AnglePose: Robust, Precise Capacitive Touch Tracking via 3D Orientation Estimation

Simon Rogers
srogers@dcs.gla.ac.uk

John Williamson
jhw@dcs.gla.ac.uk

Craig Stewart
craig@dcs.gla.ac.uk

Roderick Murray-Smith
rod@dcs.gla.ac.uk

School of Computing Science, University of Glasgow, G12 8QQ, Scotland, UK

ABSTRACT

We present a finger-tracking system for touch-based interaction which can track 3D finger angle in addition to position, using low-resolution conventional capacitive sensors, therefore compensating for the inaccuracy due to pose variation in conventional touch systems. Probabilistic inference about the pose of the finger is carried out in real-time using a particle filter; this results in an efficient and robust pose estimator which also gives appropriate uncertainty estimates. We show empirically that tracking the full pose of the finger results in greater accuracy in pointing tasks with small targets than competitive techniques. Our model can detect and cope with different finger sizes and the use of either fingers or thumbs, bringing a significant potential for improvement in one-handed interaction with touch devices. In addition to the gain in accuracy we also give examples of how this technique could open up the space of novel interactions.

Author Keywords

particle filter, touch, capacitive, mobile, probabilistic

ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: Input devices and strategies (e.g., mouse, touchscreen)

General Terms

Algorithms

INTRODUCTION

Touch sensing has rapidly become a major method of interacting with computing devices, especially mobile phones. However, a few minutes of use of even the most expensive touch phones makes clear that there are problems with the use of thumbs, as needed for single handed interaction [13], and that even with fingers, users must adjust their tapping behaviour to suit the constraints of the current sensor signal processing technology. This is because the current devices' sense of touch is impoverished compared to the quality of information that could be captured. Most capacitive touch devices can measure one or more points of contact; but fingers are not points floating in space. The pose of the finger engaged in touching contains much salient information – from static indications of grip (left hand or right hand? portrait or landscape? thumb or index finger?) – to dynamic motions involving twisting or levering. Enhanced touch sensing has been demonstrated elsewhere ([8, 20]), but such methods rely on additional non-mobile (and quite complex) sensing. In this paper we demonstrate rich pose detection using very low-resolution capacitive sensors and sophisticated probabilistic inference to extract every bit of information from the sensors.

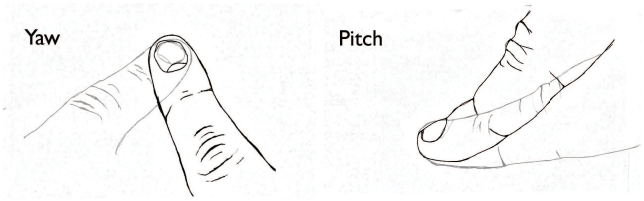


Figure 1. Yawing and pitching of the finger while the position of the intended contact point remains unchanged.

We extend the particle filtering work described in [16] to do real-time tracking of finger angle – both pitch and yaw – using a probabilistic model of the form of the finger as it appears to the sensor. Additionally, this model automatically adapts to the small but significant variation in gross finger shape among the population. The ability to infer these user- and touch-specific variables in real time allows the system to more accurately deduce the intended touch point. Experiments in [8] demonstrated that varying pitch and yaw led to consistent variation in *sensed* touch location for the same *intended* touch location. By explicitly modelling these characteristics, we demonstrate how it is possible to improve the accuracy of pointing. These improvements open up the use of smaller devices and denser control layouts, both as a result of increased accuracy in determining intended touch points, and because a range of new interaction techniques based on finger angle detection can be implemented. This technique works with multiple points of contact using small, coarse sensor arrays.

CHI 2011, May 7–12, 2011, Vancouver, BC, Canada.

The models in our system are fully probabilistic, and provide distributions over finger parameters. When the sensing is insufficient to reliably determine the position or orientation of the finger, the distribution naturally represents the increasing uncertainty. Applications using these methods can degrade gracefully in a range of ways, from simply ignoring very noisy inputs to gradually handing over autonomy to the system as control quality decreases. This handover of control, is referred to as the "H-metaphor" in [4], as a metaphor for the relation between a horse and rider, and is explored in the context of touch sensing in [16]. The availability of meaningful uncertainty estimates is one of the major advantages of this approach to touch interaction.

Current mobile touch technology exhibits reasonable accuracy in stationary settings with two-handed use. Performance rapidly deteriorates when interacting one-handed or while mobile. The ability to infer full finger pose with well-calibrated uncertainty estimates offers potential improvements in such scenarios. Improving the accuracy in tasks such as typing on small onscreen keyboards whilst walking would substantially improve the usability of many current mobile devices.

Literature

A number of papers have used sensing of finger positions using a range of technologies to explore novel interaction mechanisms. Rekimoto and colleagues' *Presense* [14] system used capacitive sensing to infer finger position above a conventional keyboard, and explored its use for previews and gestures. Alternatives to capacitive sensing in the literature include magnetic sensing [7], and vision systems [12]. [8] investigates the reliability of touch input using fingerprint scanners, and demonstrated consistent offsets ascribable to variations in pitch, yaw and roll. [10] reviews single-handed interaction. [11] describe problems with multitouch gestures. [17] discusses techniques for one-handed data entry, and [6] expands on this. [18] shows how finger rolls could be distinguished from slides, and provides a range of interaction techniques. [1] discusses dual finger techniques for precise selection, and their *SimPress* method used finger rocking detection to activate certain interaction mechanisms. Work on pen-based interaction, such as [5] is also relevant, given the ready availability of proximity measures in many pen input systems.

Particle filtering, the algorithm used in this work for probabilistic tracking, is described extensively in the literature. Particularly relevant and accessible is the work of Isard and Blake [9], who used particle filters for visual tracking. The technique is described more comprehensively in [15, 2].

THE MODEL

Key to our approach is the development of an explicit model of a finger in contact with the sensor array. The model describes the finger position, the size of the contact area and the finger's pitch and yaw and we can infer each of these parameters in real time. This is made possible by the fact that capacitive proximity sensors (such as those shown in Figure 5) detect parts of the finger that are not in physical contact with the array – they detect the *image* of parts of the finger which are not touching the sensors. With our relatively small sensors, this gives a vertical range of approximately 15mm.

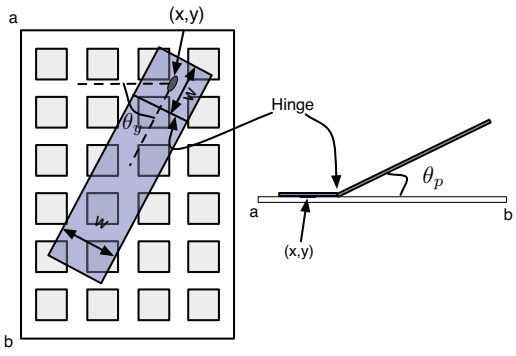


Figure 2. The finger model. The finger is represented by a hinged rectangle, here shown over a 6×4 sensor array.

In our work, the finger is modelled as a hinged rectangular surface. This can be seen in Figure 2. The model is described by 5 parameters that we shall collectively describe as $\boldsymbol{\theta}$. These five parameters correspond to $x, y, w, \theta_y, \theta_p$, as shown in Figure 2. These are (respectively) the estimated intended contact point, the finger width, the angle of yaw and the angle of pitch. All of these parameters are inferred by the system and in our experiments, the intended touch point is taken as (x, y), the center of the $w \times w$ square. Because of the size of the finger relative to the size of our hardware, we assume that the rectangle is infinitely long although an additional parameter could be used. This model gives the system the ability to infer a broad spectrum of different touch types (i.e. left/right handed users through different yaws, different finger sizes through different values of w, one- or two-handed operation through combinations of all parameters). It is possible to imagine more sophisticated models for a finger that would be suitable in this case – the model described here is relatively crude. For example, one may wish to encode more of the physical constraints on finger positions by modelling the jointed action of the finger. It is important to note that the inference framework that we describe does not preclude any such model and is certainly not unique to the model we have chosen.

ORIENTATION TRACKING

Our implementation allows for real-time tracking of the finger pose $[x, y, z, w, \theta_y, \theta_p]$ as touch is not a static process. It seems reasonable to assume that if we knew the values of $\boldsymbol{\theta}$, they would vary within a particular touch operation (as the finger approaches the sensors and then leaves) and across several touch operations (it is unlikely that a human can perfectly reproduce a particular movement). In addition, the characteristics of the touch (and therefore the values of $\boldsymbol{\theta}$) vary depending on *where* the user is touching. As an example, consider holding a touch-screen mobile phone in the right hand and using the thumb of the same hand to touch the top left of the screen and then the bottom right. The size of the contact area (the skin in contact with the sensors) and the thumb angle will be different in both cases. To further complicate matters, different users will add an extra dimen-

sion of variability to the problem (as shown in [8]). The parameters within $\boldsymbol{\theta}$ obviously vary over time as the finger pose varies, and so they must be inferred in real time from the current state of the sensors.

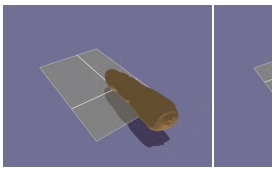


Figure 3. Finger pose estimation, with the contact point and orientation of the finger estimated in real-time.

Figure 3 shows a visualisation of the finger pose using our filter implementation. The pose compared with the actual and estimated sensor readings is shown in Figure 4. The finger angle estimation has some non-linearity in the estimated pitch angle due to the simplificiation of the finger into a flat hinged surface, but still varies monotonically with actual pitch angle. The filter gives the full distribution over the position and angles; as the finger is removed, the distribution expands as the sensing loses the ability to resolve the finger accurately. The angle estimation loses some accuracy towards the edge of the sensor array because there is insufficient coverage to distinguish different poses, but the increasing variance of the filter distribution makes this clear. The model is general enough to easily cope with multi-touch interaction, simply by introducing additional parameters to estimate (although the physical size of the hardware platform makes it hard to test multitouch beyond 2 contact points). Figure 4(d), (e), (f) shows the system estimating the orientation and position of two fingers simultaneously.

The availability of these degrees of freedom has obvious potential for designing new interfaces; effectively there is a now a joystick everywhere on the touch surface. The orientation estimation also directly improves the estimate of the finger position, because the model is a better approximation of the configuration of the the finger than orientation-less models. We anticipate that we can also improve touch interaction performance for dynamic gestures, such as distinguishing taps (which inherently involve some fingertip motion, which can be falsely interpreted as a dragging motions). Thus, the conventional 2D tracking accuracy should be improved by jointly tracking all the parameters of the model.

Inference

To infer the finger pose from the sensor values, we need a theoretical sensor model: what will the sensor "see" when the finger is in a given pose. In other words, a model that will allow us to predict $\mathbf{c}$ (the vector of sensor values) from $\boldsymbol{\theta}$. Comparing the theoretical values of $\mathbf{c}$ with the true value would allow us to rank candidate values for $\boldsymbol{\theta}$ or use more sophisticated methods (as we shall describe below). In this work, we use a method loosely based on the equation for the capacitance of a 2-plate capacitor where the plates overlap with an area A and are separated by a distance of D by a medium with dielectric constant ϵ: $C = \frac{A\epsilon}{D}$. Broadly speaking therefore, the capacitance is proportional to the overlap area and is inversely related to the distance. Given $\boldsymbol{\theta}$ and the size and position of the individual sensors, we can easily compute the area of overlap between the finger rectangle and each sensor. It is equally straightforward to compute the height of the finger rectangle above any position within a sensor (assuming that the rectangle is indeed above this position). Because the height will vary across each particular sensor, we use a single value for each sensor, computed at its centre. If the overlap for sensor i is A_i (expressed as a proportion of total sensor area, $0 \leq A_i \leq 1$) and the height of the centre is given by h_i, we compute the theoretical capacitance of sensor i (z_i) as: $z_i = A_i h_i^{-2}$ where h_i is clipped at a minimum value of 1 to ensure $0 \leq z_i \leq 1$, and compatibility with the sensor output values. h_i is squared to compensate for the nonlinear shape of the finger (it tends to get steeper) and proved to give far superior empirical results to using h_i. Much like the finger model, this sensor model is quite crude and could be improved. However, as we demonstrate later, it gives excellent empirical performance.

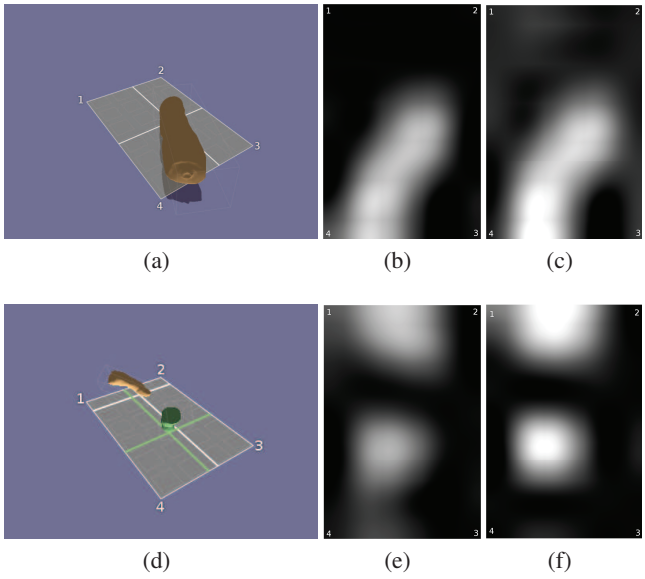


Figure 4. The finger pose (a), as shown in the visualisation, along with (b) the mean of the values the particle filter expected to observe – that is the average of the sensor "image" for each particle, given its current location and orientation, and (c) the the sensor values the that were actually observed. Values are interpolated from the 6×4 grid to more clearly show the finger shape. The numbers indicate the orientation. (d), (e) and (f) show the same visualisation for multitouch interaction.

The particle filter

Whilst it would be possible to use an optimisation routine to find an optimal value of $\boldsymbol{\theta}$ for each $\mathbf{c}$, we follow [9] and [16] and adopt a *probabilistic* approach. Rather than a single value of $\boldsymbol{\theta}$, we will work with a probability density. In particular, $p(\boldsymbol{\theta}|\mathbf{c}, \Delta)$, the density of $\boldsymbol{\theta}$ given the current sensor readings ($\mathbf{c}$) and any other necessary parameters (grouped together in Δ). The two principle reasons for this are a) that the probabilistic approach provides us with far more information than simply optimising $\boldsymbol{\theta}$; information that can be

used by the current application to, for example, assist in the handover of autonomy ('H-metaphor'; see [16] for more discussion) and b) the tools exist to allow us to *track* the density, in real-time rather than having to re-optimise each time the sensors change. This reuse of information is important given that fingers move, but probably not very far within the sampling period of the sensors (which might be ca. 15-20ms).

Due to the nature of the theoretical sensor model, the density $p(\boldsymbol{\theta}|\mathbf{c}, \Delta)$ cannot be computed analytically. As in [9] and [16], we will use a particle filter to track it in real time. Particle filters (PFs) are from a family of techniques known collectively as Stochastic Monte-Carlo techniques (SMC; see [2]). Particle filters use a set of samples (*particles*) from the density as a proxy for the actual density at a particular time. Given S particles at time t, the filter produces a set of S particles at time $t+1$ by sampling from the population at t with probability proportional to how good the particles are – how well their theoretical sensor values agree with the actual values – and modifying them according to a pre-determined model of how we anticipate the finger (and hence $\boldsymbol{\theta}$) changing – in other words a model of the dynamics of the finger. At any time instant, these particles can be passed to the current application as a proxy for the full density (as in [16]) or can be used to compute expectations. For example, if a single value of $\boldsymbol{\theta}$ is required, the particles can be used to compute the mean of $p(\boldsymbol{\theta}|\mathbf{c}, \Delta)$. Similarly, if one is interested in the covariance of the elements of $\boldsymbol{\theta}$, this is also easily computed.

1. Initialise particle population

The first step, is populating the filter with a set of S particles. In the absence of any sensor information (and because we have to start with something), these are drawn from a prior density, $p(\boldsymbol{\theta})$. The form of this density should reflect our belief in the poses that are possible. Although in reality there are clear physical relationships between the elements of $\boldsymbol{\theta}$ we assume independent priors on the different elements, $p(\boldsymbol{\theta}) = p(x,y)p(w)p(\theta_p)p(\theta_y)$. It is important to realise that this does not imply that the elements will be independent in the *posterior* density $p(\boldsymbol{\theta}|\mathbf{c}, \Delta)$. For example, we shall see that there are clear dependencies between the position (x, y) and the two angles.

2. Particle weighting

Given a current population of S particles, $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_S$, they are assigned *weights* representing how closely the particles expected sensor values match the true sensor values. For each particle (say $\boldsymbol{\theta}_s$), we compute its implied theoretical sensor value for each of the C sensors using the equation for the sensor model. This results in a vector $\mathbf{z}_s = [z_{s1}, \ldots, z_{sC}]^T$. Each $\mathbf{z}_s$ is then compared with the true sensor values $\mathbf{c} = [c_1, \ldots, c_C]^T$ $(0 \leq c_i \leq 1)$ to produce a weight value $\widetilde{w}_s$ according to a Gaussian likelihood:
$\widetilde{w}_s = K \exp\left\{-\gamma \sum_{i=1}^{C} (z_{si} - c_i)^2\right\}$ with precision (inverse variance) γ and normalising constant K. The Gaussian likelihood suffices as a simple and tractable measure of similarity. Finally, the weights are normalised so that they sum to 1 over the complete particle population: $w_s = \frac{\widetilde{w}_s}{\sum_{t=1}^{S} \widetilde{w}_t}$.

3. Particle re-sampling

Once the particles have been weighted according to their quality, they can be used to create a new generation. The new generation consists of two types of particle: R random particles (drawn from the prior $p(\boldsymbol{\theta})$, as in step 1 above) to ensure that the system can track very rapid changes and $(S-R)$ re-sampled from the current population. For each of these $(S-R)$ particles, a particle from the previous generation – say $\boldsymbol{\theta}_s$ – is chosen with probability w_s and then modified to become a particle in the new generation – say $\boldsymbol{\theta}_{s'}$ – according to some density $p(\boldsymbol{\theta}_{s'}|\boldsymbol{\theta})$. This density encodes our belief in how the finger moves, and again here we make the prior assumption of independence: $p(\boldsymbol{\theta}_{s'}|\boldsymbol{\theta}) = p(x_{s'}, y_{s'}|x_s, y_s)p(w_{s'}|w_s)p(\theta_{ps'}|\theta_{ps})p(\theta_{ys'}|\theta_{ys})$. Once the particle population has been re-sampled, the system returns to the particle weighting step (step 2) using the new current sensor input.

Particle generations

An additional feature of the filter that we use in our experiments is the *generation* of a particular particle. Randomly produced particles (either at step 1 or as one of the R random ones in step 3) are given a generation of 0. When a particle is re-sampled, its generation is incremented. We use the average generation within the population to signal a press operation - as the finger is temporarily stationary, the average generation will be gradually increasing.

The model requires the choice of several densities; priors and movement models. Rather than being a weakness, this is a strength of the proposed approach. Take for example the prior probability on particle position $p(x, y)$. There will often be information available to us from the current application that can inform this choice – the position of targets, buttons or links. Movement models are harder to pin down but it is worth remembering that all we are interested in is how far the finger is likely to move in a very small time instant. As such, we have found that Gaussians with very low variance are suitable for all parameters. This is not to say that more realistic models (such as the minimum jerk model of Flash and Hogan [3]) could not be used – the only restriction on the choice of these densities is that we can sample from them.

EXPERIMENTS

To determine the enhancements that better pose tracking can provide, we ran two experiments using the particle filter model described above. In these experiments, we sought to determine whether the pose information could be used to improve the accuracy of targeting; in other words whether the system could better interpret where users had *intended* to touch. Our hypothesis is that our pose estimating particle filter will yield higher accuracy – the estimated touch point will be closer to the intended touch point – than naive interpretation of the sensors. We did not conduct a formal study of people's ability to directly control the pitch and yaw of their fingers, partly because of the difficulty in reliably controlling such an experiment (as described in [8]), and partly because if orientation tracking works well, this will be directly reflected in the improvement in targeting accuracy. Two experiments

were performed, both intended to determine targeting accuracy on a simulated numeric keypad. Participants were asked to touch marked points on a keypad layout (see Figure 5). The data from these touches was captured, and the test algorithms were run on this data to obtain the estimated touch points. The system used in the experiments gave participants no feedback as to how either of the algorithms being tested interpreted the touch; this ensured that an entirely fair and blind comparison of the algorithms could be made.

The first experiment involved participants "entering" a sequence of digits by touching marked points on a flat surface, firstly with their fore-finger with the device resting on the table, and secondly with their thumb while their hand gripped the sensor in a phone-shaped case.

The second experiment involved participants entering sequences of four digit groups (as in PIN entry for unlocking the device, a task often done single-handedly when users are engaged with other tasks), again on the flat, marked surface. The tasks in this experiment were performed with the thumb only, and with two different sizes of target grid. The total size of smaller target grid was smaller than most participants thumbs.

Hardware

For experimental testing, an SK7 sensor pack from SAMH Engineering[1] was interfaced to the SK7-ExtCS1 capacitive sensor array (see Figure 5). This array is a capacitance-to-ground (loading mode) touch sensor with 24 individual sensing pads in a 6×4 layout. This is in contrast to the row-column sensors on many touch controllers. This has the advantage of having a sensitive region which extends further from the pads at the cost of reduced XY resolution. The sensitive region measures 52mm×34mm. Each pad is 7mm×7mm with a 2mm gap between sensors. The capacitance to ground on each pad is sampled at 5kHz with 16-bit resolution and then low pass filtered and decimated. A no-touch and full-touch level that are measured as part of an initial calibration are applied to the data and the resulting signals are sent to the PC as 8-bit resolution signals at 60Hz using a USB connection.

A plexiglass sheet of 2mm thickness was attached to the top side of the board with clear double-sided sticky tape, to ensure the absence of air bubbles. On to this sheet a 3×3 grid of target points was marked, at the junctions where four capacitive pads meet. This pad measured 22mm×22mm. A smaller 3×3 grid of 10mm×10mm was also laid out inside for testing extremely small target sizes. These grids are visible in Figure 5). The entire board was mounted a 40mm thick foam block and placed inside a plastic enclosure shaped like a mobile phone. This additional material made it possible to hold the assembly with the hands at the rear of the experimental hardware unduly interfering with the capacitive sensing. The surface of the board was smooth and featureless. The targets were only marked on visually.

Filter details

The details of the particular densities and parameters of the filter used in the experiments are given below: The sensor space is modelled as a 60×90 discrete grid (1 grid division = 0.67mm). All values given below correspond to this coordinate system. We used a total of $S = 1000$ particles, of which 20 percent were randomly sampled at each iteration and the remaining were re-sampled from the previous population. The individual components were defined as follows:

$$p(x,y) = \sum_{j=1}^{J} \frac{1}{J} \mathcal{N}_x(\mu_{xj}, 1)\mathcal{N}_y(\mu_{yj}, 1) \quad (1)$$

$$p(w) = \mathcal{U}_w(10, 20) \quad (2)$$

$$p(\theta_y) = \mathcal{U}_{\theta_r}(0, \pi) \quad , \quad p(\theta_p) = \mathcal{U}_{\theta_e}(0, \pi/2), \quad (3)$$

where $\mathcal{U}(a,b)$ is a uniform density between a and b, $\mathcal{N}(\mu, \sigma^2)$ is a Gaussian (normal) density with mean μ and variance σ^2 and Equation 1 describes a J component mixture where J denotes the number of buttons whose centres are defined by μ_{xj} and μ_{yj}. Note that elevation is restricted to be between 0 (finger flat) and $\pi/2$ (finger vertical). Also, as the users were only using the device in the upright orientation, the angle of rotation was restricted to be between 0 and π (the finger cannot approach from the top of the device).

We also assume prior independence across movement models. The individual movement models were all Gaussian densities, centred on the previous particle value with small variances. These gave excellent performance across a wide range of users. The precision of the Gaussian likelihood was set at a relatively high value of 1×10^5 although varying this made little difference to the results. In our experiments, we use the posterior average (the mean of $p(\boldsymbol{\theta}|\mathbf{c}, \Delta)$) computed after the weighting step and before the re-sampling step as:$\bar{\boldsymbol{\theta}} = \sum_{s=1}^{S} w_s \boldsymbol{\theta}_s$.

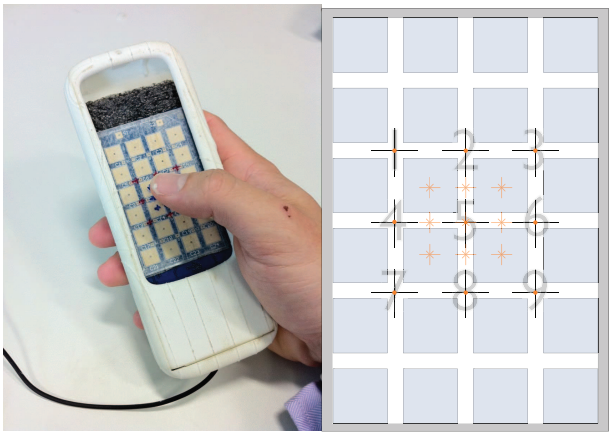
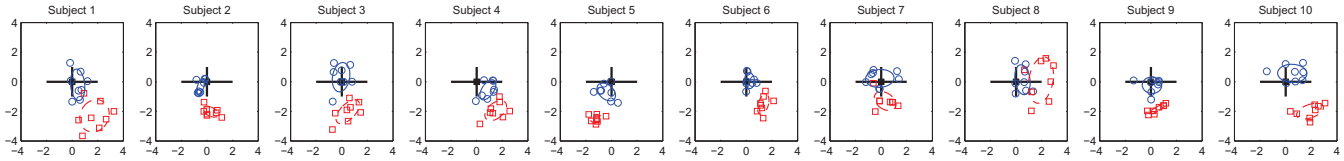


Figure 5. Left: the hardware used in this study. The 24 capacitive proximity sensors are clearly visible. Right: Experimental grid, shown to actual scale. The larger grid is marked with large dark crosses, smaller grid with small light stars.

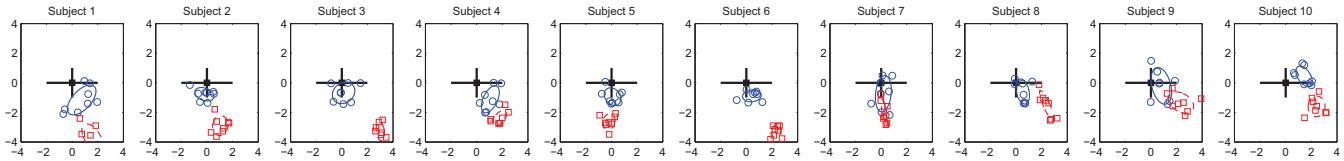
Experiment 1

Data was captured from 10 participants, 9 male, 1 female, aged between 20 and 41, all members of the local Computing Science Department. Subjects used their dominant hand for all tasks: eight of the subjects were right-handed (used their right index finger and thumb) and two (subjects 5 and 7) were left-handed (left index finger and thumb). All

[1] http://code.google.com/p/shake-drivers/



(a) All participants pressing digit '1' with their index finger.



(b) All participants pressing digit '1' with their thumb.

Figure 6. Results of ten participants touching digit '1' with their index finger. Blue circles show the model output and red squares the result of sensor interpolation. Axes are labeled in millimetres. Notice the consistent bias below and to the right for all subjects except those who are left-handed (5 and 7).

had some familiarity with touch screen devices. Users were not requested to touch in any particular manner and were free to touch however they felt comfortable. Diversity in touch styles was evident whilst observing the users and can be clearly seen in Figure 6. Each participant was asked to treat the dots marked on the larger 3×3 grid (shown to scale in Figure 5) as a numeric keypad in telephone layout, containing only digits 1 through 9. In each session, participants had to enter a pseudo-random sequence of digits. During the trial, the participant heard a numeral and were then asked to place their finger on the appropriate point until they heard a beep ($\approx$ 100ms after touching). The touch location was taken at the same time as the beep was heard. The use of audio cues is intended to remove any effects from dividing visual attention. The users received no feedback as to how the system interpreted their press.

This continued until 90 digits had been entered. Each participant completed this experiment twice, once using their forefinger with the device resting on a desk and once with their thumb whilst holding the device in the same hand (as shown in Figure 5). The system logged both the model output ($\boldsymbol{\theta}$) and the location determined by an interpolation of sensor centres: if $\mathbf{r}_i$ is the (x, y) location of the ith sensor, and $\widetilde{\mathbf{c}}$ is the vector of sensor values normalised so as to sum to 1, the interpolated position is given by: $\mathbf{d} = \sum_{i=1}^{C} \widetilde{c}_i \mathbf{r}_i$. This was chosen as a baseline for comparison because the conventional technique of finding the centroid of a blob on a binary thresholded sensor image would be unreasonably inaccurate on our low resolution grid. In the second experiment (below) we show that interpolation provides a reasonable approximation to the accuracy of state-of-the-art technology (the iPhone).

Figure 6 shows an example of the results obtained. The top row shows the data for all users pressing digit '1' with their index finger and the lower row shows them all pressing digit '1' with their thumbs. The units on the axis are millimetres. In each panel, the squares show the result of interpolation and the circles the position inferred by the model.

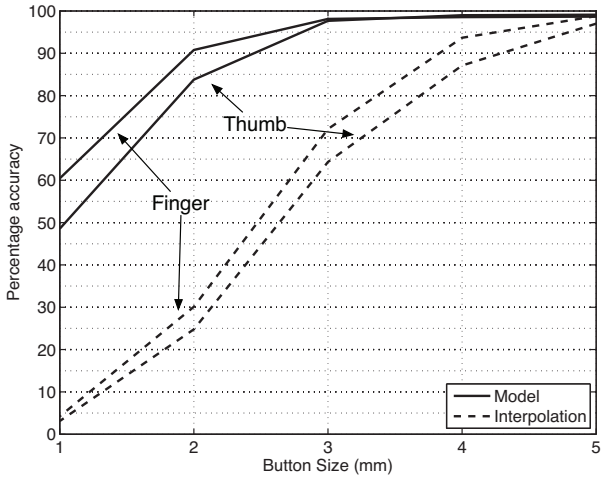


Figure 7. Percentage accuracy versus virtual button size (radius of a circular button). The model provides greater than 95% accuracy for buttons of size 3mm for use with either thumb or forefinger.

Two things are immediately obvious. Firstly, there is a great diversity amongst users (from left to right) and across finger/thumb for individual users (top to bottom). This confirms the observations in [8]. Secondly, the model provides a far more accurate position estimate than interpolation.

The improvement provided by the model is best quantified by computing the implied minimum button size that each method can tolerate. Assuming that buttons are round, with a particular radius, we can compute accuracy (percentage of times the touch would have been within the correct button) for a range of radii across all digits and subjects. The results are shown in Figure 7. The solid lines correspond to the model (for finger and thumb respectively) and the dashed lines to interpolation. The improvement offered by the model is clear: for both finger and thumb, 95% accuracy is achievable for a button with radius 3mm (note that because users sometimes entered the wrong digit, our accuracy estimates are likely to be conservative). This com-

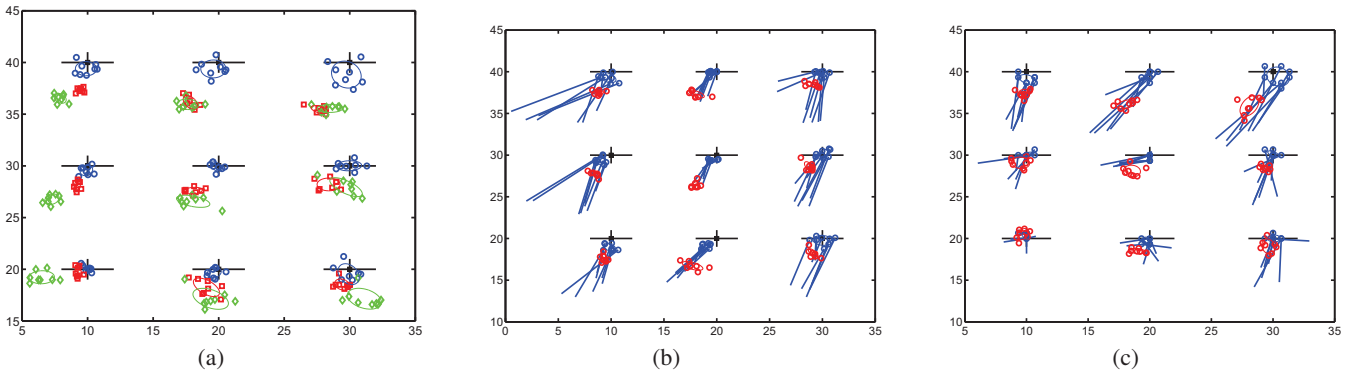


Figure 8. (a) Comparison with an implementation of a previous particle filter approach (green diamonds) for one representative user. Their model assumed that a finger could be represented as a point, as such a consistent bias is present. (b) and (c) Angles inferred for subject 5 when using finger (b) and thumb (c). Lines show projection of finger rectangle onto sensor plane (shorter lines correspond to greater pitch). This subject is left-handed. In (b) the lines are of a consistent length – when the device is on the table, the finger makes contact in roughly the same manner for each digit. In (c), the restrictions in movement caused by one-handed operation are clearly visible with pitch varying dramatically across the array.

pares very favourably with the results presented in [8] where a sensor radius of approximately 3mm (they show diameter of $\approx 5.4 \pm 1.4$mm) was achieved, when correcting for controlled pitch, roll and yaw variations. Note that interpolation gives values of approximately 70% and 65% at 3mm for finger and thumb respectively. Our results suggest that the interpolation can achieve approximately 95% accuracy for sensors of radius 5mm. This is significantly better than the value of approximately 7mm presented in [8] and is likely due to some of the extreme pitch, roll and yaw angles that the users were asked to perform in that study. The improvement that we see is a direct consequence of the explicit finger model. When a user performs a touch, parts of the their finger towards the hand will cause a response from the sensors. This response will naturally pull the interpolated value away from the intended touch location. By modelling this additional signal, our approach is able to automatically correct for this deviation.

Figure 9 plots the mean squared error for the model and interpolation across all right handed subjects. It is clear that the most consistent improvement is to be had (points further from the line $y = x$) when the users have to reach further to the target. Hence, the biggest overall improvement is for the thumb reaching across to the digit 1 with high improvements also seen for other digits towards the top and left of the grid. The smallest improvement is for those digits towards the bottom and right (9,7,6,3), where little of the thumb would be over the sensor area. The pattern for left-handers is similar (swap 1 for 3, 4 for 6, 7 for 9 etc) but not shown for space reasons.

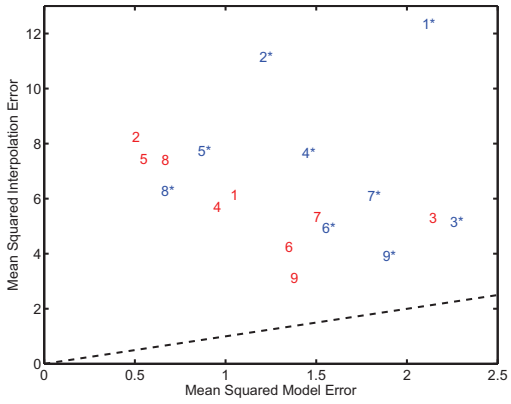


Figure 9. Mean squared error for model and interpolation across right handed participants (note the asymmetric axis scales) for each digit and thumb/forefinger. Plain (red) numbers correspond to the finger, numbers followed by a '*' (blue) correspond to thumb. Points above the $y = x$ line describe an improvement for the model over interpolation.

Experiment 2

The second experiment involved a more realistic data entry task, where participants were asked to enter a sequence of four digit numbers, again consisting of the digits 1 through 9 only. The hardware, particle filtering code and physical set up was identical to the first experiment.

The experiment consisted of two tasks, the first performed on a keypad of dimensions 18mm×18mm as in the first experiment (large grid in Figure 5), and the second performed on a grid of 10mm×10mm (small grid in Figure 5). The task was performed with the dominant thumb only, with the sensor held in the same hand in the phone-shaped case shown in Figure 5. Participants heard each four digit sequence read by a synthetic voice. Each time a touch was registered, the participants heard a beep. After four touches, the next four digit sequence was heard. This continued for 36 four digit groups (144 total digits). No feedback on the interpretation of the touches was given. 6 participants took part in this part of the study, 1 female, 5 male. 5 were right-handed and 1 left-handed, and again the tasks were performed with dominant hand only.

iPhone version

To determine whether our alternative (linear interpolation) algorithm was unreasonably poor compared to state-of-the-art commercial touch sensing, we replicated our experiment on a standard iPhone 3GS as accurately as possible. Raw sensor values are not accessible on this platform so we used

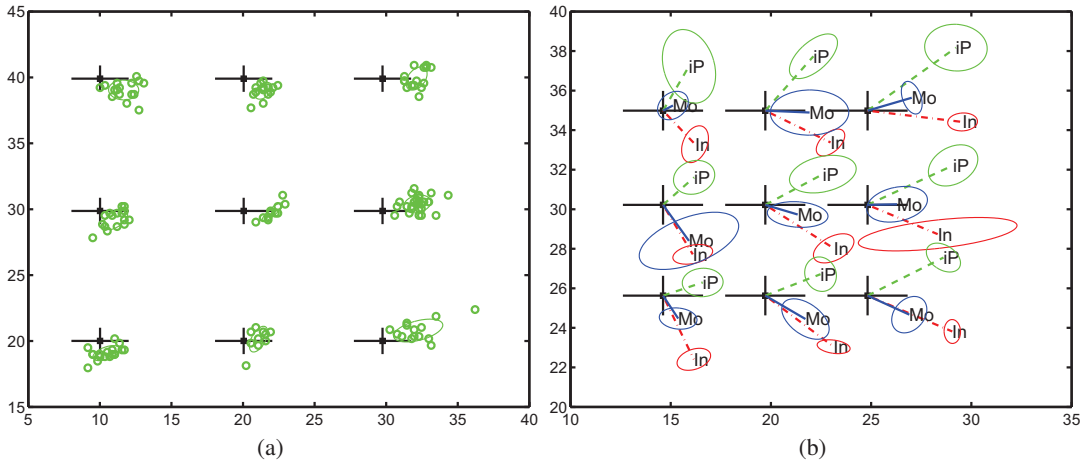


Figure 10. Comparison with the data from the iPhone (axes labeled in millimetres). (a) Raw iPhone data for subject A using their thumb on the large grid. Model and interpolation values not shown as too many points overlap. (b) Subject B using their thumb on the small grid. For clarity ellipses rather than raw data shown. Comparison of contact points for the model (Mo), Interpolation (In) and iPhone (iP). Here the model performs consistently better than both alternatives.

the contact point returned by the API and we took precautions to avoid the effect of any context sensitive target heuristics. An application which displayed an exact, to-scale image of the target grid was implemented. This was not an implementation of our particle filter but rather a test to ensure that our interpolation baseline wasn't unfairly inaccurate. The application followed an identical experimental protocol: users again held the device in their dominant hand and heard four digit sequences, whereupon they had to press on the target with their thumb and hold until a beep was heard. This was conducted for both the coarse and the fine target grid. 36 sequences were again used for each grid. The reported touch points on the iPhone were logged, and subsequently converted from pixels to mm. This gives a baseline measure of the accuracy of a high-resolution capacitive sensing system with post-processing. The results of this are shown in Figure 10. In general the iPhone performance was substantially better than the interpolation algorithm, but was not nearly as good as the pose tracking particle filter.

Although these are very preliminary results with a very small test population, it is worth noting that our system generally outperforms the iPhone, despite having a very coarse sensing array. The large sensor pad size in our hardware also makes hover tracking out to 15mm or more feasible.

DISCUSSION

The filter model clearly performs well in tracking the intended finger location with relatively coarse sensing hardware. The results indicate that the performance is much better than simple weighted interpolation of sensor locations in terms of accuracy. Although looking at plots of touch points for individual users would suggest a simple offset would give high accuracy, in reality the offsets between sensed and intended touch points vary across individuals and contexts. The pooled variance in touch location for the interpolated model is much larger than that given by our pose-tracking model. The model copes easily with the variation in digit width and orientation caused by using the device one-handedly.

Whilst in our experiments we have concentrated on touch events, the dynamic tracking performance compares competitively with the particle filter of [16], which does not model full finger pose. It is difficult to perform a rigorous comparison with the iPhone (due to differing hardware and lack of access to raw sensor values) but in the best comparison we were able to do, the accuracy achieved by the model exceeded that of the iPhone.

NEW INTERACTION POSSIBILITIES

We now explore more innovative interaction mechanisms based on finger pose dynamics – effectively giving users a subtle, context-sensitive joystick.

Rolling Context Menus Touching a surface to select an option is direct and intuitive but is limited by the size of the touching object. The initial finger position narrows down the range of options, but a secondary interaction method could open up controlled by pitch and bearing changes. For example, in Figure 12(a) we show a rolling context menu, which is controlled by changes in bearing. These techniques are similar to methods proposed in [6], but benefit from the ability to refine selection based on finger pose, rather than position in [6], or roll in [18].

Occlusion As the filter tracks the whole finger with a relatively high degree of accuracy, there are straightforward ways of automatically avoiding the common issue of finger occlusion. The work described in [19] could be enhanced with the knowledge of the orientation of the finger. Text could wrap around the "shadow" of the finger, adapting as the orientation changes.

Hidden Flaps The emotional content of messages passed between people changes the way such messages are opened. We suggest an interaction style where users conceal the message with their hand and reveal it carefully by levering the cover with their hand to expose the message (see Figure 12(c)). Secret messages passed between classmates are hidden from view and unfolded discreetly.

High-precision pointing The sensed finger contacts can be

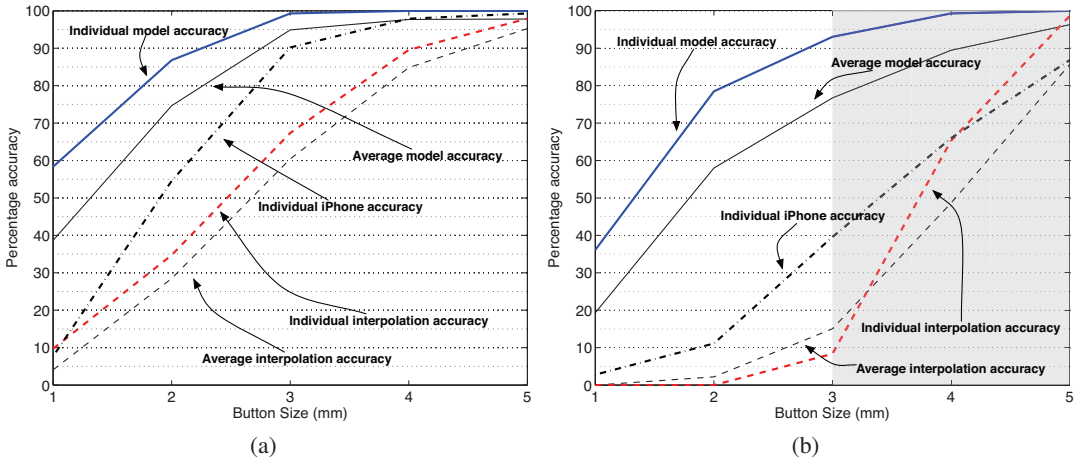


Figure 11. Effective button sizes for experiment 2. (a) Accuracy as a function of button size for the large grid. Shown are the average curves for the population as well as the specific performance for user A and the same users performance on the iPhone (data from Figure 10(a)). (b) As (a) but for the small grid and also showing the individual performance for subject B (iPhone data as per Figure 10(c)).

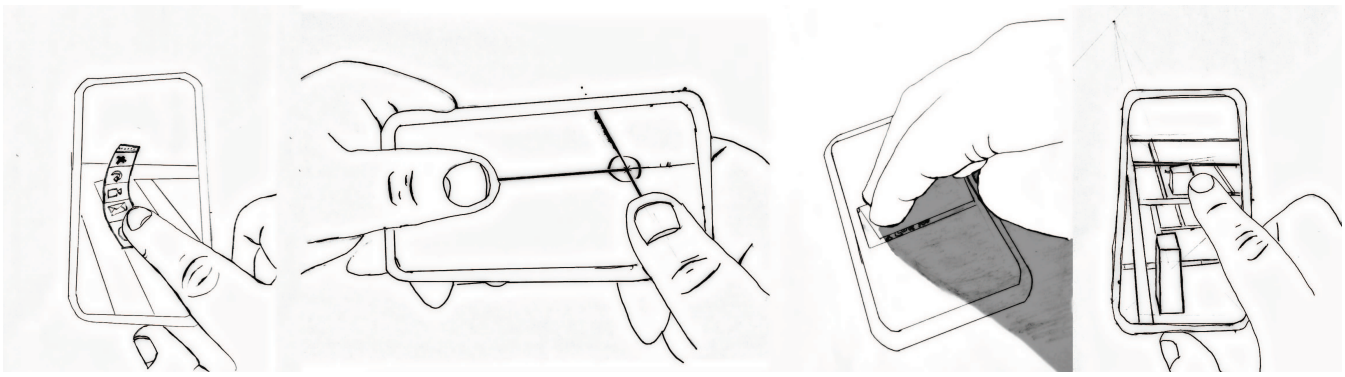


Figure 12. Interaction concepts using tilt-sensitive touch sensing on mobile devices. Left-to-right: (a) Rolling context menus; (b) High-precision bimanual pointing; (c) Peeking through message flaps using whole hand pitch angle detection; (d) 3D map navigation.

virtually "extended" like chopsticks emanating from the finger tips, similar to the work presented in [21]. The interaction techniques suggested in [20] for table-top interaction could be adapted for devices with capacitive touch screens. Figure 12(b) illustrates the concept.

CONCLUSIONS

Conventional touch sensing hardware is quite capable of extracting rich information about the pose and movement of fingers above and in contact with it. Our approach explicitly formulates a model of the finger and estimates the pose using probabilistic particle filtering. The model clearly encodes the assumptions about the physical nature of the fingers and the sensors, and is easy to extend to different contexts. This approach allows the extraction of additional degrees of freedom from the input, beyond simple 2D contact points. The probabilistic nature of our model results in meaningful uncertainty estimates as sensing degrades. It also makes for a consistent framework for integrating uncertainty from higher levels of the interface with and as a result the values measured by the sensor can be interpreted in the context of the entire interaction, from electrical and biomechanical constraints to inference about complex intentions. The potential uses for increased precision and additional degrees of freedom are numerous; we have identified several concepts which could be implemented on small-screen devices.

We have experimentally demonstrated that properly estimating finger pose with both pitch and yaw is both feasible on low-resolution capacitive sensor arrays, and significantly improves targeting performance when 2D target positions are estimated. The estimate of the point a user *intended* to touch can be improved using more complete knowledge of how the finger is posed in space. These targeting improvements mean that devices can have dense control layouts using only cheap and readily available electronics. The limitation on device size is not human finger size, but the crude interpretation of capacitive arrays in existing technologies. We have shown that buttons densely packed at a spacing of 4mm can be made usable with a sensor grid with a resolution of 6×4 pads in an overall area of 34mm$\times$55mm; even smaller layouts are possible if coupled with appropriate secondary interaction mechanisms, such as the rolling context menus. This size of button is equivalent to that required for a QWERTY key-

board on a small touch screen and improving accuracy here could have substantial usability benefits particularly when, for example, only one hand is available or the user is walking. Our hardware comprised a small number of long range capacitive sensors in contrast to the fine row-column style prevalent in current devices. Our investigation suggests that a shift towards the type of hardware that we have used could offer significant accuracy advantages at very low cost. Heterogenous sensors composed of a spectrum of coarse long-range sensors through ultra-fine but short range sensors with regular or irregular shapes can easily be used with our algorithms, unlike conventional approaches. Such sensor arrays represent a simple way of extracting detailed 3D pose information across the entire surface of a device. Our current implementation runs on a desktop system and has not been optimised for mobile technology. We expect that an efficient fixed point implementation could easily run on modern smartphones at low CPU cost.

These probabilistic models demonstrate that by simply modelling the problem in terms of the expected measurements given known constraints, surprising amounts of relevant information can be extracted even from crude sensors. Wider use of these techniques in HCI could dramatically extend the capabilities of existing and yet-to-be-imagined hardware.

ACKNOWLEDGEMENTS

This work was supported by EPSRC grant EP/E042740/1, the PASCAL2 Network of Excellence, and by a fellowship from the Scottish Informatics and Computing Science Alliance (SICSA).

REFERENCES

1. H. Benko, A. D. Wilson, and P. Baudisch. Precise selection techniques for multi-touch screens. In *CHI '06:*, pages 1263–1272. ACM, 2006.

2. A. Doucet, N. de Freitas, and N. Gordon, editors. *Sequential Monte Carlo Methods in Practice*. Springer, 2001.

3. T. Flash and N. Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *J. Neuroscience*, 5(7):1688–1703, 1985.

4. O. Flemisch, A. Adams, S. R. Conway, K. H. Goodrich, M. T. Palmer, and P. C. Schutte. The H-Metaphor as a guideline for vehicle automation and interaction. Technical Report NASA/TM 2003-212672, NASA, 2003.

5. T. Grossman, K. Hinckley, P. Baudisch, M. Agrawala, and R. Balakrishnan. Hover widgets: using the tracking state to extend the capabilities of pen-operated devices. In *CHI '06:*, pages 861–870. ACM, 2006.

6. T. J. Gunn, H. Zhang, E. Mak, and P. Irani. An evaluation of one-handed techniques for multiple-target selection. In *CHI '09*, pages 4189–4194. ACM, 2009.

7. C. Harrison and S. E. Hudson. Abracadabra: wireless, high-precision, and unpowered finger input for very small mobile devices. In *UIST '09*, pages 121–124, New York, NY, USA, 2009. ACM.

8. C. Holz and P. Baudisch. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In *CHI '10*, pages 581–590. ACM, 2010.

9. M. Isard and A. Blake. Contour tracking by stochastic propagation of conditional density. In *ECCV*, pages 343–356, 1996.

10. A. Karlson, B. Bederson, and J. Contreras-Vidal. Understanding single-handed mobile device interaction. In *Handbook of Research on User Interface Design and Evaluation for Mobile Technology*, pages 86–101. IGI, 2007.

11. M. A. Nacenta, P. Baudisch, H. Benko, and A. Wilson. Separability of spatial manipulations in multi-touch interfaces. In *GI '09:*, pages 175–182. Can. Inf. Proc. Soc., 2009.

12. T. Niikura, Y. Hirobe, A. Cassinelli, Y. Watanabe, T. Komuro, and M. Ishikawa. In-air typing interface for mobile devices with vibration feedback. In *SIGGRAPH '10*, pages 1–1. ACM, 2010.

13. P. Parhi, A. K. Karlson, and B. B. Bederson. Target size study for one-handed thumb use on small touchscreen devices. In *Mobile HCI'06*, pages 203–210, 2006.

14. J. Rekimoto, T. Ishizawa, C. Schwesig, and H. Oba. Presense: interaction techniques for finger sensing input devices. In *UIST '03:*, pages 203–212, 2003.

15. B. Ristic, S. Arulampalam, and N. Gordon. *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House, 2004.

16. S. Rogers, J. Williamson, C. Stewart, and R. Murray-Smith. Fingercloud: uncertainty and autonomy handover incapacitive sensing. In *CHI '10*, pages 577–580. ACM, 2010.

17. A. Roudaut, S. Huot, and E. Lecolinet. Taptap and magstick: improving one-handed target acquisition on small touch-screens. In *Proc. AVI '08*, pages 146–153. ACM, 2008.

18. A. Roudaut, E. Lecolinet, and Y. Guiard. Microrolls: expanding touch-screen input vocabulary by distinguishing rolls vs. slides of the thumb. In *CHI '09*, pages 927–936. ACM, 2009.

19. D. Vogel and R. Balakrishnan. Occlusion-aware interfaces. In *CHI '10*, pages 263–272. ACM, 2010.

20. F. Wang, X. Cao, X. Ren, and P. Irani. Detecting and leveraging finger orientation for interaction with direct-touch surfaces. In *UIST '09*, pages 23–32, 2009.

21. H. Wyss, R. Blach, and M. Bues. iSith - Intersection-based Spatial Interaction for Two Hands. In *3D User Interfaces, 2006*, pages 59 – 61, 2006.