Recent development of the HMM-based speech synthesis system (HTS)

Heiga Zen,[*‖] Keiichiro Oura,[*] Takashi Nose,[†] Junichi Yamagishi,[†‡] Shinji Sako,[*]
Tomoki Toda,[§] Takashi Masuko,[†**] Alan W. Black,[¶] Keiichi Tokuda[*]
[*] Department of Computer Science, Nagoya Institute of Technology, Nagoya 466–8555, Japan
[†] Department of Information Processing Interdisciplinary, Tokyo Institute of Technology, Yokohama 226–8502, Japan
[‡] The Centre for Speech Technology Research, University of Edinburgh, Edinburgh EH8–9LW, UK
[§] Graduate School of Information Science, Nara Institute of Science and Technology, Ikoma 630–0192, Japan
[¶] Language Technologies Institute, Carnegie Mellon University, Pittsburgh, PA 15213, USA
[‖] Presently, with the Cambridge Research Laboratory, Toshiba Research Europe Limited, Cambridge CB4 0GZ, UK
[**] Presently, with the Corporate Research & Development Center, Toshiba Corporation, Kawasaki 212–8582, Japan

***Abstract*—A statistical parametric approach to speech synthesis based on hidden Markov models (HMMs) has grown in popularity over the last few years. In this approach, spectrum, excitation, and duration of speech are simultaneously modeled by context-dependent HMMs, and speech waveforms are generated from the HMMs themselves. Since December 2002, we have publicly released an open-source software toolkit named "HMM-based speech synthesis system (HTS)" to provide a research and development toolkit for statistical parametric speech synthesis. This paper describes recent developments of HTS in detail, as well as future release plans.**

I. Introduction

A statistical parametric approach to speech synthesis based on hidden Markov models (HMMs) has grown in popularity over the last few years [1]. In this approach, context-dependent HMMs are estimated from databases of natural speech, and speech waveforms are generated from the HMMs themselves. This framework makes it possible to model different voice characteristics, speaking styles, or emotions without recording large speech databases. For example, adaptation [2], interpolation [3], and eigenvoice techniques [4] were applied to this system, and it was found that voice characteristics could be modified.

Since December 2002, we have publicly released an open-source software toolkit named "HMM-based speech synthesis system (HTS)" [5] to provide a research and development platform for statistical parametric speech synthesis. Various organizations currently use it to conduct their own research projects, and we believe that it has contributed significantly to the success of HMM-based speech synthesis today. This paper describes the recent developments of this system as well as future release plans.

The rest of this paper is organized as follows: Section 2 reviews statistical parametric speech synthesis. In Section 3, details of HTS are described. Other applications of HTS are presented in Section 4. Concluding remarks and future release plans are presented in the final section.

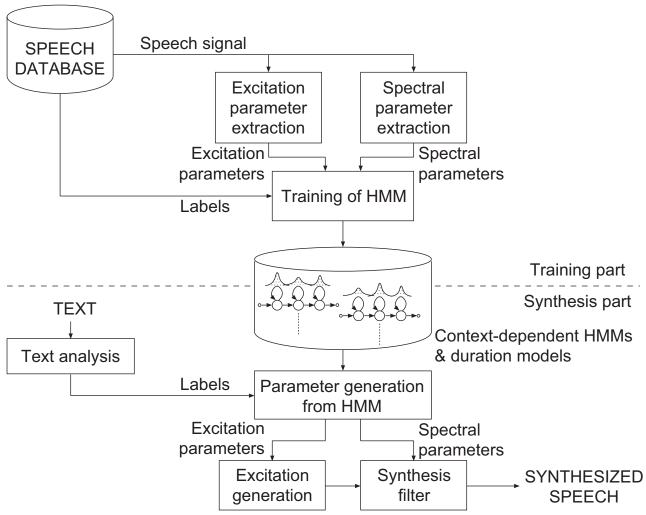

Fig. 1. Overview of HMM-based speech synthesis.

II. Statistical parametric speech synthesis

Text-to-speech synthesis can be viewed as the inverse procedure of speech recognition. The goal of any text-to-speech synthesizer is to take a word sequence, $\boldsymbol{w} = \{w_1, \ldots, w_N\}$, as its input and produce an acoustic speech waveform, $\boldsymbol{o} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_T\}$. In a typical system, contextual factors such as accent, lexical-stress, part-of-speech, and phrase-boundary contexts are assigned to a given word sequence, $\boldsymbol{w}$, by a natural language processing engine, and then $\boldsymbol{w}$ is mapped into the corresponding context-dependent sub-word sequence, $\boldsymbol{u} = \{u_1, \ldots, u_M\}$. Finally, a speech waveform, $\boldsymbol{o}$, is synthesized for $\boldsymbol{u}$.

Most state-of-the-art speech synthesis systems are based on large amounts of speech data. This type of approach is generally called corpus-based speech synthesis [6]. This approach makes it possible to dramatically improve the naturalness of synthesized speech compared with early speech synthesis systems such as rule-based ones.

One of the major approaches in corpus-based speech synthesis is sample-based one: unit-selection [7]. In this approach, speech data are segmented into small units, *e.g.*, HMM state, half-phone, phone, diphone, or syllable, and stored. Then a unit sequence corresponding to a given context-dependent sub-word sequence is selected by minimizing its total cost, consisting of target and concatenation costs [7]. These cost functions have been formed from a variety of heuristic or `ad hoc` quality measures based on features of the acoustic signals and given texts. Recently, target and concatenation cost functions based on statistical models have been proposed and investigated [8], [9], [10], [11], [12].

Another major approach is statistical parametric one: HMM-based speech synthesis [1]. It generates a speech parameter vector sequence, $\boldsymbol{o} = \{\boldsymbol{o}_1, \boldsymbol{o}_2, \ldots, \boldsymbol{o}_T\}$, with maximum a posteriori (MAP) probability given the context-dependent sub-word sequence, $\boldsymbol{u}$, as follows:

$$\hat{\boldsymbol{o}} = \arg\max_{\boldsymbol{o}} P(\boldsymbol{o} \mid \boldsymbol{u}). \tag{1}$$

Although any kind of generative models can be applied to represent $P(\boldsymbol{o} \mid \boldsymbol{u})$, currently HMMs are widely used.

Figure 1 overviews HMM-based speech synthesis. It consists of training and synthesis parts. The training part is similar to that used in speech recognition. The main difference is that both spectrum (*e.g.*, mel-cepstral coefficients and their dynamic features) and excitation (*e.g.*, $\log F_0$ and its dynamic features) parameters are extracted from a speech database and modeled by context-dependent HMMs (phonetic, linguistic, and prosodic contexts being taken into account). To model variable dimensional parameter sequences such as $\log F_0$ with unvoiced regions, multi-space probability distributions (MSDs) [13] are used for state-output distributions. Each HMM has its state-duration distributions to capture the temporal structure of speech. As a result, spectrum, excitation, and duration are modeled simultaneously in a unified HMM framework [1].

The synthesis part does the inverse operation of speech recognition. First, an arbitrarily given text to be synthesized is converted to a context-dependent label sequence, and then a sentence HMM is constructed by concatenating the context-dependent HMMs according to the label sequence. Second, state durations of the sentence HMM are determined based on the state-duration distributions. Third, the speech parameter generation algorithm generates sequences of spectral and excitation parameters that maximize their output probabilities under the constraints between static and dynamic features [14]. Finally, a speech waveform is synthesized directly from the generated spectral and excitation parameters using a speech synthesis filter. The most attractive part of this system is that voice characteristics, speaking styles, or emotions can easily be modified by transforming HMM parameters using various techniques, such as adaptation [2], interpolation [15], or eigenvoices [4].

III. HTS: A software toolkit for HMM-based speech synthesis

A. Outline

The HMM-based speech synthesis system (HTS) has been developed by the HTS working group as an extension of the HMM toolkit (HTK) [16]. The source code of HTS is released as a patch for HTK. Although the patch is released under a free software license (new and simplified BSD license [17]), once the patch is applied users must obey the license of HTK.[1] The HTS patch code can be downloaded from the HTS website [5]. After downloading HTK, HDecode, and HTS tar balls and expanding them, you can obtain HTS source codes by applying the patch code for HTK as

```
% cd htk
% patch -p1 -d . < HTS-2.1_for_HTK-3.4.patch
```

Finally, by running configure and Make scripts, HTS executable codes are generated. Note that HTS has not support the latest HTK version 3.4.1 yet at the time of writing this paper.

The history of the main modifications which we have made are listed below:

- **Version 1.0 (December 2002)**
  - Tree-based clustering based on the MDL criterion [18].
  - Stream-dependent tree-based clustering [1].
  - Multi-space probability distributions (MSD) [13].
  - State-duration modeling and clustering [19].
  - Speech parameter generation algorithm [14].
  - Demo using the CMU Communicator database.
- **Version 1.1 (May 2003)**
  - Small run-time synthesis engine.
  - Demo using the CSTR TIMIT database.
  - HTS voices for the Festival speech synthesis system [20].
- **Version 1.1.1 (December 2003)**
  - Variance flooring for MSD-HMMs.
  - Post-filtering [21].
  - Demo using the CMU ARCTIC database.
  - Demo using the Nitech Japanese database.
  - HTS voice for the Galatea toolkit [22].

B. HTS version 2.0 / 2.0.1

After an interval of three years, HTS version 2.0 was released in December 2006. This was a major update and included a number of new features and fixes:

- Terms about redistributions in binary form were added to the HTS license.
- `HCompV` (global mean and variance calculation tool) accumulates statistics in double precision. For large databases the previous versions often suffered from numerical errors.

[1] The HTK license prohibits redistribution and commercial use of source, object, or executable codes.

- HRest (Baum-Welch re-estimation tool for a single HMM) can generate state-duration distributions [19] with the -g option.
- Phoneme boundaries can be given to HERest (embedded Baum-Welch re-estimation tool) using the -e option. This can reduce computational cost and improve phoneme segmentation accuracy [23]. Subsets of boundaries (e.g, pause positions) may also be specified.
- Reduced-memory implementation of tree-based clustering in HHEd (a tool for manipulating HMM definitions) with the -r option. For large databases the previous versions sometimes consumed huge memory.
- Each decision tree can have a name with regular expressions (HHEd with the -p option). As a result, two different trees can be constructed for consonants and vowels respectively.
- Flexible model structures in HMGenS (speech parameter generation tool). In the previous versions, we assumed that the first HMM stream is for mel-cepstral coefficients and the others are for $\log F_0$. Now we can specify model structures using the configuration variables PDFSTRSIZE and PDFSTRORDER. Non-left-to-right model topologies (*e.g.*, ergodic HMM), Gaussian mixtures, and full covariance matrices are also supported.
- Speech parameter generation algorithm based on the expectation-maximization (EM) algorithm (the Case 3 algorithm in [14]) in HMGenS. Users can select generation algorithms using the -c option.
- Random generation algorithm [24] in HMGenS. Users can turn on this function by setting a configuration variable RNDPG=TRUE.
- State- or phoneme-level alignments can be given to HMGenS.
- The interface of HMGenS has been switched from HHEd-style to HERest-style.
- Various kinds of linear transformations for MSD-HMMs are supported in HERest.
  - Constrained and unconstrained maximum likelihood linear regression (MLLR) based adaptation [25].
  - Adaptive training based on constrained MLLR [25].
  - Precision matrix modeling based on semi-tied covariance matrices [26].
  - Heteroscedastic linear discriminant analysis (HLDA) based feature transform [27].
  - Phonetic decision trees can be used to define regression classes for adaptation [28]
  - Adapted HMMs can be converted to the run-time synthesis engine format.
- Maximum a posteriori (MAP) adaptation [29] for MSD-HMMs in HERest.

HTS version 2.0.1 was a bug-fixed version. The new features in this version are as follows:

- Band structure for linear transforms [30].
- Speaker interpolation [3].
- Stream-dependent variance flooring scales.
- Demo scripts support LSP-type spectral parameters.
- β version of the runtime synthesis engine API.

C. New features in version 2.1

The latest version, HTS version 2.1, was released in July 2008. This version includes four important new features: hidden semi-Markov models (HSMMs) [31], [32], the speech parameter generation algorithm considering global variance (GV) [33], advanced adaptation techniques [34], and stable version of runtime synthesis engine API.

Note that HTS version 2.1, with the STRAIGHT analysis/synthesis technique [35], provides the ability to construct the state-of-the-art HMM-based speech synthesis systems developed for the past Blizzard Challenge events [36], [37], [38], [39].

1) Hidden semi-Markov model: This section describes hidden semi-Markov models. In HMM-based speech synthesis, rhythm and tempo are controlled by state-duration distributions. They are estimated from statistical variables obtained at the last iteration of the forward-backward algorithm, and then clustered by a decision tree-based context-clustering algorithm: state-duration distributions are not iteratively re-estimated in the Baum-Welch algorithm [19]. At the synthesis stage, we construct a sentence HMM and determine its state durations so as to maximize their probabilities. Then, speech parameter vector sequences are generated. However, there is an inconsistency: although parameters of HMMs are estimated without explicit state-duration distributions, speech parameter vector sequences are generated from HMMs using the explicit state-duration distributions. This inconsistency can degrade the quality of synthesized speech.

To resolve the discrepancy, HSMMs [40], which can be viewed as HMMs with explicit state-duration distributions, were introduced into the training part [31]. The use of HSMMs makes it possible to simultaneously re-estimate state-output and -duration distributions. The adaptation and adaptive training techniques for HSMMs were also derived [32]. Zen *et al.* reported small improvements in speaker-dependent systems [31]. However, Tachibana *et al.* reported that the use of HSMM was essential to adapt state-durations distributions [41]. The HSMM was also successfully applied to speech recognition [42].

2) Speech parameter generation algorithm considering global variance: In the basic system, the speech parameter generation algorithm is used to generate spectral and excitation parameters from the HMMs [14]. By taking into account constraints between the static and dynamic features, it can generate smooth speech parameter trajectories. However, the generated spectral and excitation parameters are often excessively smooth compared with those of natural speech. This over-smoothing is due to the statistical process of model training, and causes degradation in the naturalness of synthesized speech. To avoid this problem, Toda *et al.* proposed a speech parameter generation algorithm considering global variance (GV) [33].

This algorithm iteratively maximizes the following objective function with respect to a speech parameter vector sequence $\boldsymbol{c} = \left[\boldsymbol{c}_1^\top, \ldots, \boldsymbol{c}_T^\top\right]^\top$ (static features only):

$$\mathcal{F}_{\mathrm{GV}}(\boldsymbol{c}) = w \log P(\boldsymbol{W}\boldsymbol{c} \mid \boldsymbol{q}, \lambda) + \log P(v(\boldsymbol{c}) \mid \lambda_v) \qquad (2)$$

where λ is a sentence HSMM, $\boldsymbol{q} = \{q_1, \ldots, q_T\}$ is a state sequence determined by state-duration distributions, $\boldsymbol{W}$ is a window matrix which appends delta and delta-delta features to $\boldsymbol{c}$, w is a weight for the state-output probability, $v(\boldsymbol{c})$ is the GV of $\boldsymbol{c}$ which is defined as an intra-utterance variance of $\boldsymbol{c}$, and λ_v denotes parameters of a GV distribution. The second term of Eq. (2) can be viewed as a penalty term for over-smoothing. The use of this algorithm dramatically reduces the buzziness in synthesized speech and improves the speech quality [33]. This was one of the main components of Nitech's Blizzard Challenge 2005 system [36].

3) CSMAPLR: The MLLR adaptation algorithms utilize the ML criterion to estimate linear transformation matrices. However, the amount of adaptation data is usually very limited at the adaptation stage. Therefore, we should use more robust criteria such as the MAP criterion. In the MAP estimation, we estimate the linear transformation matrices $\boldsymbol{X}$ as follows:

$$\hat{\boldsymbol{X}} = \arg\max_{\boldsymbol{X}} P(\boldsymbol{o} \mid \lambda, \boldsymbol{X})\, P(\boldsymbol{X}) \qquad (3)$$

where $P(\boldsymbol{X})$ is a prior distribution for the linear transformation matrix $\boldsymbol{X}$.

In the structured MAP linear regression (SMAPLR) [43], we first estimate a global linear transformation matrix at the root node of the tree structure using all the adaptation data, and then propagate it to its child nodes as their prior distributions. In the child nodes, linear transformation matrices are estimated again using their adaptation data, based on the MAP criterion with the propagated prior distributions. Then, the recursive MAP-based estimation of the transformation matrices from a root node to lower nodes is conducted. Yamagishi *et al.* applied the SMAP to the constrained MLLR adaptation and derived constrained SMAPLR (CSMAPLR) [34], in which the linear transformation matrices for both mean vectors and covariance matrices of state-output distributions are shared and estimated using the recursive MAP criterion. The CSMAPLR adaptation algorithm can utilize the tree structure more effectively than the constrained MLLR adaptation since the tree structure represents connection and similarity between the distributions, and the propagated prior information automatically reflects the connection and similarity. This algorithm was applied to the HMM-based speech synthesis and showed that it was better than the other linear transformation-based adaptation techniques [34]. This technique is also useful for speech recognition [44].

4) hts_engine API: Since version 1.1, a small stand-alone run-time synthesis engine named `hts_engine` has been included in the HTS releases. It works without the HTK libraries, and it is released under the new and simplified BSD license; Users can develop their own open or proprietary software based on the run-time synthesis engine and redistribute these source, object, and executable codes without any restriction. In fact, a part of `hts_engine` has been integrated into several pieces of software, such as ATR XIMERA [45], Festival [20], and OpenMARY [46]. The spectrum and prosody prediction modules of ATR XIMERA are based on `hts_engine`. Festival includes `hts_engine` as one of its waveform synthesis modules. The upcoming version of OpenMARY uses the JAVA version of `hts_engine`.

As described above, `hts_engine` has been used as a module rather than a piece of stand-alone software. This suggests that users require the `hts_engine` library, not the stand-alone program. In response to this demand, we decided to rewrite `hts_engine` in an API-style implementation. The stable version, `hts_engine API` version 1.0, was released with HTS version 2.1. It is written in C and provides various functions required to setup and drive the synthesis engine. The reference for this API is also available. It supports LSP-type parameters in addition to cepstral parameters. The speech parameter generation algorithm considering GV is also included. `Flite+hts_engine`, which is a combination of CMU Flite and `hts_engine`, was also released. It shows an implementation of English TTS for embedded devices using Flite front-end and `hts_engine` back-end.

Both `hts_engine` and `Flite+hts_engine` can be downloaded from the `hts_engine` SourceForge project website [47].

D. Demonstrations and documentation

Currently two demo scripts to construct speaker-dependent systems (English and Japanese) and a demo script to train a speaker-adaptation system (English) have been released. The English demo scripts use the CMU ARCTIC databases and generate model files for Festival and `hts_engine`. The Japanese demo script uses the Nitech database and generates model files for the Galatea toolkit [22]. These scripts demonstrate the training processes and the functions of HTS. The demo scripts to construct speaker-dependent and adaptation systems with the STRAIGHT analysis/synthesis technique [35] are also released. The demo scripts first extract STRAIGHT spectrum, F_0, and aperiodicities using the MATLAB version of STRAIGHT then they are converted spectral parameters such as mel-cepstral coefficients or LSPs, $\log F_0$, and band aperiodicities. At the synthesis stage, generated spectral parameters, $\log F_0$, and band aperiodicities are converted to STRAIGHT spectrum, F_0, and aperiodicities then the MATLAB version of STRAIGHT reconstructs a waveform from these STRAIGHT parameters.

Six voices for Festival trained by the CMU ARCTIC databases have also been released. Each HTS voice consists of model files trained by the demo script, and can be used as a voice for Festival without any other HTS tools.

Currently no documentation for HTS is available. However, the interface and functions of HTS are almost the same as those of HTK. Therefore, users who are familiar with HTK can easily understand how to use HTS. The manual of HTK

[16] is also very useful. There is also an open mailing list for discussion of HTS (`hts-users@sp.nitech.ac.jp`).

IV. Other applications

Although HTS has been developed to provide a research platform for HMM-based speech synthesis, it has also been used in various other ways, for example:

- Human motion synthesis [48], [49], [50],
- Face animation synthesis [51],
- Audio-visual synthesis [52], [53] and recognition [54],
- Acoustic-articulatory inversion mapping [55],
- Prosodic event recognition [56], [57],
- Mispronunciation detection in CALL systems [58],
- Very low bit-rate speech coder [59],
- Acoustic model adaptation for coded speech [60],
- Training data generation for ASR systems [61].
- Automatic evaluation of ASR systems [62].
- Online handwriting recognition [63].

We hope that HTS keeps contributing to progress in other research fields as well as speech synthesis.

V. Conclusions and future release plans

This paper described the recent developments of the HMM-based speech synthesis system (HTS). Internally, we have a number of variants of HTS, *e.g.*,

- Variational Bayes [64],
- Trajectory HMMs [65],
- Minimum generation error training [66],
- Shared tree construction [67],
- Eigenvoice [4],
- Multiple linear regression HMMs [68].

Hopefully, we can integrate valuable features of these variants into future HTS releases. On-line demonstrations which have been built using the above HTS version 2.1 features are available at [69].

VI. Acknowledgements

The authors thank Mr. Oliver Watts of the University of Edinburgh for his helpful comments on this paper. We are also grateful to all people who have contributed to the development of HTS. This work was partly supported by the Ministry of Education, Culture, Sports, Science and Technology (MEXT) e-Society project, the Hori information science promotion foundation, a Grant-in-Aid for Scientific Research (No. 1880009) by JSPS, and the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement 213845 (the EMIME Project).

References

[1] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura. Simultaneous modeling of spectrum, pitch and duration in HMM-based speech synthesis. In *Proc. Eurospeech*, pages 2347–2350, 1999.

[2] J. Yamagishi. *Average-Voice-Based Speech Synthesis*. PhD thesis, Tokyo Institute of Technology, 2006.

[3] T. Yoshimura, T. Masuko, K. Tokuda, T. Kobayashi, and T. Kitamura. Speaker interpolation for HMM-based speech synthesis system. *J. Acoust. Soc. Jpn. (E)*, 21(4):199–206, 2000.

[4] K. Shichiri, A. Sawabe, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura. Eigenvoices for HMM-based speech synthesis. In *Proc. ICSLP*, pages 1269–1272, 2002.

[5] K. Tokuda, H. Zen, J. Yamagishi, T. Masuko, S. Sako, T. Toda, A.W. Black, T. Nose, and K. Oura. The HMM-based speech synthesis system (HTS). http://hts.sp.nitech.ac.jp/.

[6] R. Sproat, J. Hirschberg, and D. Yarowsky. A corpus-based synthesizer. In *Proc. ICSLP*, pages 563–566, 1992.

[7] A. Hunt and A.W. Black. Unit selection in a concatenative speech synthesis system using a large speech database. In *Proc. ICASSP*, pages 373–376, 1996.

[8] N. Mizutani, K. Tokuda, and T. Kitamura. Concatenative speech synthesis based on HMM. In *Proc. Autumn Meeting of ASJ*, pages 241–242, 2002. (in Japanese).

[9] C. Allauzen, M. Mohri, and M. Riley. Statistical modeling for unit selection in speech synthesis. In *Proc. the 42nd meeting of the ACL*, 2004.

[10] S. Sakai and H. Shu. A probabilistic approach to unit selection for corpus-based speech synthesis. In *Proc. Interspeech (Eurospeech)*, pages 81–84, 2005.

[11] Z.-H. Ling and R.-H. Wang. HMM-based unit selection using frame sized speech segments. In *Proc. Interspeech (ICSLP)*, pages 2034–2037, 2006.

[12] Christian Weiss and Wolfgang Hess. Conditional random fields for hierarchical segment selection in text-to-speech synthesis. In *Proc. Interspeech (ICSLP)*, pages 1090–1093, 2006.

[13] K. Tokuda, T. Masuko, N. Miyazaki, and T. Kobayashi. Multi-space probability distribution HMM. *IEICE Trans. Inf. Syst.*, E85-D(3):455–464, Mar. 2002.

[14] K. Tokuda, T. Yoshimura, T. Masuko, T. Kobayashi, and T. Kitamura. Speech parameter generation algorithms for HMM-based speech synthesis. In *Proc. ICASSP*, pages 1315–1318, 2000.

[15] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura. Speaker interpolation in HMM-based speech synthesis system. In *Proc. Eurospeech*, pages 2523–2526, 1997.

[16] S. Young, G. Evermann, M. Gales, T. Hain, D. Kershaw, X.-Y. Liu, G. Moore, J. Odell, D. Ollason, D. Povey, V. Valtchev, and P. Woodland. *The Hidden Markov Model Toolkit (HTK) version 3.4*, 2006. http://htk.eng.cam.ac.uk/.

[17] http://www.opensource.org/licenses/bsd-license.php.

[18] K. Shinoda and T. Watanabe. MDL-based context-dependent subword modeling for speech recognition. *J. Acoust. Soc. Jpn.(E)*, 21(2):79–86, 2000.

[19] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura. Duration modeling for HMM-based speech synthesis. In *Proc. ICSLP*, pages 29–32, 1998.

[20] A.W. Black, P. Taylor, and R. Caley. The festival speech synthesis system. http://www.festvox.org/festival/.

[21] T. Yoshimura, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura. Incorporation of mixed excitation model and postfilter into HMM-based text-to-speech synthesis. *IEICE Trans. Inf. Syst. (Japanese Edition)*, J87-D-II(8):1563–1571, Aug. 2004.

[22] Galatea – An open-source toolkit for anthropomorphic spoken dialogue agent. http://hil.t.u-tokyo.ac.jp/galatea/.

[23] D. Huggins-Daines and A. Rudnicky. A constrained Baum-Welch algorithm for improved phoneme segmentation and efficient training. In *Proc. of Interspeech*, pages 1205–1208, 2006.

[24] K. Tokuda, H. Zen, and T. Kitamura. Reformulating the HMM as a trajectory model. In *Proc. Beyond HMM – Workshop on statistical modeling approach for speech recognition*, 2004.

[25] M.J.F. Gales. Maximum likelihood linear transformations for HMM-based speech recognition. *Comput. Speech Lang.*, 12(2):75–98, 1998.

[26] M.J.F. Gales. Semi-tied covariance matrices for hidden Markov models. *IEEE Trans. Speech Audio Process.*, 7(3):272–281, 1999.

[27] M.J.F. Gales. Maximum likelihood multiple projection schemes for hidden Markov models. *IEEE Trans. Speech Audio Process.*, 10(2):37–47, 2002.

[28] J. Yamagishi, M. Tachibana, T. Masuko, and T. Kobayashi. Speaking style adaptation using context clustering decision tree for HMM-based speech synthesis. In *Proc. ICASSP*, pages 5–8, 2004.

[29] J.L. Gauvain and C.-H. Lee. Maximum a posteriori estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Trans. Speech Audio Process.*, 2(2):291–298, 1994.

[30] L. Qin, Y.-J. Wu, Z.-H. Ling, and R.-H. Wang. Improving the performance of HMM-based voice conversion using context clustering decision tree and appropriate regression matrix. In *Proc. of Interspeech (ICSLP)*, pages 2250–2253, 2006.

[31] H. Zen, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura. A hidden semi-Markov model-based speech synthesis system. *IEICE Trans. Inf. Syst.*, E90-D(5):825–834, 2007.

[32] J. Yamagishi and T. Kobayashi. Average-voice-based speech synthesis using HSMM-based speaker adaptation and adaptive training. *IEICE Trans. Inf. Syst.*, E90-D(2):533–543, 2007.

[33] T. Toda and K. Tokuda. A speech parameter generation algorithm considering global variance for HMM-based speech synthesis. *IEICE Trans. Inf. Syst.*, E90-D(5):816–824, 2007.

[34] J. Yamagishi, T. Kobayashi, Y. Nakano, K. Ogata, and J. Isogai. Analysis of speaker adaptation algorihms for HMM-based speech synthesis and a constrained SMAPLR adaptation algorithm. *IEEE Trans. Audio Speech Lang. Process.*, 17(1):66–83, 2009.

[35] H. Kawahara, I. Masuda-Katsuse, and A.de Cheveigné. Restructuring speech representations using a pitch-adaptive time-frequency smoothing and an instantaneous-frequency-based F_0 extraction: possible role of a repetitive structure in sounds. *Speech Comm.*, 27:187–207, 1999.

[36] H. Zen, T. Toda, M. Nakamura, and K. Tokuda. Details of Nitech HMM-based speech synthesis system for the Blizzard Challenge 2005. *IEICE Trans. Inf. Syst.*, E90-D(1):325–333, Jan. 2007.

[37] H. Zen, T. Toda, and K. Tokuda. The Nitech-NAIST HMM-based speech synthesis system for the Blizzard Challenge 2006. In *Blizzard Challenge Workshop*, 2006.

[38] J. Yamagishi, T. Nose, H. Zen, Z.-H. Ling, T. Toda, K. Tokuda, S. King, and S. Renals. A robust speaker-adaptive HMM-based text-to-speech synthesis. *IEEE Trans. Audio Speech Lang. Process.*, 2009. (accept for publication).

[39] J. Yamagishi, H. Zen, Y.-J. Wu, T. Toda, and K. Tokuda. The HTS-2008 system: Yet another evaluation of the speaker-adaptive HMM-based speech synthesis system in the 2008 Blizzard Challenge, 2008.

[40] M.J. Russell and R.K. Moore. Explicit modeling of state occupancy in hidden Markov models for automatic speech recognition. In *Proc. ICASSP*, pages 5–8, 1985.

[41] M. Tachibana, J. Yamagishi, T. Masuko, and T. Kobayashi. A style adaptation technique for speech synthesis using HSMM and suprasegmental features. *IEICE Trans. Inf. Syst.*, E89-D(3):1092–1099, 2006.

[42] K. Oura, H. Zen, Y. Nankaku, A. Lee, and K. Tokuda. A fully consistent hidden semi-Markov model-based speech recognition system. *IEICE Trans. Inf. Syst.*, E91-D(11):2693–2700, 2008.

[43] O. Shiohan, Y. Myrvoll, and C.-H. Lee. Structural maximum a posteriori linear regression for fast HMM adaptation. *Comput. Speech Lang.*, 16(3):5–24, 2002.

[44] J. Dines, J. Yamagishi, and M. Gibson. Measuring the gap between HMM-based speech synthesis and speech recognition. In *2nd One Day Meeting on Unified Models for Speech Recognition and Synthesis*, 2009.

[45] H. Kawai, T. Toda, J. Yamagishi, T. Hirai, J. Ni, T. Nishizawa, M. Tsuzaki, and K. Tokuda. XIMERA: A concatenative speech synthesis system with large scale corpora. *IEICE Trans. Inf. Syst. (Japanese Edition)*, J89-D(12):2688–2698, Dec. 2006.

[46] M. Schröder and J. Trouvain. The German text-to-speech synthesis system MARY: A tool for research, development and teaching. *International Journal of Speech Technology*, 6:365–377, 2003.

[47] K. Tokuda, K. Oura, H. Zen, J. Yamagishi, T. Masuko, S. Sako, T. Toda, A.W. Black, and T. Nose. hts_engine. http://hts-engine.sourceforge.net/.

[48] K. Mori, Y. Nankaku, C. Miyajima, K. Tokuda, and T. Kitamura. Motion generation for Japanese finger language based on hidden Markov models. In *Proc. FIT*, volume 3, pages 569–570, 2005. (in Japanese).

[49] N. Niwase, J. Yamagishi, and T. Kobayashi. Human walking motion synthesis with desired pace and stride length based on HSMM. *IEICE Trans. Inf. Syst.*, E88-D(11):2492–2499, 2005.

[50] G. Hofer, H. Shimodaira, and J. Yamagishi. Speech driven head motion synthesis based on a trajectory model. In *Proc. SIGGRAPH*, 2007.

[51] O. Govokhina, G. Bailly, G. Breton, and P. Bagshaw. TDA: a new trainable trajectory formation system for facial animation. In *Proc. Interspeech*, pages 1274–1247, 2006.

[52] M. Tamura, S. Kondo, T. Masuko, and T. Kobayashi. Text-to-audio-visual speech synthesis based on parameter generation from HMM. In *Proc. Eurospeech*, pages 959–962, 1999.

[53] S. Sako, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura. HMM-based text-to-audio-visual speech synthesis. In *Proc. ICSLP*, pages 25–28, 2000.

[54] T. Ishikawa, Y. Sawada, H. Zen, Y. Nankaku, C. Miyajima, K. Tokuda, and T. Kitamura. Audio-visual large vocabulary continuous speech recognition based on early integration. In *Proc. FIT*, pages 203–204, 2002. (in Japanese).

[55] K. Richmond. A trajectory mixture density network for the acoustic-articulatory inversion mapping. In *Proc. of Interspeech*, pages 577–580, 2006.

[56] K. Emoto, H. Zen, K. Tokuda, and T. Kitamura. Accent type recognition for automatic prosodic labeling. In *Proc. Autumn Meeting of ASJ*, volume I, pages 225–226, 2003. (in Japanese).

[57] H.-L. Wang, Y. Qian, F.K. Soong, J.-L. Zhou, and J.-Q. Han. A multi-space distribution (MSD) approach to speech recognition of tonal languages. In *Proc. of Interspeech*, pages 125–128, 2006.

[58] L. Zhang, C. Huang, M. Chu, F. Soong, X. Zhang, and Y. Chen. Automatic detection of tone mispronunciation in Mandarin. In *Proc. ISCSLP*, volume I, pages 590–601, 2006.

[59] T. Hoshiya, S. Sako, H. Zen, K. Tokuda, T. Masuko, T. Kobayashi, and T. Kitamura. Improving the performance of HMM-based very low bitrate speech coding. In *Proc. ICASSP*, volume 1, pages 800–803, 2003.

[60] K. Tanaka, S. Kuroiwa, S. Tsuge, and F. Ren. An acoustic model adaptation using HMM-based speech synthesis. In *Proc. NLPKE*, volume 1, pages 368–373, 2003.

[61] M. Ishihara, C. Miyajima, N. Kitaoka, K. Itou, and K. Takeda. An approach for training acoustic models based on the vocabulary of the target speech recognition task. In *Proc. Spring Meeting of ASJ*, pages 153–154, 2007. (in Japanese).

[62] R. Terashima, T. Yoshimura, T. Wakita, K. Tokuda, and T. Kitamura. An evaluation method of ASR performance by HMM-based speech synthesis. In *Proc. Spring Meeting of ASJ*, pages 159–160, 2003. (in Japanese).

[63] L. Ma, Y.-J. Wu, P. Liu, and F. Soong. A MSD-HMM approach to pen trajectory modeling for online handwriting recognition. In *Proc. ICDAR*, pages 128–132, 2007.

[64] Y. Nankaku, H. Zen, K. Tokuda, T. Kitamura, and T. Masuko. A Bayesian approach to HMM-based speech synthesis. In *Tech. rep. of IEICE*, volume 103, pages 19–24, 2003. (in Japanese).

[65] H. Zen, K. Tokuda, and T. Kitamura. Reformulating the HMM as a trajectory model by imposing explicit relationships between static and dynamic feature vector sequences. *Comput. Speech Lang.*, 21(1):153–173, 2006.

[66] Y.-J. Wu and R.-H. Wang. Minimum generation error training for HMM-based speech synthesis. In *Proc. ICASSP*, pages 89–92, 2006.

[67] J. Yamagishi, M. Tamura, T. Masuko, K. Tokuda, and T. Kobayashi. A context clustering technique for average voice models. *IEICE Trans. Inf. Syst.*, E86-D(3):534–542, 2003.

[68] T. Nose, J. Yamagishi, T. Masuko, and T. Kobayashi. A style control technique for HMM-based expressive speech synthesis. *IEICE Trans. Inf. Syst.*, E90-D(9):1406–1413, 2007.

[69] http://www.cstr.ed.ac.uk/projects/festival/onlinedemo.html.

Appendix A
Modifications in Model Definition

In HTS, the HTK HMM definition (please see HTKBook [16] Chapter 7) has been modified to support MSD [13], stream-level tying, and adaptation of multi-stream HMMs. This section gives a brief description of these modifications.

First, <MSDInfo> have been added to global options of the HTK HMM definition language. The arguments to the <MSDInfo> option are the number of streams (default 1) and then for each stream, 0 (non-MSD stream) or 1 (MSD stream) of that stream. The full set of global options in HTS is given below.

```
globalOpts  =  option { option }
option      =  <HmmSetId> string |
               <StreamInfo> short { short } |
               <MSDInfo> short { short } |
               <VecSize> short |
```

```
                  <ProjSize> short |
                  <InputXform> inputXform |
                  <ParentXform> ~a macro |
                  covkind |
                  durkind |
                  parmkind
```

Second, specification of the number of mixture components has been modified to support stream-level tying structures as follows:

```
HTK                            HTS
 <State> 2                      <State> 2
  <NumMixes> 1 2
  <SWeights> 2 0.9 1.1           <SWeights> 2 0.9 1.1
  <Stream> 1                     <Stream> 1
                                    <NumMixes> 1
     <Mixture> 1 1.0                <Mixture> 1 1.0
      <Mean> 4                       <Mean> 4
        0.3 0.2 0.1 0.0                0.3 0.2 0.1 0.0
      <Variance> 4                   <Variance> 4
        0.5 0.4 0.3 0.2                0.5 0.4 0.3 0.2
    <Stream> 2                     <Stream> 2
                                    <NumMixes> 2
     <Mixture> 1 0.4                <Mixture> 1 0.4
      <Mean> 2                       <Mean> 2
        1.0 2.0                        1.0 2.0
      <Variance> 2                   <Variance> 2
        4.0 8.0                        4.0 8.0
     <Mixture> 2 0.6                <Mixture> 2 0.6
      <Mean> 2                       <Mean> 2
        2.0 9.0                        2.0 9.0
      <Variance> 2                  <Variance> 2
        3.0 6.0                        3.0 6.0
```

As can been see, <NumMixes> has been moved from state-level to stream-level. This modification enables us to include the number of mixture components in the stream-level macro. Based on this implementation, a stream-level macro was added. The various distinct points in the hierarchy of HMM parameters which can be tied in HTS is as follows:

~s	shared state distribution
~p	shared stream
~m	shared Gaussian mixture component
~u	shared mean vector
~v	shared diagonal variance vector
~i	shared inverse full covariance matrix
~c	shared Cholesky *U* matrix
~x	shared arbitrary transform matrix
~t	shared transition matrix
~d	shared duration parameters
~w	shared stream weight vector

Note that the ~p macro is used by the HMM editor HHEd for building tied mixture systems in the original HTK macro definition.

The resultant state definition of in the modified HTK HMM definition language is as follows:

```
state      = <State> short stateinfo
stateinfo  = ~s macro |
             [ weights ] stream { stream } [ duration ]
macro      = string
weights    = ~w macro | <SWeights> short vector
vector     = float { float }
stream     = [ <Stream> short ] streaminfo
streaminfo = ~p macro | [ <Stream> short ] [mixes]
             (mixture { mixture } | tmixpdf | discpdf)
mixes      = <NumMixes> short {short}
tmixpdf    = <TMix> macro weightList
weightList = repShort { repShort }
repShort   = short [ * char ]
discpdf    = <DProb> weightList
mixture    = [ <Mixture> short float ] mixpdf
mixpdf     = ~m macro | mean cov [ <GConst> float ]
mean       = ~u macro | <Mean> short vector
cov        = var | inv | xform
var        = ~v macro | <Variance> short vector
inv        = ~i macro |
             (<InvCovar> | <LLTCovar>) short tmatrix
xform      = ~x macro | <Xform> short short matrix
matrix     = float {float}
tmatrix    = matrix
```

It should be noted that <Stream> can be specified in both stream and streaminfo. This is because <Stream> in the ~p macro is essential to specify the stream index of this macro. This stream index information is used in various HTS functions to check stream consistency.

Third, to support multi-stream HMM adaptation, the HTK HMM definition language for baseclasses is modified. A baseclass is defined as

```
baseClass  = ~b macro baseopts classes
baseopts   = <MMFIdMask> string <Parameters> baseKind
             [<StreamInfo>] <NumClasses> int
StreamInfo = short { short } |
baseKind   = MIXBASE | MEANBASE | COVBASE
classes    = <Class> int itemlist { classes }
```

where <StreamInfo> is optionally added to specify the stream structure.

Appendix B
Added Configuration Variables

A number of configuration variables have been added to HTK to control new functions implemented in HTS. Their names, default values, and brief descriptions are as follows:

Module	Name	Default	Description
HAdapt	SAVEFULLC	F	Save transformed model set in full covariance form
	USESMAP	F	Use structural MAP criterion [34]
	SMAPSIGMA	1.0	Prior parameter for SMAP criterion
	SAVEALLSMAPXFORM	T	Save all (unnecessary) linear transforms estimated in SMAPLR/CSMAPLR
	BANDWIDTH		Bandwidth of transformation matrices [30]
	DURUSEBIAS	F	Specify a bias for linear transforms
	DURSPLITTHRESH	1000.0	Minimum occupancy to generate a transform for state-duration model set

Module	Name	Default	Description
	DURTRANSKIND	MLLRMEAN	Transformation kind
	DURBLOCKSIZE	full	Block structure of transform for state-duration model set
	DURBANDWIDTH		Bandwidth of transformation matrices for state-duration model set
	DURBASECLASS	global	Macroname of baseclass for state-duration model set
	DURREGTREE		Macroname of regression tree for state-duration model set
	DURADAPTKIND	BASE	Use regression tree or base classes to adapt state-duration model set
HFB	MAXSTDDEVCOEF	10	Maximum duration to be evaluated
	MINDUR	5	Minimum duration to be evaluated
HMap	APPLYVFLOOR	T	Apply variance floor to model set
HGen	MAXEMITER	20	Maximum # of EM iterations
	EMEPSILON	1.0E-4	Convergence factor for EM iteration
	RNDPARMEAN	0.0	Mean of Gaussian noise for random generation [24]
	RNDPARVAR	1.0	Variance of Gaussian noise for random generation
	USEGV	F	Use speech parameter generation algorithm considering GV [33]
	CDGV	F	Use context-dependent GV model set
	LOGGV	F	Use logarithmic GV instead of linear GV
	MAXGVITER	F	Max iterations in the speech parameter generation considering GV
	GVEPSILON	1.0E-4	Convergence factor for GV iteration
	MINEUCNORM	1.0E-2	Minimum Euclid norm of a gradient vector

Module	Name	Default	Description
	STEPINIT	1.0	Initial step size
	STEPDEC	0.5	Step size deceleration factor
	STEPINC	1.2	Step size acceleration factor
	HMMWEIGHT	1.0	Weight for HMM output prob
	GVWEIGHT	1.0	Weight for GV output prob
	OPTKIND	NEWTON	Optimization method
	RNDFLAGS		Random generation flag
	GVMODELMMF		GV MMF file
	GVHMMLIST		GV model list
	GVMODELDIR		Dir containing GV models
	GVMODELEXT		Ext to be used with above Dir
	GVOFFMODEL		Model names to be excluded from GV calculation
HModel	IGNOREVALUE	-1.0E+10	Ignore value to indicate zero-dimensional space in multi-space probability distribution
HCompV	NSHOWELEM	12	# of vector elements to be shown
	VFLOORSCALE	0.0	variance flooring scale
	VFLOORSCALESTR		variance flooring scale vector for streams
HERest	APPLYVFLOOR	T	Apply variance floor to model set
	DURMINVAR	0.0	Minimum variance floor for state-duration model set
	APPLYDURVARFLOOR	T	Apply variance floor to state-duration model set
	DURMAPTAU	0.0	MAP tau for state-duration model set [29]
	ALIGNDURMMF		State-duration MMF file for alignment (2-model reest)
	ALIGNDURLIST		State-duration model list for alignment (2-model reest)
	ALIGNDURDIR		Dir containing state-duration models for alignment (2-model reest)

Module	Name	Default	Description
	ALIGNDUREXT		Ext to be used with above Dir (2-model reest)
	DURINXFORMMASK		Input transform mask for state-duration model set (default output transform mask)
	DURPAXFORMMASK		Parent transform mask for state-duration model set (default output parent mask)
HHEd	USEPATTERN	F	Use pattern instead of base phone for tree-based clustering
	SINGLETREE	F	Construct single tree for each state position
	APPLYMDL	F	Use the MDL criterion for tree-based clustering [18]
	IGNORESTRW	F	Ignore stream weight in tree-based clustering
	REDUCEMEM	F	Use reduced memory implementation of tree-based clustering
	MINVAR	1.0E-6	Minimum variance floor for model set
	MDLFACTOR	1.0	Factor to control the model complexity term in the MDL criterion
	MINLEAFOCC	0.0	Minimum occupancy count in each leaf node
	MINMIXOCC	0.0	Minimum occupancy count in each mixture component
	SHRINKOCCTHRESH		Minimum occupancy count in decision trees shrinking
HMGenS	SAVEBINARY	F	Save generated parameters in binary
	OUTPDF	F	Output pdf sequences
	PARMGENTYPE	0	Type of parameter generation algorithm [14]
	MODELALIGN	F	Use model-level alignments given from label files to determine model-level durations

Module	Name	Default	Description
	STATEALIGN	F	Use state-level alignments given from label files to determine state-level durations
	USEALIGN	F	Use model-level alignments to prune EM-based parameter generation algorithm
	USEHMMFB	F	Do not use state-duration models in the EM-based parameter generation algorithm
	INXFORMMASK		Input transform mask
	PAXFORMMASK		Parent transform mask
	PDFSTRSIZE		# of PdfStreams
	PDFSTRORDER		Size of static feature in each PdfStream
	PDFSTREXT		Ext to be used for generated parameters from each PdfStream
	WINEXT		Ext to be used for window coefficients file
	WINDIR		Dir containing window coefficient files
	WINFN		Name of window coefficient files

Other configuration variables in HTK can also be used with HTS. Please refer to HTKBook [16] Chapter 18 for others.

Appendix C
Added Command-Line Options

Various new command-line options have also been added to HTK tools. They are listed as follows:

HInit

```
Option
 -g      Ignore outlier vector in MSD
```

HRest

```
Option
 -g s    output duration model to file s
 -o fn   Store new hmm def in fn (name only)
```

HERest

```
Option
 -b      use an input linear transform for dur models
 -f s    extension for new duration model files
 -g s    output duration model to file s
 -n s    dir to find duration model definitions
 -q s    save all xforms for duration to TMF file s
 -u tmvwapd update t)rans m)eans v)ars w)ghts
                  a)daptation xform p)rior used
                  s)emi-tied xform
                  d) switch to duration model
```

```
                        update flag
  -y s     extension for duration model files
  -N mmf   load duration macro file mmf
  -R dir   dir to write duration macro files
  -W s [s] set dir for duration parent xform to s
           and optional extension
  -Y s [s] set dir for duration input xform to s
           and optional extension
  -Z s [s] set dir for duration output xform to s
```

HHEd

```
 Option
  -a f     factor to control the second term in MDL
  -i       ignore stream weight
  -m       apply MDL principle for clustering
  -p       use pattern instead of base phone
  -r       reduce memory usage on clustering
  -s       construct single tree
  -v f     Set minimum variance to f
```

HMGenS

```
 Option
  -a       Use an input linear transform for HMMs
  -b       Use an input linear transform for dur models
  -c n     type of parameter generation algorithm
            0: both mix and state sequences are given
            1: state sequence is given,
               but mix sequence is hidden
            2: both state and mix sequences are hidden
  -d s     dir to find hmm definitions
  -e       use model alignment from label for pruning
  -f f     frame shift in 100 ns
  -g f     Mixture pruning threshold
  -h s [s] set speaker name pattern to s,
           optionally set parent patterns
  -m       use model alignment for duration
  -n s     dir to find duration model definitions
  -p       output pdf sequences
  -r f     speaking rate factor (f<1: fast  f>1: slow)
  -s       use state alignment for duration
  -t f [i l] set pruning to f [inc limit]
  -v f     threshold for switching spaces for MSD
  -x s     extension for hmm files
  -y s     extension for duration model files
  -E s [s] set dir for parent xform to s
           and optional extension
  -G fmt   Set source label format to fmt
  -H mmf   Load HMM macro file mmf
  -I mlf   Load master label file mlf
  -J s [s] set dir for input xform to s
           and optional extension
  -L dir   Set input label (or net) dir
  -M dir   Dir to write HMM macro files
  -N mmf   Load duration macro file mmf
  -S f     Set script file to f
  -T N     Set trace flags to N
  -V       Print version information
  -W s [s] set dir for duration parent xform to s
           and optional extension
  -X ext   Set input label (or net) file ext
  -Y s [s] set dir for duration input xform to s
           and optional extension
```

Please also refer to HTKBook [16] Chapter 17 for other command-line options.

Appendix D
Added Commands and Modifications in HHEd

A. Added commands

Some HHEd commands have been added in HTS. They are as follows:

```
AX filename       - Set the Adapt XForm to filename
CM directory      - Convert models to pdf for speech
                    synthesizer
CT directory      - Convert trees/questions for
                    speech synthesizer
DM type macname   - Delete macro from model-set
DR id             - Convert decision trees to a
                    regression tree
DV                - Convert full covariance to
                    diagonal variances
IT filename       - Clustering while imposing loaded
                    tree structure.  If any empty
                    leaf nodes exist, loaded trees
                    are pruned and then saved to
                    filename
IX filename       - Set the Input Xform to filename
JM hmmFile ilist  - Join Models on stream or state
                    level
PX filename       - Set the Parent Xform to filename
// comment        - Comment line (ignored)
```

B. Item listing

In many HHEd commands, we are required to specify item lists to specify a set of items to be processed. In HTS, item list specification has been modified to specify stream-level items.

```
itemList   = "{" itemSet { "," itemSet } "}"
itemSet    = hmmName . ["transP" | "state" state ]
hmmName    = ident | identList
identList  = "(" ident { "," ident } ")"
ident      = < char | metachar >
metachar   = "?" | "⋆"
state      = index ["." stateComp ]
index      = "[" intRange { "," intRange } "]"
intRange   = integer [ "-" integer ]
stateComp  = "dur" | "weights" | stream
stream     = [ " stream" index ] [ ".mix" mix ]
mix        = index [ "." ( "mean" | "cov" ) ]
```

For example,

```
TI str1 {*.state[2].stream[1]}
```

denotes tying streams in state 2 of all phonemes.

C. Mix-up

In the HHEd command MU, HTS additionally supports additive and multiplicative mixture incrementation. For example,

```
MU  6 {*.state[3].mix}
MU +6 {*.state[3].mix}
MU *6 {*.state[3].mix}
```

if the the mixture components per state is 2, the first command increases the numbers of mixtures in state 3 of all phonemes of aa to 6, the second one increases them to 8, and the last one increased them to 12.