High Rate, Low-Latency Multi-Touch Sensing with Simultaneous Orthogonal Multiplexing

Darren Leigh[1], Clifton Forlines[1], Ricardo Jota[1,2], Steven Sanders[1], Daniel Wigdor[2]

Tactual Labs[1]
New York, NY
USA
{first.last}@tactuallabs.com

Department of Computer Science[2]
University of Toronto
Toronto, ON, Canada
daniel@dgp.toronto.edu

ABSTRACT

We present "Fast Multi-Touch" (FMT), an extremely high frame rate and low-latency multi-touch sensor based on a novel projected capacitive architecture that employs simultaneous orthogonal signals. The sensor has a frame rate of 4000 Hz and a touch-to-data output latency of only 40 microseconds, providing unprecedented responsiveness. FMT is demonstrated with a high-speed DLP projector yielding a touch-to-light latency of 110 microseconds.

INTRODUCTION

Touch sensors in commercial use today typically operate at 60-85 frames per second. Such frame rates imply a "best" worst-case latency of at least 16 milliseconds, from the time a user's finger touches the sensor until the moment that information is made available to the system bus. Recent work from Microsoft and the University of Toronto has demonstrated that as little as 25 milliseconds of latency can impair performance [1], and that two milliseconds can be noticed by users of direct-touch systems [2]. Thus, it is the first goal of the present research to build a sensor capable of supporting an overall end-to-end latency of less than two milliseconds.

Ng *et al.* describe the results of analysis that suggest typical mobile devices have "end-to-end" latency, that is, a time between an input and the result being shown on the display, in the range of 75-125 milliseconds [2]. Other sources provide different measures of latency [3]. This variability is expected given that end-to-end latency includes performance characteristics of user-space software. As Ng *et al.* further describe, there are three broad sources of latency: the touch sensor, the software stack, and the display controller along with its refresh rate. The latency in these components is additive: while introducing a fast sensor will not, on its own, enable sub-2 millisecond responses, it is necessary to achieve this goal. Further, as Jota *et al.* describe, the impairment of user input to direct touch displays caused by latency is a continuous function – therefore any reduction of latency is expected to improve input performance [1].

Traditional projected capacitive (PCAP) sensors utilize a time-division multiplexing (TDM) approach to sensing: the controller continuously monitors sense traces, shown as columns in Figure 1 (top). Each drive trace, shown as a row, is then sequentially activated. If no touching object is present, an expected crosstalk signal is transferred from the active drive trace to all sense traces, and no touch is recorded. If, however, the user's finger is present on the sensor, a different amount of signal is transferred. The (*x*,*y*) coordinates of a touch are determined as follows: in the example in the figure, the *x* coordinate is the identity of the sense trace at which the signal change is detected, and the *y* coordinate corresponds to the drive trace that is active at the time of the change.

PCAP sensors offer a number of advantages over resistance-based sensors; durability and rigidity among them. Even more important in today's commercial market is their ability to detect multiple simultaneous touches on a transparent surface. Thus, their inherent limitation in sensor throughput has been tolerated.

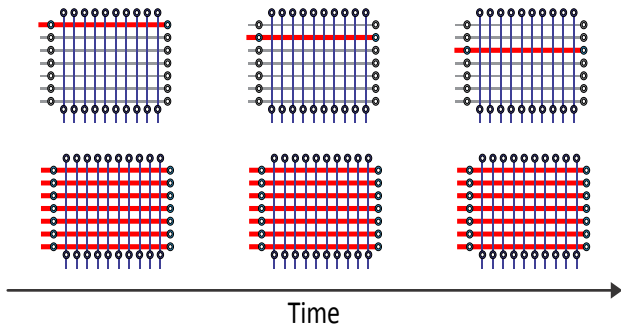

Figure 1: Top row: Traditional TDM-based projected capacitive sensors scan each row sequentially, which takes time. Bottom row: FMT sensors scan all rows simultaneously.

UIST '14, October 05-08 2014, Honolulu, HI, USA
 ACM 978-1-4503-3069-5/14/10…$15.00.
http://dx.doi.org/10.1145/2642918.2647353

Our goal was to build a projected capacitive sensor, complete with all of these advantages, which is also able to perform at a significantly higher rate. To achieve this, we eliminate the sequential scanning inherent in a TDM approach by instead activating all drive traces simultaneously using orthogonal signals: instead of looking for a signal strength change at a given sense trace, the controller examines each sense trace for the strength of every orthogonal signal transmitted on the set of drive traces. This is shown in Figure 1 (bottom).

RELATED WORK

A thorough review of the literature examining interaction latency is provided in [1], and so we omit it here. Instead, we briefly review alternative methods for sensing touch: traditional PCAP, optical techniques, and resistive methods.

The use of capacitive sensing of touch input dates to at least the mid-1960's [4], and for multi-touch to the mid 1980's [5]. More recently, SmartSkin [6] was a PCAP system that used a TDM scheme, transmitting bursts of 400 kHz square wave on each row, in turn, and looking for the response at the columns. SmartSkin had a 30 Hz frame rate, and therefore a "best" worst-case latency of about 33 milliseconds. Westerman *et al.* also demonstrated a similar method [7], and illustrated its use to compose complex multi-touch gestures on an opaque touchpad. That work formed the basis of the FingerWorks product line.

DiamondTouch [8] was a multi-user touch system, though not fully multi-touch: it could sense multiple touches, but only as projections along the rows and columns. It employed the rows and columns identically as transmitters, with receivers in contact with the users' bodies for identification purposes. It was, therefore, not PCAP, but a close cousin. DiamondTouch used a TDM scheme, transmitting a 100 kHz square wave burst on each row/column in turn. Several versions of this system were produced, with frame rates ranging from 20 to 40 Hz, implying "best" worst-case latencies from 25 to 50 milliseconds.

PCAP forms the basis of most multi-touch screens used to date. In contrast, alternative sensing techniques for touch have also been demonstrated.

A number of optical techniques have been demonstrated for touch input. As early as 1982, optical methods were employed for multitouch input using a camera for imaging and diffuse illumination for lighting [9]. More recently, frustrated total internal reflection (FTIR) systems use cameras to detect where light leaks from a sheet of plastic due to touches frustrating the total internal reflection of light that is propagating within [10]. Camera-based optical techniques are difficult to implement in thin form factors, such as mobile devices, because of the standoff necessary for the sensor – this is manifest as a large volume behind the display, or as a bezel surrounding the screen. Further, for techniques that employ 2D cameras, the latency is limited by processing time, as well as the frame rate of the sensor: standard 60 frame-per-second cameras would imply a "best" worst-case latency of 17 milliseconds, similar to traditional PCAP techniques. Bezel-mounted optical sensors enable sensing of touch [11], and do not require 2D image processing. However, they too suffer from the need for a bezel. A detailed overview of many different optical techniques is provided in [12].

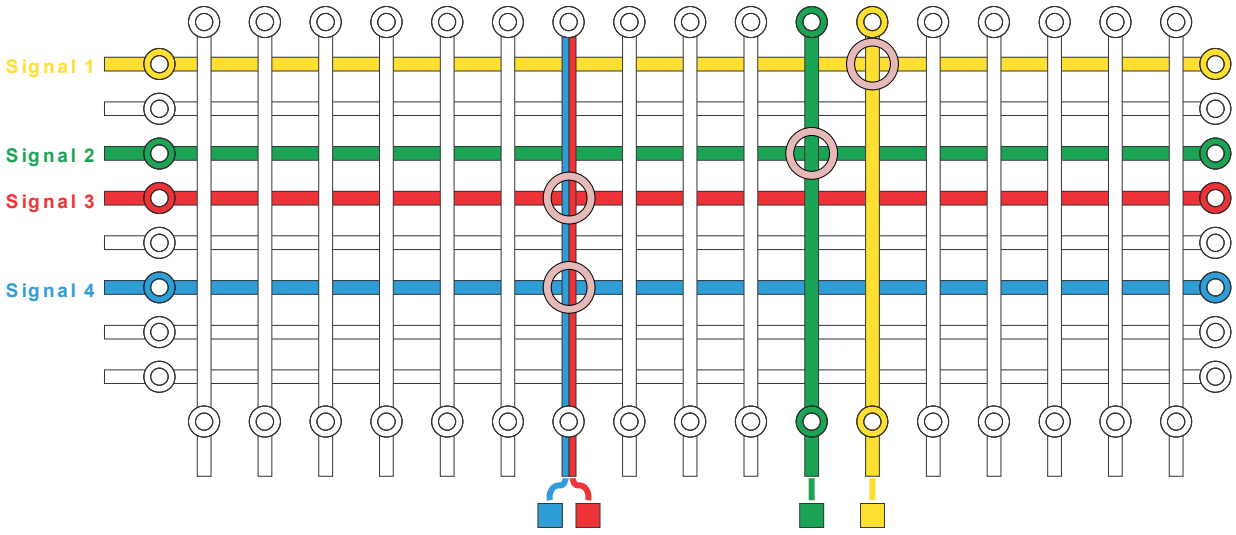

Figure 2: Orthogonal signals are simultaneously transmitted on each row. Where touches occur (the pink rings in the figure), signals from the affected rows are coupled into the affected columns. Column receivers determine the amount of each row signal present on that column to determine the affected row/column intersections.

Ng *et al.* developed a very fast multi-touch system based on a resistive technique, and used it to study and characterize low-latency touch interaction [2]. They demonstrated an end-to-end latency of about one millisecond. Unfortunately, the design and technique of their touch system is proprietary and has never been publically disclosed, beyond a cursory mention of its "resistive" nature. Because of limitations in durability and requirements of application of active pressure, resistive techniques methods have largely been supplanted by PCAP. From Ng we draw the method for enabling fast visual of output of sensor data, completing the quick round trip. We utilize the same projector, and similar FPGA-based approach to generating the imagery.

ORTHOGONAL SIGNAL CAPACITIVE TOUCH SENSING

Our goal is to detect touch events from human fingers, or other capacitive objects, on a two-dimensional manifold such as a planar surface. It is important that multiple simultaneous touch events be detected and distinguished from each other. For the reasons we have described, it is also important that the touch events be detected, processed and supplied to downstream computational processes with very low latency, i.e. on the order of one millisecond or less.

To accomplish this goal, we have developed a projected capacitive method that has been enhanced for a high update rate and low latency measurement of touch events. Our technique employs parallel processing and higher frequency waveforms to gain the above advantages. We have also developed methods to make the measurements sensitive and robust, allowing the technique to be used on transparent display surfaces and permit the economical manufacture of products that employ it.

BASIC TECHNOLOGY

The touch surface is comprised of a series of rows and columns[1], along which signals can propagate. The rows and columns are designed so that, when not being touched, a negligible amount of signal is coupled between them.

A different signal is transmitted onto each of the surface's rows. These signals are designed to be "orthogonal" in the mathematical sense, *i.e.*

$$\int f_i(x)\, f_j(x)\, dx = 0\,, \quad i \neq j \qquad \text{Equation 1}$$

so that a linear combination of them can be separated and the individual signals distinguished from one another. The use of orthogonal signals allows us to take advantage of matched-filter receiver techniques, which can be optimal under real-world conditions.

When a row and column are touched simultaneously, a small amount of the signal that is present on the row is coupled into the corresponding column. A receiver, attached to each column, is designed to receive any of the transmitted signals, or an arbitrary combination of them, and to individually measure the quantity of each of the orthogonal transmitted signals that is present on that column.

Touch events correspond to the received signals on the columns. For each column, the different signals received there indicate which of the corresponding rows is being touched simultaneously with that column. The quantity of each signal received is related to the amount of coupling between the corresponding row and column and may indicate the area of the surface covered by the touch, the pressure of the touch, etc. A touch is detected for a particular row / column intersection when that intersection's row signal is detected (above a threshold) by the intersection's column receiver.

Signal Pathway

The basic architecture of FMT is shown in Figure 2. The row signals, transmitted with amplitude V, pass through the rows, fingers and columns and then into the column receivers, where they are detected with a matched filter. The matched filter (one for each expected row signal) integrates over a time period τ, producing a measured level for that row signal on that column. That level can be expressed by the formula:

$$\text{received signal level} = \alpha A V \tau \qquad \text{Equation 2}$$

where A is the area touched by the finger and α is a scale factor that is a function of signal attenuation along the rows and columns, coupling between the finger and the touch surface, and various other gain parameters. To get the best system performance, it is important that we maximize the received signal level by making good design choices.

Choosing a Set of Orthogonal Signals

There are an infinite number of orthogonal signal sets that could be used to implement our fast touch sensor and, in theory, they are all equivalent. In practice, the specific details of each signal type have advantages and disadvantages that will cause us to choose one that is optimal for our needs. Some of these factors include:

- Ease and expense of implementation
- Dynamic range, or the ability to distinguish simultaneous weak touch signals from strong ones
- Immunity to noise and interference

A good signal set will be easy and inexpensive to implement, have high dynamic range and be relatively immune to noise and interference. A real-world implementation will require some tradeoffs between these.

[1] The nature of the rows and columns is arbitrary and the particular orientation is irrelevant. In fact, it is not even necessary that the rows and columns be in a square grid.

The choice of signal sets is similar to those used for multiplexing in communication systems, and includes:

Time Division Multiplexing (TDM), in which the measurement period is divided into segments and each channel is assigned one of those segments in which to transmit.

Frequency Division Multiplexing (FDM), in which each channel is assigned a separate frequency band.

Code Division Multiplexing (CDM), in which each channel is a assigned a separate "spreading code": a random-looking waveform that is statistically uncorrelated with the others.

Time division multiplexing, as described in the introduction, is the technique used by SmartSkin [6] and most other PCAP touch systems today. It has the advantage of being simple to implement, with a minimum of hardware. However, the inherent delays from time-slot to time-slot add latency to the system. Removing this latency is possible by decreasing the duration of the time slots, but this would lower the integration period τ of the receivers, decreasing the received signal strength (per Equation 2).A designer can compensate for the shorter integration period by increasing the transmit amplitude V – and commercial products often do this – but there are practical limits. Increasing the amplitude to achieve very low latencies might require the transmitter to put out hundreds of volts.

Frequency division multiplexing and code division multiplexing both allow the entire frame time to be used for the receiver integration period, because the orthogonal signals are transmitted *simultaneously* and there is no need to divide the frame into separate time slots.

CDM has the advantage of being easy to generate and receive (because the modulation time reference is local), and is more immune to external noise and interference due to the "random" nature of its signals. Unfortunately, CDM suffers from dynamic range problems, making it difficult to receive weak signals in the presence of strong ones.

Even worse, CDM signals are broadband and can suffer degradation if the communication channel's frequency response is not "flat" (the same at all frequencies). A PCAP system being touched at multiple points at random times by uncharacterized appendages is unlikely to present a flat frequency response. This problem will be exacerbated if the sensor's rows and columns are made from a material that is less conductive than would be desired, which is the case with transparent conductors such as indium-tin-oxide (ITO).

Frequency division multiplexing can be very robust under bad channel conditions because a channel tends to be well-behaved over the narrow bandwidth occupied by its individual frequency band. For this reason, variants of it, such as orthogonal frequency division multiplexing (OFDM), are widely used in communication systems with poorly-behaved channels, including WiFi, digital TV broadcasting and power-line communication. FDM can be more prone to external noise and interference, but there are mitigation techniques that can be employed to lessen the effects.

The dynamic range of FDM depends strongly on the receiver architecture, but it is possible to achieve results almost as good as TDM.

For these reasons, we chose to use frequency division multiplexing for FMT. In the simplest implementation, the orthogonal signals being transmitted onto the rows are unmodulated sinusoids, each of which has a different frequency. The frequencies are chosen so that they can be easily distinguished from each other in the receiver.

We use a "comb" of frequencies, where the spacing between adjacent frequencies is constant, to allow easy detection using a discrete Fourier transform (DFT). Due to the Fourier relationship between time and frequency, the spacing between frequencies, Δf, must be at least the reciprocal of the integration period τ. Otherwise the signals will not be "orthogonal" and will be confused with each other in the receiver. For example, if we desire to determine which row signals are present at a column receiver, and we wish to do so once per millisecond, then the frequency spacing $\Delta f > 1/\tau$ or greater than one kilohertz. In reality, the frequency spacing should be greater than the minimum to permit a simpler, more robust design.

We also ensure that the highest transmitted frequency is less than twice the lowest, in order to avoid any problems with harmonics. Harmonics are integer multiplies of a fundamental frequency that are created by non-linearities and other problematic physical processes. If our highest frequency is greater than twice the lowest, then it is possible that a harmonic of a low frequency will be interpreted as a legitimately transmitted signal, causing false readings. By constraining our transmitted frequencies so that none of their harmonics would overlap any of our deliberately generated signals, we can avoid this problem.

The Fourier and harmonic conditions described above combine to constrain the minimum signal frequencies that we can successfully implement. If an FMT touch sensor has n rows and a latency of L, then we must transmit n frequencies that are at least $1/L$ apart, yielding a minimum frequency bandwidth of n/L. Because the highest frequency can be no more than twice the lowest, the minimum signal we can transmit must be no lower than n/L. Therefore, for a forty row touch sensor with a latency of one millisecond, the lowest possibly frequency that can be used is 40 kHz.

Our FMT demonstrator has thirty rows and therefore requires thirty separate frequencies — one per row. In its fastest mode, the system has a frame update period of 40 microseconds, which means that our frequency spacing must be at least $1/40$ µsec or 25 kHz. We use an actual spacing of about 163 kHz, with our lowest frequency being 5.371 MHz and our highest 10.091 MHz.

Modulated sinusoids

The use of unmodulated sinusoids has two problems. First, the sinusoids might cause radiofrequency interference to other devices near the touch surface, and a device employing such might have problems passing regulatory testing (e.g. FCC, CE). Second, sinusoids in the environment, whether from deliberate transmitters or from other interfering devices (perhaps even another identical touch surface), might cause false or degraded touch measurements on our device.

An effective technique for minimizing such interference is to modulate or "stir" the signals we are transmitting in a manner such that we can demodulate ("unstir") or otherwise compensate for the modulation of the signals when they reach the receiver. Signals emitted or received under such a technique are highly uncorrelated with anything else, and so act as mere noise instead of appearing to be similar to other signals present in the environment.

Two straightforward ways of doing this are frequency modulation and direct-sequence spread spectrum modulation. FMT was built to do the latter, and includes bi-phase modulators (multiplied by +1 or -1, i.e. selective inversion) in both the row transmitters and the column receivers. These modulators share a control signal so that all of the signals, at both the transmitters and receivers, flip polarity at the same time. The control signal is pseudo-random, causing the transmitted signal to spread in a wider bandwidth on the touch surface, and then be unspread in the receiver before we attempt to detect the sinusoids.

Care must be taken not to spread the signals too much, or we will run into the channel problems described above for CDM. A good rule of thumb is to spread on the order of the row frequency spacing.

Signal Detection

To determine which rows and columns are being simultaneously touched, we need to receive any signals present on the columns and determine which of the transmitted frequencies appear. This can be done with common frequency analysis techniques, such as a Fourier transform or filter bank.

From each column's signal, we determine the strength of each of the transmitted frequencies that it contains. If the strength of a frequency is greater than some threshold, then we have determined that there is a touch event between the column and the row corresponding to that frequency. Signal strength information can be used to determine the area of the touch event, which may correspond to various physical phenomena including the size of the finger, the pressure with which it is pressing down, the fraction of row/column intersection that is being touched, etc.

Once we calculate the signals strengths for each frequency (corresponding to a row) for each column, we can create a two-dimensional "heat map" of these, with the signal strength being the value of the map at that row/column intersection.

The heat map can be thresholded to determine touch events, or can be used to infer information about the shape, orientation, etc. of the object touching the surface.

Our FMT demonstrator implements a complete radio receiver with a Fast Fourier Transform (FFT) detection scheme for every column. This digitizes the RF waveform, detects the sinusoids and performs digital signal processing on them.

POST PROCESSING

After the signal strengths from each row in each column are calculated, the system does some post-processing to convert this 2-D "heat map" into usable touch events. The process includes "field flattening", touch point detection, interpolation and touch point matching between frames.

Because we requireme very low latency, the processing steps are as optimized and parallelized as possible.

Field Flattening

Due to signal attenuation across rows and columns and other systematic error sources that affect signal strengths, we first perform field flattening by normalizing these strengths across the whole touch surface. This can be done with a known calibration object, measuring the signal strength response that it causes for every row/column intersection, and computing both an additive and multiplicative offset for that intersection. When the offsets are applied, the responses are normalized across the entire touch surface.

Our FMT demonstrator does not require much field flattening, because its touch surface uses copper rows and columns, which are very conductive. Field flattening is much more important when compromise conductors, such as transparent ITO, are used.

Touch Point Detection

Once the heat map is generated and "field flattened", we determine the coarse touch points. This is done by finding local maxima in the normalized signal strengths. We use a fast and parallelizable method for finding these, by comparing each element of the normalized heat map to its neighbors and labeling it a local maximum if it is strictly greater than all of them, and above a given threshold.

We can define the set of neighbors in various ways, but the two most useful sets will probably be the Von Neumann neighborhood and the Moore neighborhood (see Figure 3). The Von Neumann neighborhood consists of the four elements that are vertically and horizontally adjacent to the element in the center (i.e. the elements to the north, south, east and west of it). This is also called the "four-connected" neighborhood. The Moore neighborhood consists of the eight elements that are vertically, horizontally and diagonally adjacent to the element in the center (i.e. the elements to the north, south, east, west, northeast, northwest, southeast and southwest of it). This is also called the "eight-connected" neighborhood.

The neighborhood we choose will depend on the interpolation scheme we use to calculate the fine touch points.

Interpolation

Once the coarse touch points are determined, we compute the fine touch points using interpolation. A straightforward way to do this is to model the capacitive contact of a distributed touch as a second-order function in two dimensions, fitting it to a paraboloid.

For a Von Neumann neighborhood, the relevant points look like Figure 3 (left) with the central blue element being the local maximum and the subscripts being the coordinates of a particular element relative to it. The positions and signal strengths of the five elements allow us to fit them to the following equation:

$$Ax^2 + Cy^2 + Dx + Ey + F = \mathrm{z} \qquad \text{Equation 3}$$

where x and y are the position of an element, z is the signal strength of the element, and A, C, D, E and F are the coefficients of the second-order polynomial. Relative to the central point, all of the element x, y positions are constant. The z values are the measured signal strengths at each element, and thus are known. The five polynomial coefficients are the only unknowns, so we need five simultaneous equations to solve for them. Each equation represents one of the five points, including the central point and its four neighbors.

We solve for the polynomial coefficients by inverting a Vandermonde-like matrix, which yields:

$$\begin{bmatrix} A \\ C \\ D \\ E \\ F \end{bmatrix} = \frac{1}{2} \begin{bmatrix} 0 & 1 & -2 & 1 & 0 \\ 1 & 0 & -2 & 0 & 1 \\ 0 & -1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & -1 \\ 0 & 0 & 2 & 0 & 0 \end{bmatrix} \begin{bmatrix} z_{0,1} \\ z_{-1,0} \\ z_{0,0} \\ z_{1,0} \\ z_{0,-1} \end{bmatrix} \qquad \text{Equation 4}$$

Note that the polynomial coefficients are a linear combination of the signal strengths and that only simple multiplications, involving negation and a single shift, are required to calculate them. This means that they can be efficiently computed in an FPGA or ASIC. By fitting our data to a paraboloid, we are assuming that the fine touch point is at its maximum, which occurs at the point x_f, y_f where:

$$x_f = -\frac{D}{2A} \quad \text{and} \quad y_f = -\frac{E}{2C}$$

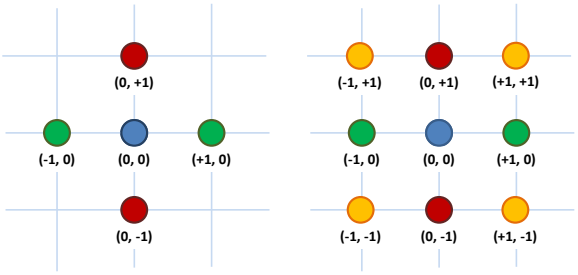

Figure 3: On the left is a Von Neumann neighborhood. On the right is a Moore neighborhood.

The values x_f and y_f are independent of each other, with x_f depending only on the signal strengths of the elements to the left and right of the center point, and y_f depending only on the signal strengths of the elements above and below it.

For the Moore neighborhood, the relevant points look like Figure 3 (right). The positions and signal strengths of the nine elements can be fit to the following second-order equation:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey + F = \mathrm{z} \qquad \text{Equation 5}$$

which is similar to the previous case, but with an added xy cross term. This is an over-determined system with nine simultaneous equations (one per element), so we must employ a least-squares technique to solve it. This yields:

$$\begin{bmatrix} A \\ B \\ C \\ D \\ E \\ F \end{bmatrix} = \frac{1}{36} \begin{bmatrix} 6 & -12 & 6 & 6 & -12 & 6 & 6 & -12 & 6 \\ -9 & 0 & 9 & 0 & 0 & 0 & 9 & 0 & -9 \\ 6 & 6 & 6 & -12 & -12 & -12 & 6 & 6 & 6 \\ -6 & 0 & 6 & -6 & 0 & 6 & -6 & 0 & 6 \\ 6 & 6 & 6 & 0 & 0 & 0 & -6 & -6 & -6 \\ -4 & 8 & -4 & 8 & 20 & 8 & -4 & 8 & -4 \end{bmatrix} \begin{bmatrix} z_{-1,1} \\ z_{0,1} \\ z_{1,1} \\ z_{-1,0} \\ z_{0,0} \\ z_{1,0} \\ z_{-1,-1} \\ z_{0,-1} \\ z_{1,-1} \end{bmatrix}$$

Equation 6

Note again that the polynomial coefficients are a linear combination of the signal strengths. The multiplications are slightly more complicated, but many of the multiplicands can be factored out and applied a single time near the end of the calculation to make the computation more efficient. Because we are interested in the maximum of the paraboloid – meaning that overall scale factors are irrelevant and we are only concerned with relative values and the arguments which maximize the function – we may be able to cancel out many of them altogether.

The fine touch point is at the maximum of the paraboloid, which occurs at the point x_f, y_f where:

$$x_f = (BE - 2CD)/(4AC - B^2)$$

and

$$y_f = (DB - 2AE)/(4AC - B^2)$$

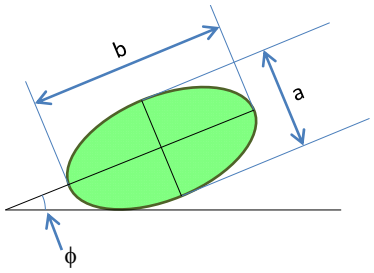

Figure 4: An elliptical fit to an asymmetric touch point. The aspect ratio and tilt angle can provide useful information about the touching object.

For the eight-connected neighborhood, the values x_f and y_f are not independent of each other. Both depend on the signal strengths of all eight neighbors. This might seem like a disadvantage because of the increased computational burden, and the possibility that certain combinations of signal strengths will produce singular values for the fine touch points. However, there are advantages as well. Because of the least-squares nature of the eight-connected calculation, it is more robust against noisy signal strength values. Small errors in one signal strength will be compensated for by the increased amount of data used in the calculation, and the self-consistency of that data.

Another advantage of the eight-connected neighborhood is that it provides an extra piece of information that might prove useful as part of a user interface. The B coefficient of the xy cross-term can be used to characterize asymmetry in the fitted paraboloid and, along with the aspect ratio information inherent in the A and C coefficients, can allow software to determine the angle at which the touch is occurring.

Figure 4 shows a notional touch point with an elliptical cross section, which can be obtained by truncating the paraboloid at a particular z value. The values of a and b can be obtained from the A and C coefficients of the polynomial, and they give us information about the aspect ratio of the object touching the surface. For example, a finger or stylus would not necessarily be circularly symmetric, and the ratio of a to b can tell us about its shape.

Knowledge of the angle ϕ can tell us how the ellipse is oriented, which might indicate which way the finger or stylus is pointing or at what angle it is tilted with respect to the touch surface. We can calculate ϕ from the eigenvalues and eignevectors of the 2 x 2 matrix M given in Equation 7**:**

$$M = \begin{bmatrix} A & B/2 \\ B/2 & C \end{bmatrix} \qquad \text{Equation 7}$$

Blob Detection

The above post-processing steps work for isolated touches, like single fingers, but don't handle extended "blob"-like touches. The FMT demonstrator currently does not do blob detection and, instead, responds to an extended touch with a swarm of single touches. We are currently investigating highly-parallel, low-latency algorithms for detecting and representing extended touches.

Frame Matching

To properly track objects moving on the touch surface over time, it is important to match the calculated touch points with each other between frames. While this is a fundamentally hard problem – and insoluble in the general case – we are aided by geometry and physics. Because the items that are in contact with the touch surface are of finite size, and move according to certain physical principles, we will not experience the worst cases.

Fingers and styluses have a minimum size and are unlikely to approach each other closely enough to cause an ambiguous case. They also travel at speeds characteristic of the motion of a human arm, which bounds the problem.

Because the our touch system has such a high update rate – well over a kilohertz – fingers and styluses touching the surface cannot move arbitrarily far or at extreme angles from one frame to the next. This makes the problem much easier, allowing us to do frame matching mostly by choosing the closest point in adjacent frames. Another useful heuristic for matching touch points between frames is the use of dynamics, calculating the time derivatives of touch point positions and using them to predict the likely position in the next frame. We can also use the signal strengths and shapes of previous touch points to distinguish between touch points and to infer likely candidates for matches. Both the Moore neighborhood calculation and extended blob matching would be useful.

A robust frame matching system would likely combine all of the above techniques, performing simple matching in most cases and more complicated matching in ambiguous ones.

IMPLEMENTATION

Our demonstration system appears in Figure 5. It was built around circuit boards and development kits that were intended for radio astronomy use. We designed the FMT demonstrator to be extremely flexible and adaptable so that we could experiment with parameters and new use cases. We can control the transmit signal strengths, independently set the transmit frequencies on a per-row basis, control the receive timing and FFT spacing, turn the CDM modulation of the sinusoids on and off, mask the analog-to-digital converter outputs to simulate different ENoBs (effective numbers of bits), etc. This flexibility is very useful in exploring ways that the system can be optimized, trying new touch substrate materials, etc.

The Touch Surface

The touch surface itself was created from a two-layer printed circuit board. It approximates a 25 centimeter diagonal screen, similar to an Apple iPad, and contains 30 rows and 40 columns. The top layer contains the rows and columns, and the bottom layer contains the ground plane (and jumpers for the columns). The row and column pattern consists of interlocking diamond shapes, similar to those used by DiamondTouch and in many commercial PCAP touch sensors. This geometry provides the maximum capacitive coupling between the touching finger and the rows and columns, while minimizing cross talk between those. The row and column pitch is 5 millimeters, which we have found is well matched to the size of human fingers.

Need for a Ground Plane

Not all PCAP touch systems use a ground plane under the rows and columns. The ground plane helps with several issues. First, it ensures that all capacitive interactions occur very close to the physical surface, eliminating "hover" and producing a very satisfying touch experience. Without the ground plane, the touch detection occurs more gradually as a finger approaches the surface, producing a "mushy" feel.

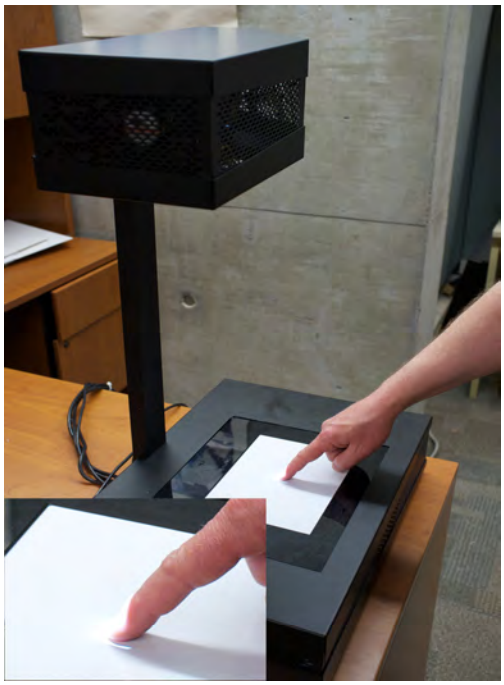

Figure 5: The FMT demonstrator. Note the dot of light on the finger, emanating from the high-speed projector.

Second, the large capacitance between the ground plane and the rows and columns dominates the smaller capacitance between them and the environment. This means that, when a row/column intersection is touched, the amount of row signal transferred onto the column increases. Without the closely coupled ground plane, the signal would actually decrease. Most commercial PCAP systems do not use a ground plane.

Signal generation hardware

Our row signal generators consist of a bank of forty Analog Devices AD9834 direct digital synthesizers that are set up to generate sinusoids in the range from DC to about 12 MHz. They are individually programmable and contain a phase selection input that can be used for bi-phase modulation. They are followed by an amplifier that lets us transmit row signals with amplitudes up to five volts peak-to-peak.

Receiver hardware

Our receiver front ends consist of forty Texas Instruments VCA8613 programmable gain amplifiers with a gain of up to 40 dB and a 12 MHz low-pass filter. These feed a bank of analog-to-digital converters, based on the TI ADS5272, which sample at a rate of 50 Msps. The ADCs have high-speed serial outputs that feed the touch processing unit.

Processing

The processing unit is a "mini-ROACH" (**R**econfigurable **O**pen **A**rchitecture **C**ompute **H**ardware) field programmable gate array (FPGA) board that was developed as part of the CASPER (Collaboration for Astronomical Signal Processing and Electronics Research) effort [13]. The mini-ROACH is based on a Xilinx Virtex-5 FPGA and includes Ethernet and a large amount of digital I/O. It performs both the row signal detection for each column receiver and also post-processing to turn the "heat map" into useful, interpolated touch events.

Our processing does not use any "tricks", such as assuming a limited number of touch points or where the touches might be. There are no fundamental limits on the number of touch points that can be detected or tracked. For convenience of implementation, the firmware is currently limited to sixteen simultaneous touches, but this can easily be changed by recompiling with new parameters.

Outputs

The FMT processor board has two outputs, including a fast, low-latency parallel port that sends processed touch events to the high-speed display, and an Ethernet port. The Ethernet port is used for control and monitoring, including being able to upload raw "heat maps" to a computer for visualization and debugging. The low-latency parallel port includes a "dial a latency" feature, allowing us to add calibrated amounts of delay to the demonstrator for testing purposes.

Display

We required an extremely fast video display to demonstrate FMT's capabilities without adding its own excessive latency. The DLP Discovery 4100 kit from Texas Instruments, which uses their digital micro-mirror technology to modulate light at a 32 kHz rate, was able to provide the necessary speed and low latency. This is the same display used by Ng *et al* [2].

PERFORMANCE

The touch sensor operates with an update rate of 4 kHz and each of those frames has a latency of 40 microseconds. The mismatch was done for implementation convenience and, during the remaining 210 microseconds of a 4 kHz frame, the processor is basically idle. A more optimized system could use the extra time to provide a 25 kHz frame rate, or to perform a longer integration (from 40 microseconds to 250 microseconds), increasing the system's signal-to-noise ratio by 8 decibels.

We have calculated the touch sensor's inherent latency from our FPGA code, and also directly measured the entire system's end-to-end (touch instant to display output) latency. We used a piezo transducer as a fast impact sensor to detect the instant of touch by attaching it to a fingertip and tapping the sensor quickly. The piezo transducer generates a voltage when it is mechanically strained, which is easily seen on oscilloscope when the transducer is tapped on a hard surface. Its metal construction registers a touch point on the surface as well as a finger alone does. We also used a photodiode sensor to measure the light level being projected onto the tapping finger. Using an oscilloscope, we measured the time difference between the tap and the displayed response. Our system's end-to-end latency, from touch to light, is approximately 110 microseconds.

Because FMT makes no prior assumptions about the number and kinds of touches, and merely reports high-speed samples of where it has been touched, measuring the latency of a single finger tap is sufficient to know he latency of any other kind of touch operation, such as dragging.

High speed video, taken at 400 fps and 1200 fps, confirms these figures and shows a dot of light keeping up with and tracking the finger very smoothly. There is no noticeable lag.

Variable Latency "Pong"

To see how changes in latency affect video game play, we implemented a version of the classic "Pong", but with one player experiencing no perceptible latency and the other burdened with 150 milliseconds of lag. While we did not do a formal user study, there was anecdotal consensus that the "zero" latency side was much easier to play. The high latency side required some getting used to, but even so, was significantly disadvantaged.

IMPLICATIONS

An extremely low-latency touch system opens up many new application areas. Opaque FMT can be used to implement highly-responsive track pads and game controls, providing unprecedented, immersive gaming experiences.

FMT is not limited to flat surfaces, and could be implemented on any two-dimensional manifold. Sports equipment, such as golf clubs and tennis rackets, could be instrumented to show where the player's grip is, and how it changes over the course of a swing.

Automobile dashboards, steering wheels and entertainment systems could take advantage of high-resolution, low latency touch to permit the driver to control the cabin environment in an intuitive, almost subconscious manner; because latency makes a user interface harder to use and causes distraction, removing that latency could keep more of the driver's attention on the road and increase traffic safety.

Transparent versions of FMT can be designed into tablets, phones and other mobile devices. Although these will not achieve the strikingly low end-to-end latencies described here, due to the delays added by their software and display, about 15-20 milliseconds of latency will be removed in a stroke, which will incrementally improve the user experience. And as latency is removed from other parts of these devices, we will approach our goal of user interfaces that have "zero" perceptible lag.

CONCLUSIONS

Our system works well and has a very good feel. It's scalable to both large and small sizes, due to PCAP's need to connect only rows and columns (which scale linearly), while the touch surface area scales as thee square.

When designing such a fast, low-latency system, we found it useful to think in terms of radio frequency design (transmission lines, signal strength, cross-talk, modulation) instead of slower, precision electronic design (capacitors, charge, electric field). Although both terminologies can be used to describe the same physical implementation, we believe that the former mindset helped to influence our design decisions to meet our goals more rapidly and with a better final outcome. FMT's rows and columns, above a ground plane, are in effect leaky transmission lines between which a finger can induce cross-talk.

FUTURE WORK

Although our demonstration surface is opaque and uses front-projection, we realize that many practical commercial products will require a transparent touch surface on top of an LCD or OLED display, as might be found on a mobile computing device. We are currently working toward a transparent version that will provide one-millisecond class performance with existing (or perhaps optimized) ITO and metal mesh touch sensors, despite their lower conductivity.

If multiple users are touching the surface at the same time, it would be useful to disambiguate one user's set of touches from the others'. This would allow proper recognition of multi-finger gestures by each user. We believe that it would be possible to implement such a capability by taking advantage of residual row signals that travel through a user's body, from each of their touching fingers to the others. We are actively investigating this area.

We are also researching the possibility of including an active stylus capability, in which the stylus transmits its own compatible set of orthogonal signals to the touch surface. We believe that this might be a method of providing a very rich, multimodal input experience, including extremely low latency.

The current FMT demonstrator is bulky and power hungry, consuming nine watts (not including the high-speed projector). We are working to move the technology onto an ASIC, which would permit ultra-low latency touch capability in a much smaller form factor that consumes much less energy. We are also investigating power saving features that would allow FMT to be used on even highly constrained battery-powered devices.

One of our goals for future work is attempting to determine if virtual interaction, implemented with a low-latency touch sensor and display, can sufficiently imitate a real-world physical interaction to fool an observer. We call this the "Touch Turing Test", because it is analogous to Alan Turing's famous thought experiment on how to determine if a computer can be considered "intelligent" [14]. In our case, we wish to test how closely virtual interactions can come to their real world counterparts. We believe that this is important because the human mind evolved with the constraints of the physical world and is more accustomed to these natural types of interaction. User interfaces designed with this in mind may behave in better and more interesting ways.

ACKNOWLEDGMENTS

We would like to acknowledge and thank Rick Raffanti of Techne Instruments, who put together the FMT sensor demonstrator, and Alex Rodionov of the University of Toronto, who programmed the high speed DLP projector to visualize the output of our sensor.

REFERENCES

[1] R. Jota, A. Ng, P. Dietz and D. Wigdor, "How fast is fast enough?: a study of the effects of latency in direct-touch pointing tasks" in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*, 2013.

[2] A. Ng, J. Lepinski, D. Wigdor, S. Sanders and P. Dietz, "Designing for Low-Latency Direct-Touch Input" in *Proceedings of UIST 2012*, Boston, MA, 2012.

[3] F. Bérard and R. Blanch, "Two touch system latency estimators: high accuracy and low overhead" in *Proceedings of the 2013 ACM international conference on Interactive tabletops and surfaces (ITS '13)*, New York, NY, 2013.

[4] E. Johnson, "Touch display–a novel input/output device for comuters" *Electronic Letters,* vol. 1, no. 8, pp. 219-220, 1965.

[5] S. Lee, W. Buxton and K. Smith, "A Multi-Touch Three Dimensional Touch-Sensitive Tablet" in *ACM CHI*, 1985.

[6] J. Rekimoto, "SmartSkin: an infrastructure for freehand manipulation on interactive surfaces" in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '02)*, 2002.

[7] W. Westerman, Hand Tracking, Finger Identification, and Chrodic Manipulation on a Multi-Touch Surface, University of Delaware, 1999.

[8] P. Dietz and D. Leigh, "DiamondTouch: a multi-user touch technology" in *Proceedings of the 14th annual ACM symposium on User interface software and technology (UIST '01)*, 2001.

[9] N. Mehta, Flexible Machine Interface, Toronto: University of Toronto, 1982.

[10] J. Han, "Low-cost multi-touch sensing through frustrated total internal reflection" in *Proceedings of the 18th annual ACM symposium on User interface software and technology (UIST '05)*, 2005.

[11] P. McAvinney, The Sensor Frame - A Gesture-Based Device for the Manipulation of Graphic Objects., Carnegie-Mellon University, 1985.

[12] J. Moeller and A. Kerne, "ZeroTouch: an optical multi-touch and free-air interaction architecture" in *ACMCHI*, 2012.

[13] CASPER: "Collaboration for Astronomy Signal Processing and Electronics Research" [Online]. Available: `https://casper.berkeley.edu/`.

[14] A. Turing, "Computing Machinery and Intelligence" in *Mind, 59*, 1950, pp. 433-460.