L18: Speech synthesis (back end)

Articulatory synthesis

Formant synthesis

Concatenative synthesis (fixed inventory)

Unit-selection synthesis

HMM-based synthesis

[This lecture is based on Schroeter, 2008, in Benesty et al., (Eds); Dutoit, 2008, in Benesty et al., (Eds)]

Introduction

TTS front end

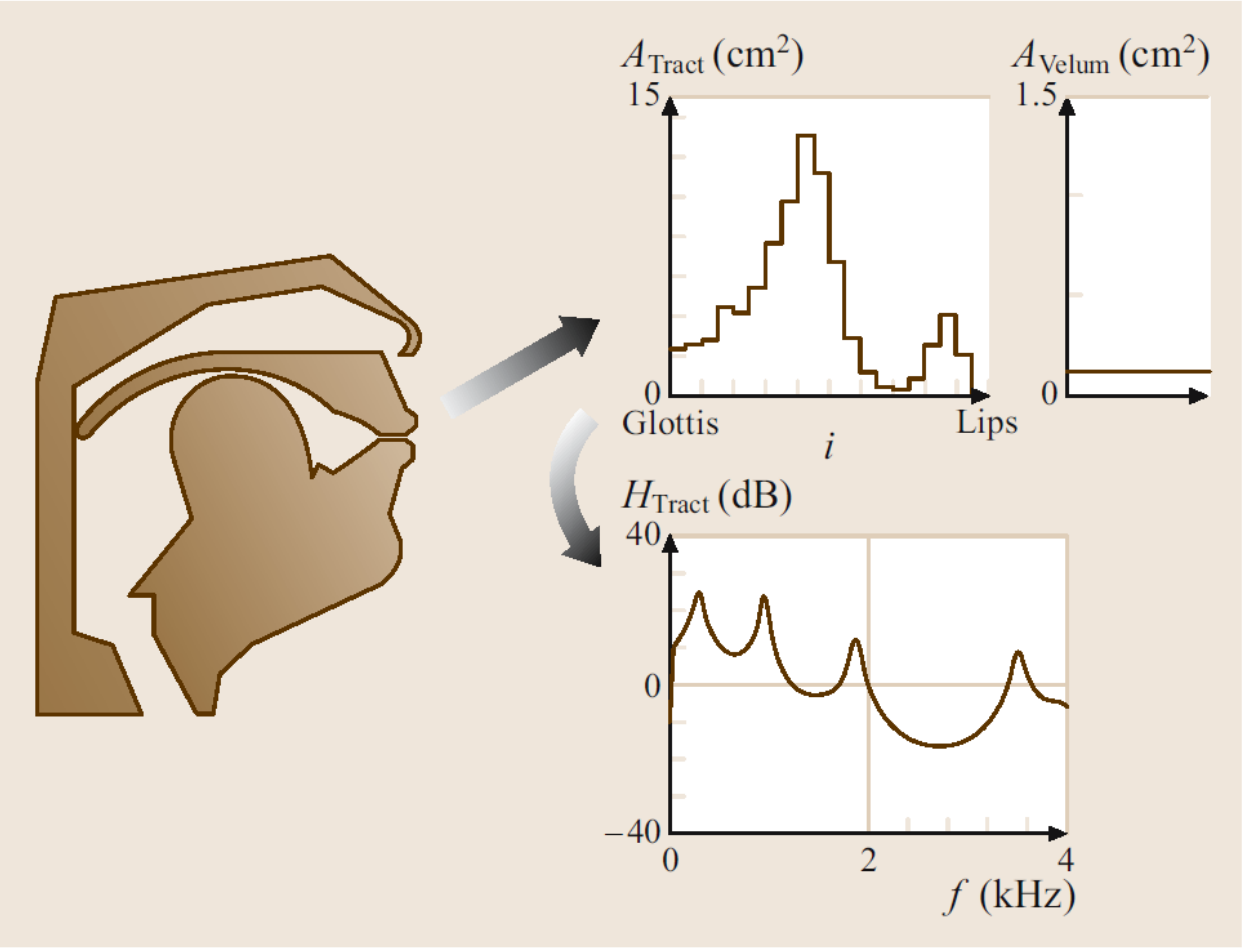
- Back-ends and front-ends are fairly independent components in TTS
  - This gives the designer flexibility, having to worry only about the interface
- A TTS back-end uses information provided by the front-end to synthesize speech using a specific method
- Traditionally, two types of synthesis methods can be distinguished
  - Rule-based methods, such as articulatory and formant synthesis
  - Corpus-based methods, such as concatenative systems
- This distinction is no longer clear, as there are hybrid methods that employ characteristics from both approaches

- In this lecture, we will focus on corpus based methods, but will also provide an overview of rule-based methods

Rule-based methods

Articulatory synthesis

- Articulatory synthesis uses mechanical and acoustic models of the speech apparatus to synthesize speech
  - Rather than describing the speech signal itself, these models employ control parameters that are meaningful for speech production
  - Parameters may include geometry and dynamics of the articulators (jaw, tongue, lips, velum) and the glottis, as well as forces and timings of all relevant groups of articulatory muscles
- Therefore, these models can be as simple as the straight tube model we saw in an earlier lecture, or as intricate as solving the Navier-Stokes PDE



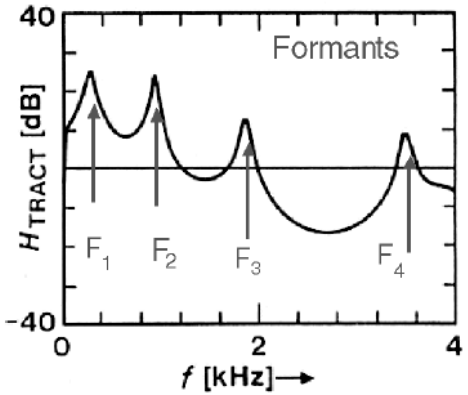
[Schroeter, 2008, in Benesty et al., (Eds)]

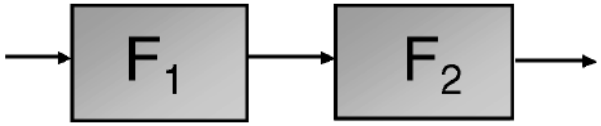
- Performance
  - Articulatory synthesis produces intelligible speech, but its output is far from natural sounding
  - The reason is that each of the various models needs to be extremely accurate in reproducing the characteristics of a *given* speaker
    - Most of these models, however, depend largely on expert guesses (rules) and not enough on observed data
    - Collecting articulatory data is an costly and fairly invasive process
  - Thus, while articulatory synthesis are appealing for scientific purposes and may one day provide completely "tunable" high-quality speech, their use remains fairly specialized

Formant synthesis

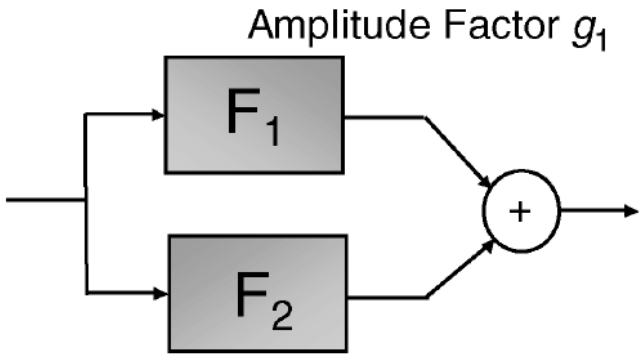
- In contrast, formant synthesis treats the vocal tract as a black-box, and aims to reproduce only its I/O characteristics
  - The goal is to approximate all VT resonances by a network of second-order filter, either in series or in parallel
- Series representation
  - Only requires frequency and bandwidth of each resonance plus a common gain
  - Approximates non-nasal sounds fairly well, but is not suited for nasals, fricatives or mixed-voicing sounds
- Parallel representation
  - Can approximate any speech spectrum
  - However, it requires individual gains for each filter
  - In addition, they introduce spectral zeros between the resonances

Vocal Tract Transfer Function:





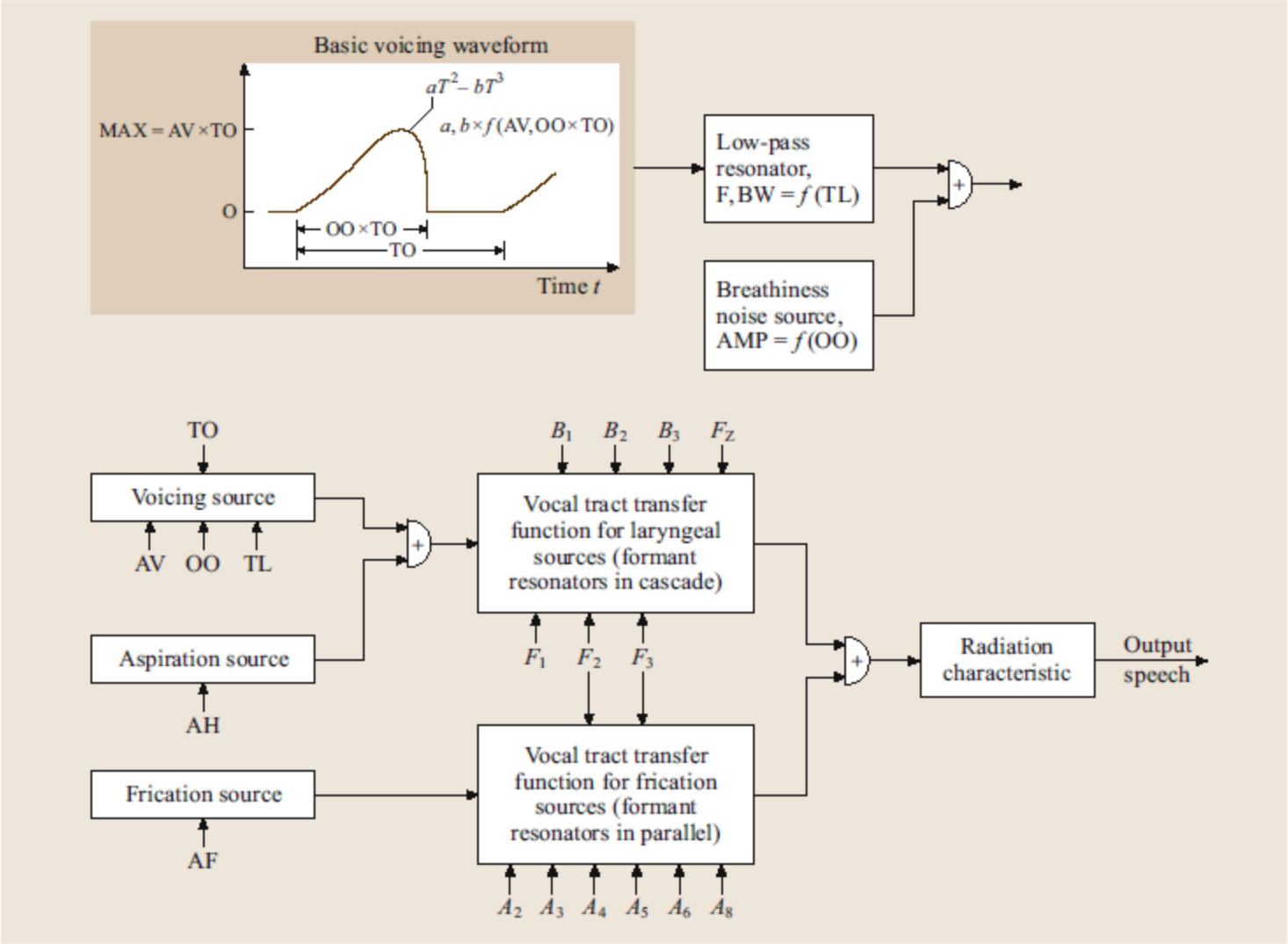
Overall amplitude set, e.g., to unity at zero frequency



[Schroeter, 2005]

- Characteristics
  - Formant synthesizers have moderate computational requirements, which make them practical for embedded applications
  - Voice quality can be controlled, but it is very difficult to match the voice of a target speaker
  - Intelligibility is generally very high
  - Formant synthesizers are highly appreciated in speech perception research, as they provide a high-level of control of the stimuli
- The main problem of formant synthesizers is deriving rules
  - Rules are needed to specify timing of the source and the dynamic values of all filter parameters
  - This is difficult enough for simple words, let alone for complete utterances
  - These rules, however, may be derived through analysis-by-synthesis

Hybrid serial/parallel synthesizer of Klatt (19 parameters)



[Schroeter, 2008, in Benesty et al., (Eds)]

Concatenative speech synthesis

Basic concept

- Concatenative synthesis techniques work by “gluing” together speech chunks that have been previously recorded
  - Concatenation is done to carefully to preserve the natural coarticulation, shimmer, jitter and inharmonic content of speech
  - Transients in speech are more important for intelligibility than stable segments, while modifying stable segments can easily affect naturalness

Types of concatenative synthesis

- Concatenative synthesis with a fixed inventory
  - These approaches generally contain one sample for each unit, and perform prosodic modification to match the required prosody
  - As a result of processing, some signal degradation is unavoidable
- Unit-selection-based synthesis
  - These approaches store several instances of each unit, thus improving the chances of finding a well-matched unit

Concatenative synthesis with fixed inventory

- These approaches generally use the diphone as the speech unit
  - A diphone starts in the middle of the stable part (if any) of a phone, and ends in the middle part of the next phone
  - Diphones reduce distortions since units are joined at their stable part
  - They also preserve coarticulation, since units contain the transition between phones

Inventory size

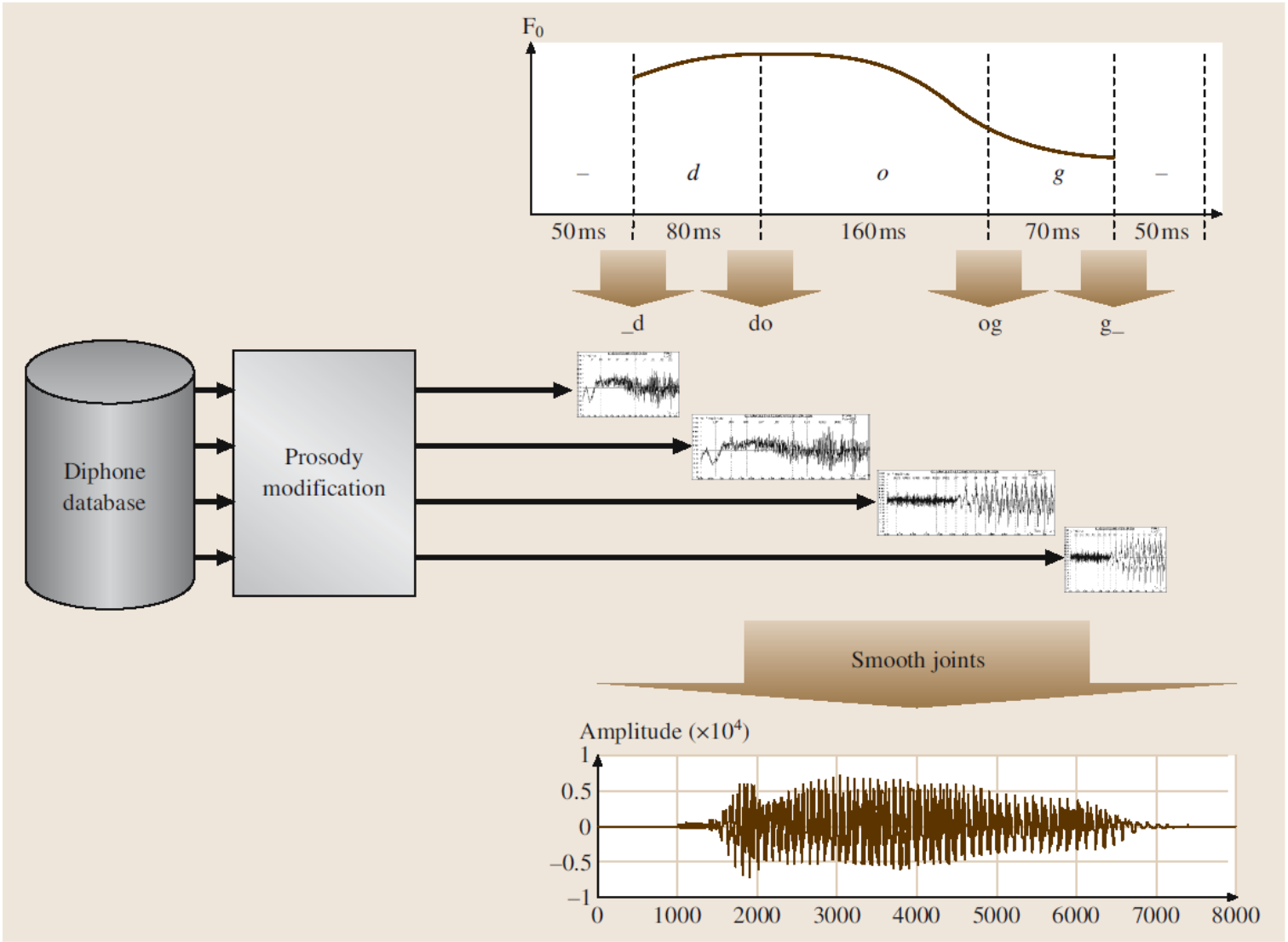
- For a language with N phonemes, up to N^2 diphones may be needed
  - In practice, the number is somewhat smaller since not all diphones are encountered in natural languages
- In the case of English, a typical diphone database contains 1,500 units
  - This represents about 3 minutes of speech $\sim$ 5MB at 16kHz/16 bits

Building the synthesizer

- Set up a list of required diphones
- Create a list of words such that each diphone appears at least once (two is better for security)
- Exclude diphones in unfavorable positions (strongly stressed syllables or strongly reduced contexts)
- Collect corpus as read by a professional speaker (avoid variations, even large pitch variations)
- Identify the elected segments, manually with the help of visualization tools, or with segmentation algorithms
- Collect segment waveforms into a diphone inventory

Running the synthesizer

- Receive phonetic input (phonemes, duration, pitch) from front end
- Perform prosodic modification
  - Diphones in the inventory will rarely match specs from the front end
- Smooth individual pairs of successive diphones
  - The end of one diphone and beginning of the next will not match in amplitude or in spectral envelope



[Schroeter, 2008, in Benesty et al., (Eds)]

Prosody modification

- Amplitude modification is straightforward
- Pitch or duration are non trivial: slowing down playback to increase duration will simultaneously decrease pitch
- Two types of prosody modification are common
  - Time-domain (TD-PSOLA)
  - Frequency domain (HNM)
- These techniques will be discussed in the next lecture

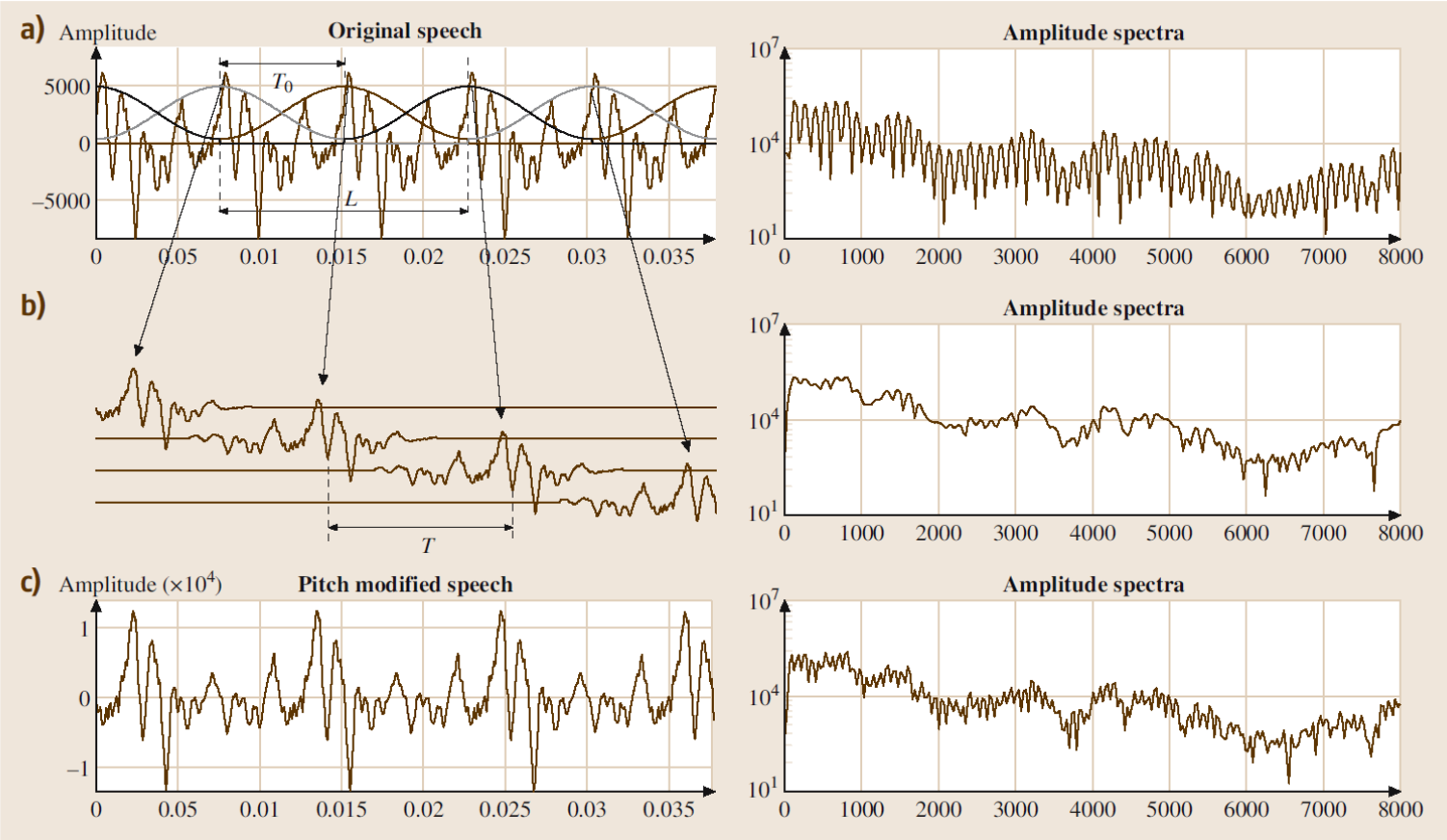


Fig. 21.2a–c The TD-PSOLA reharmonization process. The pitch-modified waveform (**c**) has the same spectral envelope as the original waveform (**a**). The center plot (**b**) shows the OLA frames and the amplitude of their Fourier transform, which is approximately the spectral envelope of the initial signal (after [21.4])

[Schroeter, 2008, in Benesty et al., (Eds)]

Smoothing

- Concatenating units from different words or phonetic contexts is not straightforward, and leads to audible clicks if not done carefully
- This is due to at least three types of mismatches between units: phase, pitch, and spectral envelope mismatches

Phase mismatches

- Occurs when OLA frames are not centered at the same place
- Several solutions are possible
  - Accurately pitch-mark all boundary frames (as needed with TD-PSOLA)
  - Adjust the position of OLA frames to maximize cross-correlation (WSOLA)
  - Measure and correct phase mismatches (MBROLA, HNS)

Pitch mismatches

- Occurs when overlapped frames have very different F_0
- This issue is difficult to resolve
  - May require recruiting a professional speaker to read the corpus with a fairly constant pitch
  - Alternatively, it may be possible to redistribute the large pitch difference across multiple frames

Spectral envelope mismatch

- Occurs whenever overlapped frames have been extracted from rather different contexts
- Also difficult to resolve; potential partial solutions may be
  - Linearly interpolating the two spectral envelopes (HNS does this)
  - Adjusting the unit boundaries on the fly to find a smoother join

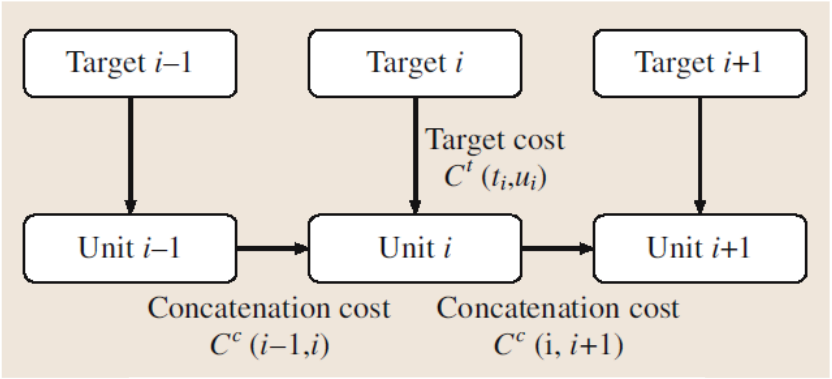
Performance

- Results from concatenative synthesis with a fixed inventory are clearly intelligible
- Unfortunately, their speech output is far from natural sounding
- This issue can be traced back to two causes
  - Storing only one units biases the recording stage towards over-articulated speech, one that fit in most contexts and improve intelligibility
  - Signal processing tricks are needed to adjust prosody, and these invariably cause some degree of signal degradation

Unit-selection-based synthesis

Basic idea

- In unit-selection, one stores several instances of each unit, and selects (at run time) which instance to use
- For every target unit t_i required (as specified by the front-end)
  - The selection algorithm proposes a list of candidate units
    - Each unit has a different context (generally not exactly that of t_i)
  - The final choice is based on minimizing the sum of two cost functions
    - A target cost $C^t(t_i, u_i)$, which estimates differences between t_i and u_i
    - A concatenation cost $c^c(u_{i-1}, u_i)$, which estimates the quality of the joint between candidate units u_{i-1} and u_i



[Schroeter, 2008, in Benesty et al., (Eds)]

- The best sequence of n units u_1^n for a given sequence of n targets t_1^n is chosen so as to minimize the total cost

$$C(t_1^n, u_1^n) = \sum_{i=1}^{n} C^t(t_i, u_i) + \sum_{i=2}^{n} C^c(u_{i-1}, u_i) + C^c(S, u_1) + C^c(u_n, S)$$

  - where S denotes the target for a silence, and the optimal sequence is found through a Viterbi search
- Prosodic modification is not needed, unless good candidate units cannot be found with the correct pitch and duration
- Some smoothing can be applied since candidate units generally do not concatenate smoothly

Challenges in unit-selection synthesis

- Efficient target and concatenation costs
- Definition of optimal speech corpus
- Efficient search algorithms

Target costs

- The optimal target cost should estimate the perceptual distance between the candidate unit u_i and the target unit t_i
  - Each candidate unit is characterized by a feature vector, and
  - feature values for the target are predicted by the front-end
- The target cost is the weighted sum of sub-costs

$$C^t(t_i, u_i) = \sum_{j=1}^{p} w_j^t C_j^t(t_i, u_i)$$

  - Feature weights w_j^t are trained during construction of each TTS voice to optimize the mapping from feature space to perceptual space
- The feature vector contains a variety of information
  - Symbolic/phonological features (phonetic context, stress)
  - Numerical features (pitch, duration)

Concatenation cost

- The ideal concatenation cost should reflect the perceived discontinuity between successive units
- As with the target cost, the concatenation cost is a weighted sum

$$C^c(u_{i-1}, u_i) = \sum_{j=1q} w_j^c C_j^c(u_{i-1}, u_i)$$

- Features for the subcosts $C_j^c(u_{i-1}, u_i)$ are based on spectral representations (formant values, LP spectra, LSFs, MFCCs…)
- Distance measures include Euclidean distance, weighted Euclidean distance (shown in the equation), Mahalanobis distance
- Concatenation costs are assigned to zero for originally consecutive units in the speech corpus
  - This allows the synthesizer to use the longest sequence of naturally occurring units in the corpus
  - The challenge is to make sure this does not happen at the expense of intelligibility (if as a result the target costs become high)

Speech corpus

- Speech obeys some sort of Zipf's law
  - In a corpus of natural language utterances, the frequency of any word is roughly proportional to its rank in the frequency table
  - In other words, speech is composed of a large number of rare events
  - The order for a diphone database to cover a randomly selected sentence in English with $p = 0.75$ is estimated to be around 150,000 (5 hours)
  - Most commercial systems to date use about 1-10h (150-1500MB) and achieve high quality *most of the time*
- One of the major problems with unit selection is data sparsity
  - For this reason, it is common to include a set of safety units, generally a complete set of diphones
  - This provides a fall-back strategy and guarantees that performance is no worse than that of fixed inventory systems
- For limited-domain TTS, unit selection is currently the best option

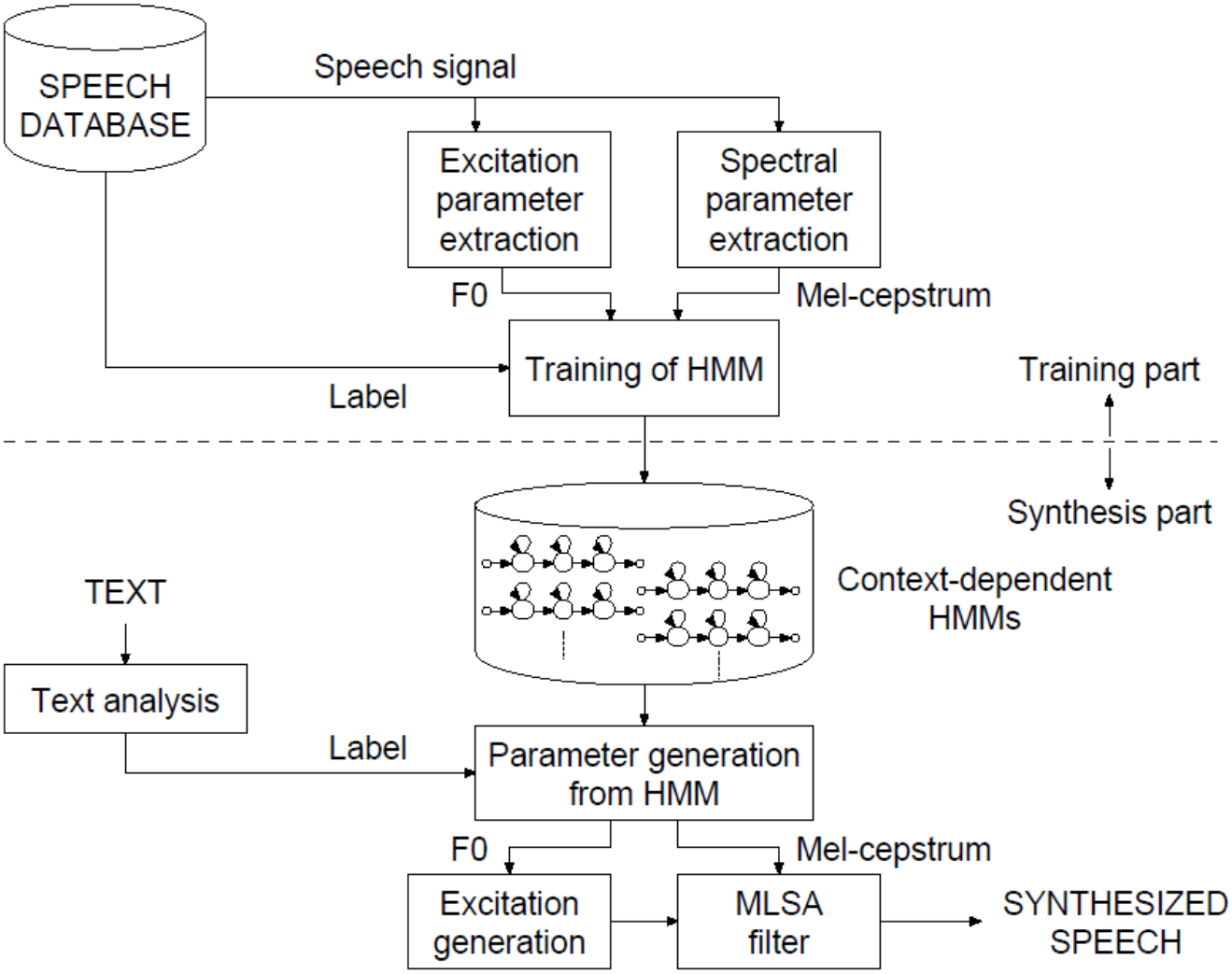
HMM-based synthesis

HMM-based synthesis borrows techniques from ASR

- HMM are used as pattern generators rather than as recognizers
- HMMs are trained with spectral vectors *as well as* with phonetic, stress and syntactic information (to be provided by the front-end)
- By using parameter tying (context clustering trees), HMM synthesis can generalize to unseen data (something unit-selection cannot do)

Each unit (a phoneme) is modeled by a three-state HMM

- Each state emits spectral feature vectors (MFCCs) according to a GMM associated with the leaves of a context clustering tree
- Pitch and duration are also generated by the HMM through separate clustering trees and GMMs



[Zen, Tokuda and Black, Speech Comm. 2009]

Generating speech

- The front-end provides a target sequence of phonemes, augmented with stress and syntactic information
- A sentence HMM is formed by concatenating context-dependent phoneme HMMs
- State durations of the phoneme HMMs are determined so as to maximize their output probability given phonetic and syntactic context
- Spectral parameters are found in a slightly different fashion
  - Spectral vectors produced by the HMM states contain static (MFCC) and dynamic (Δ, Δ^2) features
  - However, the ML solution (i.e., the mean of each Gaussian) does not guarantee that the dynamic constraints will be met
  - Remarkably, a closed form solution to this problem (maximizing ML subject to dynamic constraints) is available and leads to what is known as a trajectory HMM

Performance

- Current HMM-based synthesizers produce speech that is smooth but of poor voice quality (depending on the excitation model used)
- HMM-synthesis, however, has several advantages over unit selection
  - More flexibility, due to context clustering
  - Better coverage of the acoustic space, due to the HMM/GMMs
  - Fairly small footprint (1MB)
  - When combined with adaptation techniques, new voices can be generated with very small training sets (5-10 min)
  - Also provides a natural framework for voice conversion and modification

ex18p1.m

Demonstrate a simple MATLAB-based TTS system for “Genglish” (available at http://tcts.fpms.ac.be/projects/ttsbox)