# Requirements Overview

* Calculate implied tempo from drum strikes detected by piezo sensor connected to iPhone
* Adjustable sensitivity threshold
* Adjustable weighted average of timing over several strikes
* Configurable “strikes per bar” and “Beats per bar” to convert from “strikes per minute” to “beats beats per minute”

# Observations

The recorded audio waveform samples indicate that the sensor produces a wide range of signal values depending on where and how hard the drum is struck.

|  |  |  |
| --- | --- | --- |
| Strike type  (recorded on snare drum) | First Peak Amplitude  (dB from Full Scale) | Time signal remains over -30 dB from peak  (milliseconds) |
| Hard strike on drum surface | -6 | 140 |
| Soft strike on drum surface | -30 | 160 |
| Rim Shot | -3 | 50 |

# Strategy Overview

* Read audio in 20 ms frames
* Apply a running average energy filter over the frame
* If the energy function value exceeds a defined (but adjustable) strike-start threshold, then mark the time as a strike start detection.
* Wait for the energy function to decay to a defined value before allowing another strike to be detected.
* (In perhaps another thread) Take the time deltas between detected strikes to calculate the tempo.



Running Average Energy filter example (4 tap).

P = X(t)^2 + X(t-1)^2 + X(t-2)^2 + X(t-3)^2

Basically square the value of each sample (so as to make it positive, since each sample can be positive or negative valued) and then add a series of these squared values. Some experimentation will be necessary to see how many samples are required. I would guess that 5 will be enough.

After a strike is detected, we then need to continue to scan the buffer for additional strikes, but before another strike can be detected, the current strike needs to decay away. We will need to continue to calculate the running average energy function and when it is less than the STRIKE\_END\_THRESHOLD, we can terminate the “Strike-State.” In a more advanced implementation, we should also time out if a preset time is exceeded.

Constants & Controls

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Type | Value | Description |
| STRIKE\_START\_THRESHOLD | Constant (Int) | TBD | when the running average energy rises and exceeds this threshold we register a strike start. |
| DETECTION\_AVERAGE\_LENGTH | Constant (int) | TBD | Length (in samples) of the running energy average function – value to be determined. |
| STRIKE\_END\_THRESHOLD | Constant (int) | TBD | When the running average falls below this threshold the strike event is deemed to be over |
| FRAME\_SIZE | Constant | 960 | Frame size in samples. 20 millisecond frames will give |
| SAMPLE\_RATE | Constant | 48000 | 48k Sampling will give excellent time resolution |
| STRIKES\_PER\_BAR | Constant | 1-6 | How many strikes are expected per bar |
| BEATS\_PER\_BAR | Constant | 2-6 | Time signature |

Function Running\_Average\_Energy = (index, length, frame[])

{

For (t = start-index; t <= start-index+length; t ++)

Energy += buffer[t]\*buffer[t];

Return Energy

}

## Scan Frame for Strikes – example

Loop Start

Read Frame[]

Loop Start

For (s = 1, s <= FRAME\_SIZE, s++) {

Energy = Running\_Average\_Energy (s, DETECTION\_AVERAGE\_LENGTH, Frame[])

Case Strike-State = FALSE {

If energy > start-threshold then {

Strike-state=TRUE

Strike-delta = current\_sample\_index – Strike\_start\_index

Strike\_start\_index = current\_sample\_index

}

Loop

}

Case Strike-State = TRUE {

If energy < strike-end-threshold then {

Strike-state=FALSE

Loop

}

// note : Store last n samples in previous\_frame\_energy array so that the energy function will correctly have samples from the previous frame for the first n-1 (e.g. 4) samples.

Loop End

[Perhaps in another thread] Calculate Strikes-per-minute From detected\_strike\_array

Loop end

## Time Delta values to Tempo

The next step is to take an array of time-deltas.

The time deltas will be in sample periods (48,000 per second).

Delta\_in\_seconds = Delta\_in\_samples/SAMPLE\_RATE

Strike\_freq\_in\_hertz = 1/Delta\_in\_seconds

Strikes\_per\_minute = Strike\_freq\_in\_hertz \* 60

Bars\_per\_minute = Strikes\_per\_minute / STRIKES\_PER\_BAR

Beats\_per\_minute = Bars\_per\_minute \* BEATS\_PER\_BAR