TECHNICAL REPORT TR NoNumberYet

Building INTSINT Intonation models in the Festival Speech Synthesis System

March 2004

Deliverable Identification

Identification number	TR NoNumberYet
Type	System description
Title	Building INTSINT Intonation models in the Festival Speech Synthesis System
Status	Draft
Date	March 2004
Version	0.1
Number of pages	14
Author(s)	Aby Louw (jalouw@csir.co.za)
Project co-ordinator	
Access	Public
Key words	Intonation, Momel, INTSINT, Festival, Festvox, Speech Synthesis
Abstract	This document provides a description of the INTSINT Intonation module in the Festival speech synthesis system.
Actual Distribution	R. Tucker, K. Shalonova
Supplementary notes	None

Document evolution

Version	**Date**	**Status**	**Notes**
0.1	March 2004	Draft	Request for comments and additions.

Contents

1 Introduction

This document gives an overview of the Momel and INTSINT intonation modelling algorithms, and their use in building intonation models in the Festival Speech Synthesis system.

1.1 Overview of this Document

Section 2 gives a brief overview of the INTSINT and Momel algorithms. Section 3 describes the installation procedures for the INTSINT and Momel stand-alone programs as well as the accompanying Festival module and Festvox scripts. Finally section 4 discusses the procedures that need to be followed in order to build an INTSINT intonation module in a Festvox voice and how to use it.

2 Overview of INTSINT and Momel

Phonetic models use a set of continuous parameters to describe intonation patterns observable in an $F0$ contour [3]. An important goal is that the model should be capable of reconstructing $F0$ contours faithfully when appropriate parameters are given. However, to make it functional, a phonetic model should also be linguistically meaningful.

In fact, using certain functions, such as polynomial equations, to accurately represent F0 contour is not a difficult task. What is more challenging is developing a model whose parameters are predictable from available linguistic information. To be more specific, the mapping from various linguistic factors, which could affect intonation, to the model parameters, or vice versa, is more critical. Partly due to the availability of large corpora, more phonetic models, mainly for text-to-speech systems, have been proposed in recent years. Among the existing phonetic models, two lines of approaches are distinguished: parametric vs. non-parametric.

The model implemented in this work is a parametric model, namely INTSINT (**IN**ternational **T**ranscription **S**ystem for **INT**onation), proposed by Hirst et al [2].

2.1 INTSINT

INTSINT describes intonation with a limited set of abstract tonal symbols, which is designed such that the inventory of pitch patterns of a given language is not required. The input to the INTSINT system is a series of target points, which is estimated by a low-level $F0$ modeling technique called *MOMEL*.

The abstract symbols defined to represent the target points derived from the *MOMEL* procedure are: **T**, **M**, **B**, **H**, **S**, **L**, **U**, **D**, which stand for Top, Mid, Bottom, Higher, Same, Lower, Up-stepped, and Down-stepped respectively. Among these symbols, tones T, M, B are regarded as absolute tones, which refer to the speaker's overall pitch range. Tones H, S, L, U, D are relative with respect to the value of the preceding target point. The relative tones are further distinguished between non-iterative H, S, L and iterative U, D tones. An automatic coding scheme is used to relate the target points and the abstract symbols through a set of rules [2]. This system has been applied to a number of languages and shown to be quite robust.

INTSINT can be viewed as a hybrid phonetic/phonological model. It starts with a low-level phonetic analysis. Then a phonological description system is derived from the phonetic analysis results.

2.2 Momel

The MOMEL algorithm aims to analyze and synthesize $F0$ curves automatically. There are four basic stages:

1. preprocessing of $F0$
2. estimation of target-candidates
3. partition of candidates
4. reduction of candidates

Briefly, to estimate target candidates, first a moving window is applied to a segment of $F0$ values. Then, within each window, a quadratic regression is performed. The sequence of target candidates is partitioned by using another moving window in which the average values of two half windows are compared. The partition boundaries are determined where the values correspond to local minima and are greater than the overall average value. A final reduction procedure is taken to eliminate those outlying candidates by certain definitions.

3 Installation

This section describes how to install the Momel and INTSINT stand-alone and Festival software packages. These packages are:

- momel_tools_v1.0.tar.gz - The stand-alone Momel points calculation software
- intsint_tools_v1.0.tar.gz - The stand-alone INTSINT label calculation software
- INTSINT_Festival_v1.0.tar.gz - The Festival INTSINT module
- INTSINT_Festvox_v1.0.tar.gz - The Festvox INTSINT scripts

3.1 Requirements

In order to compile the INTSINT and Momel stand-alone tools you need the to have the *Edinburgh Speech Tools Library* installed on your machine.

The INTSINT and Momel Festival modules require a working version of Festival on your machine. This software was tested with Festival version 1.4.2, and Speech Tools version 1.2.3.

The Festvox voice building tools are also required for building the intonation models of a new voice. This software was tested with the version 2.0 release of Festvox.

The University of Macquarie's Speech Hearing and Language Research Centre distribute labelling tools for speech database is used for viewing the INTSINT labels. It is available from http://www.shlrc.mq.edu.au/emu/

3.2 Momel Stand-alone

Unpack momel_tools_v1.0.tar.gz into your home directory

```
cd
tar -zxvf momel_tools_v1.0.tar.gz
```

this will create the directory `/llsti/momel/` in your home, containing the source of the Momel stand-alone tools. Edit the `EST_HOME` variable in

```
~/llsti/momel/Makefile
```

to match that of your own Speech Tools location. Then make the executable

```
cd ~/llsti/momel
make
```

The source should compile without any warnings or errors. Finally add an environment variable named MOMELDIR with the location of the Momel stand-alone tools, e.g.

```
bash\$ export MOMELDIR=/home/aby/llsti/momel/

or

csh\% setenv MOMELDIR /home/aby/llsti/momel/
```

To see all the options available with this program run

```
~/llsti/momel/calc_momel -h
```

3.3 INTSINT Stand-alone

Unpack intsint_tools_v1.0.tar.gz into your home directory

```
cd
tar -zxvf intsint_tools_v1.0.tar.gz
```

this will create the directory `/llsti/intsint/` in your home, containing the source of the INTSINT stand-alone tools. Edit the `EST_HOME` variable in

```
~/llsti/intsint/Makefile
```

to match that of your own Speech Tools location. Then make the executable

```
cd ˜/llsti/intsint
make
```

The source should compile without any warnings or errors. Finally add an environment variable named INTSINTDIR with the location of the INTSINT stand-alone tools, e.g.

```
bash\$ export INTSINTDIR=/home/aby/llsti/intsint/

or

csh\% setenv INTSINTDIR /home/aby/llsti/intsint/
```

To see all the options available with this program run

```
˜/llsti/intsint/calc_intsint -h
```

3.4 INTSINT Festival module

Unpack INTSINT_Festival_v1.0.tar.gz into your Festival directory. This will install the following files in Festival:

```
./src/modules/INTSINT_Intonation/intsint_aux.cc
./src/modules/INTSINT_Intonation/intsint_database.cc
./src/modules/INTSINT_Intonation/intsint.h
./src/modules/INTSINT_Intonation/INTSINT_Intonation.cc
./src/modules/INTSINT_Intonation/make.depend
./src/modules/INTSINT_Intonation/Makefile
./src/modules/INTSINT_Intonation/make.include
./lib/intsint_intonation.scm
```

Because this is an additional Festival module, it must be added by adding the following to the end of your Festival config file in `festival/config/config/`, e.g.

```
ALSO_INCLUDE += INTSINT_Intonation
```

Now recompile Festival with this addition module

```
make
```

Again this should compile without any warnings or errors.

3.5 INTSINT Festvox tools

Unpack INTSINT_Festvox_v1.0.tar.gz into your Festvox directory. This will install the following files in Festvox:

```
./src/INTSINT_intonation/build_INTSINT_intonation.scm
./src/INTSINT_intonation/emu_intsint_syl.tpl
./src/INTSINT_intonation/emu_intsint.tpl
./src/INTSINT_intonation/intsint.feats
./src/INTSINT_intonation/make_dirs_intsint_intonation
./src/INTSINT_intonation/make_f0
./src/INTSINT_intonation/make_f0_ssff
./src/INTSINT_intonation/Makefile
./src/INTSINT_intonation/make_intsint
./src/INTSINT_intonation/make_intsint_intonation_model
./src/INTSINT_intonation/make_momel
./src/INTSINT_intonation/setup_intsint_intonation
./src/INTSINT_intonation/test_intonation.scm
./src/INTSINT_intonation/VOICE_intsint_f0.scm
./src/INTSINT_intonation/VOICE_intsint_intonation.scm
```

Since there is no actual source code, all the files are scripts, you don't need to recompile Festvox.

4 Building an INTSINT intonation module

To build a INTSINT intonation module a voice must first be built in the normal way as described in "Building Voices in Festival" [1].

After completing the normal voice building stage the INTSINT Festvox scripts must be installed into the current voice location. This is done with the following script

```
cd your_voice_location
\$FESTVOXDIR/src/INTSINT_intonation/setup_intsint_intonation
```

The above script will create the necessary directories and copy the needed scripts into your voice's location.

The first step will be to create $F0$ files if there aren't any present in the `./f0` directory. This is done with

```
./bin/make_f0 wav/*.wav
```

Next, Momel points must be calculated from the $F0$ files

```
./bin/make_momel f0/*.f0
```

This script makes the Momel point files in `festival/intsint/momel/` and Momel quadratic spline files in `festival/intsint/momel_qs`

From the Momel points INTSINT labels can be calculated. Note that these labels are not tied to any linguistic information, they are only optimised to reproduce the Momel quadratic splines.

```
./bin/make_intsint festival/intsint/momel/*.momel
```

The INTSINT labels for Momel points are made in `festival/intsint/intsint_lab/`

To view the "raw" INTSINT labels we first create a word labels to see where the INTSINT labels are in the waveform. This is done with as follows:

```
festival -b festvox/build_INTSINT_intonation.scm \
'(build_words_lab ‘‘etc/domain.txt’’)'
```

where "etc/domain.txt" is the file containing the voice prompts. This will create word label files in `festival/intsint/words/`. Now the INTSINT labels can be viewed with emulabel

```
emulabel etc/emu_intsint.tpl
```

Figure 1 shows an example of the output as viewed by emulabel. In the top window one can see the words, phonemes and INTSINT labels aligned to the waveform in the bottom window. The window second from the top is the Momel quadratic spline curve and the one second from the bottom is the $F0$ curve as calculated from the waveform. Note that the frequency scales on the Momel quadratic spline and $F0$ curves are not the same and can not be adjusted in the current version of emulabel.

If you are satisfied that the Momel points produced reliable INTSINT labels you can move on to building actual INTSINT labels that are aligned to some linguistic features. Currently the INTSINT Festival module calculates INTSINT labels for syllables and time aligns these labels to the middle of the vowel in a syllable. This may change in the future depending on the results of some further tests.

A few parameters which can be changed for the calculation and optimisation of the INTSINT labels are in the `./festvox/build_INTSINT_intonation.scm` file. These parameters are:

```
(set! INST_LANG_VOX::intsint_params
      '(
        (utts_dir "festival/utts/")
        (utts_ext ".utt")
        (momelqs_dir "/festival/intsint/momel_qs/")
        (momelqs_ext ".momelqs")
        (momel_dir "/festival/intsint/momel/")
```

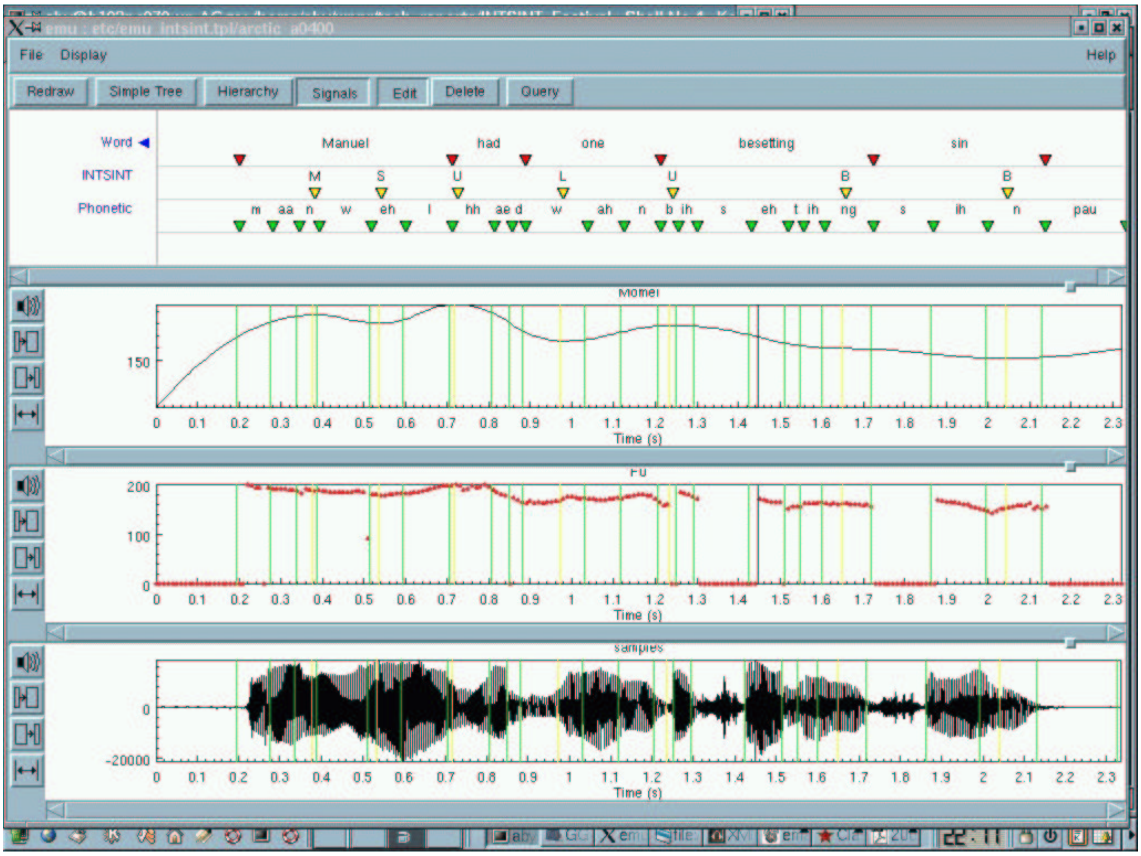

Figure 1: The raw INTSINT labels as calculated from the Momel points.

```
(momel_ext ".momel")
(intsint_dir "./festival/intsint/syl_intsint_lab/")
(intsint_ext ".intsint")
(intsint_utt_dir "festival/intsint/utts/")
(intsint_utt_ext ".utt")
(intsint_momelqs_dir "./festival/intsint/intsint_momel_qs/")
(intsint_momelqs_ext ".imomelqs")
(intsint_defs "etc/intsint.defs")
;; intsint specific stuff
(intsint_threshold 0.10)
(pause_length 0.20)
(intsint_min_range 0.5)      ;; octave f0
(intsint_max_range 2.5)      ;; octave f0
(intsint_step_range 0.1)     ;; octave f0
(intsint_mean_shift 70)      ;; linear f0
(intsint_step_shift 1)       ;; linear f0
(intsint_higher 0.5)         ;; octave f0
(intsint_lower 0.5)          ;; octave f0
(intsint_up 0.25)            ;; octave f0
(intsint_down 0.25)          ;; octave f0
(only_stressed_syls 1)))
```

The first few parameters are the locations and file extensions of the INTSINT labels and Momel quadratic spline curves that are to be calculated. These can be left as is, as the Festvox INTSINT setup creates the directories for these files. The following are specific to the INTSINT labels:

- `intsint_threshold` - This is the percentage that a given frequency can be from the top or bottom frequencies to be considered itself as a top or bottom label.
- `pause_length` - The minimum length of the preceding pause to the current label position for this label to be considered as “mid” label.
- `intsint_min_range` - The minimum search range (in octaves) from the mean database $F0$ to search for the optimum top, mid and bottom frequencies of the INTSINT labels.
- `intsint_max_range` - The maximum search range (in octaves) from the mean database $F0$ to search for the optimum top, mid and bottom frequencies of the INTSINT labels.
- `intsint_step_range` - The step value (in octaves) of the $F0$ search range.
- `intsint_mean_shift` - The shift from mean $F0$ to top $F0$, and mean $F0$ to bottom $F0$ (in Hertz).
- `intsint_step_shift` - The step value (in Hertz) of the $F0$ search range.
- `intsint_higher` - The factor used in estimating an “H” label : $last\_target + (top - last\_target) * higher$.
- `intsint_lower` - The factor used in estimating an “L” label : $last\_target - (last\_target - bottom) * lower$.
- `intsint_up` - The factor used in estimating an “U” label : $last\_target + (top - last\_target) * up$.
- `intsint_down` - The factor used in estimating a “D” label : $last\_target - (last\_target - bottom) * down$.
- `only_stressed_syls` - Whether to label only stressed syllables or to label all syllables. 1 is only label stressed syllables

The most important of these parameters is the `intsint_mean_shift`, for instance, if the mean $F0$ of the speech database is $150Hz$, the maximum $210Hz$ and the minimum $120Hz$, then set this parameter to $60Hz$, which is the maximum shift. With these parameters set one can calculate the stressed only/all syllables INTSINT labels with the following command

```
festival -b festvox/build_INTSINT_intonation.scm \
'(optimise_intsint_labels ‘‘etc/domain.txt’’)'
```

It does take a while since the INTSINT labels are optimised for *top, mid, bottom* values for the whole speech database. This operation will create INTSINT label files for the syllables in `./festival/intsint/syl_intsint_lab/`, utterance structures with the INTSINT labels as *IntEvent* and *Intonation* relations, Momel quadratic spline curves from the INTSINT syllable labels in `./festival/intsint/intsint_momel_qs/` and the optimised INTSINT *top, mid, bottom* values of the database in `./etc/intsint.defs`.

To view the INTSINT labels we need to create syllable label files for alignment with the INTSINT labels, as was done before with words. This is done by

```
festival -b festvox/build_INTSINT_intonation.scm \
'(build_syls_lab ‘‘etc/domain.txt’’)'
```

which creates the syllable labels in the `./festival/intsint/syllables/` directory. Now the INTSINT labels that are aligned with syllables can be viewed with emulabel

```
emulabel etc/emu_intsint_syl.tpl
```

Figure 2 shows an example of the output as viewed by emulabel. In the top window one can see the words, INTSINT labels and syllables aligned to the waveform in the bottom window. In this particular example the labels were aligned to stressed syllables only. The window second from the top is the Momel quadratic spline curve of the $F0$ curve in the second last window from the bottom. The third window from the top shows the quadratic spline curve as calculated from the INTSINT labels on the stressed syllables. Note again that the frequency scales on the two quadratic spline and $F0$ curves are not the same and can not be adjusted in the current version of emulabel.

Now a CART can be trained from these INTSINT labeled syllables. This is done with the following script:

```
./bin/make_intsint_intonation_model
```

this script extracts the features used for INTSINT intonation modelling from the INTSINT utterances in `./festival/intsint/utts/`, builds testing and training data sets, trains a CART with the EST tool *wagon* and dumps the tree in `./festvox/VOICENAME_intsint_tree.scm`.

The features used in the above operation are in `./festival/intsint/etc/intsint.feats` and are a starting point. These features might change in the future based on the outcome of some further tests.

The INTSINT intonation tree can now be used in a Festvox voice by manually adding a few lines to the `./festvox/INST_LANG_VOX_clunits.scm` or `./festvox/INST_LANG_VOX_ldom.scm` files depending on the type of voice.

Add the following two lines

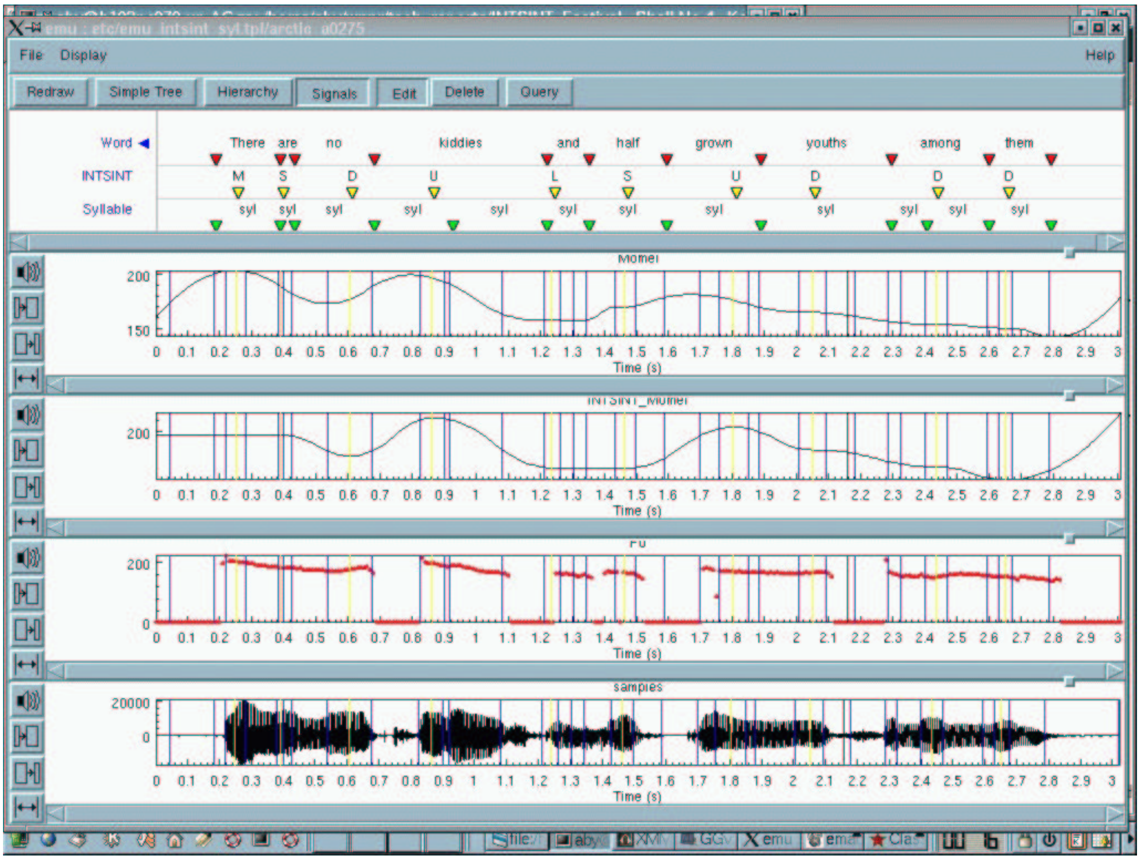

Figure 2: The INTSINT labels as calculated for stressed syllables.

```
(require 'INST_LANG_VOX_intsint_intonation)
(require 'INST_LANG_VOX_intsint_f0)
```

after the following

```
;;;  Add the directory contains general voice stuff to load-path
(set! load-path (cons (path-append INST_LANG_VOX::clunits_dir "festvox/")
                      load-path))
```

in one of the above mentioned files. This loads the INTSINT voice specific Scheme files into Festival. Now we need to set the intonation and f0 models, this is done by manually adding the following lines

```
(INST_LANG_VOX::select_intsint_intonation)
(INST_LANG_VOX::select_intsint_f0)
```

in the voice definition function, and commenting out the existing intonation and f0 models if they are defined. Now the voice will use INTSINT intonation during synthesis.

You can edit `./festvox/INST_LANG_VOX_intsint_intonation_f0.scm` to change the target voice's $F0$ range.

A scheme script `test_intonation.scm` in the Festvox INTSINT intonation directory can be used to test a trained INTSINT intonation CART with a diphone voice.

References

[1] A.W. Black and K.A. Lenzo. *Building Synthetic Voices*. Language Technologies Institute, Carnegie Mellon University, 1.0 edition, January 2003. Manual on the building of synthetic voices in the Festival Speech Synthesis System.

[2] D.J. Hirst, A. Di Cristo, and R. Espesser. Levels of representation and levels of analysis for intonation. In M. Horne, editor, *Prosody : Theory and Experiment Studies Presented*. Kluwer, Dordrecht, 2000.

[3] P. Taylor. Analysis and synthesis of intonation using the tilt model. *J. Acoust. Soc. Am.*, 107:1697–1714, 2000.