Lab10: Digit Recognition

EECS20: Introduction to Real-Time Digital Systems

By K. H. Chiang, William T. Huang, Brian L. Evans.

URL: `http://www-inst.eecs.berkeley.edu/~ee20`
News: `ucb.class.ee20`

Assigned: 16 Apr 97, Checkoff: 23 Apr 97, Writeup Due: 25 Apr 97

1 Introduction

Voice recognition systems can be categorized according to three criteria:

- speaker independent vs. speaker dependent: speaker independent systems will recognize speech spoken by any individual, whereas speaker dependent systems are usually designed to handle only the speech from a single individual. Speaker independent systems are generally harder to design than speaker dependent ones.
- small vs. medium vs. large vocabulary: the smaller the vocabulary that needs to be recognized, the easier the recognition task is.
- isolated word vs. continous: isolated word systems require pauses to occur between words. This makes recognition a much easier task.

In this lab, we will implement a speaker dependent, small vocabulary, isolated-word recognizer. Specifically, we will attempt to recognize the digits from 0 through 9.

We follow the approach outlined in Figure 1:

1. A given speech sample, or utterance, is converted to a pattern.
2. This pattern is compared to all the patterns in a database of previously calculated candidate patterns.
3. The digit associated with the candidate pattern that most closely matches the pattern of the utterance is presumably the spoken digit.

This still leaves us with the problems of how to generate patterns, and how to make the comparisons between the patterns. We first address the former.

2 Pattern Generation

The pattern generation can be broken down into four major steps, as illustrated in Figure 2: endpoint detection, segmentation, filtering, and interstate determination.

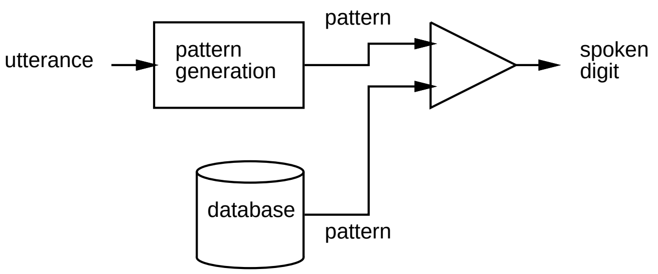

Figure 1: Content-free high level block diagram.

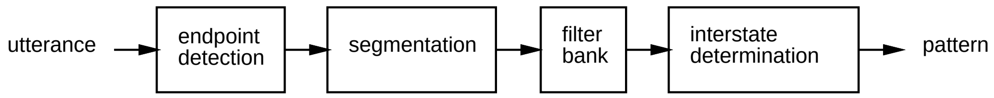

Figure 2: Pattern generation.

Exercise: Before addressing endpoint detection, refamiliarize yourself with the sampling process on the PC. Make sure that you can record a speech sample and read it into Matlab using the `wavread` command. In addition, after reading the sample into Matlab, don't forget to subtract 128 from the entire speech sample to normalize the speech around 0.

2.1 Endpoint Detection

Endpoint detection is accomplished by the process illustrated in Figure 3. The classic endpoint detector performs a number of operations:

1. The envelope of the input speech signal $s[n]$ is found by taking the absolute value of $s[n]$ and then low pass filtering through a first order LPF. This gives $e[n]$. A suggested transfer function for the LPF is:

$$H(z) = \frac{1}{1 - \frac{15}{16}z^{-1}}$$

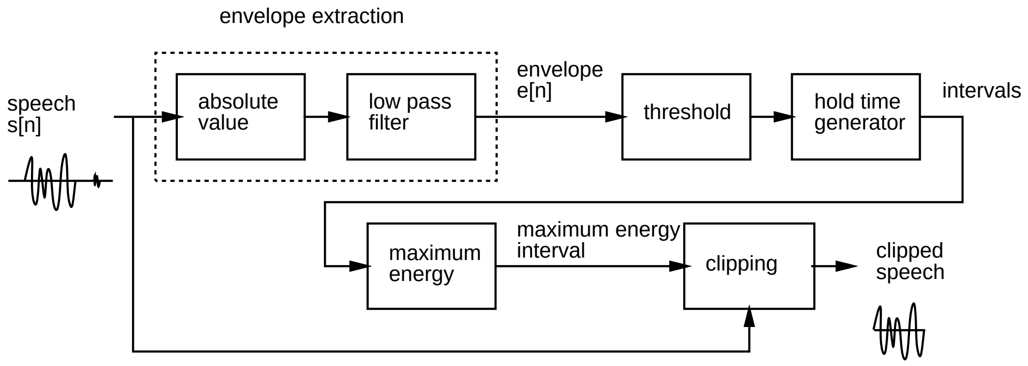

Figure 3: Endpoint detection.

Following the notation of the `filter` command, the `b` vector is `[1]` and the `a` vector is `[1 -15/16]`.

2. The envelope $e[n]$ is then passed through a threshold detector. This gives the values of n for which $e[n]$ is greater than some threshold value. This threshold value should be determined by recording a speech sample and a sample of background noise, and then finding some value greater than the noise sample's envelope and less than the speech sample's envelope.

3. Once the n that correspond to $e[n]$ greater than the threshold value have been found, a hold time generator takes those values of n and throws out all the n that are within some number of samples of each other. A value of 1000 is suggested for this hold time; if the sample rate f_s was 8 kHz, this would correspond to 125 ms.

   Starting from the beginning of the list of values of n and throwing out points will give the starting points of any intervals possibly containing speech, whereas starting from the end of the list of values of n will give the ending points of any such intervals.

4. To differentiate between intervals that contain noise and intervals that contain speech, compute the energy in each candidate interval by take the sum of the squares of the samples. The speech interval should have the maximum energy.

2.2 Segmentation

One of the problems in speech recognition is that the same word may be pronounced at different speeds. For instance, a person could say a given digit very quickly or very slowly, but we would still like to recognize that digit.

To deal with this, after removing the zeros at the beginning and end of an utterance, we segment the speech into sixteen separate, nonoverlapping segments. This is easily done in Matlab with proper indexing. We hope that even though a digit can be spoken at different speeds, the nth segment of a slowly spoken digit contains the same information as the nth segment of a quickly spoken digit.

2.3 Filter Bank

Each of the sixteen segments is then filtered with a bank of 19 filters.

The coefficients for the filters are provided for your filtering pleasure. Loading the filter bank is accomplished by the command `load filtbank`, assuming that the file **filtbank.mat** is in the current working directory. This loads the matrix `abank` and the matrix `bbank`. Each row of `abank` contains the `a` vector of a DT filter, and each row of `bbank` contains the `b` vector of a DT filter, following the naming convention of the `filter` command.

The magnitude responses of the filters are given in Figure 4.

2.4 Interstate Determination

Another problem in speech recognition is amplitude normalization; a person can say the same word with the emphasis on different syllables, or at a completely different overall volume. To remedy

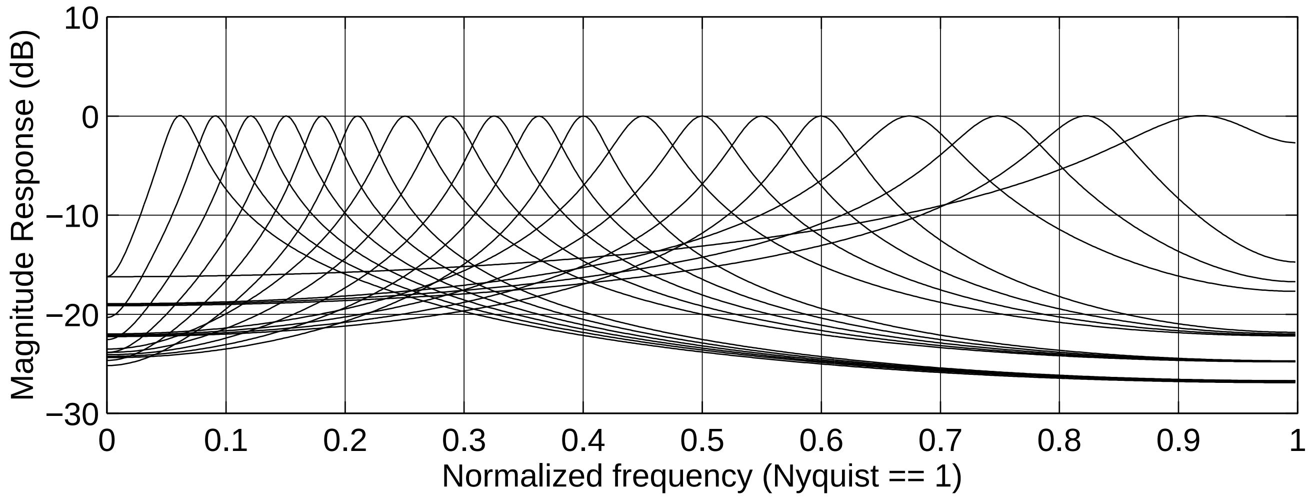

Figure 4: Filter bank magnitude response.

this, we use a technique termed "interstate determination."

After filtering, we have 19 vectors of filter outputs for each of the sixteen segments. To get that data down to a more reasonable size, we determine the energy in each of the filter outputs by taking the sum of the squares of the samples. We then compare the energy of the second filter with the first. If the energy of the second filter is greater than the first, we represent this as a 1. Otherwise, it's a 0. We repeat this pairwise comparison for the third and second filter output energies, and so on.

So, after the endpoint detection, segmentation, filtering, and interstate determination, we end up with 18 ones and zeros in a given segment. With sixteen segments, we then have 288 numbers in the pattern for a given utterance.

3 Pattern Matching

If three samples are recorded for each digit and the corresponding patterns are generated, we would have thirty patterns in a database of reference digits.

Now, we can use the database by recording a new sample, generating its pattern, and comparing this pattern to each of the candidate patterns in the database. The comparison can be performed by counting the number of ones and zeros which do not match. We hope that the digit whose pattern is the closest match is the spoken digit, closest match being defined as the lowest number of mismatched ones and zeros.

4 Questions

1. Write the endpoint detector following the description above.

2. Write a function that takes a speech sample and generates the corresponding pattern.

3. Create a database by recording three samples for each digit and generating the corresponding patterns. Note that the filters were designed for a sampling frequency of 8 kHz; use the `SoundOLE` application in the `Audio` group for recording.

4. Write a pattern matching function.

5. Test your digit recognizer. You should hope to achieve greater than a 10% recognition rate; otherwise you could replace your system with a random number generator.