Sādhanā, Vol. 21, Part 3, June 1996, pp. 345–362. © Printed in India.

Synthesis of unlimited speech in Indian languages using formant-based rules

XAVIER A FURTADO[†] and ANIRUDDHA SEN

Computer Systems and Communications Group, Tata Institute of Fundamental Research, Homi Bhabha Road, Colaba, Mumbai 400 005, India
email : asen@tifrvax.tifr.res.in

Abstract. Synthesis of continuous and unlimited speech is a matter of theoretical as well as technological interest. Independent efforts are needed for synthesis in Indian languages which are substantially different from English and other European languages. The paper discusses basic synthesis issues like text-to-phoneme and phoneme-to-speech conversion and incorporation of prosody. The three commonly adopted methodologies of concatenation, formant and articulatory syntheses are compared. The TIFR phoneme-to-speech synthesizer which utilizes a standard formant synthesizer as a speech production model is described and the methodology for evolving and organizing formant-based rules to drive the used synthesizer is emphasized. The results of some perception tests are reported and a few potential applications are suggested. The direction of the future work for enhancing the quality and expanding the scope of the synthesizer is indicated.

Keywords. Speech synthesis; computer speech; Indian language synthesis.

1. Introduction

Incorporation of human faculties like speech and vision into machines is a basic issue of artificial intelligence research. The capabilities of a computer to accept spoken input and generate speech output are termed as speech recognition and synthesis respectively.

Speech synthesis requires a profound understanding of speech production and perception, and has always been a topic of great interest in speech and cognitive sciences. Recent advances in computer, communication and information technologies have considerably enhanced its technological motivations, particularly for fast and automatic access of information through telephone.

In its simplest form, 'computer speech' can be produced by playing out a series of digitally stored segments of natural speech. The segments may be coded for storage efficiency.

[†] Deceased

The technique involved is elementary. However, it cannot be extended for producing continuous speech with unlimited vocabulary because the context sensitivity of speech makes simple merging of stored speech segments into larger utterances unacceptable. 'Unlimited' speech synthesis, to which we will restrict our subsequent discussions, has to deal with this and several other issues.

Over the years, several independent methodologies for unlimited speech synthesis in English and other European languages have evolved. Although speech synthesis research in India has greatly benefited from such work, substantial differences in phoneme sets and stress patterns in Indian languages rule out direct adoption of the techniques and makes indigenous development mandatory.

This paper presents a phoneme-to-speech synthesizer developed by the authors in the speech laboratory of the Tata Institute of Fundamental Research (TIFR). It can synthesize unlimited speech in Hindi and English as spoken by typical Indians. From among similar synthesizers in Indian languages, it stands unique in its methodology of generating speech wholly by a production model whereas other synthesizers utilize stored speech in some way or the other. The speech production model used is a standard formant synthesizer suggested by Klatt (1980) and it is driven by formant-based rules developed indigenously.

The following sections briefly discuss the issues involved in speech synthesis and various methodologies which can be applied to tackle its core problem – phoneme-to-speech conversion. The TIFR synthesizer is then described in some detail and its performance reported.

2. Issues related to unlimited speech synthesis

The basic issues related to unlimited speech synthesis can be categorized as: (a) Text-to-phoneme conversion (b) phoneme-to-speech conversion and (c) application of prosody. Such categorization breaks down the synthesis problem into fairly independent components and allows reasonably modular solution.

2.1 *Text-to-phoneme conversion*

An 'unlimited' speech synthesizer has to accept any arbitrary input message which is to be converted to speech. In most practical situations, it is convenient to specify the messages in textual form. Text is acceptable to most people and can be easily processed by a computer.

Text-to-phoneme conversion means the conversion of the textual input message into its corresponding pronunciation which is specified by means of a string of phonemes. The task is comparatively easy for the languages where the written text and the pronunciation are related in a simple and straightforward manner (e.g. Hindi) and is non-trivial where they are not (e.g. English). The conversion, in general, involves derivation of letter-to-phoneme rules, dictionary formation and look-up, and morphological analysis. As the input text is analyzed by the text-to-phoneme module, it should preferably extract information related to prosody (§ 2.3) which is derivable from such analysis. Clearly, there can be nothing like a 'perfect' analysis of text. Text-to-phoneme conversion, therefore, is an open problem and requires continous research and development.

As stated earlier, ours is presently a phoneme-to-speech synthesizer. However, it can be converted to a complete text-to-speech system by adding any appropriate text-to-phoneme module at the front end. Such modules, with acceptable quality, have already been developed for some Indian languages (Bhaskararao & Mathew 1992; Bhaskararao *et al* 1994). But until now, precisely little work has been done on text-to-phoneme conversion for Indian English.

2.2 *Phoneme-to-speech conversion*

Phoneme-to-speech conversion can be termed as the 'essence' of speech synthesis and the work presented here is mostly concerned with this. Various phoneme-to-speech conversion methodologies in general are discussed in § 3, whereas our specific work is described in §§ 4, 5 and 6.

2.3 *Application of prosody*

Prosody, in simple acoustic terms, means the variation of pitch, intensity and (intrinsic) duration of the utterances with time. Proper prosody is to be applied for making the synthetic speech natural sounding and at least a minimal amount is needed to make the speech even intelligible. Determination of proper prosody depends on analysis of syntax (grammar), semantics (meaning) and pragmatics (linguistic context). Therefore, this is also an open research problem.

In our synthesizer, only elementary prosodic rules have been applied so far. (These are discussed in § 6.7.) The immediate emphasis was on making the synthetic speech intelligible. As this was done with reasonable degree of success, incorporating better prosodic rules for making the speech more natural-sounding is the next step.

3. Various methodologies for phoneme-to-speech conversion

For phoneme-to-speech conversion in an unlimited speech synthesizer, the related problems can be broken down into the following sub-problems: (i) selection of basic units (ii) generation of the selected basic units and (iii) concatenation of the basic units for synthesizing continuous speech.

The methodologies for synthesis may vary, but in order to generate unlimited speech out of a limited corpus of stored information, (i) the selected basic units must be small enough (e.g. phonemes, syllables) so that the total inventory is limited and (ii) rules of some kind or the other are to be applied for concatenating such units into continuous speech, irrespective of the methodology adopted.

The methodologies commonly used can be divided into two broad categories: (a) synthesis by concatenating stored natural speech segments (coded or uncoded) and (b) synthesis by generating speech from a speech production model. The latter, in turn, can be classified into (i) formant synthesis, where the model is based on several formant frequency resonators and (ii) articulatory synthesis, where effort is made to mathematically model the static and dynamic characteristics of various articulators. A brief description of these three methodologies along with their comparative study follows.

3.1 *Synthesis by concatenation*

A fundamental problem of speech synthesis is that the acoustic manifestations of basic units of the utterances, i.e phonemes, are very much sensitive to the phonemic context. Also, the transition from one phoneme to another carries substantial amount of information necessary for the perception of both the phonemes.

In concatenation synthesis, a limited number of stored segments obtained from real speech are used. However, in view of the above mentioned reasons, the inventory cannot be just a set of single phonemes. A phoneme cut out from a context would most probably not fit into another and a sequence generated by pasting such phoneme splices would be completely unintelligible. In order to capture the contextual variations and transitions, splices at least equivalent to diphones are to be taken as basic units. In addition, if the transitions are not to be missed, the end-points of the splices must be in the 'steady' region of speech.

Diphones, demi-syllables are some of the basic units normally selected for the purpose (Dan *et al* 1990; Bhaskararao & Mathew 1992). For Indian languages, even 'characters' from the written scripts (consisting of CV syllables like 'koo', vowels like 'ee' and consonants like 'p' which can be represented by a single written character) were used as basic units (Rajesh Kumar *et al* 1989). In general, if n is the total number of phonemes, the number of elements in a diphone-type of inventory will be of the order of n^2. With a typical number of phonemes like 50, the inventory will run to 2500 or so. However, as all diphone combinations do not occur and as a single splice can often be made to represent 'a group' with minimal degradation of quality, sizable pruning of the inventory is possible.

The basic advantage of the concatenation method is its simplicity. As the units are taken from real speech, the cumbersome task of 'generating' them is obviated. Complicated transitions like CV are automatically 'captured' in totality by the actual speech data and the 'rules' for concatenating the basic units are therefore elementary. This allows development of acceptable systems with limited investment of time and specialized skill. If the original time waveforms are stored, the processing is basically in the time domain and the synthesis is very fast. It is then possible to make real-time synthesizers on general-purpose machines like PCs.

Some problems, however, are associated with this method. During data collection, it is not easy to ensure that all the steady-state ends of the segments (e.g. all demi-syllables ending and beginning with /i/) are exactly matching in formant frequencies etc. This can lead to 'jerks' at the joints. Also, it is obvious that this method needs some 'steady' segments of speech. However, in fluent speech with substantial co-articulation, such steady-states will often be non-existent and it is difficult to capture such fluency with this method. Many consonants are almost completely 'transitory' in nature. It is difficult to reconstruct various clusters of such consonants from segments with steady-state end-points. Another problem is that while storing segments, the allophones also should be added to the list of phonemes and this increases the inventory (proportional to the square of phonemes plus allophones) substantially. Also, in this method, the voice is 'personalized' and a totally new segment inventory is to be prepared for generating a different voice.

3.2 *Formant synthesis*

This method, adopted by us and described later in detail (§§ 4,5,6), is based on the generation of speech from a production model which has formant frequencies, energies, voice source control parameters (e.g. pitch) and few other acoustic-phonetic parameters as control variables. It is possible to generate any arbitrary speech by applying a set of context-sensitive rules on the stored phonemic data for enacting contextual modifications and for generating transition segments.

This method can overcome many of the limitations of the concatenation method. With single phonemes as the basic units, rules for smooth concatenations can be formulated. With only a few additional rules and little additional data, consonant clusters and allophones can be handled elegantly. Presence of a 'steady-state' is not mandatory, hence co-articulations can be incorporated. Also, as the model abstracts the mechanism of speech production, the resulting voice is not personalized and can be altered easily by changing a few control parameters. The effort needed to switch over to another similar language with a marginally different phoneme set is also minimal and this is very important in a multi-lingual country like India. Overall, this synthesizer has the potential to surpass a concatenation synthesizer in both quality and versatility.

However, in order to realize such potential, a set of appropriate rules are to be derived and this is not a trivial task, particularly because such rules are basically 'heuristic' in nature. The time and specialized skill needed to develop such a synthesizer is therefore high. In spite of the best of efforts, there will be approximations at two levels: in modelling the speech production as well as in capturing the model control parameter variations by a finite set of rules. A continuous effort to minimize the deviations are therefore called for. Fortunately, the synthesizer framework can be made flexible enough to allow gradual improvements. As the model uses a number of resonators, it is computationally expensive. But with the current 'hardware revolution', affordable and near real-time synthesizers using this technique are now in the realm of possibility.

3.3 *Articulatory synthesis*

For generating various utterances, the formant synthesizers control acoustic-phonetic features like formant frequencies, energies, pitch etc., whereas when we speak, we have no independent control over them. This necessitates 'heuristics' for rule formation. Articulatory synthesis strives to eliminate the problem by modelling the actual mechanism of speech production through the articulators and their movements. Thus it is expected that more elegant and conceptually clearer rules can be formulated. It should also be easier to capture the production mechanism in totality if the modelling is done on the basis of a finite and exhaustive set of articulatory movements rather than on the basis of the variation of a subjectively selected set of acoustic-phonetic features (as is done in formant synthesis).

Here, too, the problem is to realize the theoretical potential by optimizing the mathematical model and collecting sufficient data (including X-ray data of articulator movements during production of various types of utterances). This method is also computationally very expensive. Overall, although this can be described as the synthesis strategy of the future, at the moment it is more a research issue than a commercial reality and more so in India.

4. TIFR synthesizer overview

The phoneme-to-speech synthesizer developed at TIFR is completely a software synthesizer. The only special hardware it needs is a D/A converter, along with de-aliasing filters, for playing back the synthetic speech. A sampling frequency of 10 kHz and a de-aliasing low-pass filter cut-off frequency of about 4.7 kHz is used. The synthesizer was implemented on a Microvax II computer where it ran at around 24 times real time (i.e. 24 s was needed to synthesize 1 s of speech). A demonstration version was also implemented on a PC-386, where it ran at around 13 times real time.

As input, the synthesizer accepts a string of phonemes from a repertoire that is made up of 57 phonemes normally used in Hindi and Indian English, and markers for silence and question mark. Figure 1 lists these phonemes (and their classes) along with their one or two character symbols which can be entered from an English keyboard. (Note: If the second character of the symbol is '1' it can be optionally omitted. Thus 'k1' and 'k'

1. Silence	_
2. Vowels	uh, aa, eq, ee, uq, oo, अ आ इ ई उ ऊ ey, ae, oh, aw, eh ए ऐ ओ औ uh$,aa$,eq$,ee$,uq$,oo$, अं आं इं ईं उं ऊं ey$,ae$,oh$,aw$,eh$ एं ऐं ओं औं
3. Stops	k1, k2, g1, g2, क ख ग घ t1, t2, d1, d2, ट ठ ड ढ t3, t4, d3, d4, त थ द ध p1, p2, b1, b2 प फ ब भ
4. Affricates	c1, c2, j1, j2 च छ ज झ
5. Nasals	m, n1, n2, n3 म न ण ङ
6. Semi-vowels	y, w य व
7. Trills/Flaps	r, d5 र ड़
8. Laterals	l ल
9. Fricatives	ss, sh, h, z, f, v, स श ह ज़ फ़

Figure 1. Phoneme repertoire.

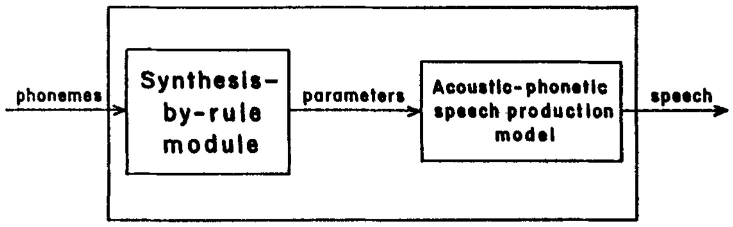

Figure 2. Block diagram of the TIFR synthesizer.

are equivalent.) The synthetic speech output can be generated in any of the four voices: (i) low-pitched male voice (ii) high-pitched male voice (iii) high-pitched female voice and (iv) low-pitched female voice. However, only the male voices have been fine-tuned.

The synthesizer (figure 2) is made up of two fairly independent modules: (i) The acoustic-phonetic speech production module or the parameter-to-speech module (§ 5) accepts a number of acoustic-phonetic parameters updated at regular intervals as input and generates corresponding synthetic speech. This is essentially the formant-based speech production model. (ii) The synthesis-by-rule module or the phoneme-to-parameter module (§ 6) generates the input parameters needed by the parameter-to-speech module. It is basically the formant-based rule module which takes the current and adjacent phonemes into consideration while generating each parameter contour. Currently, each parameter is updated at an interval of 5 ms, which is small enough to capture most of the variations in speech.

An important aspect of any synthesis or recognition methodology is to what extent it is language-independent. The extent to which the advanced techniques developed internationally can be utilized depends on this. For the currently described synthesis methodology, whereas the speech production model can be considered to be reasonably valid for any language, the phoneme to parameter conversion rules, which have to capture the variation of acoustic-phonetic features corresponding to the specific articulations of a given languages, should be language specific.

To start with, it was therefore decided to have a speech production model similar to the one described by Klatt (1980). The set of rules for generating the time variations of its control parameters was developed fully at TIFR.

5. Parameter to speech conversion

The main components of this module are: (i) Voicing source, (ii) noise source, and (iii) a series of resonators and anti-resonator(s). By proper arrangement of these components, virtually any speech sound can be faithfully synthesized.

The voicing source used is an impulse train with a periodicity corresponding to the given pitch period. It is then shaped by an appropriate low pass filter. Provision for incorporating a more 'natural' voicing source exists. This source is used to generate vowel as well as voiced consonants.

The noise-like sounds (e.g. frication or a burst from the release of a stop consonant) require a different type of source. It is implemented by a random noise generator and the output is again shaped by 'soft filtering'.

A series of resonators model the vocal tract. An anti-resonator is incorporated to facilitate the generation of nasal vowels and nasal consonants and provision exists for adding more, if the need arises.

A digital resonator is implemented (Klatt 1980) by the equation

$$y(nT) = Ax(nT) + By(nT - T) + Cy(nT - 2T), \tag{1}$$

where $y(nT)$, $y(nT - T)$ and $y(nT - 2T)$ are the current and two previous output samples and $x(nT)$ is the current input sample. The resonator coefficients A, B, C can be computed by

$$C = -\exp(2\pi B_w T), \tag{2}$$

$$B = 2\exp(\pi B_w T)\cos(2\pi FT), \tag{3}$$

$$A = 1 - B - C, \tag{4}$$

where F is the central frequency of the resonator (i.e. the formant frequency), B_w is its bandwidth and T which is 1/(sampling rate) is equal to 0.0001 second at the used sampling rate of 10 kHz.

For anti-resonators, the equation (Klatt 1980) is

$$y(nT) = A'x(nT) + B'x(nT - T) + C'x(nT - 2T), \tag{5}$$

where $x(nT - T)$ and $x(nT - 2T)$ are the previous two samples of the input $x(nT)$. The coefficients are determined by

$$A' = 1.0/A, \ B' = -B/A \text{ and } C' = -CA,$$

where A, B and C are obtained by using the antiresonance center frequency F and bandwidth B_w in (2), (3) and (4).

The resonators can be connected in cascade or in parallel. Both type of synthesizers were previously used with good effect. For example, while Klatt (1980) used cascade synthesizer, Holmes used the parallel one (Holmes *et al* 1964; Holmes 1983). As our idea was to waste least effort on the topics already worked on, we started simply by adopting the cascade synthesizer methodology used by Klatt and pursued it, as no serious problem was faced.

In this method, the resonators are connected in cascade for generating voiced sounds. In other words, the first resonator receives the source pulse train as input and then the output of a resonator is fed as the input to the next one. The output of the last resonator is the resulting voice output, having spectral poles at all the resonance frequencies. Spectral zeros can be similarly incorporated by adding anti-resonators.

A minor problem associated with cascade synthesizers is that the amplitudes of different resonance peaks cannot be directly controlled. However, it can be controlled indirectly by adjusting the bandwidths of the resonators. Furthermore, there is a provision to 'tilt' the spectrum i.e. to attenuate the higher frequency components by appropriate filtering. Proper use of these controls are essential especially for generating voiced consonants (e.g. voice bars, nasals).

For generating the noise spectra, however, the resonators are connected in parallel. This gives a more realistic noise spectra. The filtered random noise is passed through these

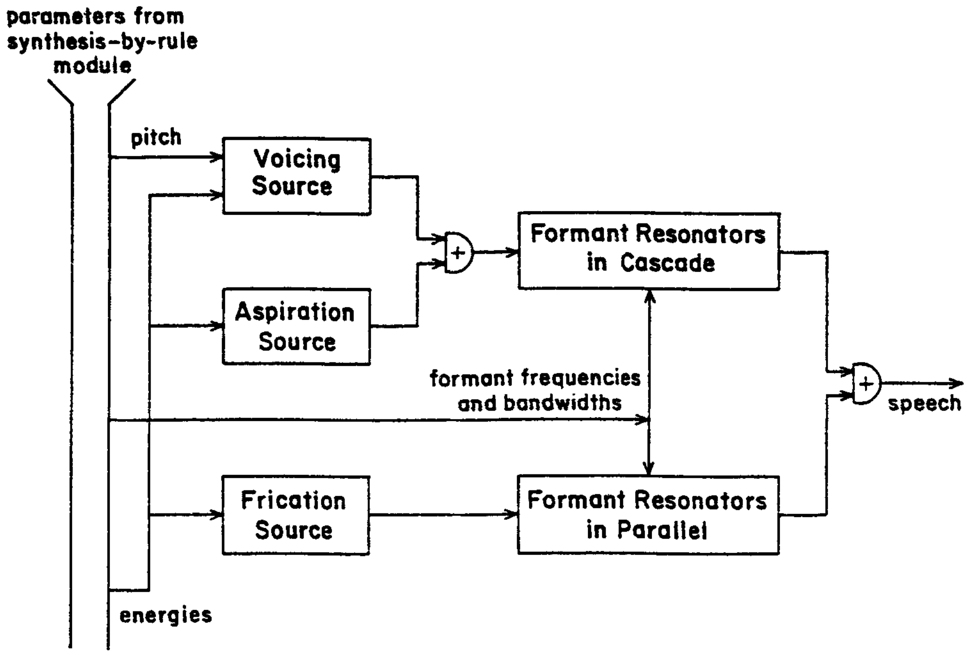
parameters from synthesis-by-rule module
pitch
Voicing Source
Aspiration Source
Fricatiion Source
Formant Resonators in Cascade
Formant Resonators in Parallel
formant frequencies and bandwidths
energies
speech
+
+

Figure 3. Acoustic-phonetic speech production module schematic.

parallel resonators and the amplitude of each resonator is tuned to generate the desired noise spectra with specific energy concentration patterns.

For generating 'aspiration' sound (the one found in /h/ or in aspirated consonants), the random noise is instead passed through the cascade resonators. As aspiration is essentially frication near the vocal fold and the noise passes through the entire vocal tract (unlike other constrictions which are made somewhere within the vocal tract), it is better modelled this way.

Currently, six resonators and one anti-resonator are being used.

Figure 3 gives a block level representation of the parameter to speech conversion scheme.

6. Formant based rules

For parameter to speech conversion, the production model needs control parameter streams whose time-varying values correspond to any arbitrary input phoneme sequence. To produce the streams automatically, a comprehensive set of rules are employed. The techniques for their formulation and organization are described hereafter in some detail.

6.1 *Indigenous development*

The rules are evidently language-dependent. To state a few differences between Indian languages and English: (i) Many Indian languages (e.g. Hindi) have separate aspirated and unaspirated consonants. (ii) Likewise, there are separate retroflexed and non-retroflexed consonants in Indian languages. (iii) The criteria for distinguishing voiced and unvoiced consonants are not identical for Indian languages and English. (iv) Most Indian languages have nasalized vowels, which do not feature in English (although they are present in some European languages like French). (v) Also, the American/British accent is not fully acceptable to native Indian listeners.

As a result, a totally indigenous set of rules had to be evolved and this activity was at the centre of our attention from the start. The task was made even more difficult due to the lack of adequate information regarding the organization of rules in other languages, presumably because of proprietary problems.

6.2 *Classifying phonemes and parameters for reducing rules*

As mentioned earlier, the acoustic-phonetic manifestations of phonemes are context dependent. In order to capture their variations, the rules must also be context sensitive. If we consider even a diphone context and consider the independent variation of each and every parameter, clearly a combinatorial explosion will result.

The speech production model has about 40 control variables. However, quality synthesis is possible by keeping many of them constant throughout the utterance and varying about 15 to 20 parameters only. Even then, if there is a rule for concatenating each varied parameter for each diphone context, more than 50,000 rules would be needed.

Fortunately, phonemes can be classified into types which exhibit similar transition characteristics. This was first shown by Liberman *et al* (1959). Rao & Thosar (1974) extended the concept to Indian phonemes. Phonemes can be classified according to their manners of articulation or places of articulation or both. We have classified them according to their manners of articulation in order to take advantage of the obvious similarities in steady-state and transition characteristics of phonemes thus classified.

Theoretically, further reduction of rules and phoneme data storage by classifying the consonants according to their places of articulation (velar, palatal, retroflex, dental, labial) is possible. The resulting reduction can be cost-effective in the methods where stored speech segments are used. However, in our method which organizes the rules in a compact manner (§ 6.4) and uses minimum amount of data for each phoneme, the gain is insignificant. For example, phonemes with the same place of articulation can use the same locus equations (§ 6.6). The saving in memory storage for the locus equation coefficients would be very small and would hardly compensate for the sacrifice of the accurate representation of the formant movements of these phonemes.

After considering the costs and benefits, further sub-classification of the phonemes according to their places of articulation was not attempted. The phoneme classes selected solely on the basis of the manners of articulation are: (i) silence (ii) vowels (iii) stops (iv) affricates (v) nasals (vi) semi-vowels (vii) trills/flaps (viii) laterals (ix) fricatives (figure 1).

Consequently, concatenation rules are to be formed according to the classes of adjacent phonemes and this results in a phenomenal reduction in the number of rules. Further reduction was also done by dividing the control parameters into groups which exhibit similar variation patterns in any given context (e.g. frequencies and bandwidths of all formants).

6.3 *Steady-state and transition segments*

For convenience of design, the duration of each phoneme is divided into a central steady-state segment and a transition segment at each end. The 'steady state' is steady in a comparative manner only. It is like a central reference segment for a phoneme from where we can proceed to construct parameter trajectories at each end. The steady-state segments may also be influenced by context and may embody transitions, but to a lesser degree in comparison with the transition segments. The transition segments, on the other hand, connect the steady-state segments of adjacent phonemes. For example, for a vowel-vowel combination the transition is done by directly interpolating the steady-state formant values of one to the other. In the case of stop consonants and affricates, the closure period is considered as the steady-state segment while the burst/frication and the formant-movement regions are collectively designated as the transition segment. Separate sets of rules are applied for constructing parameter trajectories in steady-state and transition segments.

6.4 *Compact and flexible organization of rules*

In anticipation of substantial additions and refinements of rules, these were organized in such a way as to give substantial amount of flexibility as well as compactness and this is one of the salient features of our synthesizer. This is implemented essentially by means of a set of two-dimensional tables (corresponding to each parameter class) which control the selection of 'interpolation types' (both for steady-state as well as transition segments) depending upon the classes of the adjacent phonemes. An 'interpolation type' specifies the manner in which a specific parameter in a specific context will vary. For example, it can linearly interpolate to the next target (as formant frequencies mostly do), can have a sharp change followed by a more gradual change (as does amplitude of voicing for a stop-vowel combination) or can just jump to the next target (as the amplitude of frication does at the plosive burst of a stop consonant). The interpolation type at a given context can be changed by just altering a few table entries. New interpolation types can be added, if needed, with little extra effort. It is even possible to change the classification of phonemes and parameters (e.g. if we want to put voiced and unvoiced stops as different 'types', a few more entries are to be added). In this way, a high degree of flexibility is attained. It is also clear that it makes the 'rules' very compact, as they are basically embodied into tables and the implementation of a handful of interpolation types.

However, it will be often necessary to recognize the 'independent' feature(s) of some phonemes which could not be captured by the class characteristic. Such situations are taken care of by 'exception' rules. For example, from among the fricatives, the formant frequencies of only /h/ show an affinity towards the adjacent vowels and this 'allophonic variation' is to be taken care of by rules explicit to /h/. As the number of exception rules could be kept quite small, the general organization, based on the classification of phonemes and parameters, is vindicated.

Whereas the bulk of the rules in this system are implicitly embodied into tables and their implementations, only the 'exception' rules are specified in explicit if-then-else constructs.

6.5 *Derivation of rules and extraction of phoneme data*

Rules are mainly derived using the following steps:

1. Formation of the framework for rule organization by the selection of suitable groups for phonemes and parameters.
2. Selection of a set of 'interpolation types' for capturing the variation of the parameters in the contexts of all possible phonemic classes.
3. Provision of a means of determining which 'interpolation type' is to be selected in a given phonemic context. This is done for each parameter group by means of a two-dimensional table for the steady-state segments and another for the transition segments.
4. Implementation of the 'interpolation types'.
5. Derivation of 'exception' rules.

Each of these tasks calls for suitable acoustic-phonetic knowledge. No doubt, analysis of speech and measurement of its features form the basis of acquisition of such knowledge. However, the rules and their framework cannot be directly constructed from the measured data. They are rather obtained by the interpretation of data and application of intuition (heuristics). The rules thus theorized are validated and refined by actually using them for synthesis and testing the output by means of perception experiments.

The rules need data, pertaining to each phoneme, to operate on. A table, containing steady-state parameters for each phoneme is maintained. Other important types of data stored are: (a) The burst spectra (amplitudes for each formant frequency during the noise burst) for each stop consonant. If needed, different entries are stored for front, mid and back vowel contexts. (b) Normal steady-state duration of each phoneme. (c) Transition time for each phonemic context and (d) Voice-onset-time (VOT) for each stop consonant. This is the time difference between the noise burst and the starting of voicing.

Most of these data are initially estimated from spectrographic analysis of real speech. They are however refined through trial and error for best quality synthesis.

6.6 *Locus equations*

Among the variable parameters, formant frequencies are the most important ones for the perception of particular utterances. Generating formant trajectories accurately is therefore of utmost importance.

It is observed that the terminal (entry/exit) formant frequencies for a phonemic segment are not fixed, but are considerably influenced by the context. For example, if a phonemic segment /p/ is preceded by the vowel /u/, then the second formant frequency at the onset of stop will be very low (about 600 Hz) whereas it will be considerably higher (about 1400 Hz) if the preceding vowel is /i/. Several methods to capture this variation of the terminal formant frequencies for a consonant corresponding to different vowels have been described in literature. We used Klatt's (1987) 'modified locus equation', which is given by:

$$F_{onset} = F_{locus} + k(F_{vowel} - F_{locus}). \tag{6}$$

Here, F_{onset} is the formant frequency either while entering into a vowel from a given consonant or while exiting from a vowel into the same consonant. F_{vowel} is the steady-state formant frequency of the vowel. Frequency F_{locus} and constant k are characteristic features for any given consonant. To find them, F_{onset} corresponding to as many F_{vowel} frequencies as possible are found and plotted for each consonant. As (6) is linear, the best straight line through all the plotted points gives the best estimate for F_{locus} and k. Set of values for k and F_{locus} are kept for all consonants and for each variable formant. Separate values can be stored for CV and VC contexts. Also, provision for keeping separate values for front, back and rounded vowels exists.

6.7 *Duration rules and intonation*

A few elementary duration rules which depend on the phonemic context and not much on the linguistic context were applied. These include rules for shortening of a vowel adjacent to a stop consonant and for pre-pausal lengthening of a phoneme.

As for intonation, a flat pitch is used except for the last vowel when the pitch is raised or dropped, depending on whether it is an interrogative or assertive sentence (it is determined by the presence/absence of an interrogation mark).

6.8 *Steps for generating parameters by rule*

After briefly viewing the different aspects of the rule system, now we finally come to the various steps for generating the parameter streams (figure 4):

1. The phoneme string is parsed, each phoneme is identified and associated with its class. Silence is implied both at the beginning and at the end of the phoneme string. Elementary rules like the singling out of geminates are applied. (Each geminate is replaced by a single phoneme, with increased duration).
2. The intrinsic steady-state and transition durations for each phoneme are found from tables. Rules are then applied to modify them according to the context.
3. For a given variable parameter and for a given phoneme, the transition segment is first interpolated between the steady-state of the previous phoneme and the steady-state

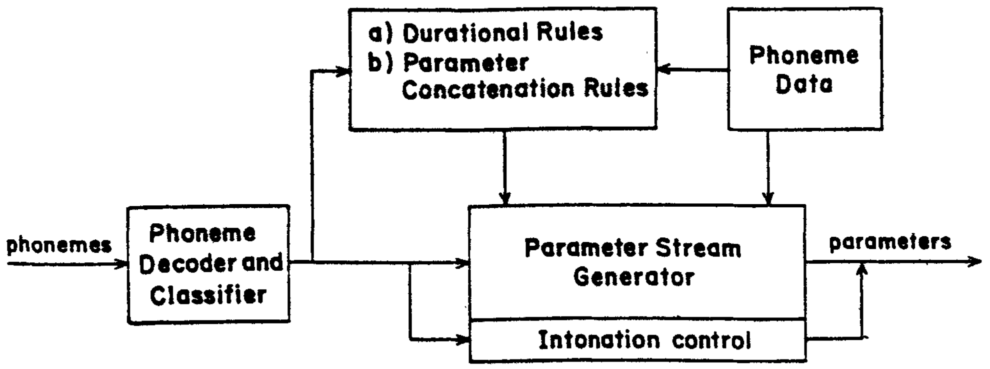
a) Durational Rules
b) Parameter Concatenation Rules
Phoneme Data
phonemes
Phoneme Decoder and Classifier
Parameter Stream Generator
Intonation control
parameters

Figure 4. Synthesis-by-rule module schematic.

of the current one. This is done by using the interpolation type corresponding to the phoneme class context (i.e. previous phoneme class and current phoneme class). Then the steady-state segment is interpolated. This too is done by taking the phonemic context into consideration.

4. Step 3 is repeated for all the variable parameters for the same given phoneme.
5. Steps 3 and 4 are repeated for the first to the last phoneme.
6. Ultimately, the pitch contour is superimposed on the utterance.

There are many important deviations from this general framework, which are taken care of independently. For example, as mentioned earlier, the allophonic variations of /h/ have to be handled as exceptions. Also, the burst energy contour for a stop consonant is fairly complex. So, these are generated by explicit amplitude rules corresponding to the place of articulation of the stop.

Figure 5 shows an example of the operation of the synthesis-by-rule module. The word 'out' in the form of the phoneme string 'aauqt' is given as input. The phoneme string is parsed into its components 'aa', 'uq' and 't' and their corresponding classes (vowel, vowel and stop) are identified. The phoneme durations (which include the steady-state durations) and the transitions are shown. Out of a set of about 20 parameters which are varied in time by rules, the variations of four selected ones (av – the amplitude of voicing and F1, F2, F3 – the first three formants) are illustrated in the figure. The spectrogram of the synthetic speech is shown at the bottom.

Figure 6 shows the spectrograms of the word 'Bambai' spoken naturally (at the top) and synthesized by our system (at the bottom) for comparison.

7. Performance

Unfortunately, no clear standard for evaluating the performance of a synthesizer has evolved internationally, with the leading laboratories conducting performance evaluations suiting their own requirements. For synthesis of Indian speech, any performance evaluation result is yet to be published.

The performance of a synthesizer can be described in terms of intelligibility, quality and speed. As we have emphasized on intelligibility at the current stage, we evaluated it in somewhat rigorous manner with the 'segmental intelligibility test' as performed by Carlson and Granstrom of KTH (Carlson *et al* 1990). For this, 102 VCV syllables were synthesized with three 'extreme' vowels /a/, /i/ and /u/ and 34 consonants (including semi-vowels) which are acceptable to our synthesizer. (We left out /v/ because in Indian languages it is quite often pronounced as /w/.) The syllables were played in a random sequence to 8 listeners who were not much exposed to synthetic speech earlier and who were given the opportunity to listen to each utterance only once. The listeners were asked to identify the consonants and the error rate in identifying them was 45.7%.

To compare with the internationally reported results, the KTH synthesizer, under similar test conditions, had an initial error rate of about 42% which was improved to about 12% over several years (Carlson *et al* 1990). Under less stringent closed response (multiple

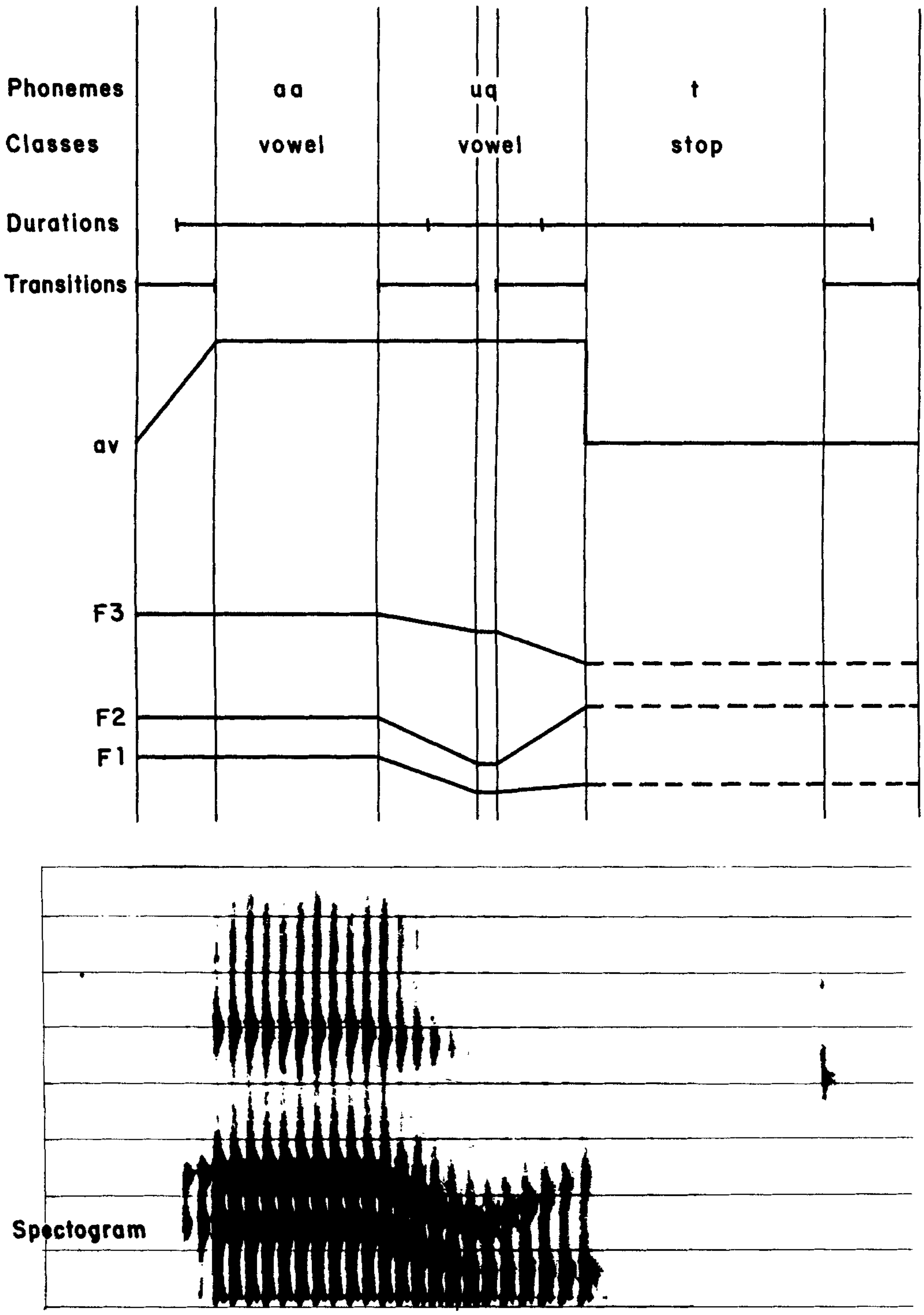

Figure 5. Synthesis-by-rule: an example.

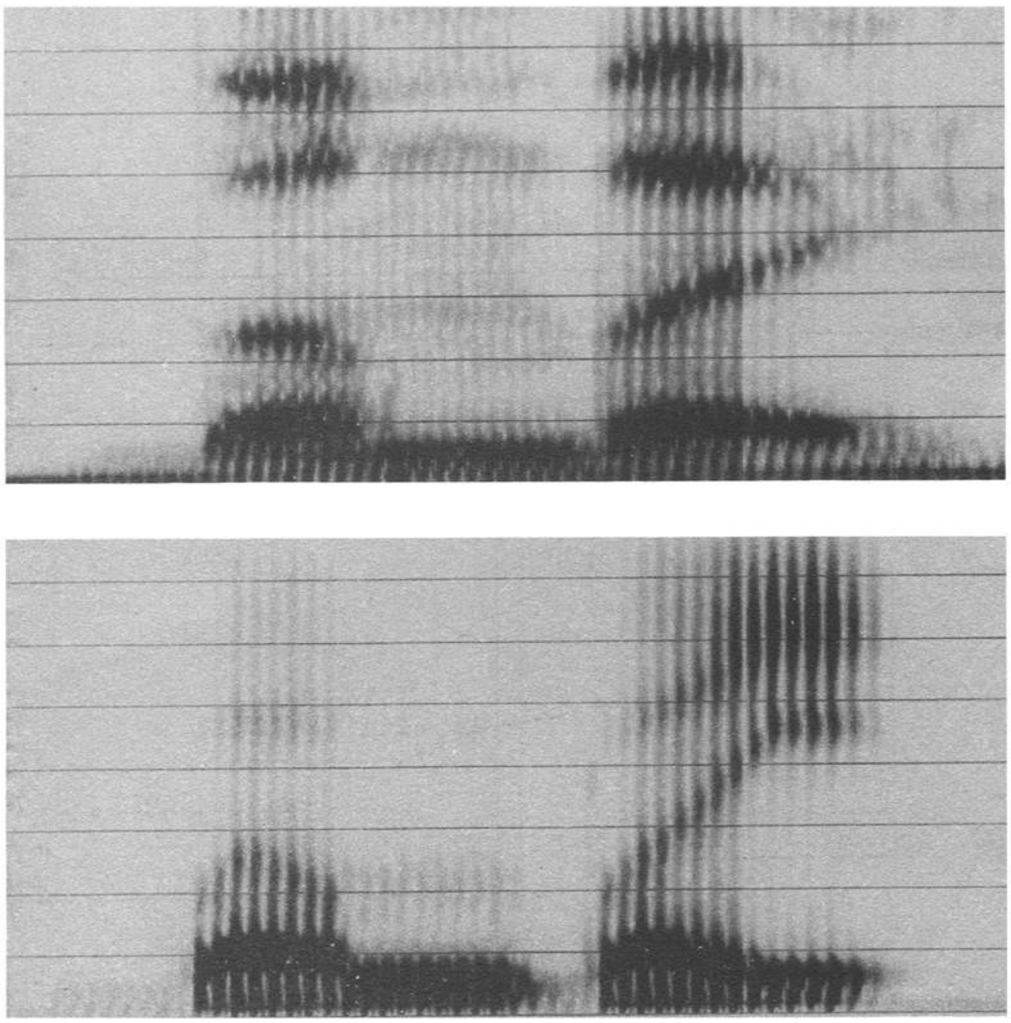

Figure 6. Spectrograms of the word 'Bambai': natural speech (at the top) and synthetic speech (at the bottom).

choice) tests, the error rate in identifying consonants varied from 27% (e.g. Type-n-talk) to 3% (e.g. DECtalk)(Klatt 1987).

As mentioned earlier, there is no previously reported result on Hindi speech to compare with. However, it can be argued that the error rates are likely to be more for Hindi due to the presence of many more consonants (e.g. there are 16 stop consonants in Hindi as against 6 in English) and their greater confusabilities (e.g. between the retroflexed and dental consonants and also between the aspirated and non-aspirated consonants). All considered, the error rate of our synthesizer is moderately high, but compares favourably with the initial error rate of the KTH synthesizer under similar test conditions.

From among the errors, some are due to inexact implementation (e.g. the error rates for nasals and affricates were high) and improvements were undertaken as per the perception test results. However, the closeness of some phoneme pairs [e.g. /r/ and the retroflexed flap (d5 in figure 1), /f/ and the voiceless, aspirated labial stop (p2 in figure 1)] puts a practical limit on the reduction of error rate at syllable level. They are expected to be better perceived at a word or sentence context.

It is expected that, in general, the perception accuracy for nonsense syllables will be much less than the corresponding value for meaningful sentences generated by the synthesizer. To test this, we played out the synthesized address of an unfamiliar building and the error rate in identifying the phonemes in this 'global acceptance test' was 6.8%. This establishes our estimation that for any practical use, the intelligibility of the synthesizer is acceptable.

8. Applications

Speech synthesizers, especially in the form of comprehensive text-to-speech systems, have a wide range of potential applications. These include public announcements, voice-mode computer tutors (specifically, language tutors) and various aids for the speech handicapped and the blind. The last mentioned category of applications is described in some detail in Klatt's (1987) review paper.

However, one category of potential applications we want to particularly emphasize is the access to information over telephone from a computerized information base. With the current 'information revolution' and the explosion of computer and communication networks, the demand for automatic access to information of public utility (e.g. railway reservation status, flight schedules, latest stock market quotations) by dialling will rapidly increase. The real-time update of the information base will rule out pre-recorded messages or even manual response. The usage of fast and quality synthesizers will be the only solution. Even when the information base is not updated very fast, a text-to-speech synthesizer may be advisable because text occupies much less storage than the speech waveform. The access to the specific information needed can be done by a multi-level menu system and can be implemented by either touchtone buttons or by a small vocabulary voice recognition system.

9. Conclusions

This paper describes our source-filter model-based synthesizer against the backdrop of the current advances in the area of speech synthesis in Indian languages. At least one concatenation-based real time text-to-speech system is now available (Bhaskararao *et al* 1994). It is expected that the work reported here will clear the way for formant synthesizers, with better potential for quality and versatility, to appear in the scene.

The major hurdle has been successfully overcome with the development of a comprehensive set of rules for synthesizing unlimited speech in some Indian language(s). A standard implementation of the source-filter model was found to be adequate as the basic tool for such synthesis. A formant-based text-to-speech synthesizer should be the next logical step.

For that, incorporation of text-to-phoneme conversion is necessary. Initiating research and development in that direction is in our future agenda. Any readily available module can also be plugged in.

For meaningful applications, the synthesis should be done in real or near real time. This is a hardware engineering problem and an acceptable solution with the current state-of-the-art is possible.

That a rule-driven formant synthesizer can generate quality speech in Indian languages has been amply demonstrated by our work. However, we feel that the full potential of the synthesizer is far from being utilized. Taking advantage of the flexible rule structure, quality is being improved continually. The work includes refinement of phoneme concatenation rules, and better incorporation of prosody and co-articulatory effects.

The satisfactory synthesis of the female voice poses many additional problems in any language. Detailed study of the female voice is being done by us for better implementation.

Taking advantage of the versatility of the synthesis techniques employed, extension of the synthesizer to other Indian languages is being contemplated.

We gratefully acknowledge the support provided through the Knowledge Based Computer Systems (KBCS) project in carrying out this work. We thank the late Dr D H Klatt, Prof. K N Stevens and other members of the Speech Communications Group, MIT, USA for helping us to learn the latest techniques of speech synthesis when we visited their laboratory. We also thank Prof. P V S Rao for his support and encouragement.

References

Bhaskararao P, Mathew S 1992 Phonemic transcription rules for text-to-speech synthesis of Hindi. *Computer processing of Asian languages* (ed.) R M K Sinha (New Delhi: Tata-Mcgraw Hill) pp 310–311

Bhaskararao P, Peri V N, Udpikar V 1994 A text-to-speech system for application by visually handicapped and illiterates. *Proc. of the ICSLP 94*, Tokyo, Japan, pp 1239–1241

Carlson R, Grantstrom B, Nord L 1990 Evaluation and development of the KTH text-to-speech system on the segmental level. *Proc. Int. Conf. Acoust., Speech Signal Process. 90*, Toronto, vol. 1, pp 317–320

Dan T, Datta A K, Mukherjee B 1990 Speech synthesis using signal concatenation. Presented at the Workshop on speech technology for man-machine interaction, Tata Inst. Fundam. Res., Bombay

Holmes J N 1983 Formant synthesizers: cascade or parallel. *Speech Commun.* 2: 251–273

Holmes J N, Mattingly L, Shearme J 1964 Speech synthesis by rule. *Language Speech* 7: 127–143

Klatt D H 1980 Software for a cascade/parallel formant synthesizer. *J. Acoust. Soc. Am.* 67: 971–995

Klatt D H 1987 Review of text-to-speech conversion for English. *J. Acoust. Soc. Am.* 82: 737–793

Liberman A M, Ingemann F, Lisker L, Delattre P, Cooper F 1959 Minimal rules for synthesizing speech. *J. Acoust. Soc. Am.* 31: 1490–1499

Rajesh Kumar S R, Sriram R, Yegnanarayana B 1989 A new approach to develop a text-to-speech conversion system for Indian languages. *Proc. of the regional workshop on computer processing of Asian languages*, Bangkok, pp. 102–109

Rao P V S, Thosar R B 1974 A programming system for studies in speech synthesis. *IEEE Trans. Acoust., Speech Signal Process.* ASSP-22: 217–225