Development of exploratory research tools based on TANDEM-STRAIGHT

Hideki Kawahara*, Toru Takahashi†, Masanori Morise‡ and Hideki Banno§
* Faculty of Systems Engineering, Wakayama University, Wakayama 640-8510 Wakayama Japan
E-mail: kawahara@sys.wakayama-u.ac.jp Tel: +81-73-457-8461
† Graduate School of Informatics, Kyoto University, Kyoto 606-8501 Kyoto Japan
E-mail: tall@kuis.kyoto-u.ac.jp Tel: +81-75-753-5992
‡ College of Information Science and Engineering, Ritsumeikan University, Kusatsu 525-8577 Shiga Japan
E-mail: morise@fc.retsumei.ac.jp Tel: +81-77-561-5075
§ Faculty of Science and Technology, Meijo University, Nogoya 468-8502 Aichi Japan
E-mail: banno@ccmfs.meijo-u.ac.jp Tel: +81-52-838-2088

***Abstract*—This article introduces a new set of tools based on TANDEM-STRAIGHT, a fundamental reformulation of STRAIGHT, a speech analysis, modification and resynthesis system introduced in 1997. STRAIGHT has been used in a wide range of speech-related research as a flexible tool for implementing experiments and applications though its scientific foundation was not well established. TANDEM-STRAIGHT introduced a solid basis to resolve this difficulty while preserving the underlying concept of STRAIGHT. TANDEM is a procedure for estimating a temporally static power spectral representation of periodic signals. Together with this representation, the consistent sampling theory enabled complete reformulation of entire algorithms of STRAIGHT and led to a new implementation. This new implementation was applied to develop a set of new tools: temporally variable multi-aspect speech morphing tools, a graphical user interface (GUI) for manipulating STRAIGHT and morphing parameters, and procedures for preparing stimuli for perceptual experiments.**

I. Introduction

This article introduces a set of new speech research tools based on a speech analysis, modification and resynthesis system, TANDEM-STRAIGHT [1], with historical and conceptual background. TANDEM-STRAIGHT is a completely new formulation of its predecessor STRAIGHT [2], [3], [4] (legacy-STRAIGHT [1]) while preserving its underlying principles [3], [4].

STRAIGHT is essentially a contemporary version of a channel VOCODER [5]. It decomposes an input speech signal into two components, source information and spectral (filter) information. The source information consists of fundamental frequency (F0) information and an aperiodicity spectrogram that represents the ratio between the random and periodic components in each frequency band. Virtually complete decomposition attained in STRAIGHT enabled flexible manipulation of speech parameters without introducing severe degradations, which were inevitable in previous implementations of a so-called source-filter model.

Partially supported by Grants-in-Aid for Scientific Research (A) 19200017 by JSPS and the CrestMuse project by JST.

[1] "STRAIGHT" is used to represent both legacy-STRAIGHT and TANDEM-STRAIGHT in this article. Only when necessary, prefixes "TANDEM" or "legacy" are used.

Conceptual simplicity and flexibility in manipulations without introducing severe degradations made STRAIGHT a convenient substrate for implementing various applications and tools. Auditory morphing [6] is the most influential tool among the others. It makes it possible to generate stimulus continuum based on two speech examples with typical manifestation of perceptual attributes, for example. Auditory morphing was extended to a temporally variable multi-aspect procedure [7] based on new TANDEM-STRAIGHT implementation.

In this latest implementation, a set of GUIs was also introduced to reduce complexity involved in morphing manipulations. The latest morphing procedure requires each of five aspects to be specified as a single or a multidimensional time series. These aspects are temporal axis mapping, frequency axis mapping, time-frequency indexed spectrographic level mapping, time-frequency indexed aperiodicity mapping and F0 mapping. Definition of mapping consists of the assignment of anchoring points on each time-frequency representation. The total number of anchoring points in the case of two-mora Japanese words is typically around 30. In addition to this anchoring information, about 60 or more numbers need to be specified to define the amount of morphing, multi-aspect temporally variable morphing rates.

The following sections begin with background motivations behind STRAIGHT. A brief history of development and an introduction to technical backgrounds are then followed by descriptions of GUIs and step-by-step introduction to manipulations using GUIs.[2] Finally, representative examples of exploratory research and future plans are discussed.

II. Background

Development of STRAIGHT has been intended to provide relevant tools for speech perception research. In other words, it was motivated by frustrations over conventional representations of speech sounds. Usual spectrographic representation suffers from periodicity in speech sounds. However, this runs counter to our everyday experiences in speech communication. Periodic sounds such as vowels play important roles in speech

[2] The body of descriptions is placed in the appendix for readers' convenience.

communication. They are perceived as smooth, stable and rich. These aspects were not well represented by conventional methods. STRAIGHT was designed to provide a representation that replicates this naive observation by removing interferences due to periodicity from spectrographic representation while preserving spectral details of periodic sounds [1], [3], [4].

The other motivation came from ecological points of view in auditory perception, represented by a research framework called "Auditory Scene Analysis"[8]. This ecological view leads to the guideline that it is crucial to use test signals that are relevant for a system to be tested when the system consists of nonlinear components inside. That is the case in doing research on speech perception. Relevant test signals for speech perception are consequently natural speech. However, this strongly restricts research strategies. STRAIGHT was designed to solve this problem by providing a tool to reproduce perceptually natural speech signals from decomposed representations that enable precise quantitative manipulations. It is important to note that such reproduction must not rely on waveform reconstruction because completely different waveforms can be perceived as identical, for example two sample fragments of white Gaussian noise.

A. Brief history of revisions

Development of STRAIGHT to date consists of various refinements and a few fundamental inventions. Legacy-STRAIGHT was based on a complementary set of time windows for obtaining temporally stable spectral representations and F0 adaptive frequency smoothing that is equivalent to piecewise linear interpolation [2]. In the next refinement, its F0 adaptive spectral smoothing was updated by introducing an inverse filtering on the spatial frequency domain based on a theory on spline functions and, at the same time, an instantaneous frequency-based F0 extraction method was introduced [3]. The subsequent refinement was made based on fixed points of frequency to instantaneous frequency mapping [9]. It refined F0 extraction and aperiodicity representation. The latest refinement of the legacy-STRAIGHT algorithm was a "nearly defect free" (NDF) F0 extractor [10]. It still provides the best performance among F0 extractors available for STRAIGHT. The morphing procedure introduced in 2003 was revised in 2005 based on this refinement. These refinements were initially implemented using Matlab [11], a high-level language and interactive environment for scientific research. Libraries and a real-time application [12] were implemented using C language, based on these Matlab implementations. The latest legacy-STRAIGHT implementation in C language was conducted by the MEXT leading project e-Society [13] and was made available [14]. Many applications have been developed based on this legacy-STRAIGHT [15], [16], [17].

The most important revision is TANDEM-STRAIGHT. TANDEM-STRAIGHT is based on two inventions, TANDEM and a new spectral envelope recovery method. TANDEM [18] is a simple method for yielding a temporally static power spectral representation of periodic signals by averaging two power spectra calculated at temporal positions $0.5/F0$ apart. The spectral envelope recovery method is based on a new formulation of sampling theory called "consistent sampling" [19], which only requires recovery at resampled points of D/A results. This revision made the STRAIGHT procedure compact and transparent and led to an advanced morphing procedure that enables temporally variable multi-aspect morphing without introducing objective and perceptual breakdown. This extension of morphing procedure introduced excessive complexity in designing morphing materials and led to the development of GUIs.

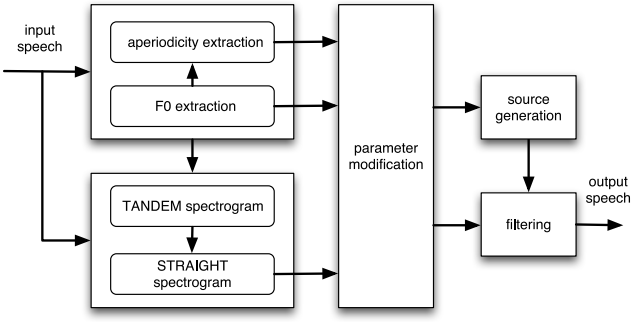


Fig. 1. Schematic diagram of TANDEM-STRAIGHT. Manipulation block modifies source parameters (F0 and aperiodicity) and STRAIGHT spectrogram as well as their coordinates (time and frequency axes).

III. Basic algorithms

This section introduces basic algorithms used in TANDEM-STRAIGHT and the extended morphing procedures. Please refer to original articles [1], [7], [18] for details.

A. TANDEM-STRAIGHT

Figure 1 shows a schematic diagram of TANDEM-STRAIGHT. TANDEM-STRAIGHT consists of two essential algorithms: the power spectral calculation algorithm TANDEM and the spectral envelope calculation algorithm. They are represented in the bottom left box of the diagram.

TANDEM is a procedure for extracting temporally static power spectra, represented by the following equations [18], [1]:

$$\begin{aligned} S(\omega,t) &= \int x(\tau)w(\tau-t)e^{-j\omega\tau}d\tau \\ P_T(\omega,t) &= \frac{|S(\omega,t-T_0/4)|^2+|S(\omega,t+T_0/4)|^2}{2}, \end{aligned} \quad (1)$$

where $P_T(\omega,t)$ represents the TANDEM spectrogram. In these equations, $x(t)$ and $w(t)$ represent the waveform under study and the time windowing function. The interval T_0 represents the reciprocal of fundamental frequency f_0. Temporally varying components in $|S(\omega,t)|^2$ are virtually cancelled by (1).

Spectral envelope estimation consists of two stages [1]: spectral smoothing and compensation for consistent sampling:

$$\begin{aligned} C(\omega,t) &= \int_{\omega_L}^{\omega} P_T(\lambda,t)d\lambda \\ L(\omega,t) &= \ln\left(C(\omega+\omega_0/2,t)-C(\omega-\omega_0/2,t)\right) \\ P_{ST}(\omega,t) &= \exp\left(\tilde{q}_1(L(\omega+\omega_0,t)+L(\omega-\omega_0,t))\right. \\ &\quad \left.+\tilde{q}_0 L(\omega,t)\right), \end{aligned} \quad (2)$$

where $P_{ST}(\omega,t)$ represents the STRAIGHT spectrogram and $\omega_0 = 2\pi F_0$ represents fundamental angular frequency. The initial two lines are an implementation of spectral smoothing

using a rectangular smoothing kernel of width f_0. The last equation approximately applies the consistent sampling concept by introducing compensation constants $\tilde{q}_0$ and $\tilde{q}_1$, which are calculated from autocorrelation of the Fourier transform of the time windowing function. This implementation assures positivity of the resultant STRAIGHT spectrogram.

1) F0 extraction: These procedures rely on F0 estimation. Several F0 extractors are implemented in the current release, and it provides optional use of other publicly available extractors such as YIN [20], SWIPE [21], and others [22], [23].

The default F0 extractor [1] of TANDEM-STRAIGHT is based on spectral division using two power spectral representations. Dividing a power spectrum consisting of periodicity information by its envelope leaves a power spectrum with periodicity information only and yields a periodicity detector specialized to its assumed F0. The default F0 extractor consists of a set of these specialized detectors setting assumed F0 candidates equidistant on the log-frequency axis covering normal F0 range.

2) Aperiodicity extraction: Aperiodicity is represented as a set of power ratios between periodic component and random component in each frequency band. It provides information to design a mixed-mode excitation source signal for resynthesis. It is estimated band-wise linear prediction using forward and backward segments one pitch period apart. Detailed descriptions of aperiodicity estimation will be submitted elsewhere.

B. Temporally variable multi-aspect morphing

Extending the morphing rate to vary temporally requires a reference time axis on which temporally variable morphing rates are defined. Let subscripts A and B represent two examples to be morphed and m represent the reference. Using this notation, let $T_{Am}(x_A)$ represent a morphing transformation of a parameter x_A of example A to parameter x_m on the morphing axis m. A temporally variable morphing rate for the parameter $r_{AB}(t)$ is defined to have the value 0 when the morphed result is equivalent to example A and to have the value 1 when the morphed result is equivalent to example B. A new morphing definition is introduced and described using this notation.

To alleviate breakdown in explorative morphing, morphing is redefined based on a logarithm of the derivative of mapping functions. This new definition of morphing also makes the morphing procedure simpler as follows:

$$\begin{aligned} T_{Am}(x_A) &= \int_0^{x_A} \exp\left(\log\left(\frac{dT_{Am}(\lambda)}{d\lambda}\right)\right) d\lambda \\ &= \int_0^{x_A} \exp\Bigg((1-r_{AB}(\lambda))\log\left(\frac{dT_{AA}(\lambda)}{d\lambda}\right) \\ &\quad + r_{AB}(\lambda)\log\left(\frac{dT_{AB}(\lambda)}{d\lambda}\right)\Bigg) d\lambda \\ &= \int_0^{x_A} \left(\frac{dT_{AB}(\lambda)}{d\lambda}\right)^{r_{AB}(\lambda)} d\lambda \ , \end{aligned} \tag{3}$$

because logarithmic conversion of the identity mapping vanishes. This formulation assures monotonicity of T_{Am} if the coordinate conversion T_{AB} from speaker A to B is monotonic.

Two morphing algorithms are formulated based on this new definition of morphing: real-time morphing and off-line morphing. In the case of real-time morphing, the morphing rates are incrementally supplied and used to update morphed parameters incrementally. This formulation is useful for interactive applications such as v.morish, a real-time singing manipulation interface [24]. In the case of off-line morphing, the morphed time axis, which is also the time axis for the morphed signal, is calculated for the first time. Then, other morphed parameters are calculated using the morphing rate on this new reference axis. This formulation is necessary for implementing an editing system for post production. Figure 2 illustrates the synthesis procedure using these parameters and transformations.

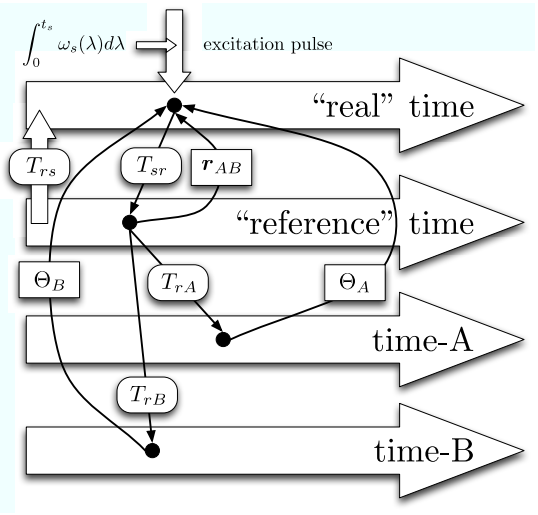


Fig. 2. Morphing procedure with a "reference" time axis for defining temporally variable morphing rates. Θ_A and Θ_B represent grouped STRAIGHT parameters of examples A and B respectively. $\boldsymbol{r}_{AB}$ represents multi-aspect morphing rates. $\omega_s(t)$ represents morphed F0 represented by angular instantaneous frequency.

IV. Design considerations

When using morphing in exploratory research, design of transformation functions and assignment of multi-aspect temporally variable morphing rates themselves are crucial steps of the research. Researchers have to have full control of design and awareness of what is assigned. These requirements exclude use of automatic procedures. However, these preparations are time-consuming and prone to error. A set of tools and GUIs is developed to solve these problems and to provide means to interactively explore physical correlates of perceptual attributes of speech. GUIs are also prepared for STRAIGHT parameter manipulations.

In designing GUIs and tools, the following difficulties found in using legacy-STRAIGHT with scripting and developing test programs were taken into account.

- Parameter conditions used to synthesize a stimulus and the generated sound file were usually separately handled and resulted in confusion and loss of records.
- Scripts and programs were sometimes updated without keeping records of older versions.
- Scripting and programming usually take so long that researchers forget their original insights.

- It is not always easy to find a relevant algorithm for the desired tests. It is also not always easy to implement the algorithm using Matlab and APIs of legacy-STRAIGHT.

Tools and GUIs were designed to remove these difficulties because scripting and programming themselves are not users' research targets.

V. Examples of operations

Step-by-step examples for using GUIs are given in the appendix to illustrate how these considerations are implemented in GUIs and tools. There are two sets of GUIs. The first set is for STRAIGHT analysis, modification and resynthesis. The second set is for morphing and refers to the first set as sub-functions.

VI. Applications and other tools

Interactive tools introduced in this article were found very useful for quickly grasping perceptual phenomena qualitatively. However, those observations have to be tested using a systematic experimental procedure. Combining GUI tools and Matlab programming is a useful strategy for preparing stimuli for such experiments.

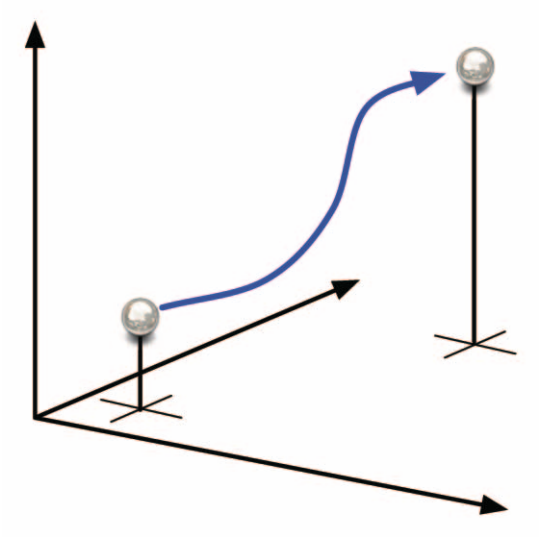
Using a file consisting of a variable "`STRAIGHTobject`" and manipulation data file saved using the GUI for STRAIGHT parameter manipulation, a loop that incrementally changes interpolation rates generates a stimulus continuum connecting simple resynthesis to manipulated speech. This strategy is applicable to investigating the contribution of each STRAIGHT parameter on perceptual attributes.

Two morphing rate definition files are prepared by the GUI for morphing rate manipulation using the same "`mSubstrates`" data, and a similar incremental loop generates a stimulus continuum connecting those two morphed sounds. This is a useful GUI application because defining two set of relevant morphing rate for a specific test requires much trial-and-error that involves a large number of parameters. Morphing-based strategies using legacy-STRAIGHT were already applied by other researchers and reported [25], [26].

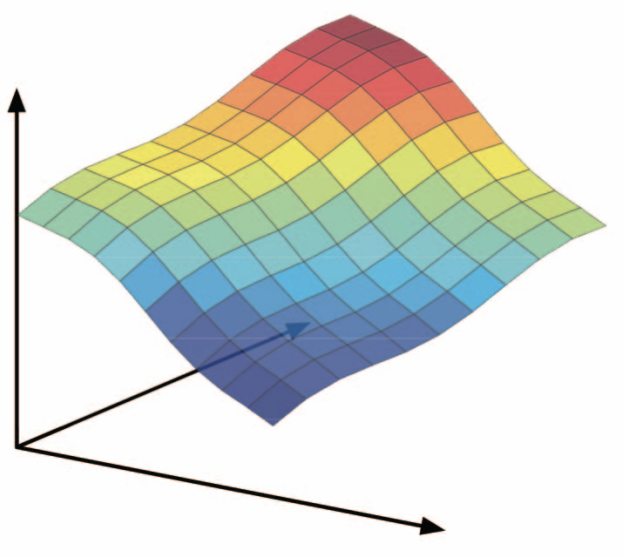
This strategy is generalized to parameterized mapping. Figure 3 illustrates schematic diagrams of such generalization. The upper panel of Fig. 3 shows how to generalize linear interpolation to nonlinear interpolation by using a parameterized mapping function. The lower panel also shows generalization to multi-dimensional nonlinear mapping using constraints of a set of attributes. A linearized version using constraints of a set of attributes was applied to develop user interfaces for singing manipulations [27], [28].

TANDEM-STRAIGHT and temporally variable multi-aspect morphing procedures, as well as their GUI tools, are implemented using Matlab. In writing Matlab code, variables and fields of structured variables are named to be understandable by using long and meaningful words. First priority in writing codes is placed on readability and is not on sophisticated tricks for speed and memory efficiency. GUI codes are also designed to provide programming examples to illustrate how to use APIs for basic constituent functions of TANDEM-STRAIGHT.

In addition to Matlab-based implementation, there are several projects and a product for implementing TANDEM-STRAIGHT algorithms in C-language [29], [30]. Also, stan-



(a) Generalized parameterized path



(b) Generalized manipulation surface

Fig. 3. Schematic illustration of extended stimuli preparation using a parameterized connecting path between examples and manipulation surface for composite control parameters.

dalone applications of these GUI-based tools are prepared using Matlab Compiler.

VII. Conclusions

This article introduced GUI-based tools for TANDEM-STRAIGHT parameter manipulations and temporally variable multi-aspect morphing. They are designed to promote exploratory interactive research in speech perception. Availability and conditions of the tools introduced in this article can be found at the following web address.

`http://www.wakayama-u.ac.jp/~kawahara/STRAIGHTadv/`

Acknowledgment

The authors would like to express their sincere appreciation to all researchers who used STRAIGHT in their research and published their scientific contributions and to all supporting

companies, institutes and organizations. Their feedback and requests were indispensable for development of STRAIGHT and tools based on it.

REFERENCES

[1] H. Kawahara, M. Morise, T. Takahashi, R. Nisimura, T. Irino, and H. Banno, "TANDEM-STRAIGHT: A temporally stable power spectral representation for periodic signals and applications to interference-free spectrum, f0, and aperiodicity estimation," in *Proc. ICASSP 2008*, 2008, pp. 3933–3936.

[2] H. Kawahara, "Speech representation and transformation using adaptive interpolation of weighted spectrum: vocoder revisited," in *Proc. ICASSP 1997*, vol. 2, 1997, pp. 1303–1306.

[3] H. Kawahara, I. Masuda-Katsuse, and A. de Cheveigné, "Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F0 extraction," *Speech Communication*, vol. 27, no. 3-4, pp. 187–207, 1999.

[4] H. Kawahara, "STRAIGHT, exploration of the other aspect of vocoder: Perceptually isomorphic decomposition of speech sounds," *Acoustic Science & Technology*, vol. 27, no. 5, pp. 349–353, 2006.

[5] H. Dudley, "Remaking speech," *J. Acoust. Soc. Am.*, vol. 11, no. 2, pp. 169–177, 1939.

[6] H. Kawahara and H. Matsui, "Auditory morphing based on an elastic perceptual distance metric in an interference-free time-frequency representation," in *Proc. 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2003)*, vol. I, Hong Kong, 2003, pp. 256–259.

[7] H. Kawahara, R. Nisimura, T. Irino, M. Morise, T. Takahashi, and H. Banno, "Temporally variable multi-aspect auditory morphing enabling extrapolation without objective and perceptual breakdown," in *Proc. ICASSP 2009*, 2009, pp. 3905–3908.

[8] A. S. Bregman, *Auditory Scene Analysis*. Cambridge, MA: MIT Press, 1990.

[9] H. Kawahara, H. Katayose, A. de Cheveigné, and R. D. Patterson, "Fixed point analysis of frequency to instantaneous frequency mapping for accurate estimation of F0 and periodicity," in *Proc. Eurospeech'99*, vol. 6, 1999, pp. 2781–2784.

[10] H. Kawahara, A. de Cheveigné, H. Banno, T. Takahashi, and T. Irino, "Nearly defect-free f0 trajectory extraction for expressive speech modifications based on STRAIGHT," in *Interspeech'05*, Lisboa, 2005, pp. 537–540.

[11] Mathworks. Matlab - The language of technical computing. [Online]. Available: http://www.mathworks.com/products/matlab/

[12] H. Banno, H. Hata, M. Morise, T. Takahashi, T. Irino, and H. Kawahara, "Implementatioin of realtime STRAIGHT speech manipulation system: Report on its first implementation," *Acoustic Science and Technology*, vol. 28, no. 3, pp. 140–146, 2007.

[13] K. Shikano, T. Kawahara, H. Saruwatari, K. Takeda, H. Kawahara, K. Tokuda, T. Nishiura, and A. Lee, "Development of advanced development of fundamental software through tight collaboration of academia and industry: Human friendly speech interface," *IPSJ Journal*, vol. 49, no. 11, pp. 1297–1301, 2008, [in Japanese].

[14] STRAIGHT-suite: a high-quality speech analysis, modification and synthesis system. [in Japanese]. [Online]. Available: http://straight-suite.sys.wakayama-u.ac.jp/

[15] H. Zen, T. Toda, M. Nakamura, and K. Tokuda, "Details of the nitech HMM-based speech synthesis system for the Blizzard Challenge 2005," *IEICE Trans., Information and Systems*, vol. E90-D, no. 1, pp. 325–333, 2007.

[16] T. Saitou, M. Goto, M. Unoki, and M. Akagi, "Speech-to-singing synthesis: Converting speaking voices to singing voices by controlling acoustic features unique to singing voices," in *Proc. WASPAA IEEE workshop*, New Paltz, NY, USA, Oct. 2007, pp. 215–218.

[17] T. Toda, A. W. Black, and K. Tokuda, "Voice conversion based on maximum likelihood estimation of spectral parameter trajectory," *IEEE Trans. Audio, Speech and Language Proc.*, vol. 15, no. 8, pp. 2222–2235, 2007.

[18] M. Morise, T. Takahashi, H. Kawahara, and T. Irino, "Power spectrum estimation method for periodic signals virtually irrespective to time window position," *Trans. IEICE*, vol. J90-D, no. 12, pp. 3265–3267, 2007, [in Japanese].

[19] M. Unser, "Sampling – 50 years after Shannon," *Proceedings of the IEEE*, vol. 88, no. 4, pp. 569–587, 2000.

[20] A. de Cheveigné and H. Kawahara, "YIN, a fundamental frequency estimator for speech and music," *J. Acoust. Soc. Am.*, vol. 111, no. 4, pp. 1917–1930, 2002.

[21] A. Camacho and J. G. Harris, "A sawtooth waveform inspired pitch estimator for speech and music," *J. Acoust. Soc. Am.*, vol. 124, no. 3, pp. 1638–1652, 2008.

[22] B. Yegnanarayana, C. d'Alessandro, and V. Darsinos, "An iterative algorithm for decomposition of speech signals into periodic and aperiodic components," *IEEE Trans. on Speech and Audio Processing*, vol. 6, no. 1, pp. 1–11, Jan. 1998.

[23] M. Morise, H. Kawahara, and H. Katayose, "Fast and reliable f0 estimation methos based on the period extraction of vocal fold vibration of singing voice and speech," in *Proc. AES 35th International Conference Audio for Games*, London UK, 2009.

[24] H. Kawahara, M. Morise, T. Takahashi, H. Banno, R. Nisimura, and T. Irino, "Singing morphing extension to temporally varying parameters for realtime morphing control interface," *IPSJ SIG Technical Report*, vol. 2008-MUS-78, no. 127, pp. 91–96, 2008, [in Japanese].

[25] T. Yonezawa, N. Suzuki, S. Abe, K. Mase, and K. Kogure, "Perceptual continuity and naturalness of expressive strength in singing voices based on speech morphing," *URASIP Journal on Audio, Speech, and Music Processing*, vol. 2007, p. 9, 2008, [doi:10.1155/2007/23807].

[26] S. R. Schweinberger, C. Casper, N. Hauthal, J. M. Kaufmann, H. Kawahara, N. Kloth, and D. M. Robertson, "Auditory adaptation in voice perception," *Current Biology*, vol. 18, pp. 684–688, 2008.

[27] H. Kawahara, T. Ikoma, M. Morise, T. Takahashi, K. Toyoda, and H. Katayose, "Proposal on a morphing-based singing design manipulation interface and its preliminary study," *IPSJ Transaction*, vol. 48, no. 12, pp. 3637–3648, 2007, [in Japanese].

[28] H. Kawahara, M. Morise, and H. Katayose, "Implementation of realtime voice morphing system using STRAIGHT," *IPSJ SIG Technical Report*, vol. 2008-MUS-75, no. 50, pp. 117–122, 2008, [in Japanese].

[29] H. Banno, S. Yoshida, S. Takahashi, F. Itakura, M. Morise, T. Takahashi, and H. Kawahara, "Development of applications of high-quality speech analysis and synthesis system by using the STRAIGHT library," in *Proc. ASJ annual spring conference*, no. 1-R-26, Tokyo, 2009, pp. 441–442, [in Japanese].

[30] "VoicebaseII," Software, Animo Limited. [Online]. Available: http://www.animo.co.jp/EN/

APPENDIX

This appendix introduces GUIs and their operations using examples. They are implemented only using Matlab m-language for avoiding platform dependency. They were tested on Windows, Linux and Mac OS X and were found functional. The following examples were prepared using Mac OS X, which is the primary environment for development of TANDEM-STRAIGHT and related tools.

A. First set: TANDEM-STRAIGHT

Typing "`TandemSTRAIGHThandler`" in the Matlab command window invokes a menu interface for TANDEM-STRAIGHT. At the very beginning, it opens a dialogue for speech file selection. Once selection is made an interface shown in the left panel of Fig. 4 appears. Only relevant buttons are made accessible depending on the state of processing.

1) F0 extraction and editing: By clicking "`F0/F0 structure extraction`" the F0 extraction GUI appears. Figure 5 shows its appearance just after the completion of F0 extraction. Several F0 extractors can be selected by using a pull-down menu user interface (UI) element on the top right of the GUI. The pull-down menu consists of the default TANDEM-STRAIGHT F0 extractor (called XSX, which stands for "eXcitation Structure eXtractor"), zero-frequency-filtering-based F0 extractor, legacy-STRAIGHT's NDF and the default F0 extractor. YIN and SWIPE may also appear on the pop-up item, if they are installed in the system and accessible from Matlab. The GUI searches for them in the initialization phase of the GUI and adds them to the menu item list. The Matlab code also consists of a prototype interface code for user-defined F0 extractors.

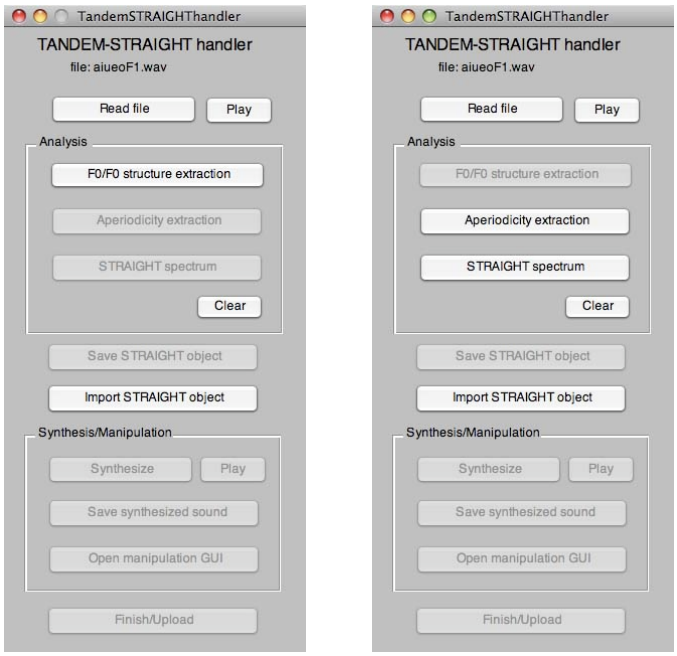


Fig. 4. Left: Initial state of the STRAIGHT handling menu. Right: State after F0 extraction. Note that only buttons that invoke relevant functions are highlighted and the sound file name under analysis is displayed just under the title.

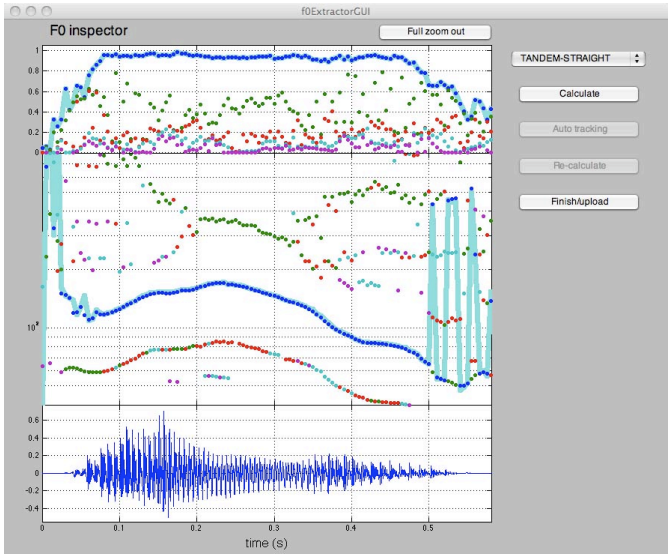


Fig. 5. F0 extraction GUI after initial F0 extraction. F0 extractor is default TANDEM-STRAIGHT F0 extractor.

The results shown in Fig. 5 were obtained using the default TANDEM-STRAIGHT F0 extractor, XSX. The material is a Japanese vowel sequence /aiueo/ spoken by a male speaker at a relatively fast speaking rate. The middle panel shows the extracted F0 trajectory (represented with a thick cyan-colored line). Other F0 candidates that represent local periodicity are plotted using colored dots. The color of the dots represents the order of the periodicity level of each candidate. The periodicity levels and the color ordering are shown in the top panel. One

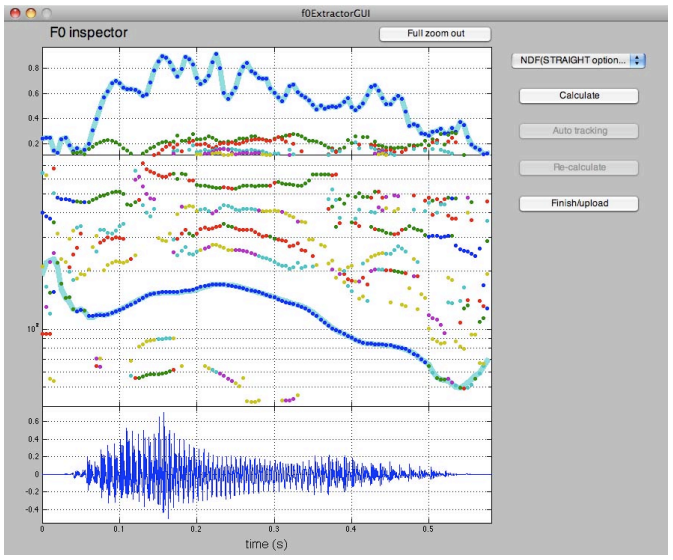


Fig. 6. F0 extraction results by NDF. Note that no defect is found in the extracted trajectory.

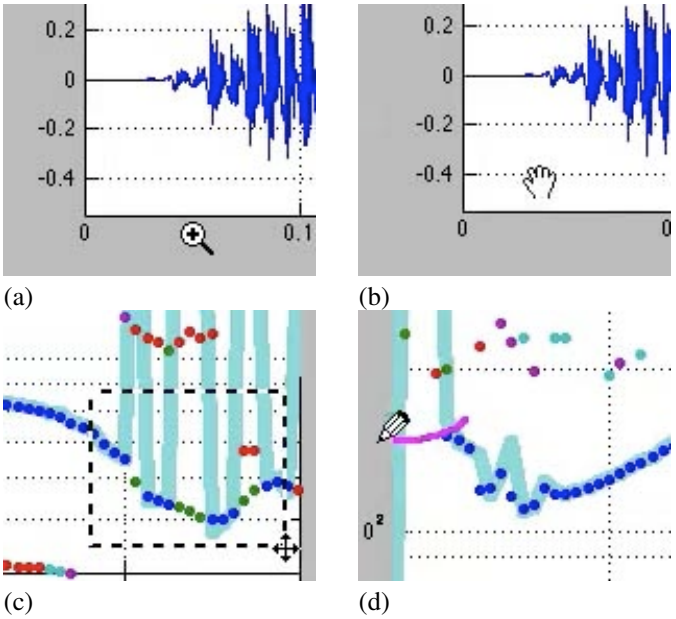


Fig. 7. F0 trajectory editing tools. Pointer shapes are dependent on graphical objects under pointer and modifier keys.

important feature of XSX is this periodicity level. If a signal is purely periodic in the observation window, its periodicity level goes to 1. The other candidates also represent actually existing local periodicity.

Figure 6 shows the results by NDF. The resulting F0 trajectory does not have any defects. However, the other candidates shown in the plots do not have physically meaningful information. It also consists of spurious candidates caused by nonlinear interactions between two internal procedures using different periodicity clues. The other problem is that NDF is slower than the other F0 extractors and cannot handle long sound files due to memory demands.

Figure 7 shows available edit functions represented by shapes of pointers. They obey common GUI practice. The time axis can be magnified by (mouse or any other positioning device) clicking when the pointer shape is shown like (a) in Fig. 7. The sign inside of the pointer changes to the minus sign while the "shift" key is hold down. With the minus sign, clicking makes the time axis shrink. Positioning the pointer inside of the waveform plot turns it into a "hand" shape. The magnified time axis can be dragged while the pointer exhibits the hand shape.

The extracted F0 trajectory can be edited in two ways. By using rubber band region selector, the search range for F0 candidates can be explicitly defined. If no candidates are found for a specific frame inside the region, the originally selected candidate at the frame is not replaced. This is the default behavior and is shown in (c) of Fig. 7. By holding down the "alt" key while the pointer is inside the F0 candidate plot, the pointer shape changes to "pen." Drawing a line using the pen provides a hint for candidate search. The closest candidate at each frame is selected when the minimum distance from the line to the candidate is less than the predetermined threshold (one half octave, this time). When no candidate is found within the threshold, the value on the drawn line is substituted to F0 of the frame and this substitution is recorded.

When this F0 extraction and editing stage is completed, clicking "Finish/upload" brings the interface menu to the front of all windows. The menu at this phase is shown in the right plot of Fig. 4.

2) Aperiodicity and STRAIGHT spectrum extraction: Two analysis buttons on the menu are active, as shown in the right plot of Fig. 4. The first one is for aperiodicity extraction and the second is for STRAIGHT spectrum extraction. These buttons can be clicked in any sequence. The "`clear`" button removes all analysis results and enables the button for F0 extraction again.

3) Resynthesis and saving results: The left panel of Fig. 8 shows how the menu looks after aperiodicity and STRAIGHT-spectral analyses. At this point, all the necessary information for resynthesizing speech is ready. The analysis results can be saved to a file by clicking the "`Save STRAIGHT object`" button. It is also possible to click the "`Synthesis`" button to resynthesize speech. The right panel of Fig. 8 shows the state after resynthesis. The synthesized speech also can be saved to a file by clicking the "`Save synthesized sound`" button.

4) Access to internal variables: The input data and analysis results with other housekeeping information are stored in a `userdata` field of a graphic object, this time the menu. By using a handle of the object, it is easy to access to this information. The following sequence provides an example:

```
>> Thandle = TandemSTRAIGHThandler;
>> userData = get(Thandle,'userdata');
```

where "`>>`" is the prompt of the Matlab system. After executing these lines, the variable "`userData`" has the fields shown in Fig. 9. Field name was made readable, meaning self-explanatory. Fields consisting of analysis and synthesis data are themselves structured variables with detailed information inside. Users can modify the content of these fields and use them to resynthesize their own manipulated sounds.

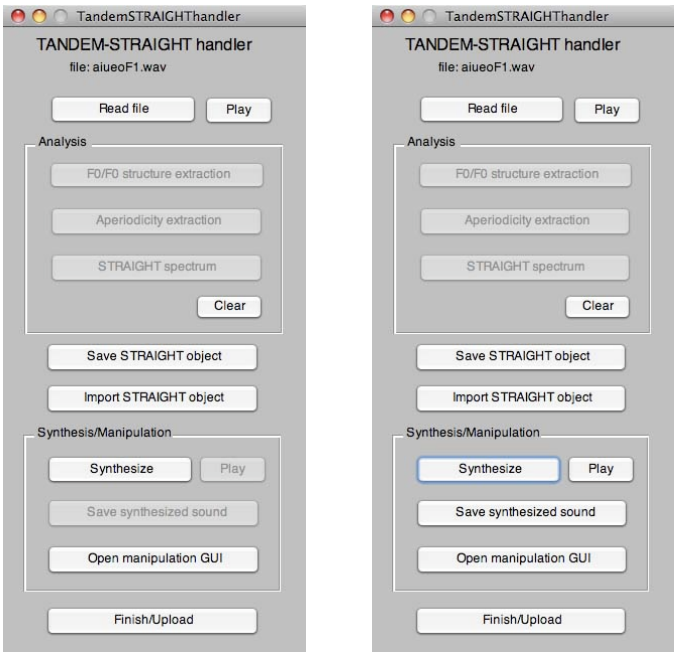


Fig. 8. Left: State of STRAIGHT handling menu after aperiodicity and spectral analysis. Right: State after resynthesis.

```
                creationDate: '20090430T110057'
               dataDirectory: [1x59 char]
                dataFileName: 'aiueoF1.wav'
            samplingFrequency: 44100
                    waveform: [25600x1 double]
                  standAlone: 1
              currentHandles: [1x1 struct]
           F0extractionDate: '20090430T110107'
     HandleOfCallingRoutine: 158.0012
        originalF0Structure: [1x1 struct]
         refinedF0Structure: [1x1 struct]
AperiodicityExtractionDate: '20090430T110138'
      AperiodicityStructure: [1x1 struct]
     SpectrumExtractionDate: '20090430T110135'
          SpectrumStructure: [1x1 struct]
        SynthesisStructure: [1x1 struct]
```

Fig. 9. Fields of structure variable `userData` after synthesis is completed.

This structure variable is also the information stored using the "`Save STRAIGHT object`" button. A structured variable named "`STRAIGHTobject`" is used to save the information using the "`save`" command of Matlab.

5) GUI for STRAIGHT parameter manipulation: A specialized GUI for STRAIGHT parameter manipulation is invoked by clicking the "`Open manipulation GUI`" button or typing the "`STRAIGHTmanipulatorGUI`" command in the Matlab command window.

Figure 10 shows the initial state of the GUI. The example uses the same speech material used in the previous sections. The top panel shows the STRAIGHT spectrogram. Similar to the waveform window in the F0 extraction GUI, this spectrographic display can be expanded and dragged horizontally. The following five plots are used for parameter manipulations. Horizontal axes of these plots are zoomed and move in synchrony with the spectrogram plot. They provide

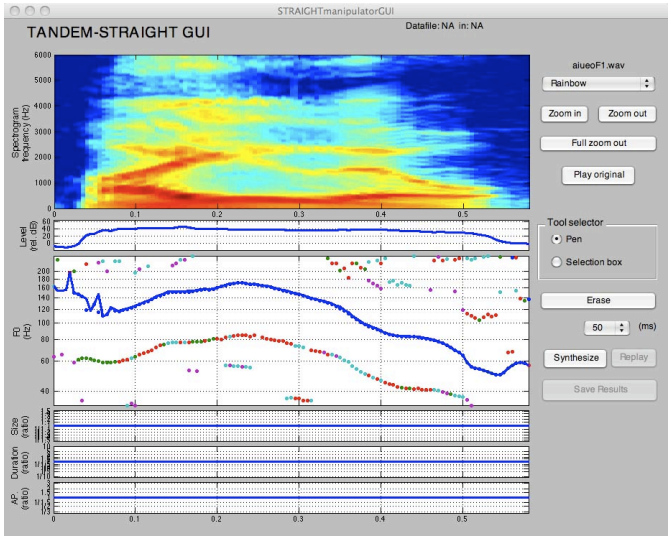
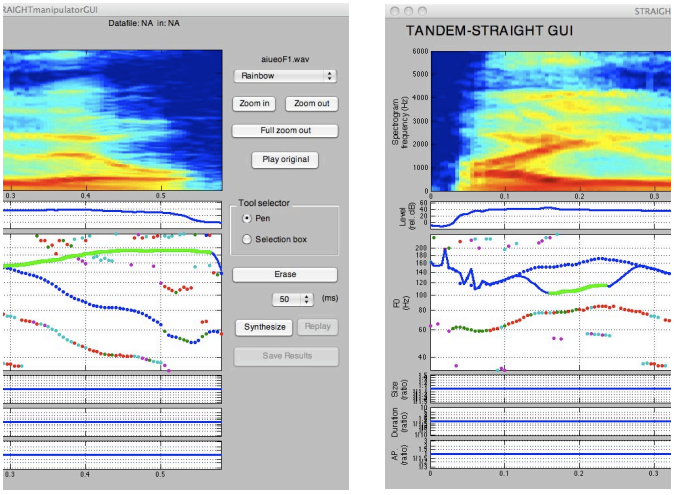
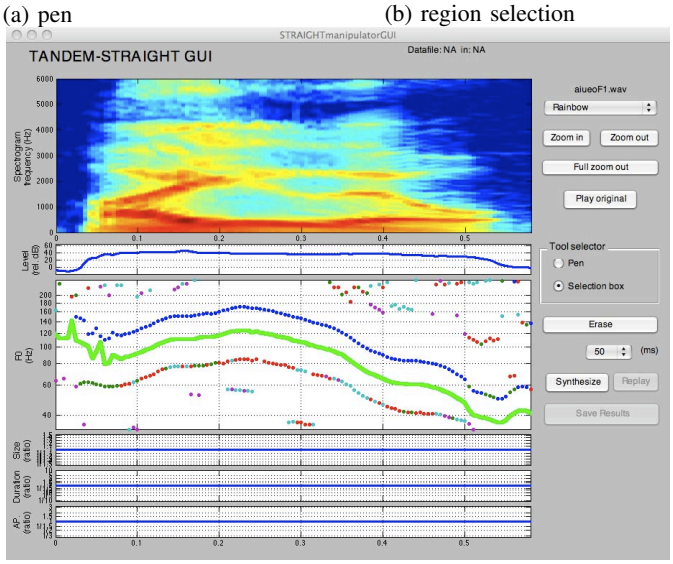


Fig. 10. GUI for STRAIGHT parameter manipulation.



(a) pen (b) region selection



(c) dragging using a whole trajectory

Fig. 11. Three types of F0 manipulation. Top left: drawing a line using a pen. Top right: selecting a region and dragging. Bottom: selecting a whole trajectory and dragging.

manipulation of power, F0, size (reciprocal of frequency axis stretch), duration and aperiodicity, from top to bottom. Only the focused parameter is displayed using vertically wide space and the other parameter plots are shrunken. The shrunken plot expands when it is clicked. Power and F0 have their values plotted in their panels and the others have plots of amount of modifications. The vertical axis of each manipulation plot can also be zoomed and dragged to precisely manipulate parameters.

Three types of interaction methods are prepared: (a) drawing a line using a pen tool, (b) vertically displacing the selected region by dragging and (c) dragging the whole trajectory. Transitional regions are prepared for both sides of the region in the manipulation types (a) and (b) to connect to the the manipulated region and the other parts.

By clicking the "`Synthesis`" button, synthetic speech is generated and reproduced using the manipulated values. The synthesized sound and the manipulations defined by the GUI are stored using "`.wav`" format and "`.mat`" format respectively. The waveform information and the manipulation information files are assigned the same name except for the extensions.

B. Second set: Extended morphing

Four types of GUIs are prepared for assisting morphing preparation: main menu, temporal anchoring GUI, frequency anchoring GUI and morphing parameter manipulation GUI.

1) Main menu for morphing: Figure 12 shows the main menu of the morphing procedure. This menu is invoked by typing "`MorphingMenu`" in the Matlab command window. Similar to the STRAIGHT parameter handling menu, user data of the menu consists of housekeeping information and STRAIGHT parameters.

In Fig. 12, the left panel shows the initial status. There are two options: start from sound files or import prepared data. When starting from sound files, "`Analyze A`" or "`Analyze B`" invokes the STRAIGHT handling menu that was described in the previous sections. The analysis result is returned to the user data of the morphing menu.

When starting from data import, necessary data for morphing is ready just after importing. The right panel of Fig. 12 shows the state of the menu after data importing.

2) Assigning temporal anchors: The next step is to assign temporal anchoring points for defining the mapping function of the time axis. Clicking "`Open anchoring interface`" on the menu invokes the GUI for temporal anchor assignment shown in Fig. 13. The top figure of Fig. 13 shows the state after distance calculation.

The two left plots represent STRAIGHT spectrograms of example A and example B with power plot and waveform plot attached. Vertical white lines represent assigned temporal anchors. The diamond-shaped plots show the distance matrix between two examples and its shrunken copy. They are rotated 45 degrees. The distance is calculated using MFCC. Other

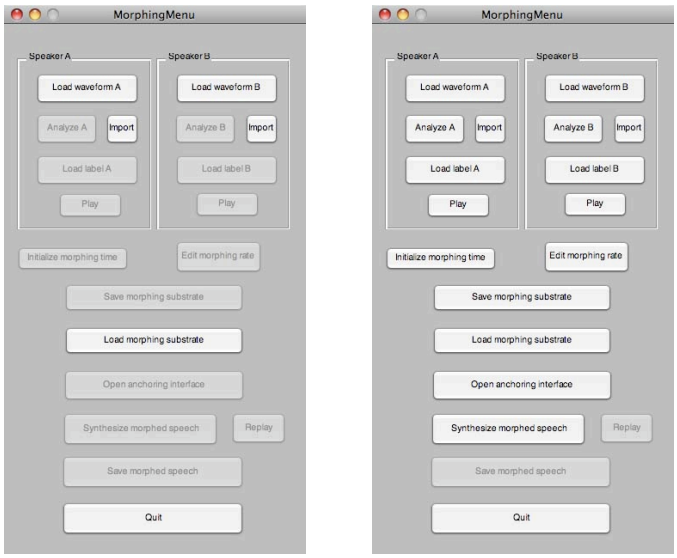
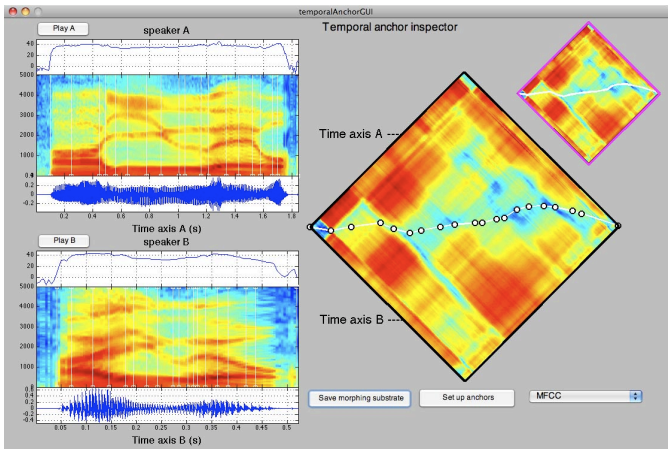
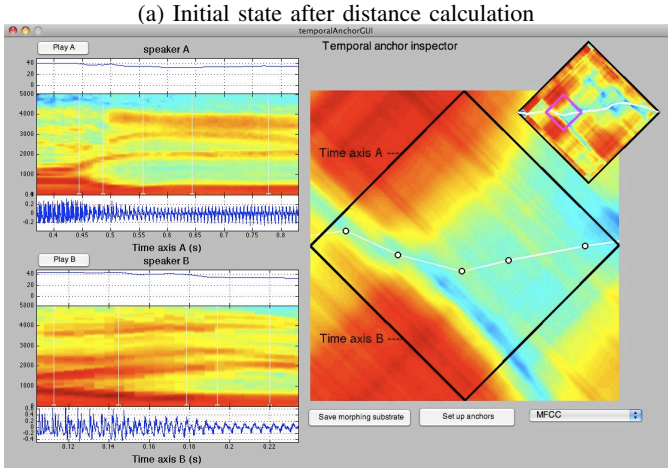


Fig. 12. Left: Initial state of morphing handling menu. Right: State after importing a morphing substrate.



(a) Initial state after distance calculation



(b) Inspection using zooming

Fig. 13. GUI for temporal anchor assignment.

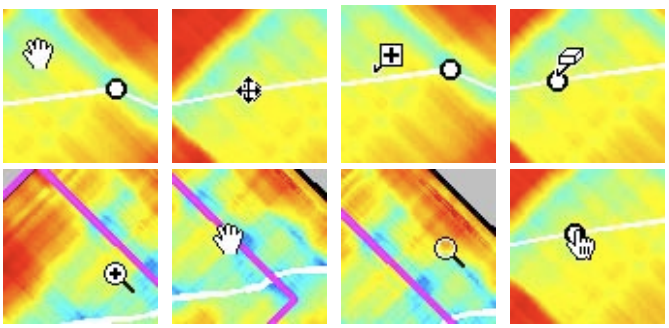


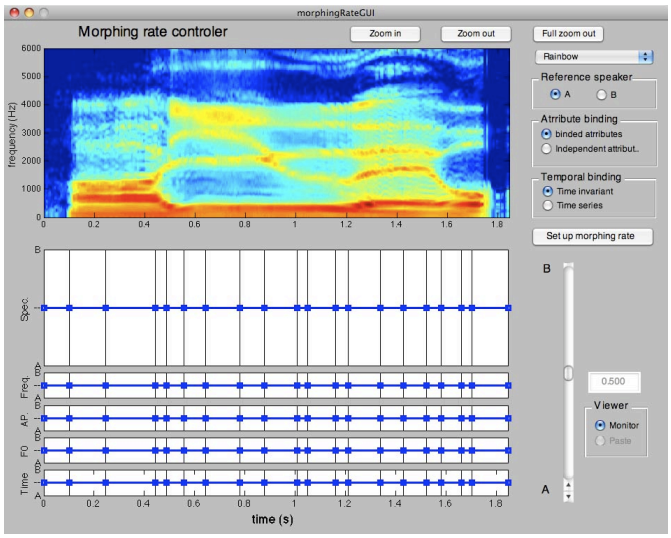
Fig. 14. Pointer shapes used in GUI for temporal anchoring.

Fig. 15. GUI for frequency anchor assignment. Blue and red lines represent STRAIGHT spectrum slice at anchoring points on example A and example B respectively.

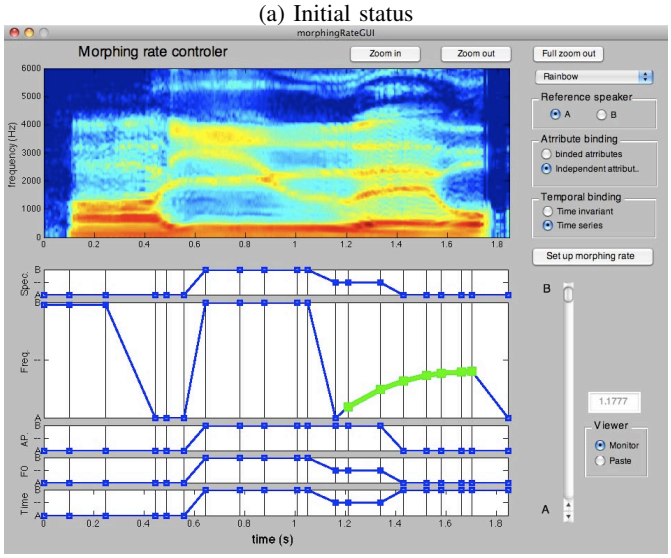
distance measures can also be used. A white trajectory with open circles represents the assigned mapping path and temporal anchors.

The bottom panel shows the zoomed distance matrix. Spectrograms on the left are also zoomed. This magnification is used to edit anchoring points. Figure 14 shows pointer shapes used in this edit mode. The shapes represent their assigned functions; from top left to bottom right: distance matrix dragging, anchor position modification, addition of new anchor point, deletion of existing anchor point, expansion of zoomed region, dragging inspection region, relocation to the pointer position and inspection of spectral slices for frequency anchoring. These functions are dependent on graphic objects under pointer and depression of modifier keys. The assigned temporal anchor points are returned to the morphing menu by clicking the "`set up anchors`" button.

3) Assigning frequency anchors: Holding down the "alt" modifier key and clicking one of the anchor points on the mapping trajectory invokes the GUI for frequency anchoring. Figure 15 shows the GUI for frequency anchoring. Similar to temporal anchoring, anchor points can be added, moved and removed by using modifier keys and pointer operation. This plot can also be zoomed and dragged horizontally. In this implementation, frequency anchoring points are set to minimize discrepancy of representative spectral features, such



(a) Initial status



(b) Morphing rate assignment example

Fig. 16. GUI for morphing rate assignment.

as formant frequencies. The assigned frequency anchor points are returned to the morphing menu by clicking the "`update frequency anchors`" button.

4) Morphing rate manipulation: When all anchoring information is ready, the "`Synthesize morphed speech`" button is enabled. Also, with the same preparation, the "`Edit morphing rate`" button is enabled. By clicking the latter button, the GUI for the morphing rate controller appears.

Figure 16 shows the GUI. The upper plot shows the initial state. The lower plot shows an example of completed editing. The GUI consists of six plots; from top to bottom: STRAIGHT spectrogram, spectrum level morphing rate, frequency axis morphing rate, aperiodicity morphing rate, F0 morphing rate and temporal axis morphing rate. Similar to the STRAIGHT parameter manipulation GUI, the spectrographic display is used as a handle for zooming and dragging on the horizontal axis. The other four plots are used to assign morphing rates. The vertical lines in those five plots represent temporal anchoring points. Morphing rates are represented by a piecewise

```
                    creator: 'morphingSubstrate'
                creationDate: '20081103T073341'
  dataDirectoryForSpeakerA: [1x57 char]
  dataDirectoryForSpeakerB: [1x57 char]
       fileNameForSpeakerA: 'aiueoL.wav'
       fileNameForSpeakerB: 'aiueoF2.wav'
         samplintFrequency: 44100
   temporaAnchorOfSpeakerA: [18x1 double]
   temporaAnchorOfSpeakerB: [18x1 double]
 frequencyAnchorOfSpeakerA: [1x1 struct]
 frequencyAnchorOfSpeakerB: [1x1 struct]
                f0OfSpeakerA: [370x1 double]
     f0TimeBaseOfSpeakerA: [1x370 double]
                f0OfSpeakerB: [105x1 double]
     f0TimeBaseOfSpeakerB: [1x105 double]
STRAIGHTspectrogramOfSpeakerA: [1025x370 double]
spectrogramTimeBaseOfSpeakerA: [1x370 double]
STRAIGHTspectrogramOfSpeakerB: [1025x105 double]
spectrogramTimeBaseOfSpeakerB: [1x105 double]
     aperiodicityOfSpeakerA: [1x1 struct]
aperiodicityTimeBaseOfSpeakerA: [1x370 double]
     aperiodicityOfSpeakerB: [1x1 struct]
aperiodicityTimeBaseOfSpeakerB: [1x105 double]
      temporalMorphingRate: [1x1 struct]
      anchorOnMorphingTime: [18x1 double]
          morphingTimeAxis: [190x1 double]
              realTimeBase: [1x1 struct]
          morphedDisplayF0: [193x1 double]
    morphedDisplayRealTime: [193x1 double]
    morphedDisplayspectrum: [1025x193 double]
                lastUpdate: '30-April-2009 04:44:19'
    waveformForSpeakerA: [81408x1 double]
    waveformForSpeakerB: [23040x1 double]
  originalSubstratePath: [1x59 char]
  originalSubstrateFile:  (intentionally removed)
    currentHandleToMenu: 158.0031
             menuHandle: 158.0031
   interfaceGHIhandle: 182.0032
```

Fig. 17. Fields and contents of "`mSubstrate`" in this example. Words `Speaker A` and `Speaker B` are used instead of `Example A` and `Example B` in this implementation.

linear function using temporal anchoring points as nodes of the function.

Two types of anchoring point allocation are prepared: drawing a line with a pen and dragging a node group selected by region selection. In the case of line drawing, the crossing point at each temporal anchor is copied to the morphing rate value of the node point. The green thick line in the lower plot of Fig. 16 shows the resulting anchor group made by pen drawing. The multi-aspect and temporally variable morphing rates are sometimes too flexible. This GUI provides a way to introduce constraints using radio buttons: attribute binding and temporal binding. When both constraints are effective, this morphing is equivalent to morphing using a scalar morphing rate.

5) Saving morphing information and synthesis: By using the "`Save morphing substrate`" button, all information necessary to redo morphing can be saved to a "`.mat`" file that can be imported using the main morphing menu. The stored data is represented by a Matlab variable "`mSubstrate`" that is one field of the user data of the menu. It is accessible by using the following commands:

```
>> Mhandle = MorphingMenu
>> userData = get(Mhandle,'userdata');
>> userData.mSubstrate
```

Figure 17 shows fields of "`mSubstrate`" in this example. Based on this information, a morphed sound is generated by clicking the "`Synthesis morphed speech`" button. The morphed speech can be saved as a "`.wav`" file by clicking the "`Save morphed speech`" button.