Using Multiple Kinects to Build Larger Multi-Surface Environments

Alaa Azazi, Teddy Seyed, Fadi Botros,
Daniel Sabourin, Edwin Chan, and Frank Maurer

Agile Surface Engineering Group, University of Calgary, Canada
{alaa.azazi,teddy.seyed,fadi.botros,
daniel.sabourin,edwin.chan,frank.maurer}@ucalgary.ca

Abstract. Multi-surface environments integrate a wide variety of different devices such as mobile phones, tablets, digital tabletops and large wall displays into a single interactive environment. The interactions in these environments are extremely diverse and should utilize the spatial layout of the room, capabilities of the device, or both. For practical multi-surface environments, using low-cost instrumentation is essential to reduce entry barriers. When building multi-surface environments and interactions based on lower-end tracking systems (such as the Microsoft Kinect) as opposed to higher-end tracking systems (such as the Vicon Motion Tracking systems), both developer and hardware issues emerge. Issues such as time and ease to build a multi-surface environment, limited range of low-cost tracking systems, are reasons we created MSE-API. In this paper, we present MSE-API, its flexibility with low-cost tracking systems, and industry-based usage scenarios.

Keywords: Multi-surface framework, multi-surface applications, API Design , kinect integration

1 Introduction

Multi-Surface Environments (MSE) incorporate many heterogeneous devices into unique interactive spaces, where interaction is typically spread across and between the devices. Mobile phones, tablets, digital tabletops and large wall displays each have unique characteristics (i.e. size, mobility, resolution) that support different types of interactions and usage scenarios [10]. The interactions can use the spatial layout of the environment (i.e. flicking to another device) or non-spatial (i.e. sending to another device via a graphical user interface) which only requires interdevice communication. To design spatially-aware multi-surface systems, the environment needs to track the location and orientation of users and devices in a room so that proxemic interactions [7] can be incorporated into the workflow of the multi-surface application.

From a developer perspective, building the multi-surface environments and spatially-aware interactions faces a number of challenges. A key challenge, motivating the work presented, is related to the cost and choice of tracking sensors. In many usage scenarios, using lower-end tracking systems (such as the Microsoft Kinect[1]) is required cost-wise

[1]Microsoft Kinect. http://www.microsoft.com/en-us/kinectforwindows/

in comparison to higher-end tracking systems (such as the Vicon Motion Tracking systems[2]) despite the tradeoff in precision and robustness. As technologies such as the Microsoft Kinect are already utilized and deployed in a number of different environments [1], extending the capability to multi-surface environments is beneficial.

A specific problem with technologies such as the Microsoft Kinect is that they are designed for use in a typical home environment, which limits its use to relatively confined spaces [2]. Most use cases for the Microsoft Kinect assume a single device in a small room. As a result, occlusion problems occur when multiple users enter the space and users become untraceable in larger rooms. In this paper, we present an approach that integrates multiple Microsoft Kinects to track users and their devices in larger rooms while overcoming occlusion issues to a substantial degree. We present MSE-API, which allows developers to rapidly build and explore larger multi-surface environments, applications and interactions using multiple Microsoft Kinects. In this paper, we discuss briefly similar frameworks and their challenges and then introduce the design of MSE-API. This is then followed by a brief overview of industry-based applications built using MSE-API and finally the next steps for enhancing MSE-API.

2 Related Work

The research literature for multi-surface environments is extremely well ex-plored, from both an HCI perspective and systems perspective. One of the earliest and canonical examples of multi-surface environments is the i-Land project by Streitz et al. [3], which connected tabletops, specialized chairs with displays and large wall displays into a novel interaction environment. Users were able to move their personal content among the different devices in the environment.

Being able to move content, highlights one of the primary purpose of interactions in multi-surface environments. Interactions such as flicking [4] or picking and dropping [5] are examples of interactions that, in a multi-surface environment, require spatial information and device communication. For instance, if a user were to flick content to another user, the environment must know where each user is and where their devices are facing, as well as the system being able to communicate between the devices. As a result, to properly integrate these types of interactions for developers in a multi-surface environment, a tracking system needs to be used and a communication framework is needed.

3MF is an example of a communication framework by Kaufmann et al [6], which allows different types of devices on a network to communicate and discover each other. For a multi-surface environment, this type of communication framework would be ideal, however, from a developer perspective, its lack of proper web based standards (e.g. REST) limits its ease-of-use, in addition to not supporting direct content transfer between devices.

Proximity Toolkit by Marquadt et al. [7], is one of the first examples of a toolkit used to build applications and environments for proxemics interactions. Proxemic interactions are based upon the relationships people have with devices and each other (i.e. distance, location, orientation). Using this toolkit, developers can rapidly build unique

[2]Vicon. http://www.vicon.com/

applications with extremely precise spatial tracking. A drawback however, is its use of the expensive Vicon Motion Capture System, which requires physical markers, also limiting its practicalness for industry-based environments. While it is capable of using a Microsoft Kinect, there is a loss in its ability to do accurate spatial tracking of both people and their devices, as well as the size of the environment itself and the occlusion problems associated with the Kinect. A solution to expanding the size of an environment without highly accurate tracking sensors is to use multiple low-cost sensor equipment and merge the data from each to create a better picture of the environment. Satyavolu demonstrated that using the Kinect as a low-cost and expanded tracking system was effective for marker-based tracking as well as solving the occlusion problems of the Kinect [8]. Later work by Schonauer et al. [2], used multiple Microsoft Kinects to successfully demonstrate tracking a single user in a wider area.

As shown, there is a significant amount of work in the multi-surface space, however, much of it is either independent or not integrated, which limits their ease-of-use for developers, or is simply not cost-effective or feasible for industry-based environments. In the next section, we describe how we introduce and describe MSE-API, designed to address some of these issues.

3 MSE-API

MSE-API is designed to integrate much of the prior and sometimes independent work in the literature – proxemics and spatial interactions, multiple sensor approaches – into a useable API for developers to build multi-surface environments and applications without worrying about low-level details such as spatial tracking. Specifically, MSE-API is useful for use cases involving spatial queries and device-to-device communication. In this section, we describe the main components of MSE-API, as well as multi-Kinect integration and calibrating the room environment.

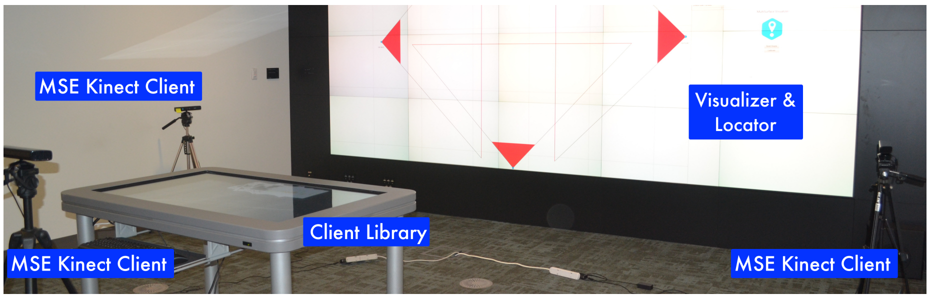

Fig. 1. *MSE Architecture.* Example environment with multiple MSE Kinect clients running, a tabletop running a client library and the MSE Visualizer and Locator service running on a wall display

3.1 API Components

The MSE-API consists of four main components: the MSE Locator service, MSE Kinect client, the MSE Visualizer, and MSE client libraries. Each of these components is explained in more detail below and shown in Figure 1.

MSE Locator. The MSE Locator service is the central component of the MSE-API. It maintains basic information about tracked devices and entities in the room space including position and orientation, which can be queried for by devices using the client libraries. The locator service is designed to obtain raw positional data from the distributed sensor clients over the local area network, and transform that data from each device coordinate space into the locators coordinate space.

MSE Kinect Client. The MSE Kinect Client uses a single Microsoft Kinect sensor to collect positional data (skeletal) of users in the room. Collected data is sent over the network using TCP connections at a rate of 30 skeleton frames per second to the MSE locator service for processing. The MSE Kinect Client also allows the API to incorporate an arbitrary number of Kinect sensors. The MSE Locator service collects tracking data from the distributed sensors over the local area network, awaiting input from the connected MSE Kinect clients. It then utilizes the received data to generate an interpretation of the entities in the room space. Following this structure allows for an easily scalable design that uses single or multiple Kinect sensors, which can be run on the same physical computer or distributed over the network. To expand the size of the observed area, multiple Kinect clients are set up covering overlapping areas of the same room. A more elaborate configuration could use multiple Kinect clients covering a larger area of the same room or distributed over multiple rooms. This is discussed in more detail in a later section. As a single Kinect sensor accurately tracks positional data from a range of 1.2 to 3.5 meters [8], incorporating multiple sensors is essential to tracking larger areas and improving the accuracy of tracking in these larger areas. In a later section, we discuss the integration of the data collected by multiple Kinect sensors, and how to calibrate MSE-API to match an arbitrary room space.

MSE Visualizer. The MSE Visualizer, shown in Figure 2, assists developers to picture and understand the MSE Locators interpretation of the position and orientation of devices and users being tracked in the environment. It presents a 2D visualization of the environment, which is updated in real-time and shows:

- The approximate area of the room space as a 2D grid (Fig 2a)
- Location and orientation of the Kinect clients currently feeding the locator service (Fig 2b)
- Devices that are currently visible within the room space (Fig 2c)
- Location and orientation of users and devices in the room space (Fig 2d & 2e)

Client Libraries. MSE-API provides client libraries in Objective-C and C# to aid developers in integrating devices (running iOS or .NET) into a multi-surface environment

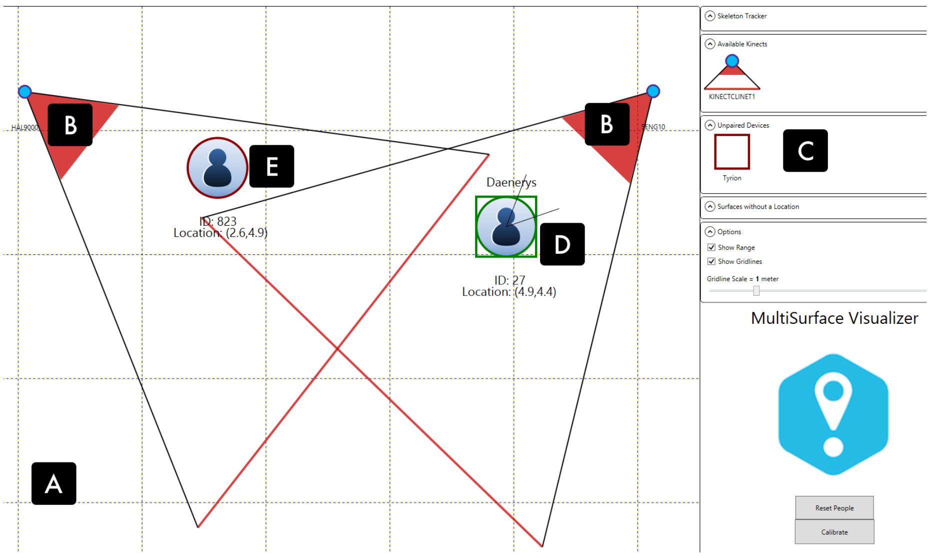

Fig. 2. *MSE Visualizer.* (A) Room Space approximation; (B) Kinect clients being used for the locator service; (C) Available device visible in room; (D) Position and orientation of a paired user and device, represented by the field of view lines and green color; (E) Unpaired but tracked user

or application. The client libraries allows device information such as orientation (from the device itself, assuming gyroscopic information is available) to be provided directly to the MSE Locator service without any extra work for the developer. Also provided is a simple interface for developers to perform spatial queries (e.g. what devices an iPad is facing) in order to obtain location and orientation information of other devices in the system. A HTTP based networking layer is used to simplify the process of sending and receiving specific data types (binary data, images, dictionaries). This allows novice developers to send and receive data throughout the system with zero understanding of the specifics of message serialization, encoding or deserialization.

3.2 Implementation.

MSE-API itself is implemented in C# (the MSE Locator service, MSE Kinect client and .NET client library) as well as Objective-C (iOS client library) and uses the Microsoft Kinect SDK to retrieve data from sensors. The Microsoft Kinect SDK also provides functionalities such as gesture recognition, which is used by the API as the mechanism to establish the pairing relationship between devices and users.

Communication: Upon starting an MSE-API based application or environment, the MSE locator service advertises its existence over the local area network using multicast. MSE Kinect clients then listen for the MSE Locator service and establish a permanent

TCP-connection with it, which is used for all future communication. The configuration of the connections of the MSE Locator and the MSE Kinect clients is established through automatic service discovery, which simplifies setup and provides plug-and-play functionality. Alternatively, for device-to-device communication, IntAirAct[3], a cross-platform networking and device discovery is used, and follows the HTTP based routing and protocols, as mentioned previously.

Integrating Multiple Kinects: Before the MSE Locator service processes any received skeletal data from a Kinect sensor, the sensor needs to be positioned physically in the room environment. When a new MSE Kinect client is detected, the MSE Locator service adds the new client to the list of connected MSE Kinect clients, and signals the client to stop transmitting skeletal frames (for performance purposes as skeletal data can be quite large). The Kinect client is presented as an icon and appears in Available Kinects in the MSE Visualizer (see Figure 2). Dragging the icon and positioning it in on the room space in the MSE Visualizer maps the approximate physical environment for the MSE Locator service. Once a MSE Kinect client is positioned, the MSE Locator signals the MSE Kinect client to start transmitting skeletal data.

Once MSE Locator receives skeletal data from a positioned MSE Kinect client, it attempts to translate and merge the data into the existing interpretation of the room space. To use the data received from the MSE Kinect client, the coordinates corresponding to each of the skeletons received are transformed from the Kinect clients coordinate space into the MSE Locators room space. The MSE Locator maintains a list of sensors tracking each user alongside the skeleton identifier each sensor is assigning for that user. After each skeleton has been transformed, the locator compares the received data with the list of users currently being tracked by the system. The locator ensures that a person that is being tracked by multiple sensors is presented only once by comparing its position with the relative position of each of the tracked users.

3.3 Calibrating the Environment

Before calibrating the Kinect sensors using MSE Visualizer, a user should ensure that the fields of view of a set of Kinect sensors are overlapping. This guarantees that no gaps are present within the tracked area of the room space. To calibrate, a user will need to stand in the mutual area observed by all the sensors, and click the Calibrate button on the MSE Visualizer, which will translate the locations of the sensors accordingly, and thus all sensors will return the same position for the tracked user. This results in the user being constantly tracked by multiple sensors and, therefore, solving the occlusion problem. The process (Figure 3) can be repeated on a different set of sensors in larger installations that cover larger or multiple rooms.

[3]IntAirAct. http://arlol.github.io/intairact.html

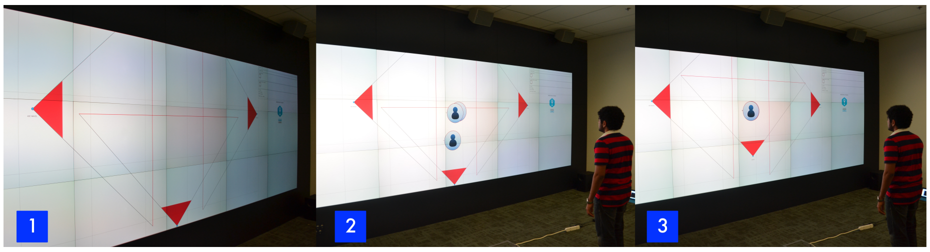

Fig. 3. Calibration. (1) Manually align the Kinects to overlap (2) A user stands in the overlapping area (3) Calibrate button is clicked on MSE Visualizer to calibrate the Kinects

4 Applications of MSE-API

4.1 ePlan Multi-Surface

C4i Consultants is a company based in Calgary, Alberta, Canada who specializes in training software for emergency response and military operations. Currently, they utilize ePlan, a software designed to simulate large scale emergencies, to train city operators on how to respond with different types and scales of emergencies. Given the collaborative nature of emergency response planning [9] and the number of individuals in the process, this was an ideal candidate for building a multi-surface environment. In collaboration with C4i Consultants, a multi-surface environment was created utilizing ePlan, that allowed larger groups of different stakeholders (e.g. fire, police, hazmat) to collaborate and communicate with a large wall display, tablets and a digital tabletop. Spatial interactions such as flick and pour are used to transfer vital emergency information amongst the different stakeholders and their devices.

4.2 SkyHunter

SkyHunter Exploration Ltd, located in Calgary, Alberta, Canada is a company who specializes in oil and gas exploration. With proprietary technology, they collect a variety of geo-spatial data, much of which is multi-disciplinary, and ultimately increases the chances of discovering oil and gas significantly. Prior to building a multi-surface application, much of the collaboration with the stakeholders in the exploration process (i.e. geophysicists, geologists) was paper-based and ineffective due to large volumes of geo-spatial data. Upon building a multi-surface environment, positive feedback was immediately received in collaboration, coordination and geo-spatial data integration.

5 Future Work & Conclusion

One of the first directions to further this research is a larger formal study of MSE-API. Several pilot usability evaluations have been performed, with relatively positive feedback, however, a formal one has not been conducted as of yet. Another unique direction for this research is to allow for MSE-API to be heterogeneous in the types of low-cost tracking sensors to be utilized. This means being able to use newer sensor technologies for more fine grained tracking in addition to the larger tracking space provided by multiple Kinect sensors.

In this paper, we present MSE-API, which provides a successful low-cost multi-tracking solution for building larger multi-surface environments and applications. We describe example applications built for industry-partners using the API and propose additional work for the future, including its evaluation.

References

1. Panger, G..: Kinect in the kitchen: testing depth camera interactions in practical home environments. In CHI '12 Extended Abstracts on Human Factors in Computing Systems (CHI EA '12). ACM, New York, NY, USA (2012)
2. Schonauer, K., Kaufmann, H.: Grid Wide Area Motion Tracking Using Consumer Hardware. In Proceedings of Workshop on Whole Body Interaction in Games and Entertainment. Lisbon, Porgual (2011)
3. Streitz, N., GeiBler, J., Holmer, T., Shin'ichi, K., Mller-Tomfelde, C., Reischl, W., Rexroth, P., Seitz, P., Steinmetz, R.: i-LAND: an interactive landscape for creativity and innovation. In Proceedings of the SIGCHI conference on Human Factors in Computing Systems (CHI'99). Pittsburgh, USA (1999)
4. Bragdon, A., DeLine, R., Hinckley, K., Morris, M.R.: Code space: touch + air gesture hybrid interactions for supporting developer meetings. In Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces(ITS'11). Kobe, Japan (2011)
5. Rekimoto, J.: Pick-and-drop: a direct manipulation technique for multiple computer environments. In Proceedings of the 10th annual ACM symposium on User interface software and technology (UIST'97). Banff, Canada (1997)
6. Kaufmann, M., Hitz, M.: 3MF - A Service-Oriented Mobile Multimodal Interaction Framework. In Proceedings of the workshop on infrastructure and design challenges of coupled display visual interfaces (PPD'12). Capri, Italy (2012)
7. Marquardt, N., Diaz-Marino, R., Boring, S., Greenberg, S.: The proximity toolkit: prototyping proxemic interactions in ubiquitous computing ecologies. In Proceedings of the 24th annual ACM symposium on User interface software and technology (UIST'11). Santa Barbara, USA (2011)
8. Satyavolu, S., Bruder, G., Willemsen, P., Steinicke, F.: Analysis of IR-based virtual reality tracking using multiple Kinects. In Proceedings of the 2012 IEEE Virtual Reality (VR'12). Orange County, USA (2012)
9. Qin, Y., Liu, J., Wu, C., Yuanchun, S.: uEmergency: a collaborative system for emergency management on very large tabletop . In Proceedings of the 2012 ACM International Conference on Interactive Tabletops and Surfaces(ITS'12). Boston, USA (2012)
10. Dzmitry Aliakseyeu, Andrs Lucero, Jean-Bernard Martens: Users' quest for an optimized representation of a multi-device space. Personal and Ubiquitous Computing 13(8): 599-607 (2009)