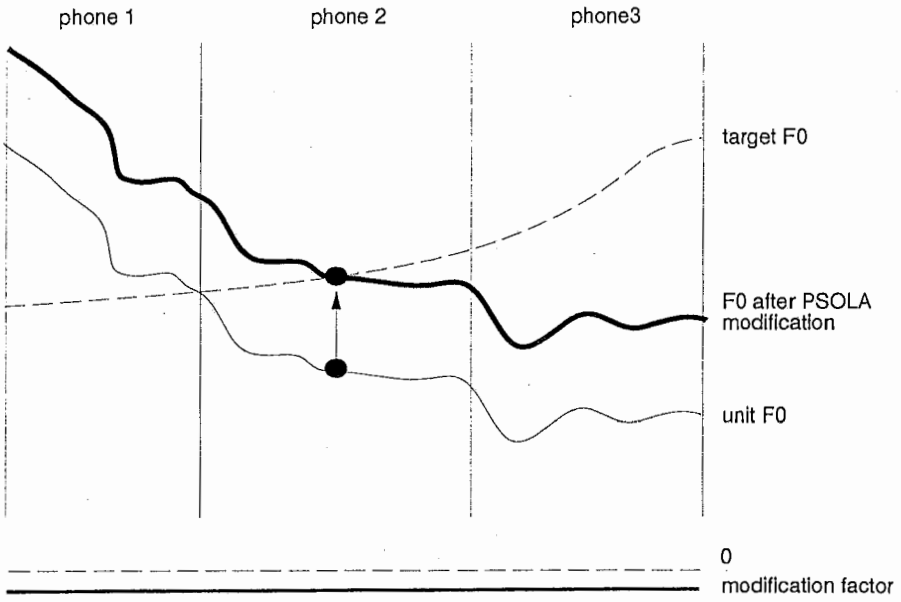

Figure 1: Unit based frequency shift

too large. In fact listening to the results of this manipulation method has not shown any such artifacts.

A main advantage of this kind of approach is its simplicity. The average unit F_0 and the average target F_0 are pre-calculated values, so you have a single division to calculate the factor (beside the limiting mechanism - see subsection 3.4). One bad thing about this method is, however, that not only micro-prosodic effects are preserved but also the slope of the F_0 within the phone. So if a unit with a falling F_0 has been chosen, but indeed we wanted to have a raising contour we would get a sort of sawtooth shaped F_0 (cf. Figure 2). It is unclear whether the right intonation is perceived and if it is, whether the distortion wouldn't be too high.

A third approach would be to *force* the waveform to the *predicted* F_0 *contour* (cf. Figure 3). Hence we get the following equation:

$$f0\_factor(t) = \frac{target\_f0(t)}{unit\_f0(t)} \tag{3}$$

In this case we would get the right slope of the F_0 which is a clear advantage upon the previous proposal. But all the naturalness that was in the original F_0 will be gone because the predicted F_0 is mostly a smooth (i.e. a sort of underlying) F_0 capturing none of the micro-prosodic effects. Additionally we would have a much higher computational cost because the single unit $F_0(t)$-values are not pre-stored in memory and thus must be collected from the F_0-files of the database. Opening and closing files, however, is a time consuming task and should therefore be avoided whenever possible. We didn't realize this method mainly because of the loss of naturalness due to the missing micro-prosody.

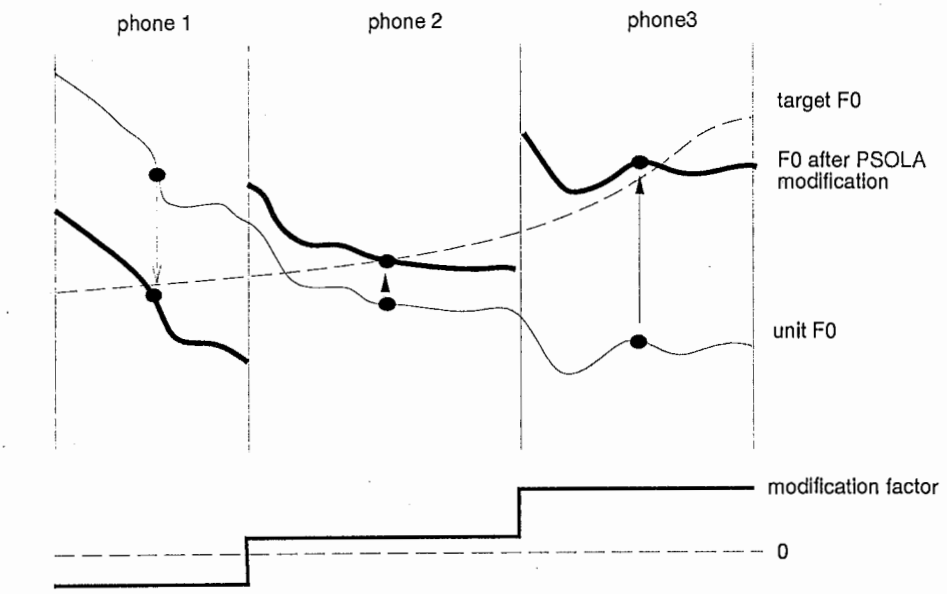

Figure 2: Phoneme based frequency shift

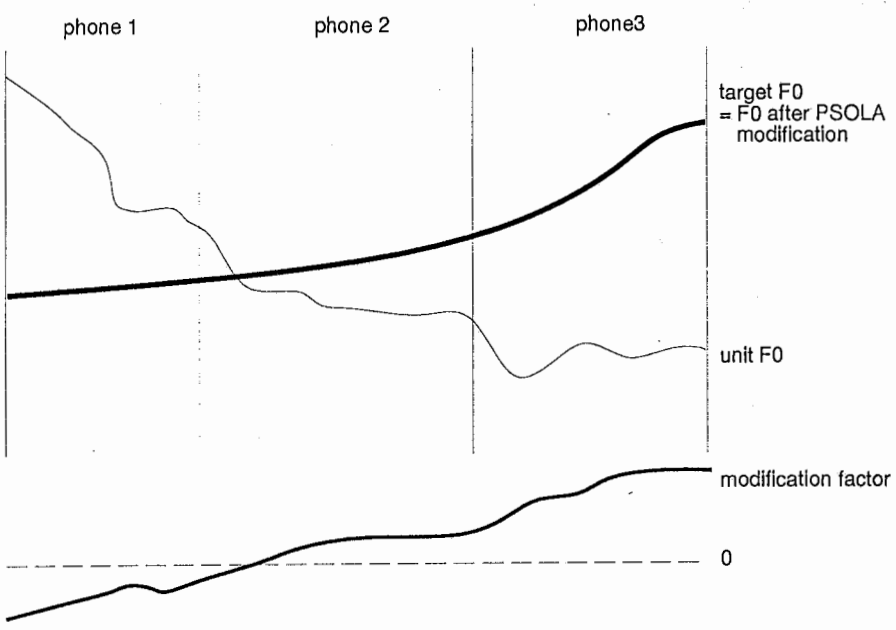

Figure 3: Imposing the predicted F_0 onto the waveform

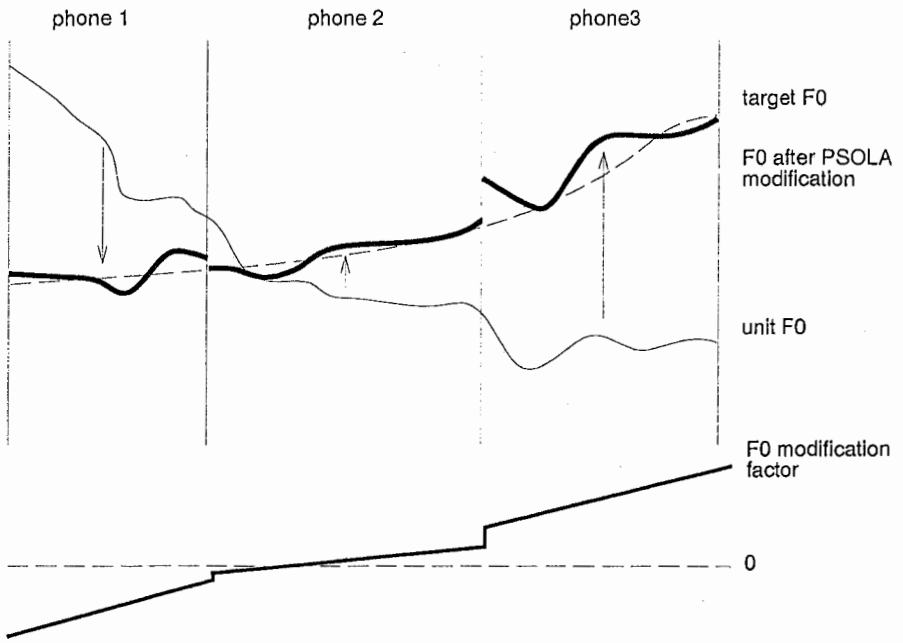

Figure 4: linear modification

One approach that preserves the micro-prosody but is capable to deal with F_0 slopes is the following one (cf. Figure 4): We calculate a linear modification curve from the factors of the previous method (e. g. by a regression line). Now we can realize the underlying slope of the F_0 within the phone. But there are drawbacks why we didn't implement this method: First, partially unvoiced phones (especially when the phone-label is not exact) and creaky voice and micro-prosodic phenomena would considerably influence the calculations and could lead to highly unreasonable results. And since what we actually want is the underlying F_0 of the original utterance, it is not clear why we don't calculate it in an off-line process, which leads us to our next method.

This last method is for sure the phonologically most motivated one (cf. Figure 4). It consists of *smoothing the* F_0 of the utterances in an off-line process. We take the original utterances and smooth them by a Hamming window glider of fixed length which gives us reasonable results for the underlying F_0. The window length is set to 180ms which is the value that smoothes the ToBI predicted F_0 contours. This method could be expressed in the following way:

$$f0\_factor(t) = \frac{target\_f0(t)}{uf0(t)} \tag{4}$$

with:

$$uf0(n) = \frac{1}{\sum_{i=0}^{l} w(i)} \sum_{j=-\frac{l-1}{2}}^{j=\frac{l-1}{2}} w(j) \cdot unit\_f0(n+j) \tag{5}$$

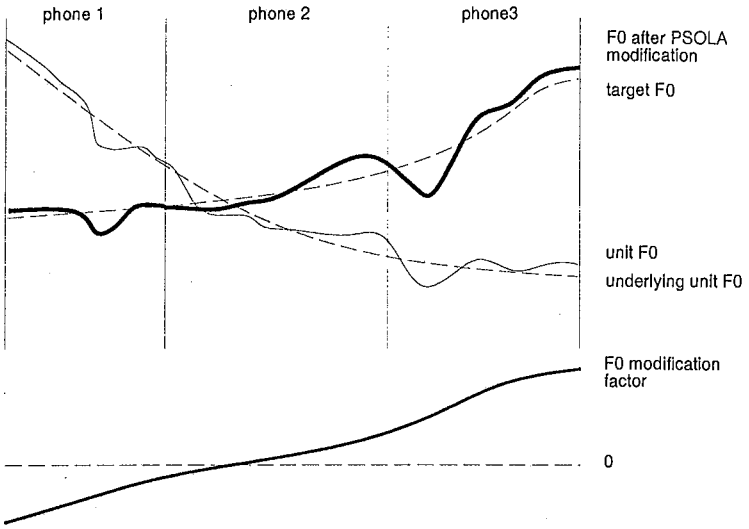

Figure 5: modification of the underlying F_0

where $uf0(n)$ denotes the underlying f0, $w(i)$ the window weights of the Hamming window and l the window length.

So, we have two F_0 contours with the same "amount of smoothness" to compare. Still we have to read the underlying F_0 of the selected units by opening a number of files which slows down the speed of the concatenation. But now the underlying F_0 of the units selected is modified to the target F_0 contour which is indeed a underlying F_0 as well. This is a phonetically correct solution of our problem.

We implemented both the second method of subunit constant shifts because of its simplicity and the last one where the underlying F_0 is corrected to the target contour because it should give us very correct results.

3.4 Factor Limitation

Due to the fact that the synthesis quality goes down for larger modification factors a limitation has been imposed on the factors. A hard limit didn't seem to be appropriate since it would suggest that the quality suddenly drops at a certain point. So a soft limitation has been chosen. As a limitation faction a 1-centered arctan-function seemed to be appropriate (cf. Figure 6). For smaller values it is approximately linear which corresponds to a direct 1-to-1 mapping between the calculated and the realized factor. For higher values a limit will be asymptotically reached. The exact function is:

$$\begin{aligned} \alpha &= 1 + \tfrac{2(max-1)}{\pi} \cdot \arctan(\tfrac{2(\alpha^*-1)}{max-1}) \quad for: \alpha^* > 1 \\ \alpha &= 1 + \tfrac{2(min-1)}{\pi} \cdot \arctan(\tfrac{2(\alpha^*-1)}{min-1}) \quad for: \alpha^* < 1 \end{aligned} \tag{6}$$

where α^* is the factor calculated by the ratio unit- to target-length/F_0 and α the factor that is finally passed to the PSOLA algorithm. max and min designate the upper and lower boundary of the arctan function.

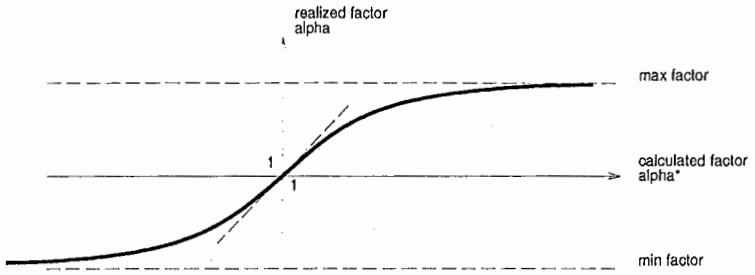

Figure 6: the arc-tan delimiter function

So far all parts of the utterance are treated equally. As mentioned earlier some parts of the utterance are more sensible to prosodic modifications than others. In these parts we would desire a higher range of modifications. In the prosodically irrelevant parts however PSOLA variations may be reduced to zero. A motivated approach to solve this problem is described in section 7

.

4 Software Description

The PSOLA algorithm could be found in the file `(chatr-dir)/src/ruc/gm_psola.c`. There is a header file with the name `(chatr-dir)/src/ruc/gm_psola.h` containing the C-structures used by the program.

4.1 The Parameter Set of the Program

The following parameters could be set by the CHATR user (see section 5 for how to set them within CHATR). Defaults are given in brackets:

f0_mode (1). Value=0 stands for subunit based F_0-variations. The F_0 modification factors are calculated once per subunit (phoneme) and stay constant over the phoneme (see section 3.3). A value=1 is referred to a modification which is based on the underlying F_0 contour.

uf0_dir ("uf0"). The directory where the files for the underlying F_0 could be found. *f0_mode* must be set to 1 to enable modifications of the underlying F_0. If no files could be found in this directory *f0_mode* is set to 0 and constant modifications are applied within subunits.

global_f0 (1), global_dur (1). Global modifications of pitch and duration, e.g. *global_f0*=2 will double the F_0 for the whole utterance. These values are overwritten by the parameters *f0_min, f0_max, dur_min, dur_max.*

f0_min (0.8), f0_max (1.3), dur_min (0.7), dur_max (1.5). These are the maximum factors that could be applied. (see section 3.4).

fact_step (0.001). As we will see in this section the PSOLA factors will be calculated for the whole utterance with a specific time interval *fact_step*, i.e. the factors are sampled at this step-width. This parameter is of more technical importance.

concat_ratio (0.7). When the units are concatenated by the DUMP+ concat-module (see below in this section) they should be spliced together at moments of low energy. The parameter *concat-ratio* specifies this moment in parts of the pitch interval, i.e. *concat_ratio* = 1 stands for concatenation at the epochs impulse and *concat ratio* = 0.7 means 30 % of of the pitch interval before the epochs impulse.

test (0). This is a parameter which is used only for debug purposes. A value=0 means no test (i.e. normal operation). For a value=1 silences are added between the units in the output signal. *test*=2 writes the ST signals separated by a short pause directly to the output, no OLA is carried out. And finally for *test*=3 a specific test signal can be specified as an input signal. Currently a sine waveform with a frequency of 100 Hz and amplitude=10000 is implemented as input. Additionally the modification values can be defined in a subroutine.

4.2 Implementation Details

The program starts by loading the parameters into the the memory (function `gm_init_params`). For unspecified parameters defaults are chosen (see section 4.1).

Then the units are loaded into the C structures by copying the waveforms and pitch-marks (function `gm_load_units`).

The modification factors (F_0 and duration) are calculated and written into the appropriate arrays by the function `gm_calc_modifications`. This is done for the whole utterance at once, even though the factor arrays are owned by the unit-structures (PSOLA acts on units - see section 3.2). The factors are synchronized with the input stream as requested by PSOLA and by default sampled at 1ms (parameter *fact_step*).

For *f0_mode*=0 the factors are calculated subunit-wise by using the precalculated values for subunit-F_0, target-F_0, subunit-duration and target-duration. Since the duration factors are defined by the ratio target-F_0/unit-F_0 without considering voicing information a mismatch could appear between unit- and target duration for unvoiced and partly unvoiced phones. After we have calculated the ratios the result is passed to the arctan-delimiter (see section 3.4) and sampled at *fact_step*.

In the case of *f0_mode*=1 the duration factors are determined the same way. The F_0 factors, however, are the calculated on the base of smoothed target F_0 and underlying F_0 (see section 3.3). The underlying F_0 of all the database utterances is calculated in an off-line process by smoothing the original F_0 contours by an 180ms Hamming window. Usually these contours are available in the directory (speaker-base)`/uf0`. A different directory can be specified by the parameter *uf0_dir*. To get the underlying F_0 of the

utterance to synthesize the values are read from the corresponding uf0-files by the function gm_make_unit_F0. It is important to mention that this function returns its values synchronized with the real unit length of the selected unit, whereas the chatr-function make_F0 returns the *target* F_0 values synchronized with the target durations. That is the reason why two timing-variables utt_time and tutt_time are used within the function gm_calc_modifications: one is synchronized with the unit timing the other with the target timing. The ratios we get by dividing underlying-F_0 and target-F_0 values are finally limited by our arctan-function and sampled at *fact_step*.

The following functions are called once per unit. First the ST signals are calculated pitch synchronously in the function gm_make_st_signals by Hanning windowing.

Then the ST signals are mapped according to the F_0 and duration factors. This is the task of the function gm_sts_mapping. How this works in detail has already been described earlier (section 2). After that the mapped ST signals are added by the OLA-algorithm in the function gm_ola.

The final process of concatenation differs in nothing from the DUMB+ algorithm used for unit-concatenation. In fact, the source code has been copied and modified to read its information out of the PSOLA unit-structures instead of the utterance-structure. The DUMB+ method works by shifting small windows around the concatenation point of the two units that should spliced together. There where the mean signal difference between the windows is the smallest the units are connected.

5 Usage of the PSOLA Module

The module described here is switched in by the following CHATR command:

```
(Parameter Concat_method GM_PSOLA)
```

The parameter described in the previous section 4.2 are set with the following command:

```
((set gm_psola_params '( (dur_max 1.2) (dur_min 0.8) (f0_max 1.2)
(f0_min  0.7)...)) )
```

Hereby, if you set only some parameters all the others are set back to default.

If verbosity is set on (command (Verbosity Warning On)) the following debug messages are printed out:

- While calculating the modification factors (*f0_mode* =0) with one line per subunit (phone):

```
unit-No./subunit-No. phone unit-F0 target-F0 -> F0-factor
   unit-dur target-dur dur-factor
```

- And for *f0_mode =1* with one line every 20ms:

```
Unit-No./Subunit-No. phone (time) unit-F0 target-F0 -> F0-factor
   unit-dur target-dur -> dur-factor
```

- After the PSOLA overlap process, once per unit:

```
unit-No. phones duration-error (unit-dur target-dur)
```

A duration error occurs if the target duration could not be reached. This is due to unvoiced segments that are not modified in duration and limitation of the factors. To keep the display clearer only the first letter of each phone is printed.

A typical debug printout looks like the following

- (*f0_mode* =0):

```
Unit 0/ 0 '#'  F0: u   0 t  69 ->     1      1  Dur: u  706 t  765 ->  1.11
Unit 0/ 1 'n'  F0: u  98 t 136 ->  1.49   1.49  Dur: u   47 t   43 -> 0.893
Unit 1/ 0 'a'  F0: u 106 t 131 ->   1.3    1.3  Dur: u   60 t   78 ->  1.38
Unit 2/ 0 'd'  F0: u   0 t 136 ->     1      1  Dur: u   78 t   47 -> 0.572
Unit 3/ 0 'a'  F0: u 140 t 134 -> 0.946  0.946  Dur: u   97 t   95 -> 0.974
Unit 3/ 1 's'  F0: u 124 t  79 ->   0.6    0.6  Dur: u   55 t   60 ->  1.12
Unit 4/ 0 'g'  F0: u   0 t  46 ->     1      1  Dur: u   80 t   88 ->  1.13
Unit 4/ 1 'e'  F0: u  78 t  59 -> 0.711  0.711  Dur: u  210 t  240 ->  1.18
Unit 4/ 2 't'  F0: u   0 t   0 ->     1      1  Dur: u   76 t   65 -> 0.821
Unit 0 '#n'     dur_err:   -47ms (u  761 t  808).
Unit 1 'a'      dur_err:    16ms (u   94 t   78).
Unit 2 'd'      dur_err:    23ms (u   70 t   47).
Unit 3 'as'     dur_err:    -7ms (u  148 t  155).
Unit 4 'get'    dur_err:    -3ms (u  390 t  393).
```

- (*f0_mode* =1):

```
...
Unit 0/0 '#'( 0.68s)  F0: u 149 t 134 -> 0.874  Dur: u  706 t  765 ->  1.11
Unit 0/0 '#'(  0.7s)  F0: u 146 t 138 -> 0.931  Dur: u  706 t  765 ->  1.11
Unit 0/1 'n'(0.719s)  F0: u 142 t 138 -> 0.964  Dur: u   47 t   43 -> 0.893
Unit 0/1 'n'(0.739s)  F0: u 137 t 134 -> 0.972  Dur: u   47 t   43 -> 0.893
Unit 1/0 'a'(0.753s)  F0: u 132 t 131 ->  0.99  Dur: u   60 t   78 ->  1.38
Unit 1/0 'a'(0.773s)  F0: u 105 t 129 ->  1.29  Dur: u   60 t   78 ->  1.38
Unit 1/0 'a'(0.793s)  F0: u 106 t 131 ->   1.3  Dur: u   60 t   78 ->  1.38
Unit 1/0 'a'(0.813s)  F0: u 109 t 133 ->  1.28  Dur: u   60 t   78 ->  1.38
Unit 2/0 'd'(0.813s)  F0: u 109 t 133 ->  1.28  Dur: u   78 t   47 -> 0.572
Unit 2/0 'd'(0.833s)  F0: u 129 t 134 ->  1.05  Dur: u   78 t   47 -> 0.572
Unit 2/0 'd'(0.853s)  F0: u 129 t 134 ->  1.05  Dur: u   78 t   47 -> 0.572
Unit 2/0 'd'(0.873s)  F0: u 130 t 136 ->  1.06  Dur: u   78 t   47 -> 0.572
Unit 3/0 'a'(0.891s)  F0: u 131 t 140 ->  1.09  Dur: u   97 t   95 -> 0.974
Unit 3/0 'a'(0.911s)  F0: u 141 t 138 -> 0.973  Dur: u   97 t   95 -> 0.974
...
Unit 0 '#n'     dur_err:   -57ms (u  751 t  808).
Unit 1 'a'      dur_err:    -2ms (u   76 t   78).
Unit 2 'd'      dur_err:    22ms (u   69 t   47).
Unit 3 'as'     dur_err:    -9ms (u  146 t  155).
Unit 4 'get'    dur_err:   -22ms (u  371 t  393).
```

6 Results

With the PSOLA tool described here further improvement of the speech output quality of the synthesis system CHATR can been achieved. If duration and F_0 modifications are constantly set to 1 (no modifications) the signal has fewer artifacts than that of the appropriate DUMB+ signal. This is due to the pitch synchronisity of PSOLA.

For the case of modifications the quality was almost unchanged until the modifications ratios reach the limits of 0.8 and 1.2 for F_0 variations and 0.7 to 1.3 for duration modifications. As mentioned earlier the quality for the latter is very much dependent on the phone that is modified. Longer vowels seem to allow larger ranges than shorter ones. This should be subject for further studies. For variations beyond the mentioned intervals the quality is still good but sounds less natural. But if a correct modeling of prosody is the goal these reduced quality has to be accepted.

The only disadvantage of the PSOLA module compared to the easy DUMB+ concatenation tool is the higher computational cost of the signal processing. Timing experiments have shown that the synthesis time is about 50% higher than with DUMB+ concatenation. This high increment might partly be due to the fact that files that must be read during the procedure to get the underlying F_0. To decrease computational costs we propose to store these values in the memory. So far, this increase of computing time appears also if no modifications are carried out (e.g. if the limits for the factors are all set to one). When in a further step we consider no modifications for prosodically irrelevant parts this should be fixed to save unnecessary computing power.

Concerning the different approaches of how to calculate the F_0-factors (section 3.3) no final decision could be made. In first listening tests no significant perceptional difference occured between the two implemented methods. But the fact that the underlying F_0 could be seen as the actual carrier of prosodic information would suggest to give preference to the method relying on the underlying F_0, which is indeed phonetically highly motivated.

7 Further Improvements

In this work only F_0 and duration modifications have been carried out on the signal. However, also *power modifications* are important for good speech synthesis quality and should therefore implemented. This is relatively easy to achieve within the module.

PSOLA modifies all voiced parts. However even within voiced segments the segment should not be modified everywhere. One example is the burst in voiced plosives. Moreover speech often contains irregularities like creaky voice that should better be left untouched. As a further improvement we would therefore suggest an off-line process that finds parts in voiced speech that should not be modified. And we could also ask whether some unvoiced elements should not be subject of duration modifications.

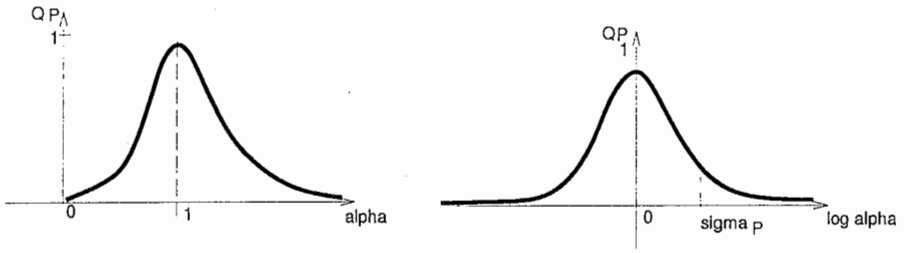

Figure 7: Quality Function of PSOLA

PSOLA depends very crucially on good pitch-mark labeling. However, the ESPS program *epochs* used for this purpose has difficulties especially in irregularities of the speech system (like creaks). Doubling of pitch-marks in such parts leads to poor synthesis results. That is why we propose another off-line process that finds such irregularities and corrects the pitch-marks.

The last improvement addresses the limitation of PSOLA F_0 variation factors. As mentioned before modifications are more important in parts of the utterance where phrase accents and boundaries occur. In other parts of the utterance modifications could be reduced to zero which means highest possible quality. An ad-hoc solution for this would play with the maximum and minimum factors that are passed to the arctan-delimiter function: Around prosodically important parts the limit is raised and in all other parts it is reduced to 1 (no modifications). A more motivated approach is the following:

7.1 An Approach to Include Prosodic Relevance into the System

First, we try to describe the process of PSOLA modifications in terms of mathematical expressions. The goal is to find a dependency between the calculated modification factor α^* and the factor α that applied in PSOLA. This function should be triggered by the relevance of prosodical modifications. In other words, it should take into consideration parts of the utterance where we definitively want the correct F_0 (around peaks and phrase boundaries) and all the other parts where this modification should only be carried out when the quality loss of a PSOLA modification is not too high.

As we have seen, the quality of PSOLA variations is highest when no modifications take place (α=1) and falls for smaller and higher factors. We suggest a quality function Q_P that runs on $\log(\alpha)$ to get a symmetric function. We choose a simple Gaussian function to describe the quality phenomena (see Figure 7). The function has the standard deviation σ_P. From listening to the PSOLA results we know that e^{σ_P} is around 1.3 (i.e. acceptable modifications between 0.7 and 1.3).

Furthermore we define a second quality function Q_I that describes the prosodic quality of a synthesized utterance. This function has a maximum when the calculated factor α^* equals to the factor α passed to PSOLA, i.e.

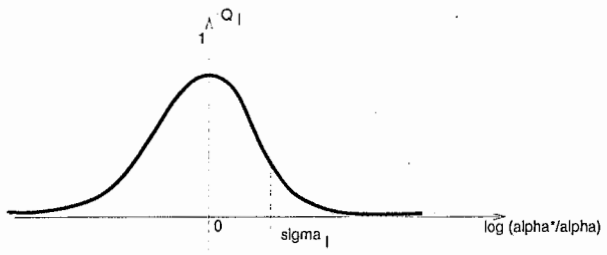

Figure 8: Prosodic Quality Function

when the waveform is totally modified to the desired one. We take $\log(\alpha^*/\alpha)$ as the independent variable (0 for $\alpha^* = \alpha$) and choose a Gaussian function as a shape (see Figure 8). The standard deviation σ_I denotes the bandwidth within modification are *prosodically acceptable*. For prosodically relevant parts σ_I will be smaller (smaller "deviations from the targets are allowed") and for all other parts it will be bigger(higher "deviations allowed").

Now, the total quality of the system regarding the two sub-qualities described above is:

$$
\begin{aligned}
Q &= Q_P \cdot Q_I && (7)\\
&= e^{-\frac{\log^2 \alpha}{\sigma_P^2}} \cdot e^{-\frac{1}{\sigma_I^2}(\log \alpha^* - \log \alpha)^2} \\
&= e^{-a^2} \cdot e^{-c(a-a^*)^2} && (8)\\
\textit{with}: & \\
a &= \frac{\log \alpha}{\sigma_P} && (9)\\
a^* &= \frac{\log \alpha^*}{\sigma_P} && (10)\\
c &= \frac{\sigma_P^2}{\sigma_I^2} && (11)
\end{aligned}
$$

Equation 8 is illustrated in Figure 9. The two quality functions shown must be multiplied to result in the total quality. We see that the maximum of the quality will be found somewhere between $\log \alpha = 0$ ($\alpha = 1$) and $\alpha = \alpha^*$ and depends on the two standard deviations which is not take into account here.

We calculate the maximum and find:

$$
\begin{aligned}
\log \alpha &= \frac{\log \alpha^*}{1 + (\frac{\sigma_I}{\sigma_P})^2} && (12)\\
\alpha &= (\alpha^*)^{\frac{1}{1+(\frac{\sigma_I}{\sigma_P})^2}} && (13)
\end{aligned}
$$

As expected we find $\alpha = \alpha^*$ for $\sigma_I \ll \sigma_P$, which is the case of high prosodic relevance; and $\alpha = 1$ for $\sigma_P \ll \sigma_I$, the case of prosodic irrelevance.

Equation (13) is illustrated in Figure 10. It gives us a motivated function of how α could be derived from α^* under the influence of different prosodic

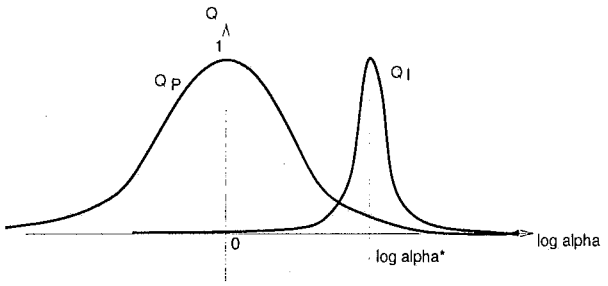

Figure 9: Total Quality

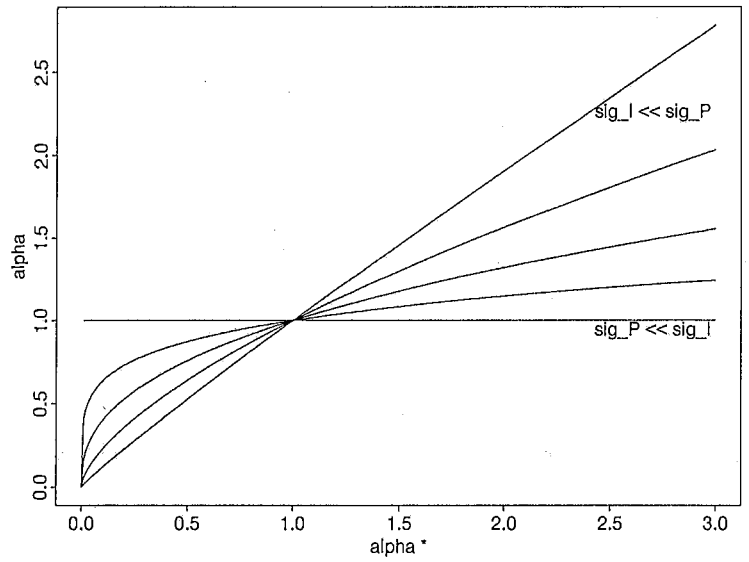

Figure 10: Dependency of the Applied Modification Factor on the Calculated Modification Factor for Different Values of σ_I and σ_P

relevance without the need of using a delimiter with an upper and lower level. In this case σ_P denoting the PSOLA quality would be a constant and σ_I the prosodic quality would a parameter depending on the prosodic relevance of the processed part of the utterance.

This is only a first approach to describe the quality of the PSOLA process. It may be necessary to expand it to other quality measures or different quality functions, e. g. there is also evidence that the derivative of F_0 plays an important role for the intonation perception.

8 Conclusion

We presented an implementation of a signal processing module that is capable to deal with the mismatch between the features of selected units and their target values. Several implementation alternatives have been described and their adequacy discussed.

The implementation showed that this module produces a signal closer to the targets with the cost of increased computation time. In most cases the intonation is perceived as smoother. The range of modification has to be limited in order to preserve high quality speech synthesis.

In the implementation all parts of the utterance are treated equally concerning the modification process. However a method has been described how prosodic relevance could be introduced into the system in order to apply modification only in parts where they are prosodically necessary, such phrase accents and phrase boundaries. This again can be expected to improve the quality of the synthesis.

9 Scripts and Stuff

This section contains a rough description of the scripts and small programs that were written during the stay. They are presented here simply to avoid double work.

(chatr-base-dir)`/db_utils/f0_smooth.c` This is the short C-program that smoothes the f0 contours by a Hamming window. It runs on ASCII-f0-files like the ones in (speaker-base)`vq/f0`. It consists of the chatr-function `f0_smooth` wrapped in a main program for outside-chatr usage. If you don't specify a window length 180ms are assumed (like in the ToBi contour smoothing).

Usage: `f0_smooth in-ASCII.f0 out-ascii.f0 [window-size-ms]`

`gregor/perl/pm_esps2chatr.pl` Takes an ESPS label file of pitch-marks and converts it to the CHATR format. Unvoiced sections are filled with equally spaced pitch-marks of 10ms distance. Unvoiced sections are either derived from an ESPS f0-file ($f0file=1) or from large "holes" between pitch-marks ($f0file=0). In the latter case an sd-file has to be supplied to get the end of file. All output files start at time 0; i.e. the ESPS start time is subtracted.

Usage: `pm_esps2chatr.pl in.pm in.f0 out.pm`, for $f0file=1

Usage: `pm_esps2chatr.pl in.pm in.sd out.pm`, for $f0file=0

`gregor/perl/xw_olay_f0.pl` Overlays several f0 contours in a xwaves window. All F_0-files have the same name but are in different directories. Specify the directories in the script.

Usage: `xw_olay_f0.pl f0-filename`

`gregor/perl/f0_ascii2esps.pl` converts ascii-f0 file to esps-f0 file (everything is considered as voiced)

Usage: `f0_ascii2esps.pl ascii-infile esps-outfile`

`gregor/perl/f0_esps2ascii.pl` Converts an esps-f0 file to an ascii-f0 file. Every output line contains only the F_0 value. Zeros are printed for unvoiced f0-values.

Usage: `f0_esps2ascii.pl esps-infile ascii-outfile`

`gregor/bin/pima_make` Shell script that calculates the pitch-marks of a speech file by using the ESPS *epochs* program. The *epochs* program is applied to the residual of the inverse filtered signal.

Usage: `make_pima_labels file.sd lab-file.pmlab`

References

[1] Hunt, Andrew. J.; Black, Alan W. *Unit Selection in a Concatenative Speech Synthesis System Using Large Databases* ICASSP-96, 1996

[2] Black, Alan W.; Campbell, Nick *Optimizing Selection of Units from Speech Databases for Concatenative Synthesis* EUROSPEECH-95. Madrid, 1995

[3] Phonetic AIMS volume 2. Word Stress. Working Papers of the Institute of Natural Language Processing, University of Stuttgart, 1995

[4] Moulines,E; Charpentier, F. "Pitch-synchronous waveform processing techniques for text-to-speech synthesis using diphones". *Speech Communication*, Vol. 9, 1990, pp.435-467

[5] Moulines, E; Verhelst, V. "Time-Domain and Frequency-Domain Techniques for Prosodic Modification of Speech" in *Speech Coding and Synthesis.* Edited by W.B.Kleijn and K.K.Paliwal. Elsevier Science B.V., 1995