Pitch Wave Generator

Design Document v1.0

Team Members:
Koohee Lee,
Guy Sivan,
Gary Lo

April 3, 2005

Introduction:

The goal is to find a way to see pitch versus time, and through this see the shape and form of the tone of the persons voice in a real-time format. So the question is, how to extract the pitch from a voice recording.

For simplicity at first let us assume that a voice consists of a single sinusoid. One way to find the pitch or frequency at a certain point in time, one can just look closely (zoom into) a wave form and measure the distance between two successive peaks. This distance corresponds to the period of the voice, and so to find the frequency, we just need to take the reciprocal. This is the fundamental concept of our algorithm.

In the real world a voice consists of many sinusoids rather than just one pure sinusoid. These sinusoids are a result of the anatomy of the human vocal system and appear all over the frequency spectrum (at many different frequencies). So what we have a much more complicated signal that consists of the fundamental harmonic and many other harmonics (depending on the sound being made, the person speaking, and other factors). The fundamental harmonic is the main frequency and it is what we define as the "pitch" of the voice. Our project is to find exactly this, and to plot it in real-time to get a graphical view of how the pitch in someone's voice is changing.

In a nutshell, as the user speaks into the microphone a line will be outputted on the screen corresponding to the pitch of their voice. Which should be fun!

Algorithm:

Take in sound wave, chop into pieces("blocks"), measure distance between peaks(fundamental period of voice, T), output value of frequency (1/T) (the pitch) for each piece.

Time/space calculations

There are several feasability calculations that are required to determine whether this can be done given our algorithm and available hardware.

Given:

CPU with clock cycle of 50MHz,
Audio sampling device @ 11kHz,
A/D convertor @ 8 bit output,

Human Voice ~ [1](100Hz –1000Hz)

1We used the following website for refernce – http://www.tnt-audio.com/topics/frequency_e.html, however

Based on these constants the following calculations were used to determine the feasability of our project.

Choosing the block size:

In order to calculate the pitch we need at least two peaks to be within the block we are measuring the pitch of. Given that the block will not always perfectly align with the waveform, the only way we can ensure that at least two peaks are within the block is to set the block size to **three wavelengths** of the lowest possible frequency. As mentioned earlier we chose the lowest possible frequency to be 100Hz. This results in a time block of 30ms:

1/(100 Hz) = 10ms,
3*10ms = 30ms

At a sampling frequency of 11kHz:

(11kHz)*(30ms) = 330 samples

==> Block Size = 30ms, 330 samples

The first step is to take in the sound wave. The A/D will take a signal from the microphone and output an 8bit digital signal at 11kHz which will be shifted into a shift register the size of one block(330 samples). Every 330 samples, the contents of the shift register will be copied into a buffer. The contents in this buffer are what we run our algorithm on to determine for a single point of the output pitch graph.

Running the Algorithm:

The first step is to perform a [2]half-autocorrelation on the data set in the buffer. The result of this autocorrelation will allow us to determine the distance between the two peaks in a time block because they will correspond to the two largest peaks in the autocorrelation. Performing this process on the data from the buffer will leave us with a new set of 330 points of data, the "half-autocorrelation graph". This data is then analyzed as follows to find the distance between the peaks:

1. Take the derivative of the data.
2. Perform the [3]sign function on the data from *step 1*.

---

we used 100Hz as our lower limit because we found this was reasonable through experimentation.

2A half-autocorrelation, is like taking the autocorrelation of a set of data but only keeping half of it. This is possible because the autocorrelation leads to a completely symmetrical function, and keeping a copy both sides of function is not necessary because they are the same. This saves us both space and processing time: the space is half, and the processing time is a quarter of a full correlation. This is a key decision in making the algorithm feasible.

3If x > 0 then x = 1, else if x < 0 then x = -1;

3. Take the derivative of data from *step 2* and multiply by -1.
4. Remove negative zero crossings.
5. Multiply results from *step 4* with data in half-autocorrelation graph one data point at a time and store in new buffer (or same buffer as *step 4* to save space.)
6. (maybe not necessary:) Remove first few points from result in *step 5*.
7. Find the first peak and record its index in the buffer.
8. Use the index from *step 7* and the block specifications to find the distance between nearest peaks and calculate value in time (period) and corresponding frequency (pitch):
   Period = T = (index / (samples per block)) * (block time)
   Pitch = f = 1 / T
9. Output the pitch

Clock Cycle calculation (Autocorrelation):

[4]Orig_N = 330
N = 165

Total number of multiplications: N(2N-1)

= 165*(330-1)
= 54285

Total number of additions: (N-1)(2N-1)

= (165-1)*(330-1)
= 53956

Clock cycles for addition: 1
Clock cycles for multiplication: 5

Total clock cycles required:

= 1*(54285) + 5*(53956)
= 324065

Process running at 50Mhz, implies time per clock cycle of:

= 1 / 50MHz
= 2 * 10^{-8} seconds per cycles

Therefore, total time for autocorrelation:

[4] Orig_N is the number of samples in a time block. However, because we need only half of the autocorrelated data, we can divide this number by two, and perform calculations on the remaining data.

$= 324065 * 2 * 10^{-8}$
$= 6.4813$ ms

This number is well within our range of 30 ms per time block.

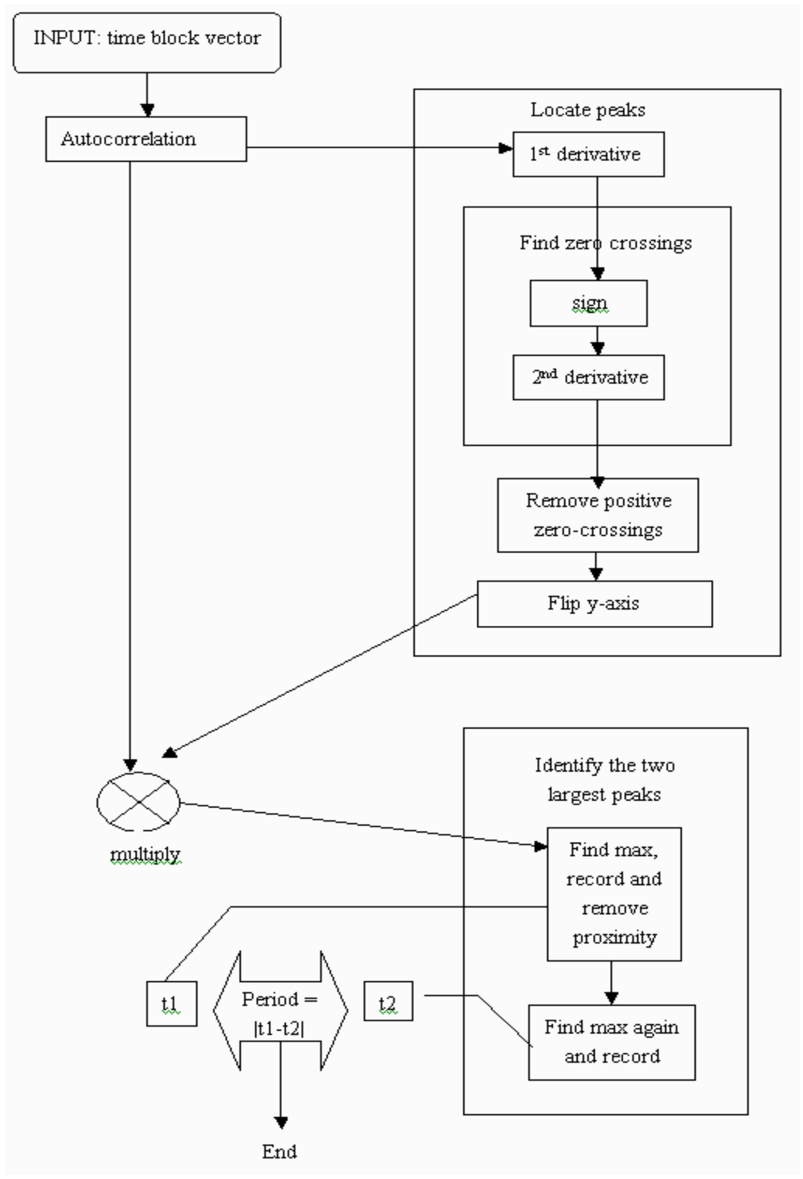

Diagram 1: Block Diagram of Algorithm

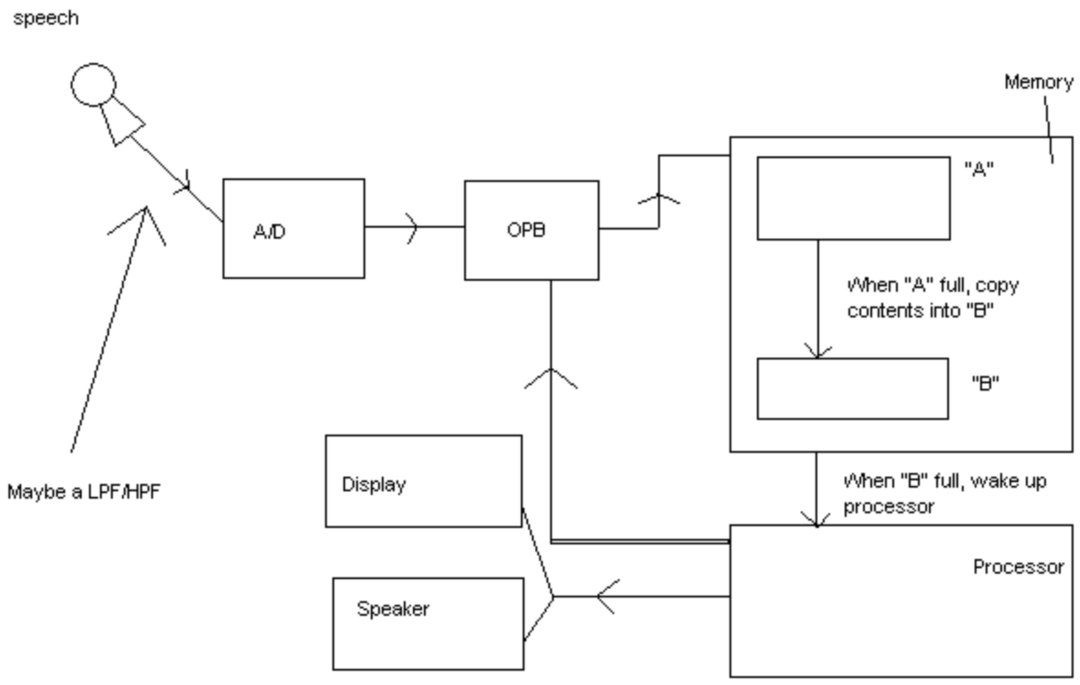
speech
Memory
"A"
A/D
OPB
When "A" full, copy contents into "B"
"B"
Maybe a LPF/HPF
Display
When "B" full, wake up processor
Processor
Speaker

Diagram 2: Block Diagram of System