Baseline tracking and touch detection

Rick Walker

May 20, 2011

1 Overview of the optical touch screen

The Corning Lightouch prototype consists of a plurality of LED light sources launching light into a glass slab at multiple points around the perimeter. A much smaller number of photodiodes are also mounted around the perimeter to monitor the light intensity from each of the LEDs to each of the photodetectors. The LEDs are strobed one at a time so that the attenuation of each optical path from every LED to every photodiode can be independently measured.

The light propagates by bouncing between the top and bottom surfaces of the glass by total internal reflection (TIR). Multiple modes propagate at different bounce angles. The LEDs can be arranged to launch light either into the edge of the glass, or into the face of the glass. Edge launched light is more likely to generate modes at low bounce angles. Face launched light generates a distribution more rich in high bounce angles.

Touch detection occurs when an object (typically a bare human finger) comes into contact with the glass surface. If the object is able to partially index match to the glass, it interferes with the TIR propagation and preferentially attenuates light which is propagating by high bounce angles. By measuring the attenuations of all the beams before the touch (generating a baseline measurement) and comparing them to the measurements during the touch, the location of a plurality of touches can be computed. A simple algorithm uses simple binary beam breaks to triangulate the touches. A more sophisticated technique using analog attenuation data can track multiple touches using a computed tomography algorithm which finds the best set of touch points to match the measured attenuation.

2 Types of touches

For a touch to be effective, it must partially absorb energy from the propagating TIR modes. Compliant materials such as rubber, bare skin, and leather gloves create easily detected attenuations. Hard plastic, steel ball bearings, and ceramic objects are examples of non-compliant objects that have little interaction with the internal mode propagation.

In addition to being compliant, it is also necessary for the object to index-match to the TIR mode. Although dry paper is compliant, it does not index match well. A rubber ball works well because it is both compliant and contains oily plasticizing agents which provide effective mode matching.

Of greatest interest is the performance of the system with a bare human finger. In general, the human finger has both compliance and mode matching capability. Dry or callused fingers create much less attenuation than a well-hydrated pad of the index finger. Areas of the skin without sweat glands such as the knuckles of the back of the hand work very poorly. A well hydrated finger tip can create an attenuation on the order of 10-50%. The higher losses are seen with face-launched LEDs that create a richer proportion of high-order modes. A very light touch, an extremely dry or calloused finger, or a touch with the back of the knuckle can create attenuations as low as 1%.

3 Temporal behavior of touch events

Figure 1. shows the first touch trace that was measured for the "Dr. Evil" prototype. This trace shows a variety of abberations which are commonly seen in human touch data on this hardware. This test setup was using edge-firing LEDs.

There are four touches shown, labelled A,B,C & D. Touch "A" is a deliberate light touch showing an attenuation of about $(.85 - .8)/.85 \approx 6\%$. Touch "B" is a firm touch showing an attenuation of $(.85 - .55)/.85 \approx 35\%$. The firm touch at "B" leaves an oily residue which abruptly drops the baseline power at "X" to 94% of the initial value. A third touch at "C" achieves roughly the same total attenuation as "B" (w.r.t. the initial power level) and deposits more oil at "Y", dropping the total baseline down to 82% of the original "clean glass" transmission. Finally, a firm "wiping" touch is swept across the beam at "D" resulting in a very short attenuation spike and partially cleaning the glass about halfway back to the original transmission level. It is likely that the residue was only partially removed and was probably just moved to another location on the glass resulting in a lower baseline for some other beam paths.

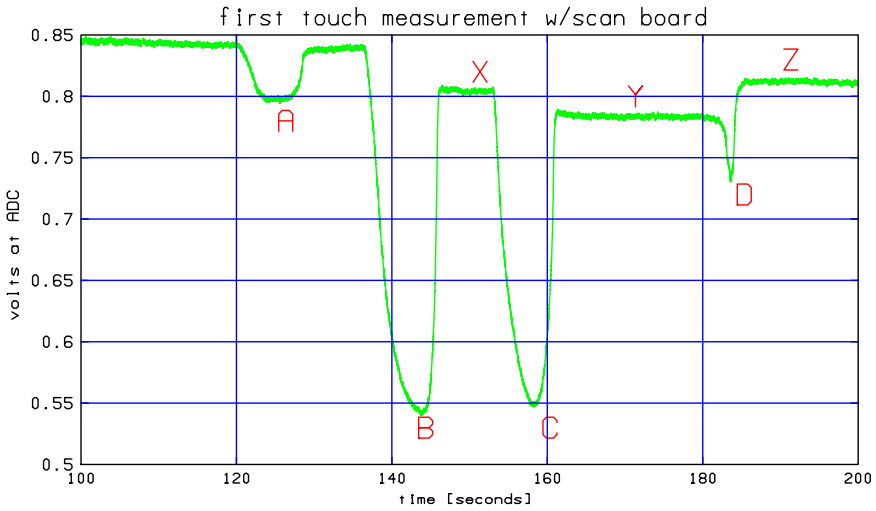

Figure 1: Single beam signal amplitude vs time for human touch taken on "Dr. Evil" prototype using PIC processor.

It is clear that at point "X" there is no touch. If we use the original beam amplitude at t=100 to set a detection threshold, there is no way to distinguish the attenuation at "X" from the attenuation at "A". Any viable baseline tracking algorithm must quickly reset itself after each touch to a potentially new value created by the deposit of oily residue or steam from the finger.

Since residue can also be wiped off, the baseline tracking algorithm must respond immediately to increases in average value such as occurs from point "X" to point "Y".

Figure 2 zooms in on touch "B" in detail. Overlayed with the attenuation data is a curve-fit exponential decay with a 4-second 10/90 risetime. The period from 137 to 138 seconds is the finger stabilizing in pressure. The initial attenuation is fairly low, perhaps 5%, however the attenuation approaches the final attenuation value of 36% with a 2 second time-constant. Given initial and final attenuation values A_i,A_f and a time constant τ, the attenuation follows the form $A(t) = A_f + (A_i - A_f)e^{-t/\tau}$.

This touch is eight seconds in duration. Shorter touches will not reach full

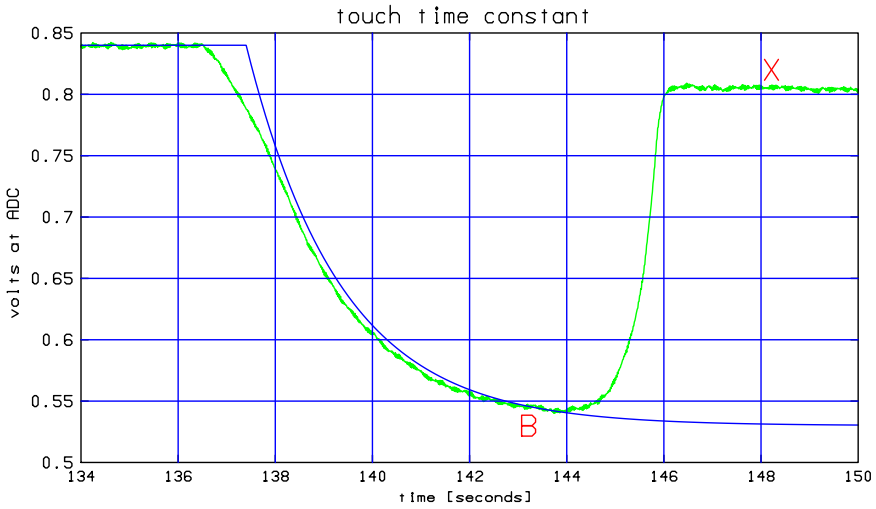

Figure 2: Enlarged plot of touch "B" showing time constant of attenuation

attenuation value, but will follow a clipped version of the above trajectory. In the above example, a one second touch would only reach 12% attenuation instead of the full 36%.

Observing finger contact with a microscope slide and appropriate side lighting shows that the initial contact of the finger occurs dry. The evolution of dark absorption points occurs over a period of 6-10 seconds. The absorption points occur at periodic points on the raised lamina of the fingerprint. Pore openings are present along the surface of the friction ridges of the finger. They are fairly evenly spaced due to the fact that one pore opening along with one sweat gland exists for each ridge "unit"[1].

An adult loses 800ml water, minimum, per day. 400ml is transepidermal[2]. The human body is estimated to have 1.8 square meter surface area[3]. The density of sweat glands varies dramatically across the body, with the finger tips being especially rich. If it is assumed that the fingers are 10x more endowed than

[1] http://ridgesandfurrows.homestead.com/friction_skin.html
[2] http://www.anaesthesiamcq.com/FluidBook/fl3_2.php
[3] http://hypertextbook.com/facts/2001/IgorFridman.shtml

the average, it can be calculated that a finger tip outgasses 2.2nL of sweat per second per square centimeter. The hypothesis is that this normal sweat gland respiration causes a super-saturated humidity in the indentation of each sweat gland, resulting in water condensation on the surface of the glass. The rate limited build-up from low humidity to saturation is probably responsible for the 1 second time constant seen.

Human sweat is approximately 99% water and 1% solids. The solid component is approximately 1/2 NaCl, and 1/2 organic compounds (amino acids, urea, and peptides) [4]. This non-water component is probably responsible for the resetting of baseline that occurs at points "X" and "Y".

Upon release of the finger, there is an almost instantaneous return to the baseline. However there appears to be another exponential process (probably the evaporation of sweat into the low humidity ambient air), which is at least 10x faster than the touch onset. The initial curvature at 145 seconds is probably the gradual withdrawing of finger pressure, however a definite exponential tail can be seen in the knee between 145.8-146. The rapid recovery can be seen very clearly at point of recovery from "D" to "Z" where the rapid finger wipe eliminates the slow variation in finger pressure.

4 Setting a detection threshold

The purpose of the baseline tracking algorithm is to provide an estimate for the non-attenuated amplitude of each beam. If the current estimate of the unattenuated beam amplitude is $A_0(t)$ in arbitrary units of photocurrent, then the current measurement $A(t)$ can be corrected to give an estimated transmission G(t) varying from 0 to 1 equal to $A(t)/A_0(t)$.

Assuming that we know something about the average noise of our measurement, we can set a touch detect threshold for reliable beam break detection. If we measure N beams per second and require a false touch detection to occur with a frequency no more often than once per K seconds, the probability of a false touch should be less than $1/(NK)$. If each measurement of G(n,t) has a variance σ^2, then the detection threshold should be set to $1 - z\sigma$, where $erf(\frac{z}{\sqrt{2}}) = \frac{NK-1}{NK}$. Table 1 shows the threshold settings as a function of the mean time to false touch (MTFT) detection for a system with 256 beams running at 50Hz, NK=12800.

The prototype system using switched integrators only achieves SNRs on the order of 100[5] so a detection threshold of 5 sigma is quite onerous, leading to

[4] Ibid 1

[5] Rick Walker, "Photocurrent signal to noise ratio", CWTC internal report, May 17, 2011.

MTFT	z
1 second	3.95
10 seconds	4.47
100 seconds	4.93
1000 seconds	5.37

Table 1: the number (z) of standard deviations at which to set a detection threshold to produce a given mean time to false touch (MTFT) detection for a system with 256 beams running at 50Hz.

a touch sensitivity of only 5%. We can reduce this requirement by defining a touch only when two or more beams are broken simultaneously

Assuming we have a reasonable algorithm for determining the normalized transmission G(t), we can bootstrap our touch detect algorithm by defining two states, IDLE and TOUCHED. IDLE is defined as $G(t) \geqslant 1-z\sigma$ and TOUCHED as $G(t) < 1 - z\sigma$. During IDLE times, we can accumulate the deviation of the new samples from G(t) and calculate σin a straightforward fashion. We can then use this computed sigma to refine our touch-detect threshold under varying illumination levels and various beam intensities and SNRs.

In the algorithm below, we measure mean deviation rather than RMS deviation and the noise multiplier is chosen empirically to reduce false triggering to an acceptable value.

5 Basic baseline tracking and touch detect algorithm

An algorithm that operates independently and in parallel on each beam and which works well in practice can be described as follows:

1. Start with a tracking step Δ, a smoothing factor ϵ, a noise multipling factor z and reasonable starting estimates of the baseline A_0 and mean channel noise s.
2. Get a new signal amplitude for a given beam
3. If the present input signal A is above the baseline estimate A_0, set the baseline estimate to the current input signal (notice that this causes A_0 to track the upper end of a two-sided noise distribution).
4. If the current input signal A is below the baseline estimateA_0 reduce the baseline estimate $A_0 = A_0 - \Delta$

5. Set the normalized signal amplitude $G = A/A_o$.
6. If $(A_0 - A) > zs$, then in state TOUCHED
7. Else if$(A_0 - A) \leq zs$, then in state IDLE, (Optionally, the estimate for s can be refined here by setting $s = (1 - \epsilon)s + \epsilon(A_0 - A)$).
8. repeat from step 2

6 Example implementation and results

The above algorithm is implemented in awk for demonstration. Step 1 is to chose and initialize the constants.

```
BEGIN {
    A0=.85              # initial tracking level
    DEL=.00005          # droop constant
    eps=.001            # pseudo sigma filter time constant
    error=40*DEL        # starting estimate for pseudo sigma
    min_err=10*DEL        # minimum allowed slicing level
    Z=4                 # number of "deviations" to test for
}
```

The next block reads the standard input list of (time, voltage) pairs and populates two arrays with the N time and voltage values

```
 // {           # read the data into voltage and time arrays
    tt[n]=$1    # time of each data point
    aa[n]=$2    # amplitude of each point
    n++         # number of points
 }
```

Next we loop through all the data and compute the baseline estimate xx[i], normalized gain G[i], the touch status touch[i], and a running estimate of the average noise deviation ee[i].

```
for (i=0; i<n; i++) {                   # for each data point
    t=tt[i]; A=aa[i]                    # simplify array access

    if (A0>A) { A0-=DEL } else { A0=A }     # track A with A0
    G[i]=A/A0                           # normalized signal

    # detect a touch
    if (A0-A > Z*error)  {
       touch[i]=1
```

```
    } else {
       touch[i]=0;
    }

    # if not in a touch, accumulate noise estimate
    # using simple RC digital accumulator
    # this couplet is optional...
    if (touch[i]==0) {
        error = (1-eps)*error+eps*abs(A0-A)
        # avoid locking up on a noise free signal
        if (error < min_err) error = min_err
    }

    xx[i]=A0          # save the tracking value
    ee[i]=error       # save the error
}
```

The signals are then plotted in Figures 3 and 4.

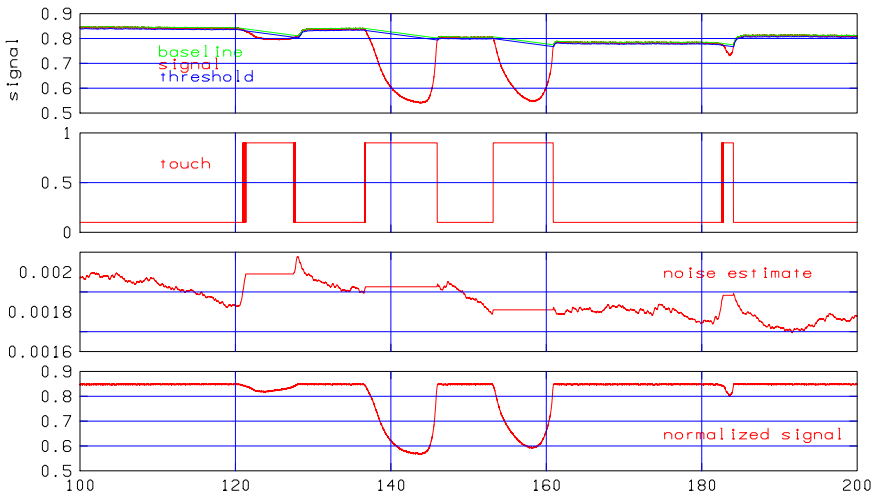

Figure 3: Plot showing all the state variables of the baseline tracking algorithm

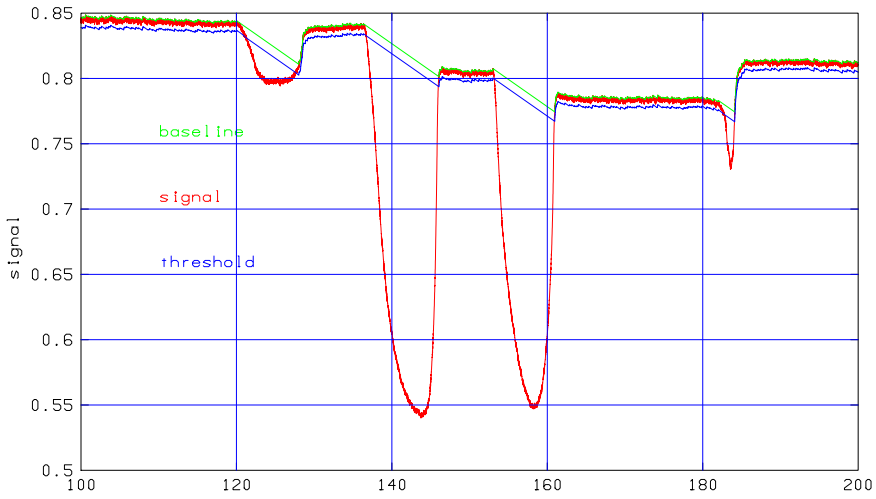

Figure 4: Enlarged view of signal, baseline estimate and threshold offset by a multiple of the noise estimate

There are some tradeoffs in the selection of constants. DEL should be set to minimize droop during a typical touch interval, but large enough to pick up the release of a dirty touch (eg: the second touch of figure 4 at time 145sec). The algorithm could be refined to allow very long touches by keeping track of the droop and not allowing any further reduction if a maximum droop is reached. However, this would prevent the display from recovering from something like an attached "Post-it" pad. A compromise could be to reduce the droop factor after a certain amount of drop. This could allow longer touches, but still ensure an eventual recovery from a screen contaminant.

The value for Z should be chosen just high enough to eliminate false beam break events. If the optional noise tracking code is implemented, the screen should adapt somewhat to varying interference levels. The code above tracks mean deviation from the peak value. It might be better to track average peak deviation to better handle interferers with high peak to average ratios, rather than just gaussian measurement noise. It might be worthwhile to let the user twiddle this factor to tune the screen to noisy environments. If the gain of each channel is normalized in hardware (by, for example, setting the integration time

of each measurement), it might not be necessary to track the noise estimate. The simplest system, of course, would be a fixed threshold for detecting a beam break. The Z value and tracking code is easy to remove if it is not needed.

A hack was added to put a limit on the minimum error estimate using "min_err". Without this term, the loop might zero out the error estimate completely on a saturated ADC channel. Getting the noise estimate bootstrapped back again without even a partially working touch detect might be problematic. Best to be safe.

The touch detect logic is shown without any hysteresis. If it is objectionable to have chatter on the this signal, then hysteresis may be added to the code.

The noise filter uses a smoothing factor "eps". This is used in a digital approximation to a first order RC filter. The value of "eps" is approximately equal to the measurement step size divided by the desired smoothing RC time constant.

7 Improved algorithm with resistance to impulse noise

The basic baseline algorithm is subject to errors caused by impulsive interference. Because the algorithm always sets the baseline to the peak value of the incoming signal, any impulse can cause an unnaturally high level in the baseline estimate. One way simple way to improve this is to change the line in the code from "if (A0>A) { A0-=DEL } else { A0=A }" to "if (A0>A) { A0-=DEL } else { A0+=DEL*M}". A value of M=5 works well. This allows the tracker to rapidly recover when the finger is removed from the screen, but provides a limited loop response for a large impulsive interference event.

The approach taken in the PIC prototype is to track the mean of the noise by a symetrical bang-bang loop. Several regions are defined based on the relative distance between the incoming signal and the baseline estimate. When the incoming signal is within +/- ΔV_1 of the baseline, the loop uses +/- DEL1 to track the signal. In a similar fashion, several different bands can defined based on the difference between the signal and the current baseline, and each band can have a different response.

```
// This code replaces the line:
//      if (A0>A) { A0-=DEL } else { A0=A }     # track A with A0
// in the example above

if (abs(A-A0) < Z*error) {
```

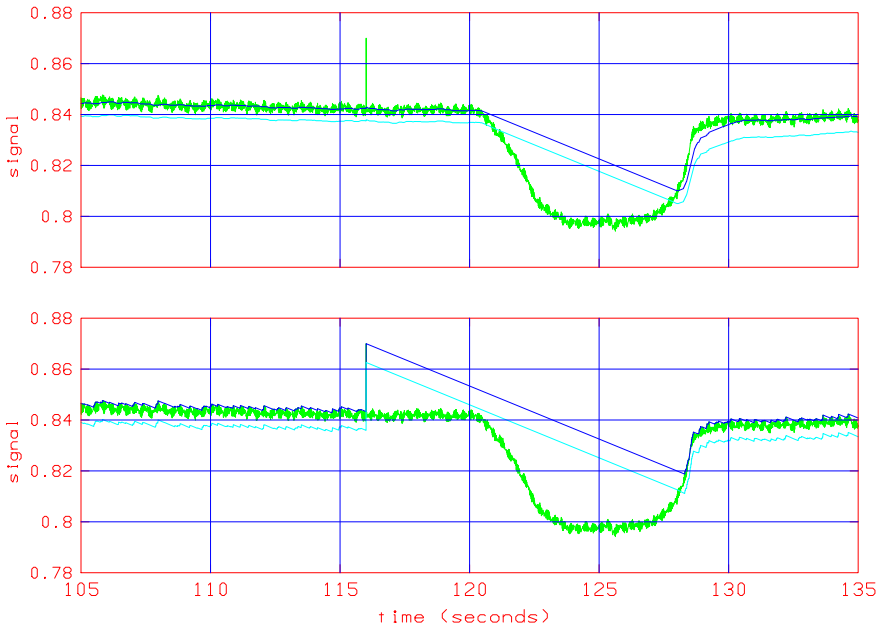

Figure 5: Comparison of enhanced algorithm (top) to basic algorithm (bottom) in the presence of an impulse interferer at t=116 seconds. The top system tracks the median and has a much attenuated response to isolated impullses. The bottom system sets the baseline to the peak value of the signal and can be fooled by isolated pulse interference.

```
    if (A0>A) {          # symmetrically track mean of A with A0
        A0-=DEL
    } else {
        A0+=DEL
    }
} else if (A0 > A) {
    A0-=DEL                              # droop during a touch event
} else if (A0 < A) {
    A0+=10*DEL                       # catch A0 up to A after a touch
}
```

Figure 5. shows a comparison between the basic algorithm which tracks peak value and the more complex algorithm that tracks the median of the noise.

A disadvantage to the improved algorithm is that the normalized amplitude

G, can go slightly above 1. This can either be clamped at unity, or G can be redefined as $G = 0.98 * A/A_0$.

8 Summary

The proposed algorithm has several properties that work well with typical touch signals. It accomodates a shift in baseline that follows an oily touch that leaves a deposit. It equilibrates to screen contamination. It reliably detects touch events with a means to set sensititivity based on measured system noise.

Several suggestions have been made for adapting or refining the algorithm based on customer input, or accumulated experience with the system.