Segment Connection Networks for Corpus-Based Speech Synthesis

Geert Coorman

TTS Research Department
Nuance Communications, Belgium
geert.coorman@ieee.org

Abstract

In this paper an efficient segment selection method for corpus-based speech synthesis is presented. Traditional unit-selectors use dynamic programming (DP) to find in a fully connected segment network the most appropriate segment sequence based on target- and concatenation costs. Instead of performing a full DP search, the presented unit-selector applies fast transformations on binary valued segment connection matrices. It is further also shown that this technique can be expanded to supra-segmental unit-selection without altering the segment size in the database.

Index Terms: speech synthesis, unit selection, connection matrix

1 Introduction

A classical corpus-based speech synthesizer [3] includes a large, prosodically rich speech segment database, containing speech units and modules for linguistic processing, prosody prediction, unit selection, segment concatenation and prosody modification. In the first step, the linguistic modules convert the input text into a sequence of abstract target segments, each having a feature vector. The feature vectors associated with the target segments are used to describe the target message in a linguistically motivated way. Therefore, the speech segment database is segmented and labeled with features that are directly related to the features used in the unit selector. In the second step, the unit selector decides which sequence of speech segments is selected from the space of possible segment sequences. Therefore it minimizes the distance between the target- and the actual segment sequence by using feature dependent cost functions and by considering inter-segment continuity by calculating a concatenation cost for each segment combination. In order to avoid disturbing concatenation artifacts, the RealSpeak® unit selector uses masking functions [3] to stimulate the rejection of segment combinations that cause artifacts. This optimization problem fits well in a dynamic programming framework. Finally speech is synthesized by retrieving, decoding, modifying and concatenating the selected speech segments.

Unfortunately there are some drawbacks associated with the dynamic programming (DP) framework used in corpus-based speech synthesizers:

- The DP optimization does not guarantee the optimal speech quality that can be obtained with a given segment database because the process of cost function calculation and combination is an oversimplification of a much more complex perceptual process.
- The DP consumes considerably more processing power than most other TTS components.

The system described in this paper avoids dynamic programming by pre-calculating viable segment combinations, efficiently stored in segment connection matrices.

2 Smooth Synthesis

Before tackling the real problem we will hypothesize a system with an infinite segment database (only conceptually). It is assumed that it is constructed by segmenting and labeling an infinite series of different single-speaker speech messages. Furthermore we assume that the infinite segment database is correctly labeled and annotated. An implicit feature of the infinite segment database is that for each well constructed target description there will be at least one natural sounding segment sequence that will fulfill the target and continuity requirements. All concatenation- and target costs will therefore be zero. In what follows we will refer to *smooth synthesis* if the synthesis result is free from concatenation artifacts (i.e. all concatenation costs are zero). In this particular case the set of optimal solutions can be found without resorting to DP. The set of all possible segment sequences can then be created by constructing a graph while traversing the target sequence from left to right. In what follows we will discuss an efficient representation for the set of smooth synthesis results based on real-world segment databases.

In reality the segment database is of finite duration. Therefore it is not always possible to find smooth synthesis results for a given target. However, by means of moderate inter-segment smoothing [1] it is possible to build segment sequences without concatenation artifacts while keeping the signal distortion resulting from the smoothing process below the hearing threshold. The requirement for smooth synthesis simplifies the unit selector significantly. For each concatenation feature *f*, the smooth synthesis condition implies that the transparency threshold P_T^f, which defines the tolerance limit for a feature mismatch, and the quality threshold Q_T^f which defines the maximum permissible feature mismatch, coincide (see [3] and fig.1). In other words the piece-wise linear masking function defined by its two thresholds is reduced to a binary classifier (fig 1). The masking function maps a small feature difference to 0 (i.e. when the unit selector foresees no audible concatenation artifact) and 1 in the other case.

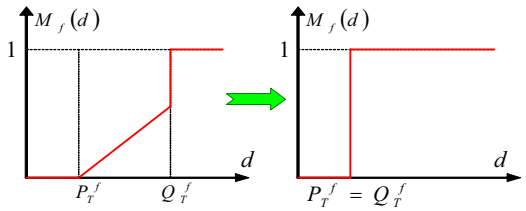

Figure 1: *Degeneration of the masking function M_f to a binary classifier*

Hence, smooth synthesis is obtained if and only if the cumulative concatenation cost associated to the path through the segment candidates is zero. As a consequence we will be able to use fast and efficient Boolean operators to construct the set of smooth synthesis results. Once the graph representation of the set of smooth synthesis results is known, it is straightforward to find the segment sequence with the lowest cumulative target cost. This procedure is more efficient than the traditional DP approach because calculation and accumulation of concatenation costs are avoided.

3 The Segment Connection Network

3.1 The Fully Connected Network

In traditional corpus based TTS, each unit from the speech segment database can be regarded as a single state in a complex fully connected state transition network in which the unit selector has to find the path of lowest cost that approximates the target [4].

In what follows it will be assumed that the target segment sequence has length L. Let N_i be the number of segment candidates for target segment $i \in \{1,2,..,L\}$. The number of all possible segment sequences C_L grows exponentially with increasing length L and is given by $C_L=N_1 N_2...N_L$. The DP algorithm used in traditional unit selectors is an efficient way to traverse through the different segment combinations in order to find the segment sequence that minimizes the imposed criteria.

3.2 The Segment Connection Matrix

In the following paragraphs it will be shown that an efficient representation of the closed set of all acceptable realizations can be easily obtained by introducing binary segment connection matrices.

Let's consider a target segment sequence of length L. The segment database provides a candidate list for each target segment. Candidate list $k \in \{1,2,..,L\}$ contains N_k segment candidates. The smooth synthesis restriction as defined above defines for each pair of adjacent candidate lists *(k, k+1)* a directed graph G_k that represents the transparent segment transitions (allowable connections) between segments of both candidate lists. Therefore we can represent all transparent segment transitions of each pair of candidate lists *(k, k+1)* by means of a binary valued *connection matrix* $\mathbf{T}_k$ which is associated to the graph G_k. Because each graph G_k is by construction bipartite, the connection matrix $\mathbf{T}_k$ is essentially a $N_k \times N_{k+1}$ matrix. Each element $t_{i,j}^k \in \{0,1\}$ of connection matrix $\mathbf{T}_k$ is representative for the connectivity between segment i of candidate list k and segment j of candidate list $k+1$ and can be written as a function of the F concatenation features:

$$t_{i,j}^k = \overline{M}_1\left(d_1\left(u_{k,i}^1, u_{k+1,j}^1\right)\right) \otimes .. \otimes \overline{M}_F\left(d_F\left(u_{k,i}^F, u_{k+1,j}^F\right)\right)$$

with

- $u_{k,j}^f$: concatenation feature $f \in \{1,2,..,F\}$ of segment candidate j,
- $d_f(:,:)$: the distance operator associated to concatenation feature f,
- $\otimes$: the logical and-operator,
- M_f : the Boolean masking function for concatenation feature f ($\overline{M}_f$ maps small feature distances to 1 and larger feature distances to 0)

It should be noted that the elements of the connection matrix may not be interpreted as concatenation costs. A "1" corresponds to a transparent transition (i.e. low concatenation cost) while a "0" means that there are audible discontinuities (i.e. high concatenation cost).

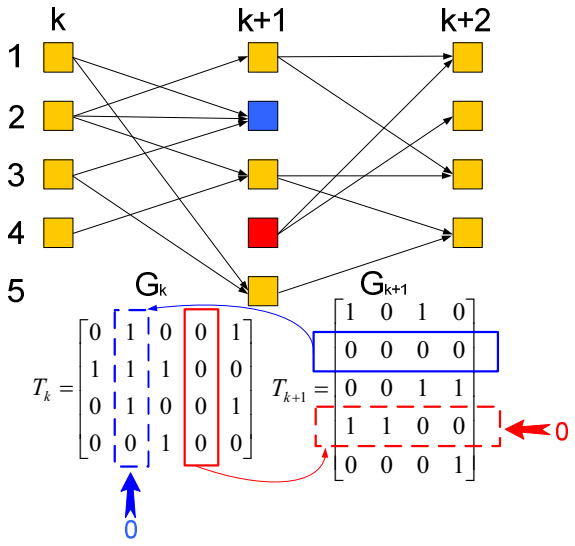

Figure 2: *Excerpt of a segment connection network. Associated connection graphs G_k and connection matrices $\mathbf{T}_k$*

The sequence of *L-1* connection matrices $\mathbf{T}_1,\mathbf{T}_2,...,\mathbf{T}_{L-1}$ is a complete and compact representation for the connections of the bipartite graphs $G_1G_2...G_{L-1}$ that describes the segment connection network under smooth synthesis conditions. Some of the segment sequences represented by $\mathbf{T}_1,\mathbf{T}_2,...,\mathbf{T}_{L-1}$ will span the entire target length while other sequences will contain less than L segments. Smooth synthesis sequences of length L will be referred to as *complete sequences*. Complete sequences start with a segment from the first candidate list and end with a segment of the last candidate list. By means of elementary matrix operations we can transform the sequence of connection matrices to a new sequence of sparser connection matrices $\mathbf{S}_1,\mathbf{S}_2,...,\mathbf{S}_{L-1}$ that uniquely describes all complete sequences. These new matrices are sparser because nodes that give rise to incomplete sequences are removed from the graph representation $G_1G_2...G_{L-1}$ (These nodes are marked in red and

blue in fig. 2). The node removal is simple. A segment sequence is incomplete if it does not start with a segment from the first candidate list or if it does not end with a segment from the last candidate list. This means that we have to check for two situations:

1. An incomplete segment sequence ends with the j^{th} segment of segment candidate list $k \neq L$ if and only if an index $i \in \{1..N_k\}$ can be found that fulfills:
$$\begin{cases} t_{i,j}^{k} = 1 \\ t_{j,l}^{k+1} = 0 \quad \forall l \in \{1,2,..,N_{k+2}\} \end{cases}$$
2. An incomplete segment sequence begins with segment candidate i of candidate list $k \neq 1$ if and only if an index $j \in \{1..N_k\}$ can be found that fulfills
$$\begin{cases} t_{i,j}^{k} = 1 \\ t_{l,i}^{k-1} = 0 \quad \forall l \in \{1,2,..,N_{k-1}\} \end{cases}$$

The actual implementation will not check for each element *(i,j)* if the above conditions are fulfilled; it will rather operate at a column or row basis. It should be noted that these operations are efficiently implemented when the connection matrices are represented as connection lists.

The two conditions above lead to the following two node removal rules:

- **A.** If the i[th] column of $\mathbf{T}_{k-1}$ is zero than set row i of $\mathbf{T}_k$ to zero
- **B.** If the j[th] row of $\mathbf{T}_{k+1}$ is zero then set column j of $\mathbf{T}_k$ to zero

It's clear that rule A (red in fig. 2) should be executed from left to right starting with the second connection matrix ($k=2$) while rule B (blue in fig 2) should be executed from right to left, starting with the penultimate connection matrix ($k=L-2$). The algorithmic complexity to find all complete sequences is linear in L. The node removal cannot be done off-line because it depends on the sequence of the connection matrices $\mathbf{T}_k$. However, by pre-calculating and caching the connection matrices $\mathbf{T}_k$ the generation of the $\mathbf{S}_k$ matrices can be made extremely fast.

3.3 Controlling connectivity

Before we continue with the discussion it is interesting to get an idea about the number of full path sequences, compared to C_L. For a large database and in the assumption that for each feature $f \in \{1,2,..,F\}$ the feature distances $d_f(:,:)$ between all viable segment combinations are uniformly and independently distributed, there exists a simple relationship between the F transparency thresholds P_T^f and the average number of full path sequences N_L.

$$N_L \cong \alpha^L \prod_{k=1}^{L} N_k = \alpha^L C_L$$

with
$$\alpha = \prod_{f=1}^{F} \frac{P_T^f}{\max\{d_f(:,:)\}} << 1$$

The above relation shows that the average number of full path sequences can be slightly controlled by changing the transparency thresholds within an acceptable range.

The segment database of a corpus-based speech synthesis system can contain rare speech segments. Therefore it is not always possible to find transparent segment connections between the candidate lists. This problem can be solved by dynamically adapting the transparency threshold while creating the connection matrices to guarantee at least a predefined number of connections for each viable candidate list combination.

4 Some Properties of Connection Matrices

4.1 Multiplication properties

It can be easily verified that the number of different paths that connect the i^{th} segment of the candidate list n to the j^{th} segment of candidate list m is given by the (i,j)[th] element of the $N_n \times N_m$ matrix:

$$\mathbf{P}^{n,m} = \mathbf{S}_n.\mathbf{S}_{n+1}\ldots\mathbf{S}_{m-1} \quad n<m$$

As a consequence $\mathbf{P}^{n,m}$ can be used to calculate the total number of different paths that arrive at each segment of candidate list n by means of the following matrix multiplication:

$$\mathbf{a}_n = \mathbf{u}_{N_1} \mathbf{P}^{1,n}$$

with $\mathbf{u}_{N_1} = [1..1]$ a row vector of dimension N_1.

Analogously, we can write $\mathbf{d}_n$ as the column vector of dimension N_n that represents the total number of different paths that depart from each segment from candidate list n:

$$\mathbf{d}_n = \mathbf{P}^{n,L} \mathbf{u}_{N_L}^T$$

To cover the full range of index n the definition of matrix $\mathbf{P}^{n,m}$ can be extended to the case where $n=m$: $\mathbf{P}^{n,n} = \mathbf{I}_{N_n}$

The total number of full path sequences N is given by the scalar product of $\mathbf{a}_n$ with $\mathbf{d}_n$ and is independent of the index of the candidate list n.

$$N = \mathbf{a}_n \mathbf{d}_n \quad \forall n \in \{1,..,L\}$$

4.2 Connectivity Index

Based on the properties of the connection matrices, the number of different paths that pass through each segment can be calculated by means of a series of matrix multiplications. The total number of paths that include a given segment is the product of the number of paths that arrive at the segment with the number of paths that depart from that segment. The number of complete sequences that pass through each segment of a candidate list n is given by the component-wise multiplication of $\mathbf{a}_n$ with $\mathbf{d}_n$ expressed as $\mathbf{a}_n \bullet \mathbf{d}_n^T$. This allows us to define for each candidate list n the following dimensionless parameter vector:

$$\mathbf{c}_n = \frac{\mathbf{a}_n \bullet \mathbf{d}_n^T}{\mathbf{a}_n \mathbf{d}_n}$$

which we will define as the *connectivity index* vector of candidate list n. The connectivity index $\mathbf{c}_{n,i}$ of the i^{th} segment of candidate list n can be seen as the relative contribution of segment i to the number of full path sequences that pass through candidate list n.

5 Pruning Techniques

In this section we discuss two pruning techniques for segment database reduction that are based on connection matrices only (i.e. without referring to target costs).

5.1 Connectivity pruning

The connectivity index of a segment is a good indicator to detect which segments contribute most to the smooth synthesis results. By synthesizing a large text corpus one can calculate the average connectivity index of each segment of all candidate lists. As opposed to standard statistical reduction techniques [5], the connectivity index vector of candidate lists corresponding with rare segments will not be directly affected by the poor distribution of those segments in the text corpus. Only rare segment combinations will be removed from the segment database as a result of the pruning process.

5.2 Similarity pruning

Segments that are used in corpus-based synthesis are generally phoneme sized. Typical segments are diphones [3], demi-phones [1] and phonemes [2]. Because of their short length, these segments do not embed much supra-segmental information. Their segment combinations define the supra-segmental representation of the synthesized speech signal. If two short speech segments have similar acoustic features at both of their boundaries (i.e. fulfill similar continuity constraints) then these segments can be exchanged without changing much to the supra-segmental realization of the sequence. Such situations occur when all paths that pass through one segment also pass through another segment. In this case we can remove one of the two segments.

The search for redundant segments based on their acoustic similarity can be efficiently implemented by means of connection matrix operations. Assume that $r_{k,i}$ and $r_{k,j}$ are two segment candidates from candidate list *k*. If all paths that pass through $r_{k,i}$ are the same as the paths that pass through $r_{k,j}$ then columns *i* and *j* of matrix $\mathbf{S}_{k-1}$ are identical and the i^{th} and j^{th} rows of matrix $\mathbf{S}_k$ are identical. In reality, few segments will fulfill this condition. In practical situations, we will find more rows and columns that are "almost" identical. The normalized correlation between two rows or two columns can be used as a measure for the degree of overlap between the rows or columns. If the degree of overlap between two columns in one combination matrix is close to one and the correlation between the two corresponding rows of the following combination matrix is also close to one than the segment that corresponds with one of the rows can be removed from the candidate list. The removal of a segment is equivalent with the deletion of a column in one connection matrix and the deletion of the corresponding row in the next connection matrix.

6 Supra-segmental unit-selection

Once all complete sequences are represented by the series of connection matrices, the best segment sequence can be selected. Inspired by traditional unit selection, the target costs at the segmental level can be used to find a minimal cost path. In what follows we will sketch a unit selection technique that assigns target costs at the syllable level by means of prosody scoring.

The speech segments from a standard segment database are too small to carry interesting prosodic information. It's only the combination of the segments that makes up the final prosody. Traditional unit-selection uses a linear combination of cost functions to model the perceptual relevance of feature vector mismatches. The unit-selector uses many different parameters to control the prosody. Some of these parameters are derived at the syllable level where after they are transplanted to the segmental level. Because of the speaker dependent prosodic variation, the selection of segments from different syllables with the same or similar symbolic features does not always yield the expected prosodic pattern. Therefore it is interesting to extend the segmental based unit selection to a supra-segmental based unit selection without resorting to features defined at the segmental level but by considering prosodic patterns that span the entire syllable. By combining the connection networks of each syllable separately, a number of syllable candidate lists can be constructed. It is clear that the number of candidate lists drops because of the agglutination of the segmental units to supra-segmental units. However, because of the many segment combinations that are possible, the syllable candidate lists can become much larger than the initial segment lists. The syllable candidates created by this composition method define, through their acoustic realization, prosodic features at the supra-segmental level. The candidate syllables can now be scored by traversing a tree containing sets of syllable-sized prosodic templates at its leaves. In a similar way, the syllables with the best scores can be combined into bi- or tri-syllable networks where after a similar scoring can be done at a lower time resolution. The agglutination to successively larger segments can be continued as long as there is enough reference data to score the combinations.

7 Conclusions

In this paper we have presented a fast unit selection method that is based on the calculation of combination matrices and the selective removal of unnecessary nodes. The technique is entirely based on the concept of smooth synthesis. Besides its computational efficiency, the combination matrices can be used for segment pruning and for supra-segmental unit selection in a segmental framework.

8 References

[1] M. Balestri, A. Pacchiotti, S. Quazza, P.L. Salza, S. Sandri, "*Choose the best to modify the least: a new generation concatenative synthesis system*," Proc. Eurospeech '99, Budapest, pp. 2291-2294, Sept. 1999

[2] A.W. Black, N. Campbell, "*Optimizing selection of units from speech databases for concatenative synthesis*," Proc. Eurospeech '95, Madrid, pp. 581-584, Sept. 1995

[3] G. Coorman, J. Fackrell, P. Rutten & B. Van Coile, "*Segment selection in the L&H RealSpeak laboratory TTS system*," proc. ICSLP-2000, Beijing, Vol. 2, pp. 395-398, Oct. 2000

[4] A. J. Hunt & A. W. Black, "*Unit Selection in a Concatenative Speech Synthesis System using a Large Speech Database*", proc. ICASSP-96, Atlanta, pp. 373-376, May 1996

[5] P. Rutten, M.P. Aylett, J. Fackrell & P. Taylor, "*A statistically motivated database pruning technique for unit selection synthesis*," proc. ICSLP-2002, Denver, pp. 125-128, Sept. 2002