# Automatic generation of synthesis units for trainable text-to-speech systems

**Conference Paper** *in* Acoustics, Speech, and Signal Processing, 1988. ICASSP-88., 1988 International Conference on · June 1998

DOI: 10.1109/ICASSP.1998.674425 · Source: IEEE Xplore

Hsiao-Wuen Hon
Microsoft
**89** PUBLICATIONS **6,929** CITATIONS
SEE PROFILE

Xuedong Huang
Microsoft
**124** PUBLICATIONS **9,055** CITATIONS
SEE PROFILE

Jinzhen Liu
Tiangong University
**376** PUBLICATIONS **12,375** CITATIONS
SEE PROFILE

# AUTOMATIC GENERATION OF SYNTHESIS UNITS FOR TRAINABLE TEXT-TO-SPEECH SYSTEMS

*H. Hon, A. Acero, X. Huang, J. Liu, and M. Plumpe*

Microsoft Research
One Microsoft Way
Redmond, Washington 98052, USA

## ABSTRACT

Whistler Text-to-Speech engine was designed so that we can automatically construct the model parameters from training data. This paper will describe in detail the design issues of constructing the synthesis unit inventory automatically from speech databases. The automatic process includes (1) determining the scaleable synthesis unit which can reflect spectral variations of different allophones; (2) segmenting the recording sentences into phonetic segments; (3) select good instances for each synthesis unit to generate best synthesis sentence during run time. These processes are all derived through the use of probabilistic learning methods which are aimed at the same optimization criteria. Through this automatic unit generation, Whistler can automatically produce synthetic speech that sounds very natural and resembles the acoustic characteristics of the original speaker.

## 1. INTRODUCTION

In [4][7], we have presented Whistler: Microsoft's Trainable Text-to-Speech (TTS) System. In contrast to most other TTS systems (including both formant and concatenative synthesizers) [1][2][12] which require human experts to hand-craft and fine-tune the synthesis units (or unit parameters), Whistler uses an automatic procedure to configure and generate the scaleable synthesis units directly from any speech database. In this paper, we will describe in detail the design issues and improvements we made to the construction the synthesis unit inventory automatically from speech databases.

Whistler is based upon a concatenative synthesizer whose unit inventory is generated by cutting speech segments from a database recorded by a target speaker [4][7][11]. There are typical three phases in the process of building a unit inventory:

1. Determine the synthesis units and derive the conversion between a phoneme string and a unit string.
2. Segmentation of each unit from spoken speech.
3. Selection of one (or a few) good unit instance when many are available in the corpus.

The synthesis unit needs to be scaleable and reflect spectral variations of different allophones, so one can build the optimal TTS systems under various resource requirements. Diphones [5] which are the most popular synthesis unit employed today, however, fail to meet both criteria. Whistler uses the decision-tree clustered phone-based unit [9][4][7] as the synthesis unit. The decision-tree clustered phone-based unit not only can effectively model contextual variation, but also provides an ideal scaleable mechanism for tradeoff between voice quality and memory size, generality and specificity.

Manual segmentation and unit selection phases are typically very labor-intensive for concatenative synthesizers because it involves subjective judgement for thousands of synthesis units. They are often based on a trial and error process, which doesn't usually address the potential distortion at any concatenation point. Moreover, it is very difficult to optimize the segmentation and unit selection phases with the choice of synthesis units. Whistler uses a probability framework for both segmentation and unit selection phases, which is consistent with the decision-tree clustered phone-based unit. This unified unit generation procedure will yield synthesis sentence with optimal probability and hopefully optimal quality.

This paper is organized as follows. In Section 2 we present Whistler's synthesis unit - the decision-tree clustered phone-based unit. In Section 3 we then discuss how to segment the speech recording into units. In Section 4 we describe the process of automatic unit selection, including multiple instance case. Finally we summarize our major findings and outline our future work.

## 2. SYNTHESIS UNIT

### 2.1 Diphone

Diphone [5][12], which contains the transitions between two phones, has been chosen as the synthesis unit for concatenative synthesizers. There are about 1500 to 2000 diphones in English, and the diphone mapping for a phoneme string is straightforward. The word HELLO /hh ah l ow/ can be mapped into the diphone sequence: /sil-hh/, /hh-ah/, /ah-l/, /l-ow/, /ow-sil/. While diphones retain the transitional information, there can be large distortions due to the difference in spectra between the stationary parts of two units obtained from different contexts. For example, there is no guarantee the spectra of /ah-l/ and /l-ow/ will match at the junction since /ah-l/ could have a very different right context than /ow/ or /l-ow/ could have a very different left context than /ah/. As evidenced in the diphone-based systems, naturalness of synthetic speech can be sometimes significantly hampered by the context mismatch between certain diphone units.

### 2.2 Senone

To achieve a more natural voice quality, one must take more contexts into account, going beyond diphones. However, simply modeling *triphones* [9] (a phone with the immediate left and right contexts) already requires more than 10,000 units for English. Fortunately, effective clustering of similar contexts modeled in a sub-phonetic level, to allow flexible memory-quality compromise, has been well studied in the speech recognition community [6]. A senone [6][3] is a context-dependent sub-phonetic unit which is equivalent to a HMM state in a triphone. Senones can usually model spectral variation in a fine sub-phonetic level. However, they introduce many more junctions than a diphone system. Those excessive junctions create potential quality degradation for synthesis systems.

### 2.3 Decision-Tree Clustered Phone-Based Unit

To achieve the same rich context modeling as senones while not introducing more junctions, the decision-tree clustered phone-based unit is used as our synthesis unit. First we construct the inventory of context-dependent phone units. The context-dependent phones could be triphones, quinphones (a phone with two immediate left and right contexts), stress-sensitive phones, word-dependent phones (the same phone in particular words are considered distinct context-dependent phones), or a combination of the above. In order to reduce the total number of synthesis units down to a manageable number while incorporating more contexts, we need to utilize clustering decision trees [9][8] to cluster similar context-dependent phone units together.

The decision tree is a binary tree with a categorical question associated with each branching node. Traversing the trees is equivalent to following the answers for the branching nodes from root to leaves, which determines the clusters for similar context-dependent phones. The decision trees are generated automatically from the analysis database to obtain minimum within-unit distortion (or entropy) for each split. The distortion measure could be obtained through the correspondent HMM's parameters [9]. Therefore, one needs to acquire a large inventory of context-dependent phone HMM's with decent coverage of the contexts one wishes to model. This criterion will assure minimum spectral variation for the context-dependent phones within each cluster. Therefore the context-dependent phone unit can be substituted by any other unit within the same cluster. Since these clustering decision trees should be consistent across different speakers, the use of a large speaker-independent database (like the DARPA's WSJ database) instead of a limited speaker-dependent database allows us to model more contexts (like quinphones) as well as to build deeper trees to generate high-quality TTS voices.

Unlike senones, our phone-based units require no more junctions than diphone-based systems and yet assure consistency within each unit to achieve better concatenation through rich context modeling. Moreover, the use of decision trees will generalize to contexts not seen in the training data, based on phonetic categories of neighboring contexts. Finally, the structure of the decision tree allows maximum scalability. One can decide how many leaves (synthesis units) to use based on resource constraints.

## 3. AUTOMATIC SEGMENTATION

To segment the speech corpus into sequences of phone-based units, we used the automatic Viterbi alignment developed in Whisper [6]. We used Whisper to align the input waveform with phonetic symbols that are associated with correspondent HMMs. HMMs are trained from the speaker-dependent data of the target speaker. We observed that 4% of the sentences in the database contain some gross segmentation errors (larger than 20 ms) when compared to hand labeled data, and most of the errors were caused by incorrect transcriptions. Nevertheless, good context coverage and consistent segmentation by HMMs typically overcomes the drawback of an imperfect automatic segmentation when compared to manual segmentation.

### 3.1 Non-uniform HMM Topologies

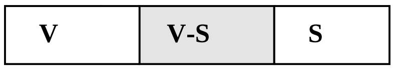

**Figure 1**. Diagram showing a vowel to stop concatenation. The shaded area represents vowel-stop transition.

During error analysis, the biggest discrepancy between automatic segmentation and hand-labeled segmentation is in vowel-stop, stop-vowel, fricative-vowel, and vowel-fricative regions. The spectrum change of stops and fricatives is typically much smaller when compared to that of vowels. Since Whisper [6] use a uniform topology for all units, the beginning and ending states of a stop or fricative tend to model the transitions from vowels or to vowels in order to achieve HMM optimization. This results in V-S (vowel-stop) parts in Figure 1 usually segmented together with S instead of V; where manual segmentation often associates the V-S part with V because the V-S has already enough vowel characteristics. From synthesis point of view, the V-S transitions is better tied to V than stop S to assure smooth transition since the spectral mismatch between V and V-S is more perceptive than that of V-S and S. Therefore we decide to use simple topologies for stop and fricative units (1-state HMM for fricative units and 2-state HMM for stop units) to alter the segmentation. As expected, the new segmentation does improve the concatenation quality for stop/fricative-vowel and vowel-stop/fricative transitions.

Using simple topologies for stop/fricative units for decision tree generation will also result in trees with fewer leaves allocated for stop/fricative units and more leaves allocated for non-stop/fricative units. This change could potentially improve the quality because more resources are allocated for units with high spectral variation. It is also possible that perceptual degradation is not linear to the spectral mismatch. Incorporating more perception-based features into our probability framework could be a important direction to further improve TTS voice quality.

### 3.2 Pitch-Synchronized Segments

The segmentation derived from HMM's Viterbi alignment is based on uniform time frames rather than pitch-synchronized frames (in sync with exact epoch boundaries). However, pitch-synchronized units are absolutely necessary for concatenative TTS in order to do arbitrary concatenation and pitch/duration stretching. To obtain pitch-synchronized segments, one needs to perform pitch/epoch estimation [4][7] on recording speech in addition to HMM segmentation. Figure 2 illustrates HMM Viterbi alignment segments two units at time m which is between time $T_i$ correspondent to epoch i and time $T_{i+1}$ correspondent to next epoch i+1. The epoch points are derived by a pitch/epoch estimator. Now the task is to realign the boundary m to either $T_i$ or $T_{i+1}$, so the units will be pitch-synchronized. One solution is to adjust the segmentation boundary m to the nearest epoch point. A better one is to adjust the new boundary to either the immediate left or right epoch point so that the pitch changes within each unit is minimized. In other words, if absolute pitch difference $|P(F_{i+1}) - P(F_i)|$ is greater than $|P(F_i) - P(F_{i-1})|$, the segmentation boundary m will be adjusted to $T_i$; otherwise adjusted to $T_{i+1}$Moreover, for unvoiced frame, the P(F) will take the value of negative infinity. Thus the above algorithm will segment at voiced/unvoiced boundary for voiced/unvoiced transition.

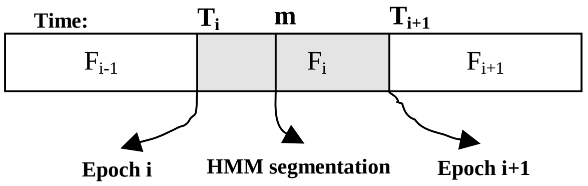

**Figure 2**. Diagram showing how to align HMM boundaries to pitch-synchronized boundaries

## 4. UNIT SELECTION

As described in Section 2, every leaf in the decision trees represent a coherent set where any particular instance of a particular context-dependent phone unit could represent the entire cluster. However, it is necessary to have a comprehensive unit selection procedure to (1) discard bad units caused by mispronunciation or bad segmentation; (2) select good unit instances when many are available for better concatenation and prosody stretching.

When reliable statistics can be obtained from enough instances, the bad unit instances should reside on the outlier of the distribution. To discard the bad units residing in the outlier, we first compute unit statistics for duration, amplitude, pitch, and voiced frame percentage (how many percentage of frames are voiced within a unit), and remove those unit instances residing more than one standard deviation away from the unit mean. In addition to pruning the bad instances, keeping the selected instances close to the cluster mean of duration, amplitude and pitch is important because prosody modification during synthesis of such units should on the average require less stretching which will lead to better synthesis quality. For the next two subsections (single-instance and multiple-instance systems), the selection procedure only consider surviving candidates after outlier pruning.

### 4.1 Single Instance System

At minimum, we need to select one unit instance for each decision-tree cluster to be able to synthesize any text. The advantage of a single-instance system is its simplicity and compact size. The disadvantage is that the instance chosen needs to be very robust for all-context concatenations and arbitrary prosody stretching. We have demonstrated that a good single-instance TTS system can be produced with enough (around 3,000) decision-tree leaf clusters.

The question now is which unit instance one should choose after discarding the outliers. In order to solve this question, let's start with a simple experiment. By randomly choosing one unit instance for each decision-tree leaf cluster from the surviving candidate pool, we get an acceptable but not so great system. The system sometimes generates large glitches for some concatenations. This experiment shows that not all surviving candidates are good and we definitely need some criterion for choosing good unit instances.

Since our unit generation is based on a unified HMM framework, HMM matching score alignment (normalized by the length of the segments) generated by Viterbi indicates how well the individual instance matches the distribution of the decision tree leaf cluster. It thus should be a good measurement of how well the instances can represent the cluster. By choosing the unit instance with highest HMM score to represent the cluster, we are able to achieve a very high concatenation quality. In another experiment to further verify this selection criterion, the instance with the worst HMM score was chosen and the resulting TTS quality was dismal.

### 4.2 Multiple Instance System

The motivation for using a multiple-instance system is obvious because the system can dynamically select the best unit instances to minimize the concatenation distortion and prosody stretching. To support this statement, we designed an experiment to randomly select the unit instance with the 10th highest HMM scores. The resulting TTS quality is almost as good as the best single-instance system described in Section 4.2. This result demonstrates that instances with good enough HMM scores are comparable to the instance with highest HMM score.

Inspired by the supporting experiment, a small number of unit instances can be pre-selected through the use of an objective function. The objective function could be based on a combination of HMM scores, variety of contexts, and variety of different prosody attributes. At synthesis time, the synthesizer will compare all instances of the required synthesis units, selecting the instances optimizing the joint objective function. Hopefully this will translated into the best overall speech quality. By increasing the number of instances in the database, we can make a trade-off between speech quality and database size and CPU utilization.

The first multiple instance system was similar in concept to ATR's system as described in [13]. The original database consisted of approximately 32,000 unit instances that contained the instances with the top-10 highest HMM scores after discarding outliers. A cost function was developed to indicate the synthesis quality. One key feature of the cost function is that for two units occurred in sequence in the training database, synthesizing them in sequence incurs a zero cost. This encourages the selection of sequential strings of unit instances, a flexible form of long units. If two units were not sequential in the original speech, their synthesis cost was based on the spectral distortion at the boundary, the expected spectral distortion at the boundary calculated from the original speech, and the type of unit (e.g. larger spectral mismatch is tolerated for fricatives than vowels). If large memory resources and a large speech database are available, it is possible to use a multiple-instance system to construct long-units for frequent words and phrases that will undoubtedly achieve optimal concatenation quality. Since the units making up this long unit can also be used individually for other contexts, the multiple-instance framework can be viewed as an extension of the simple-instance system with additional instances kept for optimal concatenation for important and/or frequent contexts.

To reduce the size of the database, a greedy algorithm was used to minimize the synthesis cost for the training sentences. Starting with all the instances (and hence a zero cost), the unit which caused the least increase in total synthesis cost was removed. This is repeated until the desired database size is achieved. Database sizes down to about 10,000 instances, or on average three instances per unit, have quality that is similar to the original quality. Below this, the quality degrades more quickly. We feel that the reason for this is that the cost function and statistics are good at identifying bad units, but are not as successful as in picking the best units. Once all of the clearly bad units are discarded, the algorithm is less successful at identifying which units to keep. For similar reasoning, starting with a minimal system and adding unit instances was not successful.

Another way to reduce the size of the database is to partition the instance distribution for each cluster into several regions, which can hopefully have good coverage for all the real concatenation conditions. To cluster the data, we use the Line Spectral Pair representation of the filters for the first and last frame of the instances. The instance representing the mean of the cluster is chosen from a subset of all instances. At run-time, dynamic programming can be applied to find the instance sequence that has minimum sum of juncture Line Spectral Pairs. One possible extension will be including the prosody attributes (duration, pitch, and amplitude) for partition criteria. Thus at run-time we could dynamically find the instance sequence minimizing a joint cost function which is the combination of the sum of juncture spectral mismatch and stretching cost (the amount of modification required to stretch it during synthesis).

One might increase the number of decision tree leaf clusters to simulate multiple-instance system without actually using multiple-instance system. However, it won't be able to reduce prosody stretching since our decision trees has yet taken prosody attributes into account. Moreover, more leaf cluster means fewer unit instances per cluster. This results in less reliable statistics which could cause ineffectiveness for outlier pruning. Therefore, how to balance the number of decision tree leaf clusters and instances per cluster when building a scalable TTS system will become an important research topic for multiple-instance TTS systems.

## 5. Conclusion

Our data-driven unit generation takes advantage of the analysis of large amounts of natural speech and avoids the over-generation problem found in traditional hand-tuned systems. We have demonstrated that a high quality voice inventory can be automatically configured and built, so the resulting TTS voice is not only natural but also similar to the original donor speaker. The voice inventory is also highly scaleable, and can tradeoff between voice quality and memory size and CPU utilization. A version of the Whistler TTS system was described in [4] and can be downloaded from Microsoft Research's web site [10].

Future work will need to focus on how to further unify the whole process of unit generation, including unit configuration, automatic segmentation and unit selection. Some obvious directions include incorporating duration, pitch information in decision tree generation; unifying the criteria for unit configuration, segmentation, unit selection and smoothing.

## 6. REFERENCES

[1] Allen J., Hunnicutt S., and Klatt D. *From text to speech: the MITalk system*. MIT Press, Cambridge, Massachusetts, 1987.

[2] Bailly G. and Benoit C., editors. *Talking Machines: Theories, Models, and Designs*. Elsevier Science, 1992.

[3] Donovan R.E. and Woodland P.C. "Improvements in an HMM-Based Speech Synthesizer". *Proceedings of Eurospeech Conference*, Madrid, Spain, 1995, pages 573-576.

[4] X. Huang, A. Acero, H. Hon, Y. Ju, J. Liu, S. Meredith, M. Plumpe "Recent Improvements on Microsoft's Trainable Text-To-Speech System - Whistler". *IEEE International Conference on Acoustics, Speech, and Signal Processing. pp. 959-962, Munich, Germany, April, 1997*.

[5] Klatt D. "Review of text-to-speech conversion for English". *Journal of the Acoustical Society of America*, 82(3):737-793, 1987.

[6] Huang X., Acero A., Alleva F., Hwang M.Y., Jiang L. and Mahajan M. "Microsoft Windows Highly Intelligent Speech Recognizer: Whisper". *IEEE International Conference on Acoustics, Speech, and Signal Processing*. Detroit, May 1995.

[7] Huang X., Acero A., Adcock J., Hon H., Goldsmith J., Liu J., and Plumpe M. "Whistler: A Trainable Text-to-Speech System". *International Conference on Spoken Language Processing*. Philadelphia, Oct, 1996

[8] Hwang, M.Y. and Huang, X. and Alleva, F. "Predicting Unseen Triphone with Senones". *IEEE International Conference on Acoustics, Speech, and Signal Processing,* Minneapolis, MN, pages 311-314. April, 1993.

[9] Hon, H., and Lee, K. "CMU Robust Vocabulary-Independent Speech Recognition System", *IEEE International Conference on Acoustics, Speech, and Signal Processing,* Toronto, CANADA, May, 1991.

[10] Microsoft's Speech Technology web page: http://www.research.microsoft.com/research/srg/.

[11] Nakajima S. and Hamada H. "Automatic generation of synthesis units based on context oriented clustering". *IEEE International Conference on Acoustics, Speech, and Signal Processing*. New York, April 1988, pages 659-662.

[12] Sproat, R. and Oliver, J. "An Approach to Text-to-Speech Synthesis". *Chapter 17 in book "Speech Coding and Synthesis*", Elsevier, 1995.

[13] Sagisaka Y., Kaiki N., Iwahashi N. and Mimura. K. "Unit Selection in A Concatenative Speech Synthesis System Using Large Speech Database". *International Conference on Spoken Language Processing*. Philadelphia, Oct, 1996.