A Snack Implementation and Tcl/Tk Interface to the Fundamental Frequency Variation Spectrum Algorithm

Kornel Laskowski [1], **Jens Edlund** [2]

[1] Language Technologies Institute, Carnegie Mellon University, Pittsburgh PA, USA
[2] KTH Speech, Music and Hearing, Stockholm, Sweden
kornel@cs.cmu.edu, edlund@speech.kth.se

Abstract

Intonation is an important aspect of vocal production, used for a variety of communicative needs. Its modeling is therefore crucial in many speech understanding systems, particularly those requiring inference of speaker intent in real-time. However, the estimation of pitch, traditionally the first step in intonation modeling, is computationally inconvenient in such scenarios. This is because it is often, and most optimally, achieved only after speech segmentation and recognition. A consequence is that earlier speech processing components, in today's state-of-the-art systems, lack intonation awareness by fiat; it is not known to what extent this circumscribes their performance. In the current work, we present a freely available implementation of an alternative to pitch estimation, namely the computation of the fundamental frequency variation (FFV) spectrum, which can be easily employed at any level within a speech processing system. It is our hope that the implementation we describe aid in the understanding of this novel acoustic feature space, and that it facilitate its inclusion, as desired, in the front-end routines of speech recognition, dialog act recognition, and speaker recognition systems.

1. Introduction

Intonation is an important aspect of vocal production, used for a variety of communicative needs. Its estimation and modeling is therefore crucial to speech classification and understanding systems. This is particularly true of those systems operating in real-time, in which inference of speaker intent should be achieved with low latency. At the present time, the overwhelming majority of such systems reconstruct intonation contours from frame-level estimates of *pitch* (F_0), computed by a pitch-tracking component; these estimates are then normalized to eliminate absolute, speaker-dependent values.

Due to its sensitivity to noise and voicing occlusions, pitch is computed and modeled only *after* speech is segmented and, often, also recognized. The resulting system flow makes intonation contours unavailable to early processing components, where they are likely to be at least as useful as in downstream processing. It is therefore of some import that an instantaneous, frame-level characterization of intonation in speech, available as early as speech detection itself, be developed. Although pitch is visually appealing, the long-observation-time requirements for accurate pitch estimation do not recommend it as a basis for frame-level modeling.

The fundamental frequency variation (FFV) spectrum, designed to address these concerns (Laskowski et al., 2008a), offers a computationally tractable alternative to characterizing intonation which obviates the need to first estimate absolute pitch, and then to normalize it out its average. It is based on a simple observation, namely that the rate of change in F_0, across two temporally adjacent frames of speech produced by the same speaker, can be inferred by finding the dilation factor required to optimally align the harmonic spacing in their magnitude frequency spectra. This can be achieved without knowledge of the frequency scale (provided it is the same in both frames). Unfortunately, FFV processing entails a significant deviation from traditional, pitch-centered conceptualizations of intonation, presenting a steep learning curve for researchers who may wish to use it in their work.

This paper addresses the latter problem by presenting our current implementation of the FFV algorithm (briefly described in Section 3.) within a free, publicly available and commonly used speech processing toolkit. The implementation exposes, via a Tcl/Tk interface, the critical parameters driving the FFV spectrum computation. We describe these parameters, some of their effects, and their defaults in Section 4.; Section 5. discusses several possibilities for modeling the spectrum. In Section 6., we present a graphical comparison between the spectrum and the output of a popular pitch tracker (Talkin, 1995), for singing voice. Before concluding, we enumerate in Section 7. several applications in which FFV processing has been shown to be useful. We expect that this evidence, collectively, may facilitate the inclusion of FFV feature computation in general speech analysis software, such as WaveSurfer (Sjölander and Beskow, 2000) and Praat (van Heuven, 2001), as well as in the front-ends of dedicated speech understanding systems.

2. The Snack Sound Toolkit

The Snack Sound Toolkit (Sjölander, 2001) is developed and maintained by Kåre Sjölander. It is designed to be used with a scripting language, and currently has bindings for Tcl/Tk (Ousterhout, 2008), Python (van Rossum, 2008) and Ruby (Matsumoto, 2008). It ships with the standard ActiveState© distributions of Tcl and Python. Snack allows users to create multi-platform audio applications with just a few lines of code, with commands for basic sound handling and primitives for waveform and spectrogram visualization. Snack supports WAV, AU, AIFF, MP3, CSL, SD, SMP, and NIST/Sphere file formats. The toolkit is designed to be extensible, and new commands, filters, and sound file formats can be added using the Snack C library. The freely available Open Source tool WaveSurfer (Sjölander and Beskow, 2000) provides a graphical user interface and visualizations for the functionality in Snack; it also can be extended with new custom plug-ins, and be embedded in other applications.

3. The FFV Spectrum

The fundamental frequency variation spectrum is a recently introduced representation (Laskowski et al., 2008a) which captures instantaneous, per-frame variation in fundamental frequency. The algorithm relies on the comparison of the frequency magnitude spectra $\mathbf{F}_L$ and $\mathbf{F}_R$ of the left and right halves of an analysis frame, respectively. The comparison is implemented as a dot product following frequency dilation by a factor ρ of one of $\mathbf{F}_L$ or $\mathbf{F}_R$. The dot product, expressed as $g(\rho)$, yields a continuous spectrum when computed over a range of ρ; an example is shown in Figure 1.

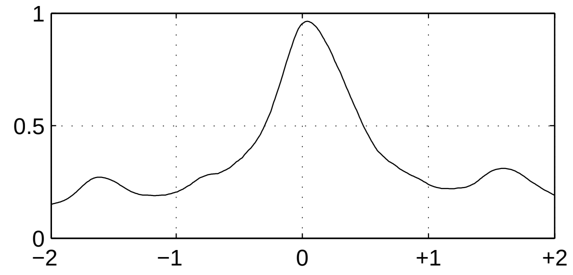

Figure 1: The FFV spectrum for a randomly chosen voiced speech frame; the magnitude of $g(\rho)$, shown along the y-axis, is a function of the dilation factor ρ shown along the x-axis in octaves per 0.008 s (the temporal separation between the maxima of the two window functions used in computing $\mathbf{F}_L$ and $\mathbf{F}_R$).

FFV processing offers several advantages over other representations of variation in fundamental frequency; most notably, it is a *local* estimate, *independent* of the absolute fundamental frequency, and it does not require dynamic programming as employed in most pitch tracking applications. This makes the representation directly amenable to hidden Markov modeling (HMMing). Examples of successful usage in this way (cf. Section 7.) include speaker change prediction in dialogue systems, speaker recognition, and dialog act classification in meeting data.

4. The Snack FFV Interface

4.1. Synopsis

Given the path name `$file` of a file containing a snippet of audio, the following Tcl code frames and prints out the FFV spectrum for each frame:

```
snack::sound s
s read $file
foreach line [s ffv
        -tFra t_fra -tSep t_sep
        -tInt t_int -tExt t_ext
        -winShapeInt <string-literal>
        -winShapeExt <string-literal>
        -Ng N_g -tSepRef t_sep^ref
        -filterbank <string-literal>
] {
        puts $line
}
```

The positive real-valued parameters `tFra`, `tSep`, `tInt`, `tExt`, `tSepRef`, the positive integer-valued parameter `Ng`, and the string-valued parameters `winShapeInt`, `winShapeExt`, and `filterbank` in the above command are described individually in the following subsections.

4.2. Framing Parameters

As mentioned in Section 3., the FFV algorithm relies on an estimate of the frequency magnitude spectra of the left and right halves of each analysis frame. These spectra are computed using two asymmetrical analysis windows, placed symmetrically about the frame's center, as shown in Figure 2. Parameters governing window shape can be modified from their default values using arguments to the Snack `ffv` function:

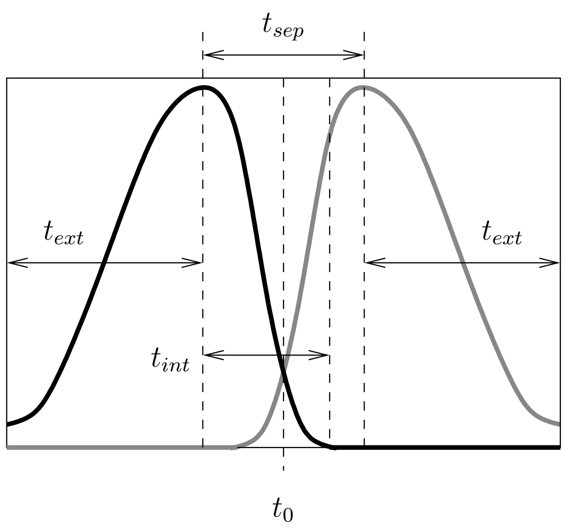

Figure 2: Relative placement of the left and right window functions, h_L and h_R, in a single analysis frame of duration $2t_{ext} + t_{sep}$ centered on instant t_0. The parameters t_{int}, t_{ext}, and t_{sep}, all in seconds, are as described in the text. Successive frames are placed such that their respective instants t_0 are t_{fra} seconds apart, not shown.

`-tFra` The frame step between successive analysis frames, in seconds; the default value is 0.008.

`-tSep` The temporal separation between the maxima of the left and right window functions in seconds; the default value is 0.014.

-tInt The temporal extent of the left and right window functions towards the center of the analysis frame, in seconds; the default value is 0.011.

-tExt The temporal extent of the left and right window functions away from the center of the analysis frame, in seconds; the default value is 0.009.

-winShapeInt The shape of the left and right window functions from their maximum towards the center of the analysis frame; currently implemented alternatives include `Hamming` and `Hann`. The default value is `Hann`.

-winShapeExt The shape of the left and right window functions from their maximum away from the center of the analysis frame; currently implemented alternatives include `Hamming` and `Hann`. The default value is `Hamming`.

4.3. Discretization Parameters

FFV spectra such as that shown in Figure 1 are continuous; their discretization requires a specification of a sampling frequency and range of interest. This is achieved by modifying the parameters `Ng` and `tSepRef`:

-Ng An even integer-valued parameter governing the number of values of $g(\rho)$ which are to be computed, at equi-spaced intervals $\Delta\rho$ (described below). $N_g/2$ is the number of $g(\rho)$ values computed for $\rho < 0$ and the number of $g(\rho)$ values computed for $\rho > 0$; $g(0)$ is always computed, for a total number of $N_g + 1$ values of $g(\rho)$. The default value of N_g is 512.

-tSepRef A positive real-valued parameter, expressed in seconds, governing the separation $\Delta\rho$ between successive values of ρ at which $g(\rho)$ is sampled,

$$\Delta\rho = \frac{4}{N_g}\frac{t_{sep}}{t_{sep}^{ref}} \quad . \qquad (1)$$

The default value is 0.008.

4.4. Filterbank Specification

As with other signal representations in speech processing, estimates of the FFV spectrum can be passed through a smoothing filterbank to improve the robustness of downstream modeling. Snack's default `ffv` filterbank can be replaced by user-specified variants as desired, using the `filterbank` parameter.

-filterBank Allows specification of a filterbank. The currently implemented alternatives include `NONE`, `DEFAULT`, and *fileName*, where *fileName* specifies the path name of a file containing a filterbank structure description. The default value is `DEFAULT`, corresponding to a filterbank of `Nf` $= 7$ filters defined as follows (the default filters are indexed by the integers $\{-3, -2, -1, 0, +1, +2, +3\}$ rather than $\{1, \ldots, \texttt{Nf}\}$ for convenience):

$$f_{-3}[i] = \begin{cases} 1, & \text{if } 117 \le i \le 139 \\ 0, & \text{otherwise} \end{cases} \qquad (2)$$

$$f_{-2}[i] = \begin{cases} 1, & \text{if } 246 \le i \le 250 \\ 1/2 & \text{if } i = 245 \text{ or } i = 251 \\ 0, & \text{otherwise} \end{cases} \qquad (3)$$

$$f_{-1}[i] = \begin{cases} 1, & \text{if } 250 \le i \le 254 \\ 1/2 & \text{if } i = 249 \text{ or } i = 255 \\ 0, & \text{otherwise} \end{cases} \qquad (4)$$

$$f_{0}[i] = \begin{cases} 1, & \text{if } 255 \le i \le 257 \\ 1/2 & \text{if } i = 254 \text{ or } i = 258 \\ 0, & \text{otherwise} \end{cases} \qquad (5)$$

$$f_{+1}[i] = \begin{cases} 1, & \text{if } 258 \le i \le 262 \\ 1/2 & \text{if } i = 257 \text{ or } i = 263 \\ 0, & \text{otherwise} \end{cases} \qquad (6)$$

$$f_{+2}[i] = \begin{cases} 1, & \text{if } 262 \le i \le 266 \\ 1/2 & \text{if } i = 261 \text{ or } i = 267 \\ 0, & \text{otherwise} \end{cases} \qquad (7)$$

$$f_{+3}[i] = \begin{cases} 1, & \text{if } 373 \le i \le 395 \\ 0, & \text{otherwise} \end{cases} \qquad (8)$$

Source domain filter limits must lie in $[0, N_g]$. A `filterbank` value of `NONE` will output all $N_g + 1$ values of $g(\rho)$.

4.5. Error Handling

Invalid parameter values, as well as errors due to a missing or a mis-formatted filterbank specification file (when such is specified), are posted as exceptions by returning the `TCL_ERROR` error code to the Tcl interpreter.

5. Modeling Alternatives

We list several possible ways in which the information available in the FFV representation may be modeled.

5.1. Peak Localization and Tracking

The value of $g(\rho)$ at ρ is the cosine distance between the magnitude frequency spectra $\mathbf{F}_L$ and $\mathbf{F}_R$ in each analysis frame, following frequency dilation by ρ. When $\rho < 0$, it is the left-half spectrum $\mathbf{F}_L$ which is frequency-dilated by $2^{+\rho}$; when $\rho > 0$ is positive, the right spectrum $\mathbf{F}_R$ is frequency-dilated by $2^{-\rho}$. The cosine distance is computed over a sub-range of the frequencies spanned by the original $\mathbf{F}_L$ and $\mathbf{F}_R$ because, after dilation, $\mathbf{F}_L$ and $\mathbf{F}_R$ differ in domain extent.

This interpretation of $g(\rho)$ recommends an obvious application. For voiced frames in which only one person is vocalizing, we can localize the peak in the spectrum,

$$\rho^* = \arg\max_{\rho} g(\rho) \quad . \qquad (9)$$

If desired, the peak can be tracked from frame to frame, using a dynamic programming approach such as that found in most pitch detection algorithms.

5.2. Density Modeling and Estimation

In contrast to localizing and optionally tracking the supremum of $g(\rho)$, we can model the entire FFV spectrum, with or without a filterbank. Mathematically, this is similar to modeling the magnitude frequency spectrum or Mel filterbank filter responses, instead of formant center frequencies (which correspond to ρ^* in this analogy). The filterbank described in Section 4.4. reduces the dimensionality of the feature space, and improves robustness by averaging $g(\rho)$ over perceptually similar rates of change in fundamental frequency. However, the outputs of the default filterbank are correlated, and mixture modeling may benefit from decorrelation prior to model estimation. This has been shown to be the case in all applications in which FFV spectra have been employed (cf. Section 7.). Decorrelation, if desired, needs to be performed externally to Snack's `ffv` function.

5.3. FFV Modulation Spectrum

Finally, a method that is promising but has not been previously explored is the computation of the modulation spectrum over the FFV representation. The magnitude modulation spectrum characterizes the frequencies with which specific rates of change in fundamental frequency appear. For example, it may reveal that increases in pitch of 1 octave per second (computed for a particular analysis frame size) are observed once per second (see Section 6.2.).

6. An Example: Singing Voice

We now present several graphical examples of the Snack `ffv` output, and a comparison with the output of the Snack `ESPS` pitch tracker (Talkin, 1995). We do so for singing voice (Lindblom and Sundberg, 2005), for three reasons. First, the application of FFV processing to singing voice has not been previously explored, and the current work provides a convenient opportunity to do so. Second, more importantly, singing involves long stretches uninterrupted by an absence of voicing, allowing us to ignore what happens at the onset and offset of voicing. This is important when comparing pitch tracker to `ffv` output because the two methods behave differently in these environments; in particular, the output of a pitch tracker is numerically undefined in unvoiced regions. Finally, demonstration of singing voice is easy to grasp, especially visually, and transcends potentially language- and culture-dependent definitions of contours which are considered prosodically meaningful.

6.1. Glissando

We begin with an example of glissando (or portamento), in which the voice "slides" across a range of pitches. Our glissando recording was made by a professional female vocalist in a home environment on a laptop; it was downsampled from 44.1 kHz to 16 kHz prior to processing. The output of the Snack `ESPS` pitch tracker, invoked using the Tcl command

```
s pitch -method ESPS
        -minpitch 60
        -maxpitch 1000
        -framelength 0.008
        -windowlength 0.0075
```

is shown in Figure 3(a). To the right of panel 3(a), in panel 3(b), we show the slope $\dot{p}[t]$ of the pitch curve $p[t]$, in octaves per second,

$$\dot{p}[t] \quad = \quad \frac{1}{t_{sep}} \log_2 \frac{p\left[t + {}^{t_{sep}}/_{2t_{fra}}\right]}{p\left[t - {}^{t_{sep}}/_{2t_{fra}}\right]} \; . \tag{10}$$

As can be seen in the Snack `pitch` syntax above, t_{fra}, the frame step, is 0.008 s; we have chosen to estimate $\dot{p}[t]$ in Equation 10 over intervals of $t_{sep} = 0.112$ s. (Division by t_{sep} in Equation 10 yields rates in octaves per second rather than in octaves per t_{sep} seconds).
The graphical result in Figure 3(b) can be directly compared to `ffv` output. In Snack, we invoke

```
s ffv -tFra 0.008 -tSep 0.112
        -tInt 0.088 -tExt 0.072
        -filterbank NONE
```

using the same parameter values for `tFra` (t_{fra}) and `tSep` (t_{sep}) as are used in Equation 10 (namely, $t_{fra} = 0.008$ s and $t_{sep} = 0.112$ s; the values for `tInt` and `tExt` are their default values scaled by a factor of 8, which is the ratio of $t_{sep} = 0.112$ s to its default value of $t_{sep} = 0.014$ s). This leads to $N_g + 1 = 513$ values of $g(\rho)$ per frame; we display a contour plot of only the 23 values for each frame, centered on $\rho = 0$, in Figure 3(c). As can be seen, the agreement with Figure 3(b) is very high; differences in y-value are due to the fact that the FFV algorithm computes variation in F_0 using windows of $t_{int} + t_{ext} = 0.160$ s without dynamic programming, rather than over windows of 0.0075 ms with dynamic programming.

6.2. A Scale

Our second example is an 8-note scale, from the same vocalist. We show the corresponding diagrams in Figure 4. All Snack commands used the same syntax as described earlier. As can be seen in panel (b) of the figure, the slope $\dot{p}[t]$ of the `ESPS` pitch trajectory indicates seven instants of fast change, corresponding to inter-note transition. The maxima in Figure 4(c) follow an almost identical trajectory, as in our example of glissando in the previous section.
We also show, in Figure 4(d), the modulation spectrum of the FFV spectrum, computed using a Hamming window over 256 values of $g(\rho)$ at fixed ρ, for all values of ρ shown in panel (c). The modulation spectrum indicates how frequently specific rates ρ of F_0 variation (along the y-axis) appear, expressed in Hertz along the x-axis. As the darkest region in panel (d) suggests, rates of approximately $+1$ octave per second (along the y-axis), corresponding to note transitions, appear at a rate of just over 1 Hz (along the x-axis). The remaining modulation frequencies, for positive rates of F_0 change, are harmonics of this 1 Hz frequency.

6.3. A Scale with Vibrato

Finally, we show a similar scale sung by the same vocalist, this time with vibrato, in Figure 5. This is an effect of small-scale F_0 variation, superimposed on the underlying note. As for our previous two examples, F_0 variation computed from pitch tracker output, in Figure 5(b), exhibits the same features as that expressed directly by the `ffv` output (Figure 5(c)).

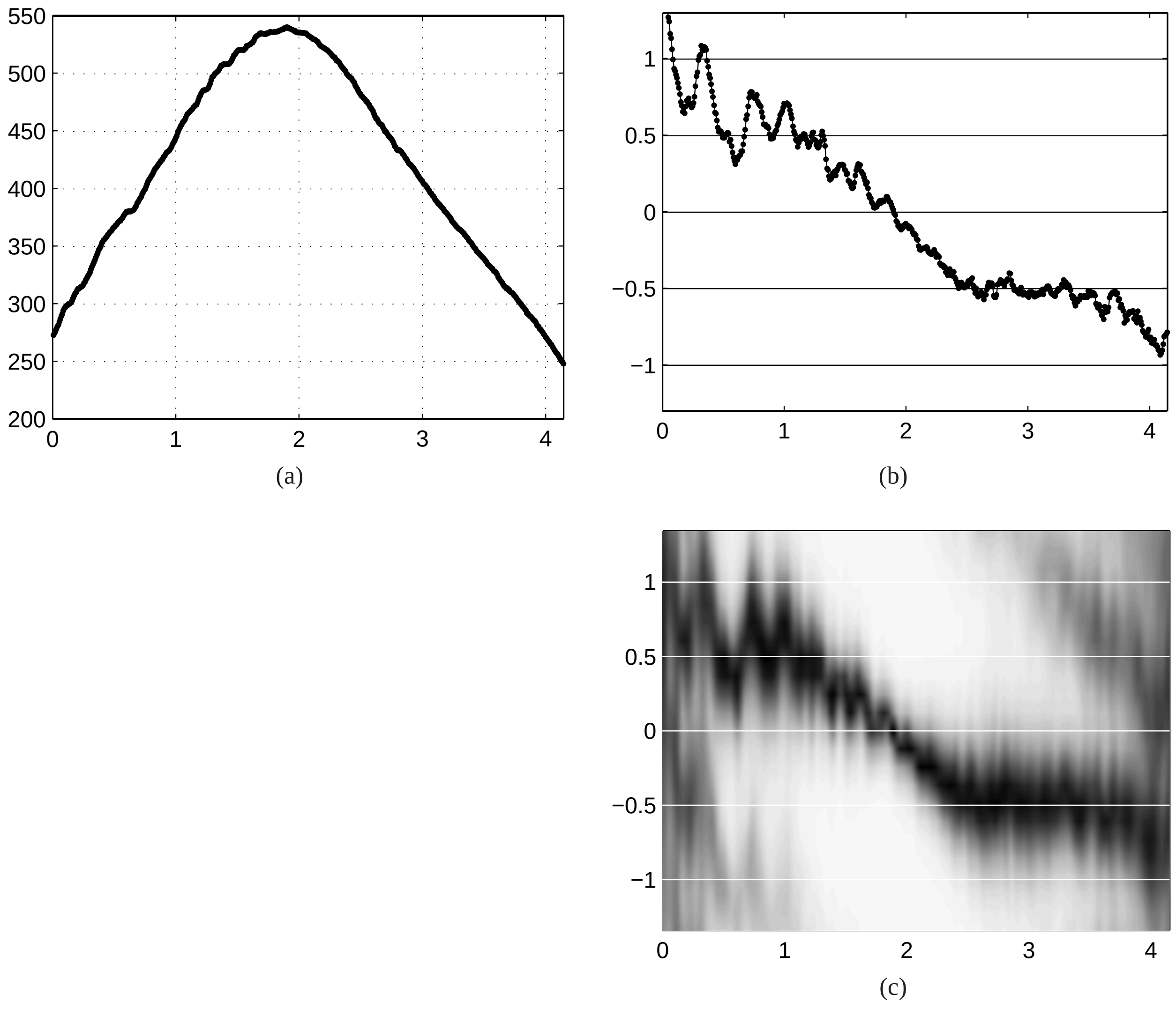

Figure 3: Glissando. Clockwise from the upper left: (a) $p\left[t\right]$, the output of a pitch tracker, in absolute frequency along the y-axis, as a function of time in seconds along the x-axis. (b) $\dot{p}\left[t\right]$, computed from $p\left[t\right]$ using Equation 10, in octaves per second along the y-axis, as a function of time in seconds along the x-axis. (c) Bird's eye view of consecutive FFV spectra $g_t\left(\rho\right)$, with time t in seconds along the x-axis, and rate of F_0 variation in octaves per second along the y-axis; darker shades of gray indicate larger magnitudes of $g_t\left(\rho\right)$.

The modulation spectrum in Figure 5(d) indicates that rates of F_0 change in the range $(+0.5, +1.0)$ octaves per second appear at a frequency of just over 1 Hz, as observed also in Figure 4(d). However, in addition, there is a dark patch at ρ values of approximately ± 0.5 octaves per second, which appears at the much higher modulation frequency of 3.5 Hz. This up and down change, symmetric about $\rho = 0$ and distinct from the mostly positive rates of change at approximately 1 Hz, is due to vibrato.

7. Applications in Automatic Speech Processing

FFV processing has been applied to a number of tasks in speech understanding, and in a limited few cases a comparison has been made with information available from a standard pitch trajectory. It is important to note, however, that the computational flexibility afforded by adopting FFV features cannot be duplicated when using their pitch-trajectory-derived counterparts. This section enumerates several applications in which FFV processing has been shown to be beneficial and/or enabling in that regard.

7.1. Text-Independent Speaker Change Prediction

Speaker change prediction is the task of deciding whether a currently vocalizing speaker is signaling the intent to continue, following an incipient pause. It is an important functionality in dialogue systems which aim to exhibit human-like response times, avoiding long hold-over waits before initiating a turn (Edlund and Heldner, 2005). From a system design perspective, real-time speaker change prediction would ideally be incorporated into speech activity detection; this calls for frame-level acoustic features characterizing instantaneous intonation sub-phrases.

The FFV representation was designed primarily with this task in mind, and was tested (Laskowski et al., 2008a) on the Swedish Map Task Corpus (Helgason, 2006). Graph-

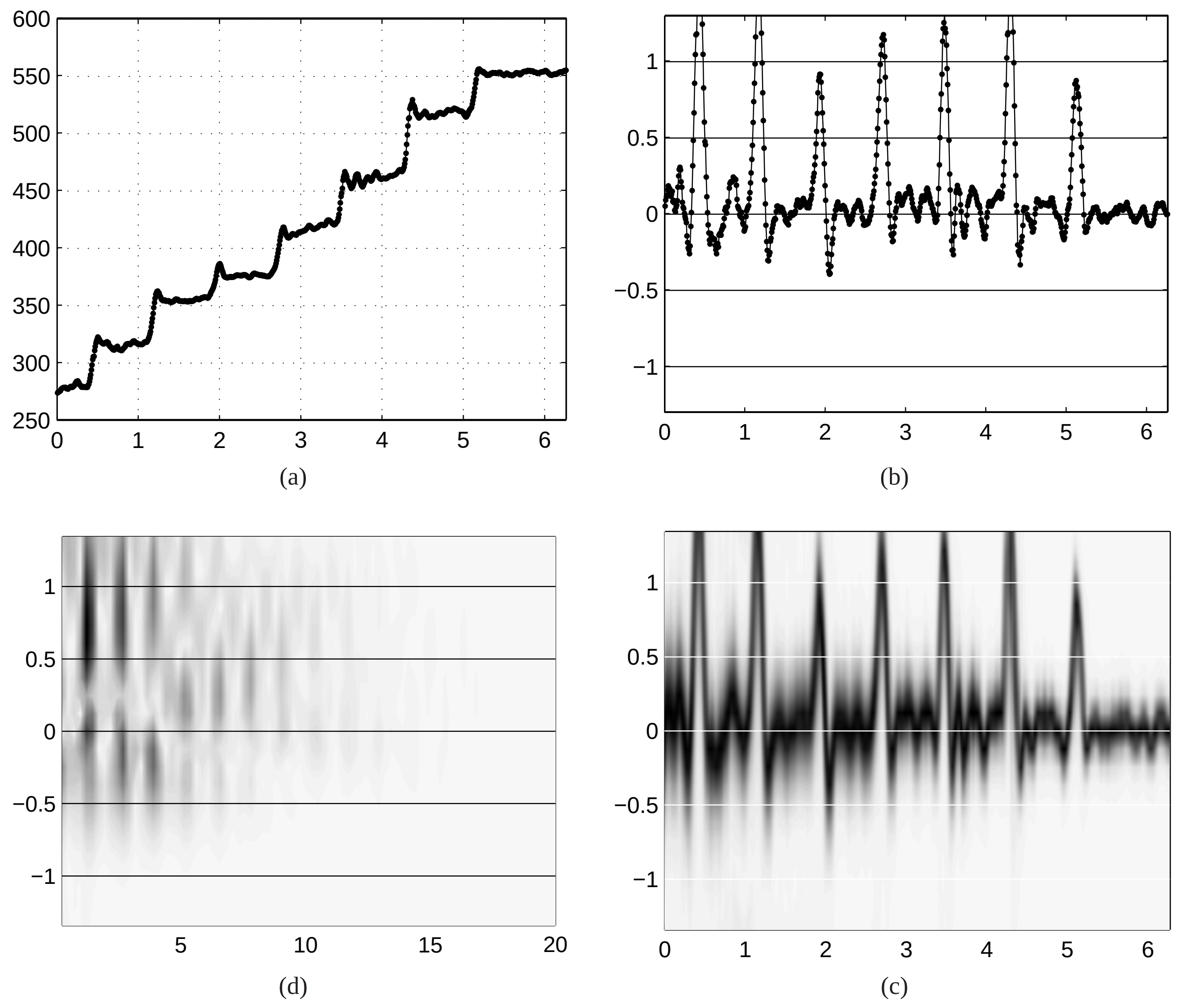

Figure 4: A scale. Clockwise from the upper left: (a) $p\left[t\right]$, the output of a pitch tracker; (b) $\dot{p}\left[t\right]$, change in pitch computed from $p\left[t\right]$; (c) consecutive FFV spectra $g_t\left(\rho\right)$; axes as in Figure 3. (d) Modulation spectrum of $g_t\left(\rho\right)$ for specific values of rates ρ of F_0 variation, shown along the y-axis as for panel (c); modulation frequency shown along x-axis in Hertz; darker shades of gray indicate larger magnitudes.

ical depiction of what HMMs learn for this task, namely that speakers employ predominantly flat intonation contours to signal a desire to continue speaking, was shown in (Laskowski et al., 2008b). In (Heldner et al., 2008), it was additionally shown that intonation contours prior to speaker change differ as a function of speaker role; instruction *givers* employ more falls than rises, while *followers* use the opposite strategy. Finally, (Laskowski et al., 2008c) considered several refinements to the FFV representation for the speaker change prediction task.

7.2. Text-Independent Dialogue Act Recognition

The FFV representation has also been applied in multiparty conversational settings (Laskowski et al., 2009a), initially to detect floor mechanism dialog acts in the ICSI Meeting Corpus (Janin et al., 2003). We reported that *floor holders* and *holds*, signaling floor reservation (and therefore quite similar to the phenomena in Section 7.1.), can be separated from all other dialog act types with a discrimination of 70.6 to 72.2%. FFV model analysis revealed that talkspurts implementing these dialog acts are initiated with flat intonation contours, and terminated with slower speech. Several modifications, and augmentation with other prosodic features, are presented in (Laskowski et al., 2009b).

Most recently, the acoustic prosodic vector defined in (Laskowski et al., 2009b) has been implemented in a full-scale text-independent HMM dialog act decoder (Laskowski and Shriberg, 2010), which segments and classifies audio into 8 types of dialog acts, with 3 types of dialog act termination. The decoder operates without reliance on word boundaries, typically used for inferring models of intonation, and is therefore deployable in privacy-sensitive settings in which spectral envelopes may not be computed. For several dialog act types and termination types, its performance approaches that of a contrastive lexical system which uses human-transcribed words.

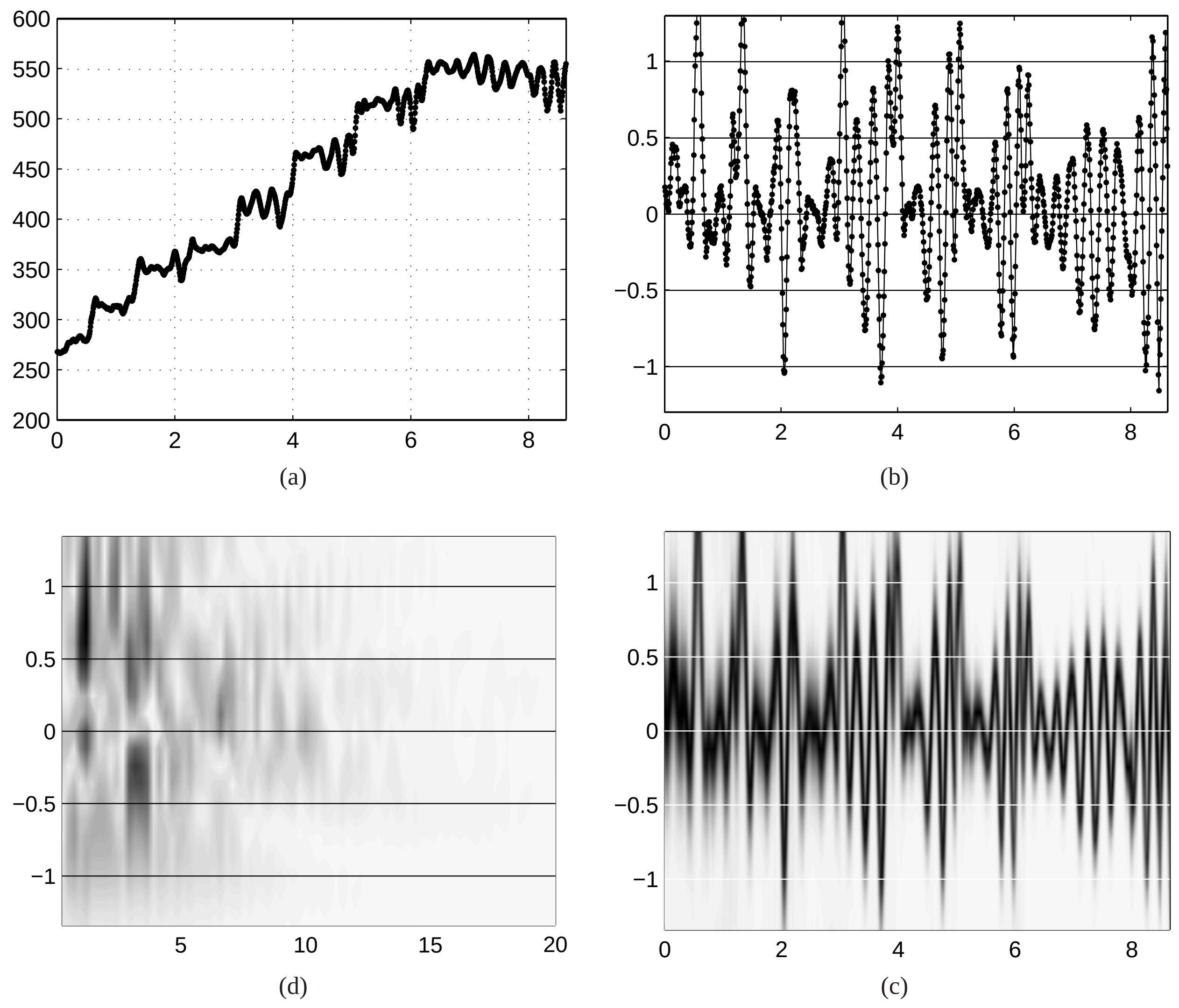

Figure 5: A scale with vibrato. Clockwise from the upper left: (a) $p\,[t]$, the output of a pitch tracker; (b) $\dot{p}\,[t]$, change in pitch computed from $p\,[t]$; (c) consecutive FFV spectra $g_t\,(\rho)$; (d) modulation spectrum of $g_t\,(\rho)$; axes as in Figure 4.

7.3. Pitch Detection

An approach which appears conceptually identical to FFV processing and which may be computationally quite similar was proposed for the purposes of improving pitch estimation in (Martin, 2008). Harmonic similarity between adjacent spectra was shown to benefit pitch tracking in several examples of utterances with low signal-to-noise ratios.

7.4. Text-Independent Speaker Recognition

Finally, the FFV representation has also been shown to aid in the task of speaker recognition, in both nearfield (Laskowski and Jin, 2009) and farfield (Jin et al., 2010) scenarios. The temporal locality of the FFV features makes possible the construction of single-state Gaussian mixture model systems, such as those used with standard Mel-frequency cepstral coefficient systems. Because these systems do not model trajectories, the observed improvements in baseline speaker recognition system performance suggest that speakers differ in their preferences of rate of F_0 change, in addition to differences in absolute F_0 (which FFV features, and therefore the systems in (Laskowski and Jin, 2009; Jin et al., 2010), do not model).

8. Conclusions

We have implemented and presented an interface to the computation of a novel representation of macro- and micro-intonation, the fundamental frequency variation (FFV) spectrum, within the popular and publicly available Snack Sound Toolkit. The interface exposes the majority of parameters governing FFV behavior, and our description makes it possible for speech researchers, practitioners and voice pathologists to explore its potential use in their work without needing to understand and to re-implement the signal processing internals.

This work has also compared the FFV representation to the slope of the F_0 trajectory, as estimated via a frequently used pitch-tracking method. For relatively clean signals of extended intervals of continuous voicing, the representation maximum appears to contain the same information as do such slopes, over a wide range of magnitudes of rate of F_0 change. This fidelity is achieved without reliance on dynamic programming, as employed in most pitch trackers, making FFV suitable for deployment in HMM decoding scenarios, such as speech recognition for tonal languages.

9. Acknowledgments

We would like to thank Marietta Fischesser and Florian Metze for contributing their voices and recording expertise, as well as Timo Baumann who implemented the first stage of the FFV port to Snack.

10. References

J. Edlund and M. Heldner. 2005. Exploring prosody in interaction control. *Phonetica*, 62:215–226.

M. Heldner, J. Edlund, K. Laskowski, and A. Pelcé. 2008. Prosodic features in the vicinity of silences and overlaps. In *Proc. Nordic Prosody*, pages 95–105, Helsinki, Finland.

P. Helgason. 2006. SMTC — A Swedish Map Task Corpus. In *Proc. Fonetik*, pages 57–60, Lund, Sweden.

A. Janin, D. Baron, J. Edwards, D. Ellis, D. Gelbard, N. Morgan, B. Peskin, T. Pfau, E. Shriberg, A. Stolcke, and C. Wooters. 2003. The ICSI Meeting Corpus. In *Proc. ICASSP*, pages 364–367, Hong Kong, China. IEEE.

Q. Jin, R. Li, Q. Yang, K. Laskowski, and T. Schultz. 2010. Speaker identification with distant microphone speech. In *Proc. ICASSP*, pages 4518–4521, Dallas TX, USA. IEEE.

K. Laskowski and Q. Jin. 2009. Modeling instantaneous intonation for speaker identification using the fundamental frequency variation spectrum. In *Proc. ICASSP*, pages 4541–4544, Taipei, Taiwan. IEEE.

K. Laskowski and E. Shriberg. 2010. Comparing the contributions of context and prosody in text-independent dialog act recognition. In *Proc. INTERSPEECH*, pages 5374–5377, Dallas TX, USA. IEEE.

K. Laskowski, J. Edlund, and M. Heldner. 2008a. An instantaneous vector representation of delta pitch for speaker-change prediction in conversational dialogue systems. In *Proc. ICASSP*, pages 5041–5044, Las Vegas NV, USA. IEEE.

K. Laskowski, J. Edlund, and M. Heldner. 2008b. Learning prosodic sequences using the fundamental frequency variation spectrum. In *Proc. Speech Prosody*, pages 151–154, Campinas, Brazil. ISCA.

K. Laskowski, M. Wölfel, M. Helnder, and J. Edlund. 2008c. Computing the fundamental frequency variation spectrum in conversational spoken dialogue systems. In *Proc. Acoustics*, pages 3305–3310, Paris, France. ASA.

K. Laskowski, M. Heldner, and J. Edlund. 2009a. Exploring the prosody of floor mechanisms in English using the fundamental frequency variation spectrum. In *Proc. EUSIPCO*, pages 2539–2543, Glasgow, UK. EURASIP.

K. Laskowski, M. Heldner, and J. Edlund. 2009b. A general-purpose 32 ms prosodic vector for hidden Markov modeling. In *Proc. INTERSPEECH*, pages 724–727, Brighton, UK. ISCA.

B. Lindblom and J. Sundberg. 2005. *The Human Voice in Speech and Singing*. Springer.

P. Martin. 2008. A fundamental frequency estimator by crosscorrelation of adjacent spectra. In *Proc. Speech Prosody*, pages 147–150, Campinas, Brazil. ISCA.

Y. Matsumoto. 2008. Ruby Programming Language 1.8.7. `http://www.ruby-lang.org/en/`. (last accessed 24 March 2010).

J. Ousterhout. 2008. Tcl Programming Language and Tk Graphical User Interface Toolkit 8.4. `http://www.tcl.tk`. (last accessed 24 March 2010).

K. Sjölander and J. Beskow. 2000. WaveSurfer — An open source speech tool. In *Proc. ICSLP*, pages 464–467, Beijing, China. ISCA.

K. Sjölander. 2001. Snack Sound Toolkit 2.2.10. `http://www.speech.kth.se/snack/`. (last accessed 24 Marchi 2010).

D. Talkin, 1995. *Speech Coding and Synthesis (Kleijn & Paliwal, eds.)*, chapter A Robust Algorithm for Pitch Tracking (RAPT), pages 495–518. Elsevier.

V. van Heuven. 2001. Praat, A system for doing phonetics by computer. *Glot International*, 5(9–10):341–345.

G. van Rossum. 2008. Python Programming Language 3.0. `http://www.python.org`. (last accessed 24 March 2010).