Voice Verification System

Tian He

Indy Yu

November 19, 2004

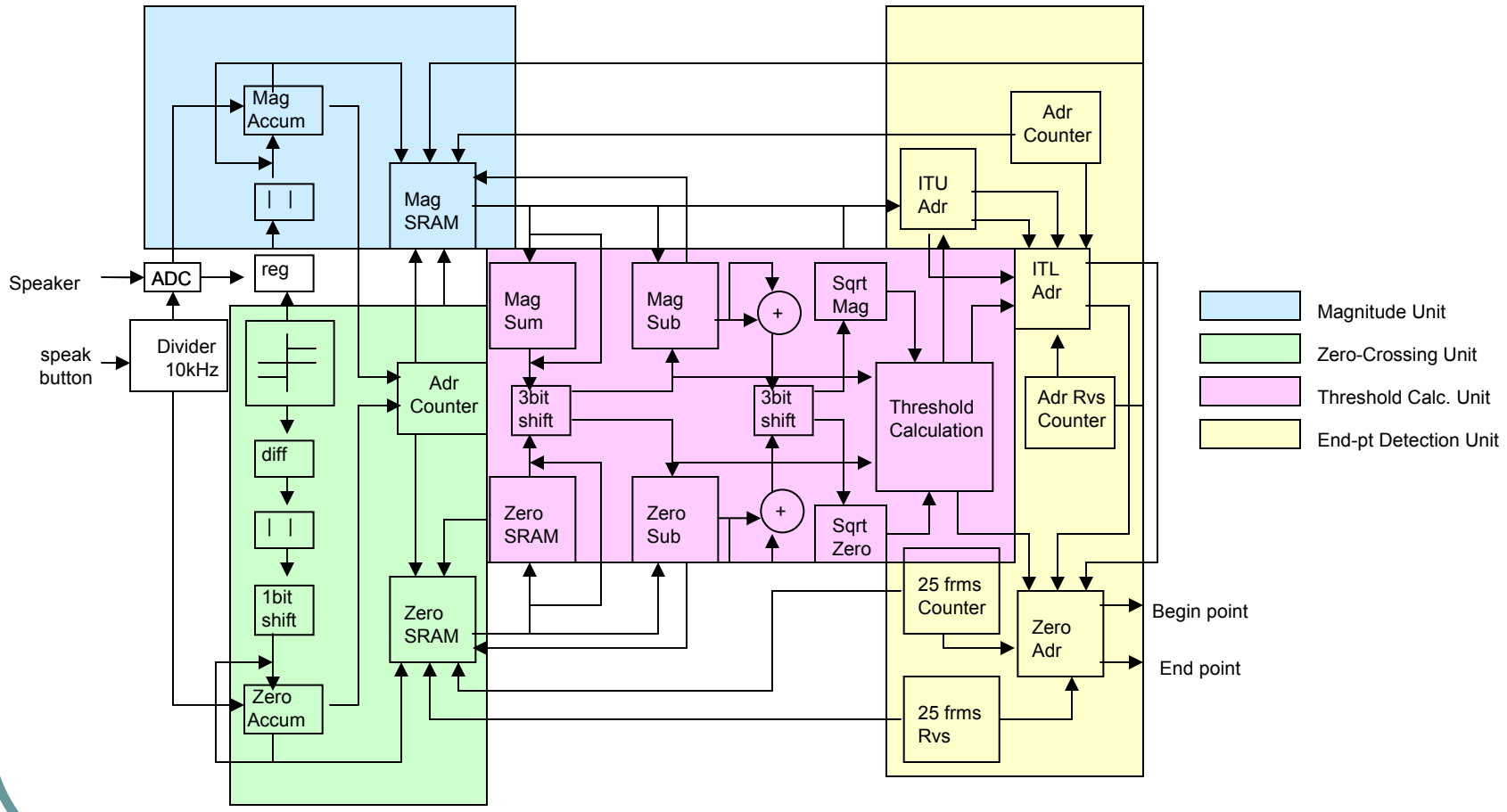
Overview

- The Voice Verification System takes in a speech waveform of a particular password, and performs the necessary computations in time-domain to verify the person's identity.
  - The system is speaker-dependent.
  - The password can be reset at anytime.
  - The system is sensitive to different noise levels.
- Overall Implementation: The speech spectrum is extracted from a microphone input and compared with a password template.

Overall Front-End Function

- The speech spectrum x(t) is sampled at 10kHz by the ADC
- The Magnitude Unit uses x[n] and calculates the magnitude of the speech spectrum in real-time.
- The Zero-Crossing Unit uses x[n] and calculates the rate at which the speech spectrum crosses the horizontal axis in real-time.
- The Threshold Calculation Unit uses noise data to determine threshold values needed to determine the location of the speech spectrum.
- The End-Point Detection Unit uses Zero-Crossing data, Magnitude data, along with the threshold values to find the endpoint locations of the speech spectrum.

Overall Front-End Block Diagram



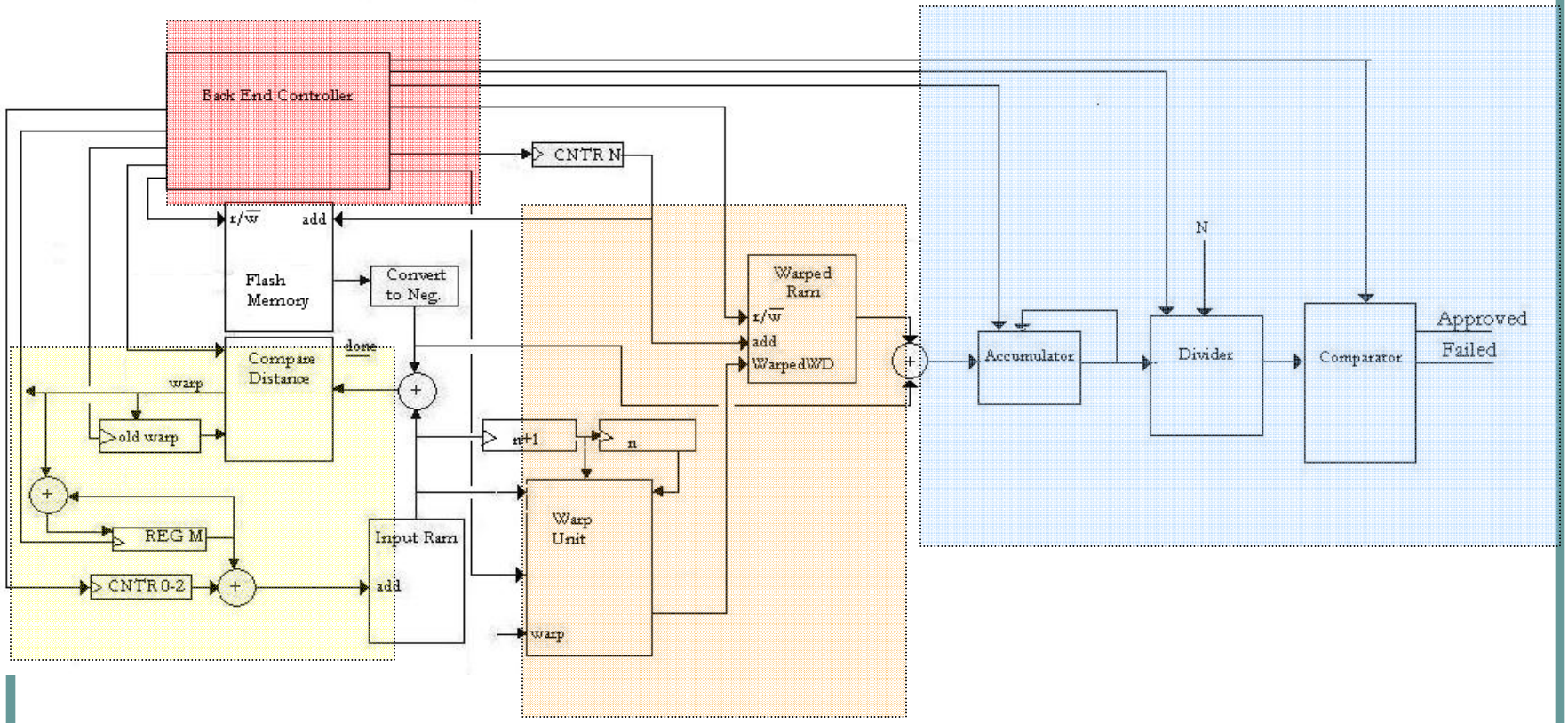
Overall Backend Function

- Takes in processed featured data from the front end
- Time warp the input data to best fit the template data
- For each template value T[n], a value Z[n] is derived from the input data X[m], X[m+1], or X[m+2] to represent the input value for the error calculation T[n]-Z[n]
- An average error is calculated between the template data and time-warped Input data. $\Sigma(T[n]-Z[n])/N$

.

- If calculated error is under a set threshold, then the waveform is verified, otherwise system fails.

Overall Backend Block Diagram



Magnitude Unit

- Convolve |x[n]| with impulse window
  - M[n] = sum(x[m]*w[n-m])
  - w[n] = 1 0<n<100
- Equivalent Implementation: Accumulating the magnitude of 100 samples and storing the weighted sum in a SRAM at 100 Hz
- **Input**: sampled speech signal
- **Output**: magnitude SRAM data

Zero-Crossing Unit

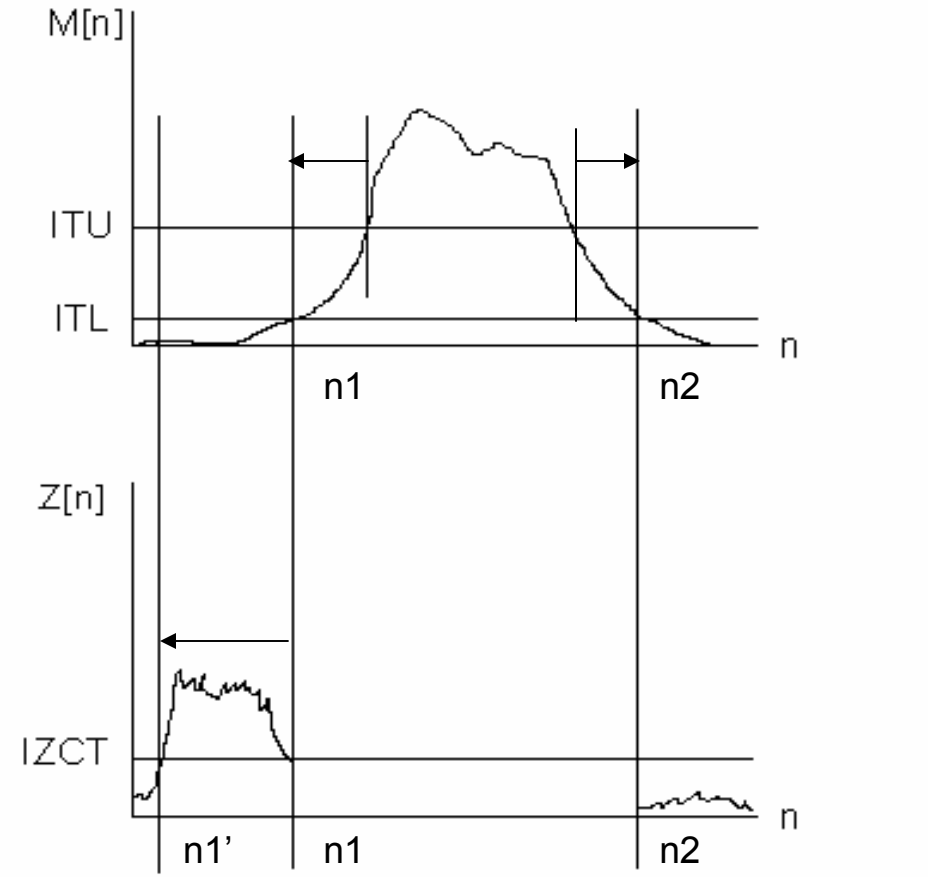
- Convolve |sign(x[n])-sign(x[n+1]| with impulse window
  - Z[n] = sum(|sign(x[m])-sign(x[m+1])|*w[n-m])
  - w[n] = 1/2 0<n<100
  - sign(x[n])=
    - 1 if x[n]>0
    - -1 if x[n]<0
- Equivalent Implementation: Accumulating the number of times the input sample changes sign within 100 frames and storing the value in a SRAM at 100 Hz
- **Input**: sampled speech signal
- **Output**: zero-crossing rate SRAM data

Threshold Calculation Unit

- Three threshold values needed
  - Magnitude spectrum upper and lower thresholds
    - ITU: noise mean* C1
    - ITL: noise mean + C2*noise standard deviation
  - Zero-crossing spectrum threshold
    - IZCT: noise mean + C3*noise standard deviation
- **Input**: first 80ms of magnitude and zero-crossing calculations from SRAMs
- **Output**: ITU, ITL, IZCT

End-Point Detection Unit

1. Determine a point for which the Magnitude Spectrum will begin to exceed ITU, the upper threshold.
2. Shift the point back to where it just exceeds ITL if the magnitude is decreasing. The new value is temporally considered the begin-point of the speech spectrum.
3. Looking back 25 frames in the zero-crossing spectrum from the temporary begin-point, if three of more values exceed IZCT, the begin-point is shifted to the new location where the value was first exceed.
4. Same algorithm is applied to determine the end-point.



Backend Data Processing

The Goal:

Match the newly processed input waveform with the password waveform and decide whether or not the input word matches closely enough with the password.

The Problem:

There is variance in word pronunciation length between samples. The system should be able to recognize ‘Terman’ and ‘Terrrman’ as the same word, within a reasonable time window of course.

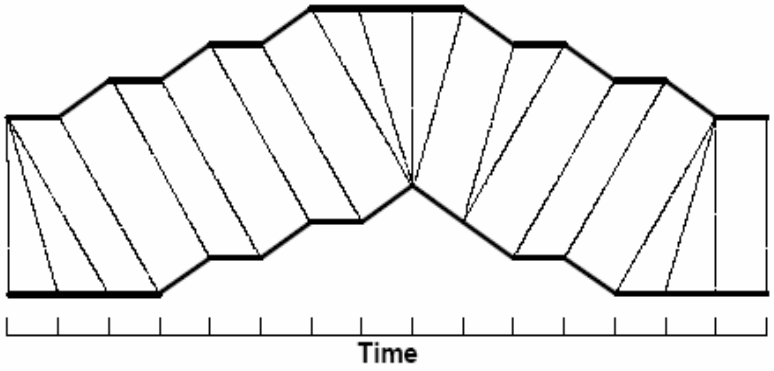
The Solution:

Use the Itakura Method for time warping data.

Itakura Method Overview

-For each sample of the template, map a value determined from the input sample to it.

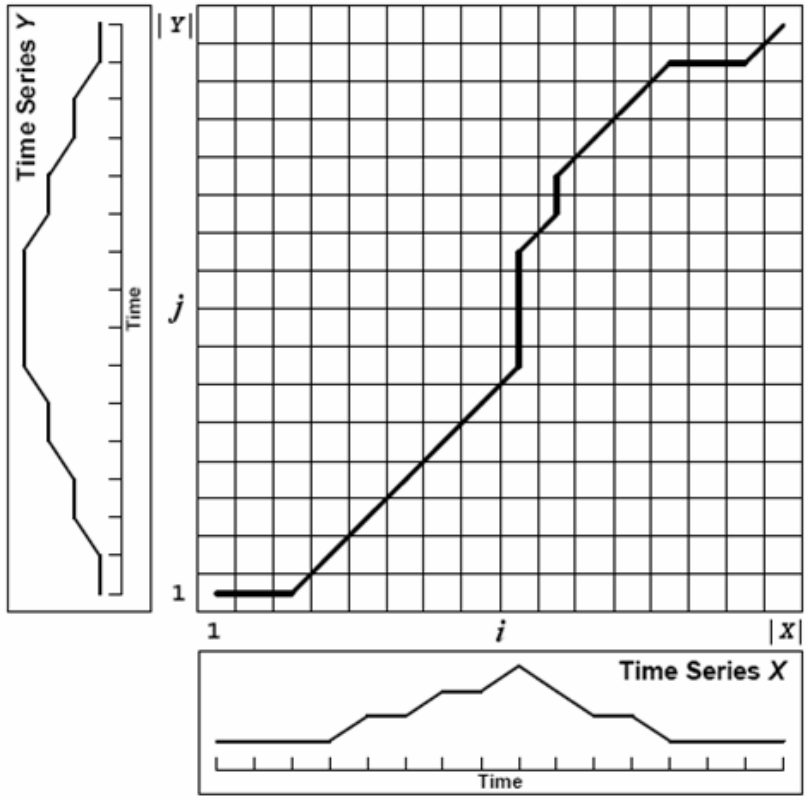
-Constraints: Beginning and ending value of template must be matched to the beginning and end value of the input data.

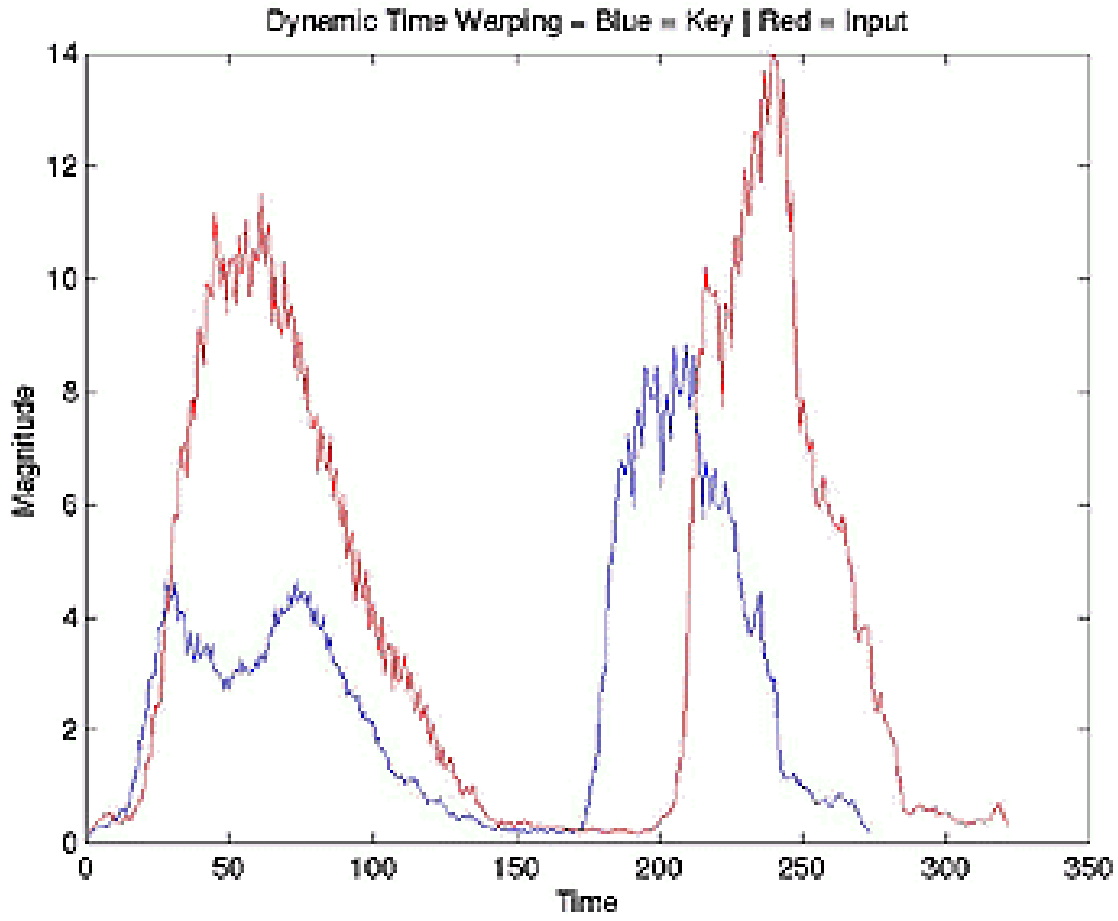


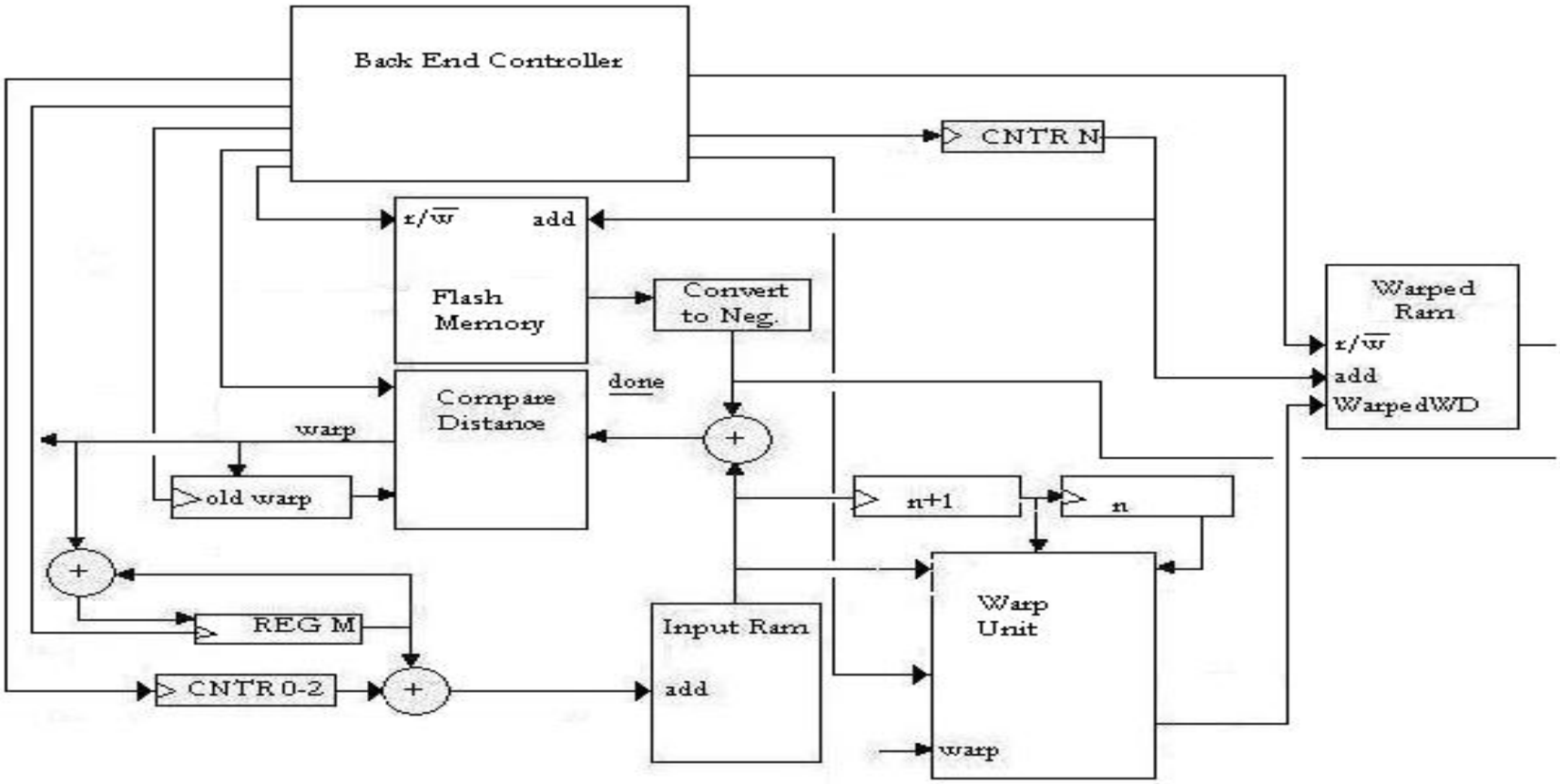
-To select the next point to be mapped to the corresponding template[j], look at all possible paths that can stem from either template [i], [i+1], [i+2] and look for the minimum distance in magnitude.

-To achieve the best result, system must consider minimizing error of overall path, not just minimizing error for each point j.

-For now use local point minimization, expand to more accurate time warping.




Dynamic Time Warping - Blue = Key | Red = Input
Magnitude
Time
0
2
4
6
8
10
12
14
0
50
100
150
200
250
300
350



n=Key_Index
m=Input_Index

Compare Distance

Input:
1. Locale Distance Diff between d(m) & d(n)
2. Previous warp value (0, 1, 2)

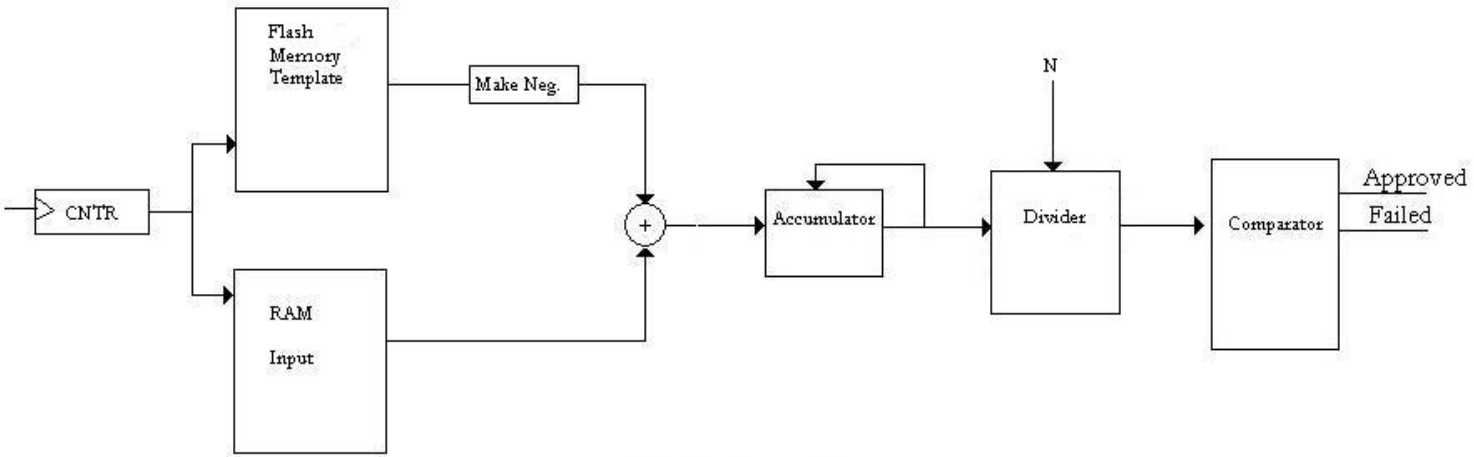
Output:
Best Distance Fit
if(old warp==2) warp either 0 or 1
if(old warp==0) warp either 1 or 2

Controller

Input:
1. warp - shift size
2. Current Ram Value
3. Previous Ram Values

Output:
WarpedWD
One of the Reg Values or
Average of two Reg Values
if there is a skip.

Decision Making Module



Calculate Average Error

Make Neg	Accumulator	Shiftbits	Comparator
Input: TempData Output: ~TempData[MSB]	Input: AccumData Data Output: AccumData+Data	Input: AccumData Output: AccumData[MSB]^N ,AccumData[MSB-1] ,AccumData[MSB-2]	Input: E/N Output: if((E /N) <Thshold) Approved else Failed

Pitch Detection (Optional)

The Goal:

Determine the Pitch Period as another means of verifying the user's voice and spoken password

The Problem:

There are many different noise factors that aren't related to the Pitch Period that could show up in the waveform. How can irrelevant peaks and valley be eliminated?

The Solution:

Lowpass filter and then go through six different period detection impulse trains to look for the fundamental pitch period.

Pitch Detection Diagram