Synchronous Gestures for Multiple Persons and Computers

Ken Hinckley
Microsoft Research, One Microsoft Way, Redmond, WA 98052
{kenh}@microsoft.com

ABSTRACT
This research explores distributed sensing techniques for mobile devices using *synchronous gestures*. These are patterns of activity, contributed by multiple users (or one user with multiple devices), which take on a new meaning when they occur together in time, or in a specific sequence in time. To explore this new area of inquiry, this work uses tablet computers augmented with touch sensors and two-axis linear accelerometers (tilt sensors). The devices are connected via an 802.11 wireless network and synchronize their time-stamped sensor data. This paper describes a few practical examples of interaction techniques using synchronous gestures such as dynamically tiling together displays by physically bumping them together, discusses implementation issues, and speculates on further possibilities for synchronous gestures.

Keywords
Distributed sensor systems, context awareness, ubiquitous computing, multi-user interfaces, input devices, sensors

INTRODUCTION
Humans have evolved to function within a fabric of social connections and collaboration. People work on problems in groups, and indeed the entire field of computer-supported collaborative work (CSCW) is devoted to technological support of such groups. Many user tasks and activities revolve around communication, which inherently involves at least two persons. Furthermore, with the burgeoning use of the internet, and research trends in ubiquitous computing and distributed systems, human-computer interaction often involves more than one computer. Yet the literature offers few examples of real-time interaction techniques that leverage the simultaneous data streams generated by multiple users and multiple computers.

This research introduces the general concept of synchronous gestures, which as defined above are patterns of activity spanning a distributed system that take on a new meaning when they occur together in time. These patterns could literally occur in parallel and in exact synchrony, or they just might be partially overlapped or even occur in a particular sequence. The key is that complementary portions of a signal are contributed by different devices or participants, and that the signal can only be recognized when these portions are brought together.

UIST '03 Vancouver, BC, Canada

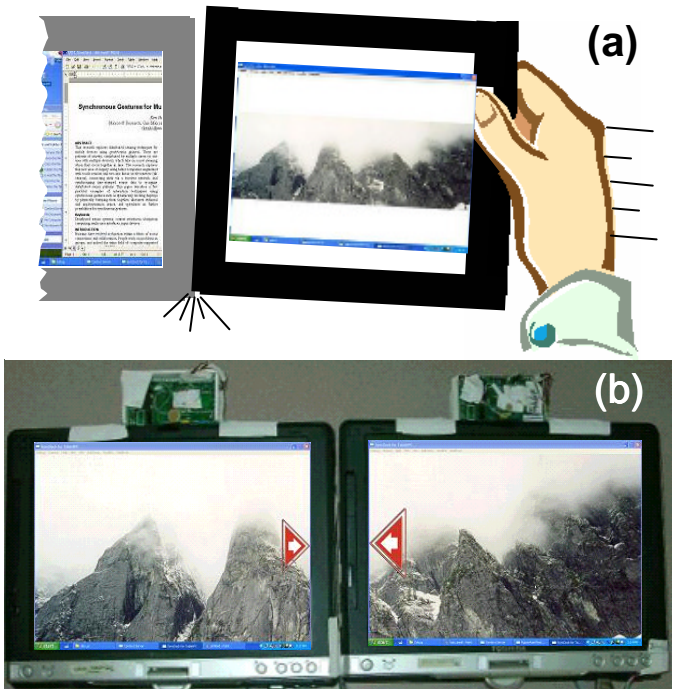

Fig. 1 **(a)** A user can bump a tablet into another one resting on a desk. The software recognizes the gesture by synchronizing the two accelerometers across a wireless network. **(b)** The tablet moved by the user annexes the display of the stationary tablet, allowing a panoramic image to span both displays.

Sensor fusion is a related concept, but generally sensor fusion refers to the use of multiple sensors in a single system to robustly detect properties of the physical environment. Synchronous gestures also involve multiple sensors, but the sensors come from multiple devices which are physically disaggregated, and may belong to different persons and thus compromise a unitary system only in the sense that they can communicate via a wireless network.

To demonstrate the practical utility of synchronous gestures this work proposes *dynamic display tiling*, a new technique which enables users to tile together the displays of multiple tablets just by physically bumping a tablet into another one lying flat on a desk (Fig. 1). Bumping generates equal and opposite hard contact forces that are simultaneously sensed by the accelerometer on each tablet. The two orthogonal sensing axes of each accelerometer provide enough information to determine which edges of the tablets have collided, allowing tiling of displays along any edge (left, right, top, or bottom). Picking up a tablet removes it from the shared display. Users can also exchange information by

bumping tablets together just as people at a dinner table might clink glasses together for a toast. This is distinguished from display tiling by sensing that both tablets are held (as opposed to one being stationary on a desk). Finally, one user can "pour" data from his tablet into that of another user by angling the tablet down when the users bump their tablets together.

Synchronous gestures represent a little-explored area that may offer much promise. This paper contributes an articulation and discussion of the general concept of synchronous gestures that has not appeared in the literature. We also contribute implementation issues and novel techniques which may spur further interest in and development of real-time distributed sensing techniques.

RELATED WORK

There are few previous examples of synchronous gestures. One example of a real-time interaction technique based on multi-device synchrony is the "Smart-Its Friends" technique [18], which allows a user to connect a pair of accelerometer-augmented handheld devices by holding the two devices together and shaking them. The device shares the data with other devices within range of its radio frequency transmitter, and looks for an identical pattern on the accelerometers to infer a shared context and establish a privileged connection between the two shaken devices.

There have been several efforts to sense social interactions between people. The Sociometer [7] is a wearable computing platform that uses post-hoc data analysis to sense when face-to-face interactions have occurred; for example, the system can sense that two people talked to one another using the mutual information of the voice streams.

Instant messaging and videoconferencing systems (such as the Portholes system [10]) often benefit from sensing user presence and making this available to other users. The Notification Platform [19] coordinates sensor information (mouse and keyboard activity, microphones, and cameras) from multiple devices to optimize delivery of messages. Distributed simulations and networked games must synchronize and serialize events from multiple computers. Chen et al. [6] use microphones from multiple wirelessly connected PDA's to perform sound localization via time difference of arrival. Synchronous gestures require synchronization and serialization for correct implementation, but go beyond these mechanisms to support new interaction techniques.

ConnecTables [25] are wheeled tables with mounted LCD displays that can be rolled together so that the top edges of two LCD's meet. The devices then recognize one another through an RFID tag and RFID reader mounted on each device. The Triangles system [12] employs special connectors along edges of triangular elements to achieve communication between discrete objects. Connecting Triangles allows the user to construct a storyline or express a series of operations.

The ConnecTables technique allows a pair of displays to be shared along one edge only. Triangles can be connected along any edge, but require multiple electrical elements along each edge. Dynamic display tiling allows multiple tablets to be connected along any edge using a single inexpensive accelerometer and touch sensors on each tablet. These sensors are not dedicated to this purpose, and have other compelling single-user applications, such as using tilt to scroll documents or automatically change the screen orientation (e.g. [14][16]). Note also that neither ConnecTables nor Triangles explore synchronous gesture detection. On ConnecTables, for example, each table's RFID reader alone can determine all the information necessary to know which display to communicate with.

Grudin reports typical use and design issues for multi-monitor display systems [13]. Interaction techniques that tile multiple tablets together might benefit from some of these design insights.

Several techniques have been proposed to share information between devices. Pick and Drop [23] uses a stylus with a unique ID to perform copy and paste operations between computers. The HyperPalette [2] uses a tilting motion to drop data from a PDA onto a work surface.

A number of technologies and techniques for discovering and using nearby devices have been proposed. Proximal selection [24] allows the user to choose from nearby devices to perform an action (e.g. print on a nearby printer). Wireless connection standards such as Bluetooth include discovery mechanisms. The RADAR technique senses location via triangulation of 802.11 wireless network RF signal strengths [3]. Synchronous gestures such as bumping tablets together or shaking devices (Smart-Its Friends [18]) represent an additional indication of user intent, beyond mere proximity, to form a privileged connection. By the same token, proximity sensing or other mechanisms to limit the scope of potential pair wise synchronies that must be computed may form an important building block for practical implementation of synchronous gestures.

SYSTEM COMPONENTS AND SENSING HARDWARE

The system is implemented on Toshiba Portege Tablet PC's with built-in 802.11 wireless networking. The techniques described in this paper use the tablets exclusively in the slate mode. The sensing hardware uses a custom sensor board (evolved from [16]) with an Analog Devices ADXL202 two-axis linear accelerometer [1] to sense left-right and forward-back tilting of the tablet relative to gravity. This sensor registers sharp spikes when two tablets are bumped together. To reliably sense the direction and magnitude of such spikes, the system maintains a sampling rate of 120Hz, and sets the accelerometer's bandwidth to 10Hz. The system also uses a pair of touch sensors [4][17] to determine when a user is holding the tablet. These are simply strips of conductive material on the left and right sides of the tablet.

BUMPING AS A SYNCHRONOUS GESTURE

To understand the possibilities for and limitations of interaction techniques based on bumping, it is helpful to see what types of signals are generated, as well as to consider undesired signals that may potentially look like bumping.

Fig. 2 shows example data from a user holding one tablet ("local device") and bumping it into another tablet lying on a desk ("remote device"). Each tablet experiences a roughly equal but opposite pattern of forces. Note that striking the left side of a tablet versus striking the right side results in a similar sensor pattern, but with spikes in the opposite direction. This allows the system software to determine which side of each tablet has made contact.

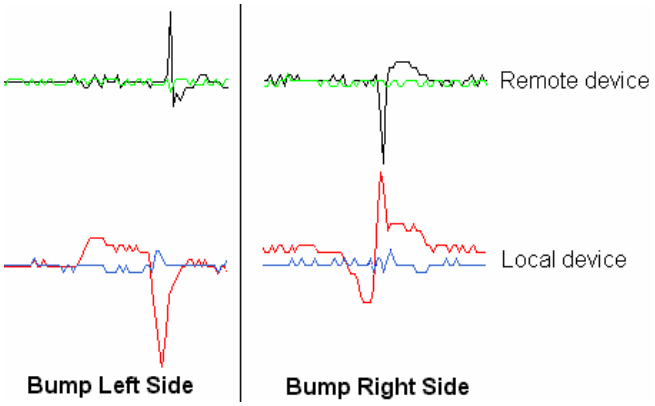

Fig. 2 Accelerometer signatures for bumping two tablets together, with forward-back and left-right accelerometer axes for the local and remote devices. **Left**: Bumping the left side of a device flat on the table with a held device. The left-right accelerometer axes exhibit characteristic spikes resulting from equal and opposite contact forces. Note the forward-back axes exhibit no significant response. **Right:** Bumping the right side of the device flat on the table. The spikes are in the opposite direction of a left-side bump.

Because the tablets are rectangular, bumping an edge of the tablet will primarily excite only one axis of the accelerometers. The second orthogonal forward-back sensing axis is also plotted in Fig. 2, but it has almost no response to the left-right impacts shown in the figure. Hence, the sensing axis with the largest response to the bump distinguishes top and bottom bumps from left and right side bumps.

With the current prototypes, it is awkward to bump the bottom edge of one tablet into the top edge of another, because of the protruding sensor board (on top) and USB connector (on bottom). However, bumping along this edge is still possible by bringing together those portions of the top and bottom edges without any protrusions. In future refinements we plan to eliminate these protrusions.

As a practical matter for gesture recognition, the signatures may not always be as clean as those shown in the plots of Fig. 2. For example, if the tablets are angled slightly as they bump together, there may be one impact from one corner of a tablet making contact, followed very quickly by a second impact from the other corner striking the same edge of the tablet. Or, if a user is a bit over-zealous and bumps the tablets together forcefully, there can be a significant bounce back in the opposite direction as well.

Another potential set of problems results from unintentional signals that may resemble intentional bumping together of tablets (Fig. 3). For example, spikes, peaks, or humps in the accelerometer data can occur if the user bumps into a desk surface that both tablets are resting on, or if two users move their tablets at the same time. Requiring spikes that are tightly synchronized in time filters out many such signals. Another simple criterion the software uses to reject undesired signals is to ignore any spikes that occur when neither of the pair of tablets is being held (as detected by the touch sensors). Further details of the recognition algorithm appear later in this paper.

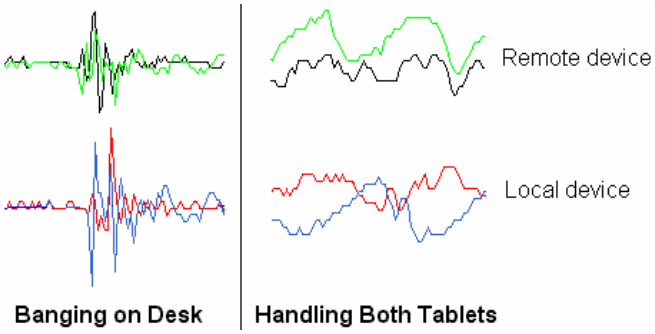

Fig. 3 Example accelerometer patterns for some non-bumping signals. **Left**: Banging on a desk causes multiple spikes exciting both axes of both accelerometers. **Right**: Handling both tablets at the same time produces waves or peaks on both sensing axes.

DYNAMIC DISPLAY TILING VIA BUMPING

Interaction with mobile devices is often constrained by the available screen real estate. Dynamic display tiling is a new interaction technique that enables multiple users (or one user with multiple devices) to combine independent devices to create a temporary larger display. This can be done horizontally (Fig. 4) or vertically (Fig. 5). Users can tile tablets together by bumping one tablet against another one resting on a desk or in a user's lap. Dynamic display tiling offers a promising application area for the synchronous gesture of bumping two devices together. Users can easily break the connection by removing one of the tablets.

For dynamic display tiling, one tablet (the *base tablet*) rests flat on a desk surface, and a second tablet (the *connecting tablet*) is held by a user and bumped into the base tablet along one of the four edges of its screen bezel. Note that this creates a hierarchy in the connection. The interaction metaphor is that the connecting tablet temporarily annexes the screen real estate of the base tablet; if either tablet is removed, the base tablet reverts to its previous state. The system distinguishes the base tablet from the connecting tablet using the touch sensors, since there is no need to hold

the tablet resting flat on the desk. If both tablets are being held, the system instead uses bumping to support sharing information between the tablets (described later).

Fig. 4 Time sequence for tiling two displays. **Top Row**: One display contains a panoramic image that does not fit the screen well, while the other is displaying a document. **Middle**: When the user bumps the displays together, the connecting tablet temporarily appropriates the screen real estate of the base tablet. **Bottom**: When the user removes one of the tablets, the connection is broken and the base tablet reverts to its previous state.

The combined information from the accelerometers allows the system to sense the *command* (tile the displays) as well as its parameters: the *edge* to tile along (left, right, top, bottom) and, in combination with the touch sensors, the *direction* (which computer "sends" and which "receives" information, using the connecting vs. base tablet distinction). Bumping naturally phrases together all of these parameters in a simple physical act that seems like a single cognitive chunk from the user's perspective [5].

When the system recognizes the docking of the tablets, the tablet which recognizes the synchronous gesture makes a short metallic clicking sound suggestive of a connection snapping together. It also displays an arrow pointing to the edge of the screen that the other tablet will dock to (Fig. 6, left). It then sends a message to its remote partner telling it to dock along the corresponding edge of its display. When the remote partner receives this message, it makes a different popping sound to indicate the completion of the connection. It also displays a smaller arrow pointing at the edge that the other display will dock to; both arrows automatically disappear after about two seconds. The size disparity of the arrows is meant to convey the hierarchy of the connection (connecting tablet vs. base tablet), although in informal demonstrations thus far this size difference seems to be a subtle cue that users can easily miss. We are experimenting with an animated arrow that appears to slide across the boundary between the two displays as a potentially more salient alternative.

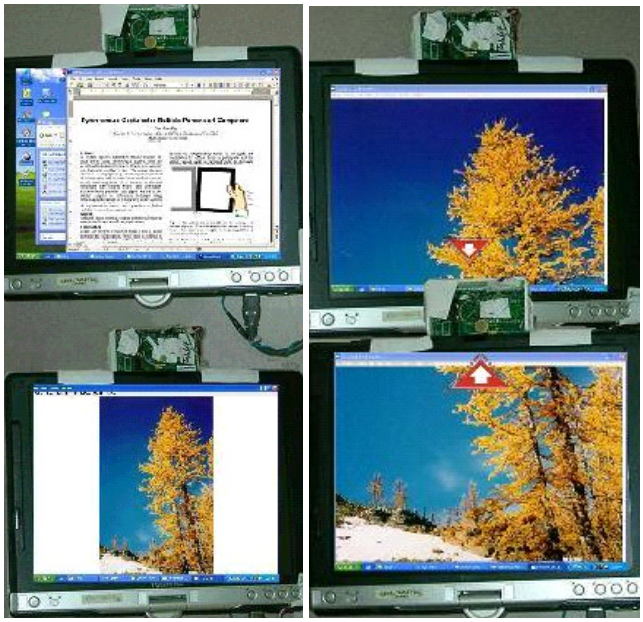

Fig. 5 Docking two displays vertically (before on left, after on right). The bottom tablet is the connecting tablet.

Note that it is important for the remote device to provide its own feedback to let the user(s) know when both devices have agreed to the connection. The feedback often may appear to be simultaneous and redundant, but when there is a brief wireless networking drop-out the confirming feedback from the other device may appear a second or two later. If it does not appear then the user(s) know that the connection may have been lost. It also seems important to provide audio feedback in addition to visual feedback. Because the technique can involve more than one user, one user's attention may not be on the tablets when docking occurs. Also, if a user is handling a tablet and a false positive recognition of docking were to occur, it seems important for the user to know about this so that the connection can be broken if desired.

As stated previously the user can break the connection by removing one of the tablets. In practice this means each tablet looks for the connection to be broken by monitoring its local tilt and touch sensor data. If significant movement

(sensed by comparing a time-decaying measure of changes to the tilt values to a simple threshold) occurs while the user is holding the tablet, the local system infers that the tablet has been moved away. If, however, the user releases the tablet before motion stops, this indicates that the user may simply be letting go of the tablet after docking, so the docking state is maintained in this case. Furthermore a short time-out is used after docking to ensure that the movement required to dock the tablets does not trigger an immediate (false positive) recognition of undocking.

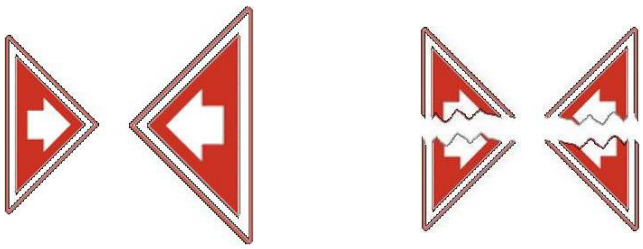

Fig. 6 Feedback for docking and undocking displays. **Left**: An arrow at the docking edge of each tablet shows that a connection has been recognized. The connecting tablet has a larger arrow than the base tablet to show the hierarchy in the connection. **Right**: When a tablet is moved away, a "shattered" arrow is shown on each tablet to give the user feedback that the association has been broken.

Once a tablet recognizes undocking, it sends a message telling its remote partner to also undock. The local tablet immediately plays a short and distinct breaking sound and shows a broken arrow on the screen (Fig. 6, right). The remote tablet provides the same feedback when it receives the message telling it to undock.

Applications and Metaphors for Connecting Displays

The author has implemented a photo viewer application that displays a small version of a photo on a single display. When docked to a second tablet, if the aspect ratio of the image is such that it can take advantage of the second display, the image expands to encompass the other display (covering whatever was previously on the screen there). If the image cannot make use of the additional screen real estate (e.g., the user docks another display horizontally to the left, but the photo viewer is currently displaying a vertical-format picture) the same image is shown on the other display for the other user to see more clearly. As another example, if a Microsoft Word document is displayed on the connecting tablet, upon docking the network path name to the document is sent to the base tablet. The base tablet displays the document and scrolls it to show the next page. Finally, another fun demo shows a ball that bounces off the edges of the screen; if two tablets are docked, the ball passes under the screen bezel and eventually appears on the screen of the other tablet.

All of these examples create one large display space, with the connecting tablet acting as the "boss" that tells the base tablet what to display. Other metaphors for creating a shared display are possible, and selecting the best one may depend on the particular application or type of devices involved.

Face-to-face Collaboration

In the ConnecTables system, the displays can only be connected along the top edge. Since each display faces its user, the displays must start out rotated 180° from one another when they are connected. When joined, the ConnecTables create a shared workspace in which users can pass objects back and forth.

In ongoing work we are exploring adding support for face-to-face collaboration metaphors [15] such as that suggested by ConnecTables, but with a pair of mobile Tablet PC's (the ConnecTables displays are fixed in place and cannot easily be picked up or moved around). When users bump the tops of two tablets together, instead of dynamic tiling, a shared whiteboard application for face-to-face collaboration is brought up on both tablets. With a pair of moveable tablets, it is possible to support fluid transitions between public, shared work when the tablets both rest on a desk, versus personal work when one or both users pick up their tablets to work with them separately. In this case, moving one tablet away does not break the connection; rather, it is maintained until a user explicitly disconnects, or walks out of the room.

Another possible metaphor for collaboration is to share a document on the screen with another user when two tablets are bumped together. In the case where the "the same image is shown on the other display," the photo viewer application used this metaphor. However, it does not explore this metaphor in depth, nor does it implement collaboration techniques to really take advantage of it.

MUTUAL AND ONE-WAY SHARING OF INFORMATION

Recall that dynamic display tiling uses the touch sensors to detect when one display is not being held, but is just resting on a flat surface. But what if both tablets *are* being held? The system uses this distinction to support a couple of techniques for sharing information between devices.

Two held tablets can be bumped together to indicate a mutual desire to share information. Here, the metaphor is that of clinking glasses together for a toast. The technique supports a digital version of exchanging business cards. When both users bump their tablets together, each device passes the user's home page to the other device and brings up that page in a web browser (Fig. 7). In informal demonstrations of this technique thus far, I have observed that users tend to just bump the corners of the two tablets together. When holding a tablet in midair users typically support it with both hands; thus for users standing side-by-side, bumping an entire edge together may not be practical because one or both users' hands may be in the way. It is hard to see how a system that relies on physical electro-mechanical links (such as Triangles [12]) could support virtual connection of devices with this kind of casual "clinking" gesture.

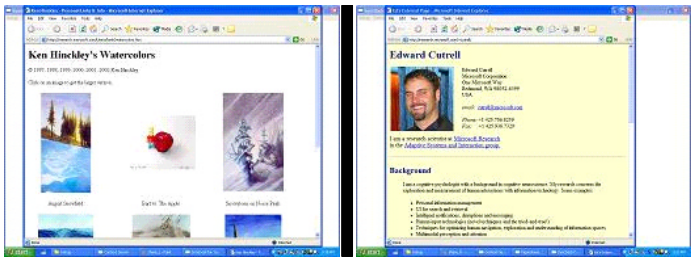

Fig. 7 Mutual sharing of information. If two users holding their tablets bump them together, the tablets perform a mutual exchange of information. Each device sends its owner's home page address to the other tablet. The image shows the screens of the two tablets after bringing up the web page for the other user.

Upon recognizing the synchronous gesture, the system makes a brief sound suggestive of teleporting information to the other device. The system currently uses visual feedback similar to that for docking two tablets together (Fig. 6), but instead draws both arrows at the same size. However, in this case there is no hierarchy to the connection, and the edge used to bump the tablets together does not matter, so the system probably should show more clearly distinct feedback in this case.

A variation of this technique allows one-way sharing of information suitable for copy/paste operations. A user can "pour" information from his tablet into another user's tablet by tilting his tablet up when bumping with the other user's tablet. If the tablet is tilted more than 15 degrees relative to the other tablet when the users bump tablets, the software sends the clipboard information to the other device. The tablet receiving the clipboard data makes the sound of water dripping in a pool to suggest the pouring metaphor. The system does not yet implement sending the actual clipboard contents to the other device; currently a web page showing hypothetical clipboard contents is brought up on the receiving device to demonstrate the technique.

SOFTWARE IMPLEMENTATION ISSUES

The current system implementation is limited to a pair of devices. Synchronous gesture recognition only needs to occur on one of the two devices. When a synchronous bump is detected, that device informs its remote partner of the gesture and its various parameters. Readers not interested in the mechanics of how synchronous gestures are implemented and recognized can safely skip ahead to the DISCUSSION section.

Time Synchronization

Synchronous gestures must determine which samples from a local and remote device are truly "synchronous" in the presence of network delays. Our implementation uses a simple message passing scheme to synchronize time to within about 50 ms, but much better algorithms can be implemented for wireless devices. The Reference Broadcast Synchronization (RBS) algorithm enables synchronization to within less than a millisecond [11]. Implementing RBS may allow our system to more easily reject undesired accelerometer signals that might randomly arrive from separate machines close together in time, but which do not actually occur at the same instant, as with intentional bumping.

Alignment of Time Frames during Gesture Recognition

The remote device buffers its own accelerometer samples and, every 100ms, sends whatever new samples have accumulated to the "server" tablet which performs synchronous gesture recognition. The server tablet buffers several seconds worth of time stamped accelerometer data from the remote device and compares it to time stamped accelerometer data from the local machine. Since reception of remote data may be subject to random network delays and momentary drop-outs in wireless connectivity, gesture recognition occurs by discrete event simulation rather than in real time. That is, the software delays handling of local samples until samples with the same synchronized time stamp arrive from the remote machine. It then uses the time stamps to handle individual samples in order, until there are no more samples of the same age or older than the most recently received remote sample.

From the user's perspective, however, unless there is an abnormally long network drop-out, recognition appears to occur in real time. The system will only miss a synchronous gesture if the drop-out lasts longer than the available buffer space to store local and remote samples.

Synchronous Bumping Recognition Algorithm

The system currently employs a deterministic algorithm to recognize synchronous spikes in the accelerometer data from a pair of devices. Given natural variations in user performance, the algorithm requires some flexibility to reliably sense when the user intentionally brings two tablets together, but the algorithm cannot be too flexible or else other naturally occurring patterns (shaking, handling, or putting down devices) may be interpreted as "bumping."

The algorithm operates by keeping track, for each sensing axis on each local-remote pair of devices, of whether or not the signal is currently above a running average by a threshold (10 degrees) or below the running average by the same threshold. Every time the signal passes through these thresholds, it generates a transition. These transitions are then used to gradually fill out the following data structure as local and remote samples are handled in sequence.

```
typedef struct {
   bool rising_edge;   // edge passed thresh?
   double rise_peak_t; // time of peak magnitude
   float peak_mag;     // peak magnitude
   bool found_one_spike; // seen a spike?
   double spike_found_t; // time spike found
   float sync_bump_mag;  // magnitude of sync bump
} BumpInfo;

BumpInfo transitions[2][2][2];
```

The `transitions` array of `BumpInfo` structures creates one instance of the data structure for each of 8 cases: spikes

on the local or remote device, on the left-tight or forward-back axis, and above or below the running average.

Once a signal transitions past the running average threshold, the `rising_edge` flag is set true and the `peak_mag` (peak magnitude of a spike after crossing the threshold) and `rise_peak_t` (time stamp of the peak magnitude) are initialized using the current sample. If subsequent samples rise further beyond the threshold then these members are both updated.

If a peak above the running average is wider than the 62.5ms timeout TOUT_TOO_LONG, it is ignored. This restricts the algorithm to look for narrow, sharp spikes typical of impact forces. Otherwise, as soon as the peak drops below the running average, the `found_one_spike` flag is set true and `spike_found_t` is set to the time that this spike was recognized. At this point, the code checks the `BumpInfo` structure to see if a spike on the other tablet has been recorded. If so it calculates the synchronicity of the spikes by subtracting the respective `spike_found_t` members. If the synchronization of the spikes falls within a 50ms window `TOUT_MAX_SYNC`, the spike is recorded as a *candidate* for a true "synchronous bump" with `sync_bump_mag` set to the `peak_mag` of this spike.

Once a candidate for a synchronous bump is identified, the algorithm continues to look at additional samples for the following 200ms timeout `TOUT_CHOOSE_BEST_SYNC`. If a candidate with a larger peak magnitude is found, it replaces the previous candidate. This behavior allows the algorithm to seek out the most significant spike, rather than taking the first one which exceeds the threshold. If it did not wait to seek further candidates, the algorithm would face a race condition where the first sample over threshold always "wins" even if other spikes are more representative of the actual gesture made by the user. This is a real concern because there may be some excitation of the orthogonal axis of the accelerometer. If this happens to exceed the running average threshold, it may lead the software to incorrectly determine which edge of the tablet the user bumped into; waiting for the best candidate in a small time window allows most such errors to be avoided.

The algorithm introduces a couple of other criteria to weed out false positives for synchronous bumps that may be the result of handling two tablets at the same time. If both sensing axes of the accelerometer observe large peaks, this suggests the contact force is not primarily along any one axis, so it is ignored. If a large number of candidates are observed, this indicates a repeated vibratory signal, and is also ignored. With an actual "synchronous bump" typically only 3-4 viable peaks will be observed. Bumps are also ignored if neither device is being held (touched) by a user. Finally, to avoid responding to accidental double-strikes, once a synchronous bump is recognized, all further bumps are ignored for `TOUT_IGNORE_AFTER_SYNC` = 750ms.

A final detail is choosing when to re-initialize the data structures. We currently completely clear out the array of `BumpInfo` structures and the list of synchronous bump candidates whenever a synchronous bump is recognized as the best candidate within the `TOUT_CHOOSE_BEST_SYNC` time window, or if no synchronous bump is detected, whenever `TOUT_INITIALIZE` = 1000ms have passed since the last observed transition.

Once a synchronous bump has been identified, the edges of the tablets involved in the collision can be identified using the table of Fig. 8. Here, the Device column refers to either the connecting device held by the user, or the base device resting on a desk (struck by the held device). Direction refers to the direction of the spike with the highest magnitude above or below the running average threshold.

Device	Axis	Direction	Result (edge to dock)
Connecting	Left-Right	Above	Right edge
Base	Left-Right	Below	
Connecting	Left-Right	Below	Left edge
Base	Left-Right	Above	
Connecting	Fwd-Back	Below	Bottom edge
Base	Fwd-Back	Above	
Connecting	Fwd-Back	Above	Top edge
Base	Fwd-Back	Below	
Connecting	Fwd-Back	Above	Face-to-face collaboration [15]
Base	Fwd-Back	Above	

Fig. 8 Table giving mapping of observed synchronous spikes, above or below the running average threshold, for each device and accelerometer sensing axis.

Limitations

The proposed recognition algorithm and system hardware is certainly not perfect. When intentionally bumping tablets together, the system occasionally recognizes the correct axis of the bump, but gets the direction wrong. This can occur if significant positive and negative spikes both result from the impact. Since the algorithm does not respond to the first spike, but only to the spike with the greatest magnitude within a small time window, if the "bounce back" from the impact has a greater magnitude than the initial impact itself, this problem will occur. Although it occurs very infrequently, it may be necessary to increase the sampling rate of the hardware to completely eliminate this problem.

Since the algorithm is based on simple crossings of a running-average threshold, a more significant limitation is that it is susceptible to recognition of "spikes" in signals that are not truly representative of bumping a pair of tablets together. One simple way to eliminate many false-positives is to keep timeout windows such as `TOUT_TOO_LONG` and `TOUT_MAX_SYNC` as narrow as possible, yet still wide enough to admit variations in user performance, and to accommodate errors in time synchronization. We expect

that implementation of RBS [11], as well as further refinements to filter out undesired signals, will be necessary for a more robust implementation.

DISCUSSION

Informal Demonstrations

We have not conducted careful usability studies, but in informal demonstrations to date with research colleagues, people readily grasp the idea and find it easy to bump two objects together in a manner that can be sensed by the system. People seem to like the idea of gaining more screen real estate by combining multiple tablets. People also seem to like the visual feedback and the brief, distinctive sounds. Only more careful user testing can help reveal what kind of metaphors users would expect when sharing displays, or if usability problems might arise from the technique of one device annexing the display of another.

A common concern people raise is that bumping might damage the hard disk on the tablet. But hard disks on mobile devices are designed to survive day to day handling of the device, and a fairly gentle tap is enough for our sensors to pick up, so great force is not necessary. The system currently uses a 10-degree deviation in tilt as the minimum threshold for a "bump", which translates to an acceleration of 0.17 units of gravity (g). A bumping threshold as small as 2 degrees (.035g) can be sensed, which is an almost imperceptibly light tap from the user's perspective. However, such a low threshold currently causes the software to recognize too many "false" synchronous bumping signals, so we default to a higher, more deliberate force.

We have found that relying solely on the touch sensor to determine the direction (the base device vs. connecting device distinction) usually works well, but it sometimes leads to problems. For example, if one user rests her device on the desk, but leaves her hand in contact with it, and another user then connects to it, this is interpreted as sharing information (since both devices are being touched) rather than dynamic display tiling. By looking more carefully at the accelerometer data before and after a synchronous bump, it should be possible to determine which device was moving and which was stationary, which seems to more closely match user expectations.

Extension to More Than Two Devices

At this time the software only implements synchronous gesture detection between a pair of devices, but we are working to support additional devices. For example, it should be possible to bring four tablets together to form a 2x2 tiled display. One implementation hurdle is to correctly handle sequences of connections, which we call the "transitive bumping" problem. For example, if two devices are tiled side by side, and then a third display is added to make a 1x3 tiled display, the force of bumping the third tablet may be transmitted through the middle display and picked up by the display at the other end (which was not physically bumped). However, it should be easy to determine which device was actually struck by keeping track of the topology of existing connections.

Scope of Synchronous Partners

With a large number of devices, looking for synchrony could overload limited wireless network or mobile device CPU resources, and also increase the possibility of false positive recognition of "synchrony" through pure chance. If a set of n devices attempts to synchronize with every other device, then there are n^2 one-way connections to consider. In practice, each pair of devices only needs to compute synchrony once, so there are $n(n-1)/2$ possible synchronies. Computing synchrony seems feasible for on the order of 10 devices, but may be computationally prohibitive or subject to increasing rates of false positive recognition if hundreds or thousands of devices must be considered.

The current synchronous gestures implementation does not address this problem, but clearly some way to restrict the scope of potential partners for synchrony would be required for a large-scale implementation of synchronous gestures. Forcing users to explicitly select a set of connected devices to form a "synchronous gesture group" could work, but places responsibility for this task on the users.

Automatic techniques for restricting the scope of potential synchronies might be as simple as limiting communication to the set of wireless devices visible to the 802.11 wireless network access point with the highest signal strength. This might be further refined through location triangulation using 802.11 signal strengths, thus limiting searches for synchronous gestures to other proximal devices. A possible difficulty comes from uncertainty in this data, e.g., two co-located devices might fail to search for synchrony due to error in location estimates.

Synchronous Bumping vs. Location/Proximity Sensing

As discussed above, synchronous gestures seem to depend on some form of scoping or proximity detection as an enabling technology to restrict the number of devices which must be considered. If proximity or location sensing exists in support of synchronous gestures, then why bother with computing synchronicity at all? It is possible to connect two devices if they are merely proximal to one another. However, if a multitude of devices exists in the same proximity, how does a user specify which devices to connect to, and how does the system know the spatial relationship between them?

Merely having two devices near one another does not necessarily indicate that the users have any desire or intention to share information. What bumping via synchronous gestures offers on top of proximity detection is an *explicit step of intentionality that the user has control over*. Bumping removes the need for the user to select a device from a list of numerous devices (as required by proximal selection [24]); with the synchronous gesture of bumping two tablets together, the selection is done in the physical world by manipulating the actual objects of concern. It also provides additional information: with

proximity, it is not clear which edge of a tablet should be docked, nor is there any hierarchy in the connection. Finally, the technology enables support for actions other than display tiling, such as sharing or pasting information between tablets, or establishing face-to-face collaboration [15]. It is hard to see how proximity alone could support all of these behaviors without burdening the user with additional decisions and selection actions.

Software Infrastructure Issues

Mechanisms implemented in the Context Toolkit [9] or the Event Heap [21] to support distributed sensors might simplify implementation of synchronous gestures. Our system infrastructure is based on passing a message in real time whenever a sensor changes values, but synchronization between local and remote devices implicitly means that the software must deal with the *history* of recent sensor values, rather than just the most recent value. In this regard, the mechanisms proposed by DEMIS [20] for saving and accessing a history of sensor values using time intervals seem like they would be useful. Platforms for distributed multimedia sensing and control such as Aura [8] may also contain useful lessons and abstractions for synchronous gestures.

In our architecture, a Context Server resides on each machine and handles all of the local sensor data [16], as well as all TCP/IP network communication with remote devices. Client applications can send messages via the Context Server to a remote partner established through a synchronous gesture. This makes it easy for client applications to exchange information and insulates them from all details of the network connection. This is useful not only to reduce complexity in the client applications, but also allows the Context Server to hide the identity of the remote device, while still allowing messages to be passed.

Additional Possibilities for Synchronous Gestures

A number of future possibilities for synchronous gestures, beyond bumping, exist:

Synchronous gestures based on human-human communication and body language. In human-human interaction, there are many examples of familiar, naturally occurring synchronous or sequenced behaviors in body language and interpersonal communication. These naturally occurring gestures could be sensed and used as implicit contextual cues to augment human-human or human-computer interaction. For example, accelerometer-augmented watches could sense when two users shake hands, and use that information to exchange digital business cards. Other examples include two persons bowing to one another, people turning to face one another, or sensing when one user's body language imitates that of another. It may be possible to derive measures of group activity at meetings by sensing the activity of users carrying sensor-augmented devices. Interpersonal behaviors might also suggest points of departure for analogous but artificial synchronous gestures to facilitate human-computer interaction. The Sociometer [7] is one example of work that senses these kinds of cues, although at present the Sociometer system does not use them to support real-time interaction techniques.

Synchronous Gestures from Commonly Available Devices. This paper primarily considers synchrony between mobile devices augmented with custom sensors, but there may be opportunities to exploit synchronicity using ordinary input devices. For example, two users could press and release buttons at the same time to connect their devices, or perhaps mice from two different computers could be brought together until they collide, with the system software looking for simultaneous cessation of movement on both mice. We are also investigating the use of pen input to enable gestures that span multiple Tablet PC's. Such actions might be useful to simulate our synchronous bumping gesture on systems without any special sensors, to implement techniques such as Pick and Drop without requiring a unique ID on the stylus [23], or to support completely new techniques.

CONCLUSIONS AND FUTURE WORK

This research proposes synchronous gestures as a new interaction metaphor for distributed sensing systems. There have been few interaction techniques exploiting synchrony between multiple users and multiple devices, so identifying this area and demonstrating practical applications contributes to the literature and may spur development of further techniques. Synchronous gestures implemented in the current system include dynamic display tiling, sharing information by bumping together mutually held tablets, or pasting information from one tablet to another by angling one tablet down while making contact. The present system demonstrates an implementation of synchronous gestures and suggests avenues for further development.

Although the techniques explored in this paper employ real-time recognition strategies, if sufficient memory exists to store all samples, synchrony could even be determined in the absence of constant wireless connectivity by detecting synchrony at a later time when both devices become connected to the network again. When bumping two tablets together to share information, for example, delayed recognition of synchrony might be useful to users who want to protect their privacy and explicitly give the OK to pass the information at a later time.

Although the present system explores synchronous bumping only for tablet computers, it should also be applicable to bumping between mulitiple dissimilar devices such as PDA's, cell phones, watches, or digital cameras. The user's expectations and possible semantics of bumping with dissimilar devices may be very different from bumping homogeneous devices together. Pierce et al. are currently exploring *opportunistic annexing* [22], which will enable users to dynamically bind together multiple input/output resources in a ubiquitous computing environment. Bumping objects together may offer a compelling and simple way for a mobile user to combine,

for example, a PDA, a keyboard, and a large display to quickly enter text and view documents while away from his or her desk.

Disparity in mass may make bumping between dissimilar objects harder to detect, but simple tests of bumping a PDA into our sensor-augmented tablet computer, for example, suggest that the accelerometer on the tablet can detect this signal even though the PDA is much less massive. Hence there seem to exist many promising but as yet unexplored possibilities for opportunistic annexing using bumping or other synchronous gestures.

ACKNOWLEDGEMENTS

Thanks to Andy Wilson, Patrick Baudisch, John Krumm, and Dimitris Achlioptas for brainstorming ideas and discussing implementation issues. Thanks to Dave Thiel for video production.

REFERENCES

1. Analog Devices Inc., *Low-Cost +/-2g Dual-Axis Accelerometer with Duty Cycle Output*, http://www.analog.com/UploadedFiles/Datasheets/567227477ADXL202E_a.pdf, 2002.
2. Ayatsuka, Y., Matsushita, N., Rekimoto, J., *HyperPalette: a Hybrid Computing Environment for Small Computing Devices*, CHI 2000 Extended Abstracts, 2000, 133-134.
3. Bahl, P., Padmanabhan, V., *RADAR: An In-Building RF-Based User Location and Tracking System*, IEEE 19th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2000), 775-784.
4. Baxter, L.K., *Capacitive Sensors: Design and Applications*. IEEE Press Series on Electronics Technology, ed. R.J. Herrick. 1997, New York: The Institute of Electrical and Electronics Engineers.
5. Buxton, W., *Chunking and Phrasing and the Design of Human-Computer Dialogues*, Information Processing `86, Proc. of the IFIP 10th World Computer Congress, 1986: Amsterdam: North Holland Publishers, 475-480.
6. Chen, J.C., Yip, L., Wang, H., Maniezzo, D., Hudson, R.E., Elson, J., Yao, K., Estrin, D., *DSP Implementation of a Distributed Acoustical Beamformer on a Wireless Sensor Platform*, Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2003), 2003, Hong Kong, China.
7. Choudhury, T., Pentland, A., *The Sociometer: A Wearable Device for Understanding Human Networks*, Conference on Computer Supported Cooperative Work (Workshop: Ad hoc Communications and Collaboratin in Ubiquitous Computing Environments), 2002.
8. Dannenberg, R., *Aura as a Platform for Distributed Sensing and Control*, Symposium on Sensing and Input for Media-Centric Systems (SIMS 02), 2002, University of Santa Barbara: Center for Research in Electronic Art Technology.
9. Dey, A., Abowd, G., Salber, D., *A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications.* Journal of Human-Computer Interaction, 2001. **16**(2-4): p. 97-166.
10. Dourish, P., Bly, S., *Portholes: Supporting Awareness in a Distributed Work Group*, Proc. CHI'92, 1992, 541-547.
11. Elson, J., Girod, L., Estrin, D., *Fine-Grained Network Time Synchronization using Reference Broadcasts*, Proc. Fifth Symposium on Operating Systems Design and Implementation (OSDI 2002),, 2002, Boston, MA.
12. Gorbet, M., Orth, M., Ishii, H., *Triangles: Tangible Interface for Manipulation and Exploration of Digital Information Topography*, Proc. CHI'98 Conference on Human Factors in Computing Systems, 1998, 49-56.
13. Grudin, J., *Partitioning Digital Worlds: Focal and Peripheral Awareness in Multiple Monitor Use*, CHI 2001, 2001, 458-465.
14. Harrison, B., Fishkin, K., Gujar, A., Mochon, C., Want, R., *Squeeze Me, Hold Me, Tilt Me! An Exploration of Manipulative User Interfaces*, Proc. ACM CHI'98 Conf. on Human Factors in Computing Systems, 1998, 17-24.
15. Hinckley, K., *Distributed and Local Sensing Techniques for Face-to-Face Collaboration*, ICMI-PUI'03 Fifth International Conference on Multimodal Interfaces, 2003, Vancouver B.C., Canada.
16. Hinckley, K., Pierce, J., Sinclair, M., Horvitz, E., *Sensing Techniques for Mobile Interaction*, ACM UIST 2000 Symp. on User Interface Software & Technology, 2000, 91-100.
17. Hinckley, K., Sinclair, M., *Touch-Sensing Input Devices*, ACM CHI'99 Conf. on Human Factors in Computing Systems, 1999, 223-230.
18. Holmquist, L., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M., Gellersen, H., *Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts*, Ubicomp 2001, 2001: Springer-Verlag, 116-122.
19. Horvitz, E., Kadie, C., Paek, T., Hovel, D., *Models of Attention in Computing and Communications: From Principles to Applications.* Comm. of the ACM, 2003. **46**(3).
20. Jiang, H., Kessler, D., Nonnemaker, J., *DEMIS: A Dynamic Event Model for Interactive Systems*, Proc. ACM Symposium on Virtual Reality Software and Technology (VRST 2002).
21. Johanson, B., Fox, A., *The Event Heap: A Coordination Infrastructure for Interactive Workspaces*, Proc. of the 4th IEEE Workshop on Mobile Computer Systems and Applications (WMCSA-2002), 2002, Callicoon, New York.
22. Pierce, J., Mahaney, H., Abowd, G., *Opportunistic Annexing for Handheld Devices: Opportunities and Challenges*, Tech Report #GIT-GVU-03-31, 2003. http://www.cc.gatech.edu/gvu/research/techreports.html.
23. Rekimoto, J., *Pick-and-Drop: A Direct Manipulation Technique for Multiple Computer Environments*, UIST'97 Symp. on User Interface Software & Technology, 31-39.
24. Schilit, B.N., Adams, N.I., Want, R., *Context-Aware Computing Applications*, Proc. IEEE Workshop on Mobile Computing Systems and Applications, 1994, Santa Cruz, CA: IEEE Computer Society, 85-90.
25. Tandler, P., Prante, T., Müller-Tomfelde, C., Streitz, N., Steinmetz, R., *Connectables: dynamic coupling of displays for the flexible creation of shared workspaces*, UIST 2001, 11-20.