Enhancing Multi-User Collaboration with Powerwalls

Vít Rusňák
Masaryk University
Botanická 68a
Brno, 602 00, Czech Republic
xrusnak@fi.muni.cz

Lukáš Ručka
Masaryk University
Botanická 68a
Brno, 602 00, Czech Republic
xrucka@fi.muni.cz

ACM 978-1-4503-1952-2/13/04.

Abstract
Powerwalls and large tabletops can be viewed as native group collaborative environments. Further integration of multi-touch overlay enables concurrent multi-user interaction with displayed content. In our work, we focus on extending basic multi-touch sensor with capability of distinguishing individual users and their association with touch input events. Our goal is an unobtrusive system for the user based on the multi-sensor integration of commodity hardware – multi-touch overlay sensors, MS Kinect or other devices.

Author Keywords
Enhanced multi-touch interaction; user distinguishing; group collaborative environment.

ACM Classification Keywords
H.5.2 [User Interfaces]: Input devices and strategies, Interaction styles

General Terms
Design, Human factors

Introduction
With the advent of various multi-touch overlay frames, panels or foils the direct multi-touch interaction becomes feasible in the domain of powerwalls and large

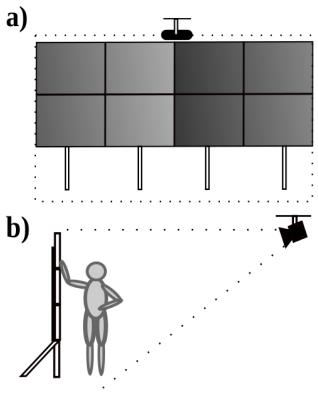

Figure 1: Powerwall scenario with a depth sensor capturing the display area; a) front view, b) side view

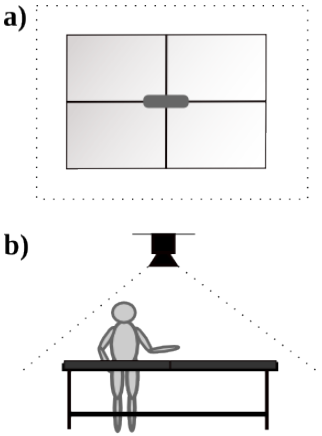

Figure 2: Tabletop scenario with a depth sensor is placed above the surface; a) top view, b) side view

tabletops [7, 15]. Although multi-user gist of such input layer enables concurrent work of multiple people, current multi-touch sensors are neither capable of distinguishing individual users nor their association with input events. Extending mere multi-touch capability with metadata for association of touch events with individual users could significantly improve collaborative work. For instance, resolving conflict situations where the activities of two users standing close to each other mutually interfere [8].

We explore general methods for coupling of sensors and multi-device orchestration in powerwall and large tabletop scenarios. The overall goal is to develop an ubiquitous interactive space providing features such as continuous user distinguishing and content personalization whilst avoiding prior user orchestration (e.g., special wearables or markers for user tracking). This paper elaborates on the novel modular framework for rapid development of such interactive user interfaces which focuses on high scalability of multi-sensor integration and provides seamless association of input events with users.

Further we give a brief overview of current approaches for user distinguishing in powerwalls. Then, the basic concepts of our framework are presented followed by a brief description of underlying algorithms. Discussion of our prototype implementation and its evaluation follows. Conclusion and outline of the future work conclude the paper.

Related Work

The topics of distinguishing users and associating them with actions they performed are studied broadly in tabletops – e.g., [4, 1, 2, 12]. In powerwall scenario the focus is laid mainly on a plain multi-user interaction [6, 15] and user distinguishing and consequent association of input events with them remain largely an unexplored topic.

Touch-less direct interaction was presented in [14]. In this installation, multiple cameras placed below the display wall canvas are used to triangulate positions of objects (i.e., fingers and hands) in a plane parallel to the display. Association of input events with users is possible only when users stand at fixed positions in front of the screen. Hutama et al. [5] presented a method for distinguishing multiple smartphone interactions on a multi-touch display wall. Each smartphone is equipped with two prongs and an integrated accelerometer. Whenever the smartphone touches the interactive surface of the display wall, accelerometer data and registered touch points are combined and the phone is identified.

Optical motion capture systems provide a reliable method for continuous distinguishing of users and other objects but require prior user orchestration. Ballendat et al. [3] use this technique for detection of users and personal devices such as cell phone or pen equipped with retro-reflective markers. The system is able to identify tracked objects based on their absolute and relative positions. The WILD Room project [9] also utilizes optical motion capture to track devices serving as remote controllers of the powerwall. Although the object tracking excels in terms of precision, the requirement of additional wearables (e.g., a cap with retro-reflective markers) may be obtrusive for the users.

LightSpace [16] allows users to interact on, above and between several surfaces within small-sized rooms. It utilizes multiple MS Kinect devices to track users without the need for special wearables but it does not scale for large setups.

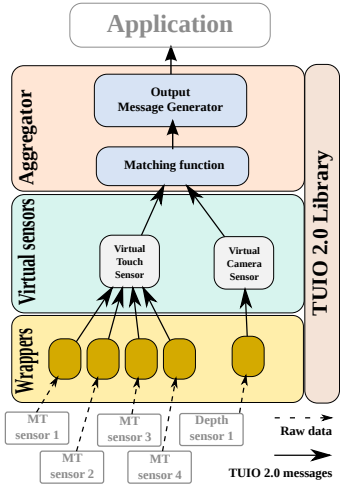

Figure 3: The framework diagram for the system consisting of 4 touch sensors and one depth sensor

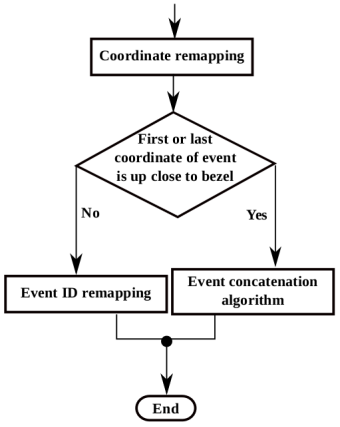

Figure 4: Virtual touch sensor algorithm

In general, individual setups are unique and differ from each other. Although there are several toolkits for developing interactive (mainly desktop) applications – such as ICon [10] – there is no tool focused on developing complex interactive interfaces or environments. These facts initiated our work on Multi-Sensor framework (MUSE) which will provide a scalable multi-sensor integration (for both same and various sensor types) together with a seamless association of input events with users.

Framework Design

The purpose of MUSE is to provide a general interaction layer to existing high-resolution visualization systems (e.g., SAGE [11]). The Figure 3 shows the setup that we use for experiments. The framework is highly modular with three-layer structure. These features enable integration of new modules to all layers easily in future.

Wrapper layer is the bottom layer which provides an interface between various physical sensor devices and unified environment of the MUSE. Each type of device has its own wrapper module; every physical device is represented as a separate wrapper. Wrappers transform heterogeneous input data into the unified messages of TUIO 2.0 which is used as the communication protocol.

Virtual sensor layer ensures assembling sensor wrappers of the same kind into a seamless *virtual sensor*. For example, *virtual touch sensor* makes several multi-touch overlay panels behave as a single device. The concept of virtual sensor is also applicatble to other devices – *virtual depth sensor* could combine multiple MS Kinect devices in order to achieve higher resolution, thus increased precision.

Aggregating Layer consists of two main modules – touch-point matching and TUIO 2.0 output message generator. Touch-point matching associates data from virtual touch sensor with 2D image blobs representing user, given from (depth) camera-based tracker. The description of an input event is then labeled with particular user. Message generator produces outgoing messages in extended TUIO 2.0 format and passes them to an application (visualization middleware, gesture recognizer, etc.).

Algorithms

The ideas of two crucial algorithms are demonstrated on virtual touch sensor and touch point matching algorithms.

Virtual touch sensor algorithm provides two operations:

a) remapping of touch events from received messages to dimensions of virtual sensor,

b) and concatenation of the touch events that are performed over two or more overlay panels as a single stroke.

The top-level structure of the algorithm is shown in Figure 4. When the algorithm receives a touch event message which starts and ends not too close to bezels (i.e., there is still some space untouched to the bezel), it modifies the coordinates of the touch event according to the position of the touch event projected in virtual sensor dimensions and changes the event identifier. Events which end or start close to the surface bezel are considered as parts of cross-sensor stroke. Thus, they are marked for further processing in concatenation algorithm. Due to the inaccuracy caused by crossing bezels, it is necessary to set up two values for time (Δt) and spatial (Δs) thresholds. Two strokes are concatenated when the latter starts in time $< \Delta t$ and its first coordinate is in Δs range from the last position of the former stroke. The concept of virtual sensor is an approach to overcome limitation of

single-device multi-touch sensors which do not scale well. They are constrained by number of concurrent touch points or maximal dimensions. Virtual touch sensor handles time synchronization and concatenation of related consequent strokes. One consequence is the possibility to use existing gesture recognition algorithms instead of developing new ones, adapted for a distributed environment of multiple multi-touch sensors.

Touch point matching algorithm processes high-precision data representing input events from virtual touch sensor with image blobs representing users from camera tracking device. The core principle is to find position of touch event within one of recognized image blobs. The premise for the algorithm is that each installation is static and its configuration (position of sensors and their parameters) is stored in a configuration file. In the first step, coordinates of touch points from virtual sensor and positions of blobs representing users are transformed to a joint coordinate system in which the algorithm operates. Due to low-resolution of depth sensors is problematic to recognize individual fingers so the algorithm represents the palm as an ellipse. Matching function is currently simplified to finding mutual position of touch point and this ellipse defined according to the standard equation:

$$\frac{(x-m)^2}{a^2} + \frac{(y-n)^2}{b^2} = f(z).$$

The match is recognized when touch point coordinates are inside the ellipse area ($f(z) < 1$). When touch event is matched with one of the blobs, its description is extended with identification of corresponding user.

Prototype Implementation

The prototype implementation of the MUSE framework consists of *a)* two wrapper modules – a general wrapper for multi-touch sensors and MS Kinect wrapper, *b)* virtual sensor module for multi-touch wrappers, *c)* simple touch point matching module and the output message generator. The physical arrangements of individual sensor devices as well as mutual position of camera-based tracking sensors and touch input layer are stored in an XML file. For benchmarking purposes we created an application for rendering touch points and blobs of users' hands.

Multi-touch wrapper supports various single- and multi-touch devices by taking advantage of Linux kernel input layer API. Wrapper captures raw events received from a physical sensor and transforms them to TUIO 2.0 messages. MS Kinect wrapper processes a depth video stream acquired from a device. Its purpose is to extract blob descriptions that represent individual users from the image frames. Blobs are continuously tracked using the tracking algorithm proposed by Senior et al. [13].

Touch point matching module receives TUIO 2.0 data streams from virtual touch sensor and from the MS Kinect wrapper. Data streams contain descriptions of touch events and blobs respectively.

Evaluation

We evaluated processing speed of individual modules, concatenation and matching accuracy in order to identify key issues for the next iteration of the development process.

Processing speed. Table 1 summarizes processing speed of individual parts of the framework. For the wrapper modules we measured the time elapsed from receiving data (touch event/image frame) from the device to the moment when TUIO 2.0 messages were dispatched. Slowness of the MS Kinect wrapper is due to complex image processing operations (multi-touch wrapper

Wrapper modules	
Multi-touch	**MS Kinect**
12.5 ± 8.6	22.6 ± 8.4

Virtual sensor module	
Localhost	**LAN**
0.063 ± 0.003	0.066 ± 0.004

Aggregation module	
Total	**Matching alg.**
0.3 ± 0.1	0.044 ± 0.02

Table 1: Processing speed of the MUSE framework modules in [ms]

Δs \ Δt	**0.3 s**	**0.5 s**	**1 s**
300 px	65	70	68
375 px	69	67	72
450 px	68	78	89

Table 2: Concatenation algorithm accuracy for various Δt and Δs [%]

transforms input raw data into TUIO 2.0 messages only). The average processing time for a single frame is approx. 23 ms – i.e., the processing is still real-time with almost 10 ms reserve for further improving of the user tracking algorithm.

Virtual sensor module processing time was measured in: *a)* Localhost scenario that represents small-scale powerwall or tabletop (sensors were plugged into a single PC), *b)* and LAN scenario representing large-scale wall-sized powerwall (sensors were plugged into different PCs connected over LAN). We performed several types of touch events that were spread across two, three and four sensors. As we can see, there is no significant slowdown in network scenario. Touch point matching was benchmarked from two perspectives – processing time of the whole MUSE framework aggregator layer and matching algorithm itself. The aggregator layer processed messages sequentially which resulted in higher computational overhead.

Concatenation accuracy. We investigated the influence of time and spatial threshold values (Δt and Δs) on concatenating function. Proper threshold values highly depend on physical parameters of sensor overlays (e.g., frame rate, width of a bezel area) and have to be obtained by experimental calibration for each installation. To determine the threshold values we conducted an informal experiment based on drawing lines across the bezels. Table 2 shows 9 combinations of threshold values and relevant accuracies.

When concatenating consecutive touch events whose trajectories and the bezel form an acute angle $< 20^\circ$ the algorithm results in false-negatives. Thus the future versions of the virtual sensor module will have to keep tracking not only each contact position, but velocity also.

Matching Accuracy. Due to using only a depth sensor of the MS Kinect we struggled with inaccurate positions of blobs. This resulted in reduced matching precision and occurrence of approx. 10 % of false negatives for touch points that were close to the edge of an ellipse (approx. 10-pixel diameter). Further, the "ellipse-matching" was too simplistic. The false positive touch point might occur under the wrist or in the middle of the user's palm where user usually does not touch. Since user usually does not touch in these areas, it could be someone's finger touching the sensor occluded by the hand of another user.

In future versions we are going to combine data from both depth sensor and VGA camera which will increase the accuracy and will reduce losses of blob details (e.g., fingers). Further improvements are expected in the next generation of depth sensors having IR sensors with higher resolution. The other issue was the occlusion of hands resulting in merging the relevant blobs into a single one. We identified the problem in the blob tracking algorithm which will be one of the focal points of our future work too.

Conclusion and Future Work

Extending mere multi-touch input layer with an association of input events with users is the next step towards a natural interaction of the real-world environment. In our work, we concentrate on techniques for sensor coupling and multi-device orchestration. We presented the basic concepts of our framework for building complex interactive systems utilizing multiple commodity sensors (both same and different kinds).

Our initial observations are quite encouraging. The proof-of-concept evaluation showed the processing speed of the framework is high enough to ensure real-time event

processing. This enables further research of gesture recognition in the area of multi-sensor group collaborative systems based on powerwalls and large tabletops.

The principles of the framework and algorithms are not limited to touch-based interaction only, but may serve as complex multi-modal interaction input layer. The algorithms showed to be promising for further exploration and they open a wide range of possible directions for future work, some of which we only touched upon in the paper.

References

[1] Agarwal, A., et al. High Precision Multi-touch Sensing on Surfaces using Overhead Cameras. In *Proc. ITS '07* (2007), 197–200.

[2] Annett, M., et al. Medusa: A Proximity-Aware Multi-Touch Tabletop. In *Proc. UIST '11* (2011), 337–346.

[3] Ballendat, T., Marquardt, N., and Greenberg, S. Proxemic Interaction: Designing for a Proximity and Orientation-Aware Environment. In *Proc. ITS '10* (2010), 121–130.

[4] Dietz, P., et al. DiamondTouch: A Multi-User Touch Technology. In *Proc. UIST '11* (2011), 219–226.

[5] Hutama, W., Song, P., Fu, C., and Goh, W. B. Distinguishing multiple smart-phone interactions on a multi-touch wall display using tilt correlation. In *Proc. CHI '11* (2011), 3315.

[6] Jagodic, R. *Collaborative Interaction And Display Space Organization In Large High-Resolution Environments*. Ph.d. thesis, 2011.

[7] Jagodic, R., et al. Enabling multi-user interaction in large high-resolution distributed environments. *Future Generation Computer Systems* (2010), 914–923.

[8] Marshall, P., et al. Rethinking 'multi-user': an in-the-wild study of how groups approach a walk-up-and-use tabletop interface. In *Proc. CHI '11* (2011).

[9] Nancel, M., Wagner, J., Pietriga, E., Chapuis, O., and Mackay, W. Mid-air Pan-and-Zoom on Wall-Sized Displays. In *Proc. CHI '11* (2011), 177–186.

[10] P., D., and J.-D., F. Support for Input Adaptability in the ICon Toolkit. In *Proc. ICMI '04* (2004), 212–219.

[11] Renambot, L., et al. SAGE: the Scalable Adaptive Graphics Environment. In *Proc. WACE '04* (2004), 8 pp.

[12] Richter, S., Holz, C., and Baudisch, P. Bootstrapper: Recognizing Tabletop Users by their Shoes. In *Proc. CHI '12* (2012), 4 pp.

[13] Senior, A., Hampapur, A., Tian, Y.-L., Brown, L., Pankanti, S., and Bolle, R. Appearance models for occlusion handling. *Image and Vision Computing 24*, 11 (2006), 1233–1243.

[14] Stødle, D., Hagen, T.-M. S., Bjørndalen, J. M., and Anshus, O. J. Gesture-based, touch-free multi-user gaming on wall-sized, high-resolution tiled displays. In *Proc. PerGames 2007* (2007), 75–83.

[15] Westing, B., Urick, B., Esteva, M., Rojas, F., and Xu, W. Integrating Multi-touch in High-resolution Display Environments. In *State of the Practice Reports*, SC '11 (2011), 8:1–8:9.

[16] Wilson, A. D., and Hrvoje, B. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proc. UIST '10* (2010).