Dynamic Tiling Display: Building an Interactive Display Surface using Multiple Mobile Devices

Ming Li and Leif Kobbelt
Computer Graphics & Multimedia Group
RWTH Aachen, AhornStr. 55
Aachen, Germany
http://www.graphics.rwth-aachen.de/

ABSTRACT

Table display surfaces, like Microsoft PixelSense, can display multimedia content to a group of users simultaneously, but it is expensive and lacks mobility. On the contrary, mobile devices are more easily available, but due to limited screen size and resolution, they are not suitable for sharing multimedia data interactively. In this paper we present a "Dynamic Tiling Display", an interactive display surface built from mobile devices. Our framework utilizes the integrated front facing camera of mobile devices to estimate the relative pose of multiple mobile screens arbitrarily placed on a table. Using this framework, users can create a large virtual display where multiple users can explore multimedia data interactively through separate windows (mobile screens). The major technical challenge is the calibration of individual displays, which is solved by visual object recognition using front facing camera inputs.

Categories and Subject Descriptors

I.3.2 [**Computer Graphics**]: Graphics Systems - Distributed /network graphics; H.5.2 [**Information interface and presentation**]: User Interfaces - Graphical user interfaces; B.4.2 [**Input/Output and Data Communications**]: Input /Output Devices - Image Display

General Terms

Algorithm

Keywords

Mobile devices, front facing camera, pattern recognition, interaction, multiple display, calibration, evaluation

1. INTRODUCTION

Interactive Tables, like Microsoft PixelSense [7], provide an interactive display surface and computing platform that allows multiple users to visually share and interact with digital content in a collaborative setting. Special features like multi-touch techniques and large display area make it a nice platform to share multimedia data, but due to the size, weight and cost its mobility is limited and it is not easily available for normal customers. On the contrary, mobile devices are much cheaper and easily accessible. It is a good medium to display/interact with multimedia data, like browsing photos and playing games. However, limited by its resolution and screen physical size it is not suitable for interactively sharing visual content to a group of users simultaneously.

The recent advance in mobile industry integrates the front facing camera to most mobile devices. Many applications utilize the front facing camera (FFC) in a single device scenario, e.g. gesture/face recognition, augmented reality or photo/video applications. The research effort has not been invested into FFC for multiple device scenarios, especially using FFC to expand mobile display surface. In this paper we present a framework that creates an interactive table using multiple mobile devices, see Figure 1. Instead of displaying unique visual patterns on corresponding mobile screens, we detect common visual features (e.g. a fiducial marker) in the current field of view of the devices' FFC and compute the relative position and orientation between devices. Based on that, every device can display a small part of the multimedia data, and together they create a large display surface. The benefits of this approach are: First, since the FFC is integrated in devices, no extra camera system is needed; Second, the pattern is not displayed on mobile screens, so the pattern recognition is not affected by screen reflection; Third and most important, moving devices after calibration will not break the viewing experience of one large display surface. Since FFC visual object recognition

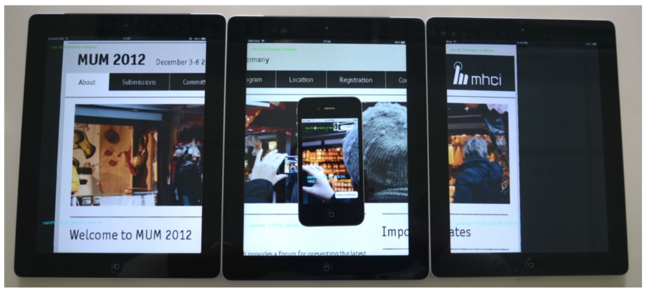


Figure 1: Our framework enables an interactive display surface using multiple mobile devices for novel forms of interaction.

is independent from displaying multimedia on screens, devices can be tracked even after calibration, i.e. the content on a screen is synchronized with its physical position on the table, see Figure 2. Using this framework, we can create new interactive table applications and games. For example, multimedia data, high definition images or a graphical user interface, can be presented on a virtual table surface, where users could focus on the data by moving separate windows (mobile screens) around the table. Moreover, new game concepts can be created to associate the position of the device on the table with its visual content or game logic, e.g. moving a game character.

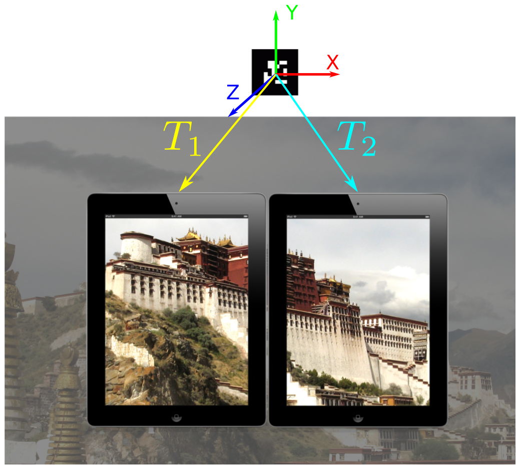


Figure 2: Visual pattern recognition will provide the position and orientation of the device in the marker coordinate system, from which we can compute the correct viewport of each device.

2. RELATED WORK

Many researchers have proposed approaches to combine multiple screens into a large display surface. Some methods focus on creating a fixed layout of devices. Merrill et al. [6] presented Siftables that uses several very small displays to form a tangible user interface. Their system allows novel input metaphors, but their devices are not off-the-shelf hardware and the limited computing power is not suitable for general multimedia data. Another example is iPodWall [10] which manually aligned a matrix of iPods to display an image slide show. Hinckley et al. [3] introduced Stitching that combines several laptop screens in a grid configuration and supports pen gesture spanning multiple displays. All these approaches allow novel interaction in multi-display scenarios, but lack effective calibration methods, thus do not support arbitrary device layout.

In order to distribute the correct view for individual mobile screen, existing approaches include a separate calibration process which displays unique id patterns on each mobile screen and uses an extra camera system to detect the position and orientation of the corresponding devices. For example, [8, 2] encode device ids in fiducial markers and use marker detection based automatic calibration or optional manual calibration methods. Other researchers [5, 9] encode every device id in a color sequence flashing on the corresponding screen. An extra camera system records and decodes the sequence to find each individual device. Theoretically as long as the size of a device appearing on the image frame is larger than one pixel, it could be detected. The differences between these two approaches are: The fiducial marker based calibration allows accurate registration of each device, but requires a high resolution camera so that every single marker can be detected clearly. If the number of devices increases or the distance to the camera increases, the detection will fail. On the contrary, the color sequence based calibration provides less accurate registration, but scales very good with lots of mobile devices.

Although the existing approaches offer good results for different scales of mobile devices, they have two issues not solved yet. First, strong ambient light or reflection on mobile screens can affect the detection results or increase the failure rate of calibration. Second, after calibration devices are not allowed to move, otherwise the aligned view is broken and users need to re-calibrate. Our proposed method can solve these issues by using the front facing camera of mobile devices and visual object recognition algorithms.

3. SYSTEM DESCRIPTION

3.1 System Structure

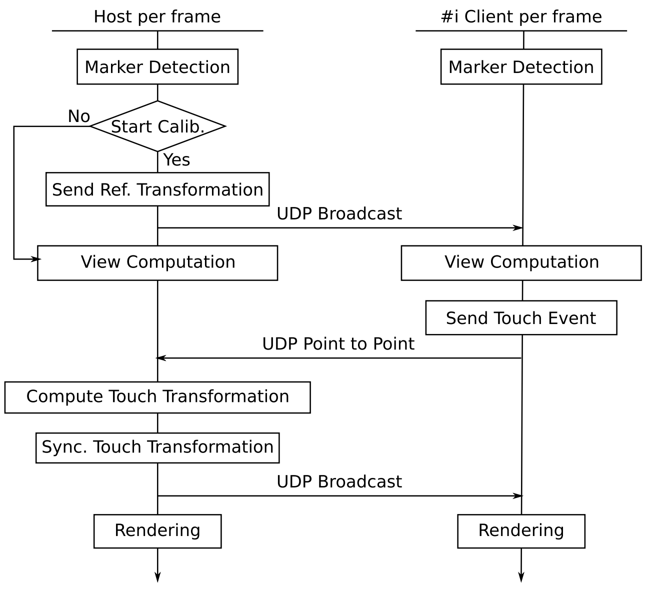


Figure 3: System state: Host broadcasts calibration data to all clients, collects touch events from clients and synchronizes the touch gesture back to clients.

Our system utilizes host-client structure, see Figure 3, where TCP/UDP connection is built via a Wi-Fi access point. Each device renders the same multimedia data locally and its individual view of the rendering is determined by visual recognition. The host sends calibration data (its initial transformation in the marker coordinate system) to all clients as a reference point, so that they can compute their relative transformation (view matrix) to this reference. Notice here the host only needs to send this reference point once in the beginning of the calibration. Later on all devices (host/clients) locally update their view relative to this reference. No extra calibration data is transmitted. In addition, the multi-touch handling is similar to [8]. The host collects touch events (pan, pinch and rotate) from clients, computes a transformation matrix (model matrix) and synchronizes that back to all clients. The computational complexity of the system depends on the marker detection algorithm. In the current prototype, the performance varies from 10 to 15 FPS on an iPad2.

3.2 View Computation

Since every mobile device renders the same multimedia data, e.g. an HD picture, we need to show the correct view on each individual screen, so that together they can give users a viewing experience that there is one global virtual canvas placed on the table. The goal of view computation is to determine the relative view between devices.

Our approach was inspired by structure from motion concept, i.e. if two cameras can see a common feature, their relative position and orientation can be computed. In the current prototype, a fiducial marker is used. The marker detection algorithm is provided by ARToolKitPlus [1] which is based on ARToolKit [4], a widely used tracking library.

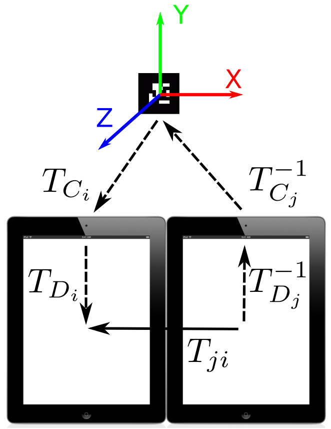


Figure 4: View Computing: The transformation between the center of two devices is computed utilizing marker detection.

In the beginning of the calibration, a marker is placed in the field of view of front facing cameras of devices arbitrarily placed on a flat table. For each device's camera C_i, a transformation matrix T_{C_i} in the marker coordinate system is computed by the marker detection, which describes the position and orientation of a camera relative to the marker, see Figure 4. As long as two cameras C_i, C_j can see the same marker, we can compute the relative transformation from C_j to C_i by:

$$T_{C_jC_i} = T_{C_i}T_{C_j}^{-1}$$

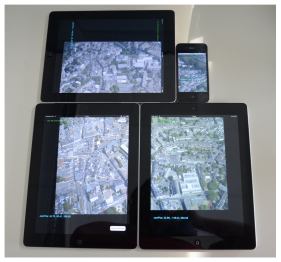
Since a view is centered at a mobile device center, we want the relative transformation between screens' center rather than cameras. Normally a mobile camera is not located at its screen center, therefore a constant offset transformation T_{D_i} (from the camera center to the screen center) is applied for each device D_i:

$$T_{ji} = T_{D_i}T_{C_i}T_{C_j}^{-1}T_{D_j}^{-1}$$

T_{ji} gives us the view of D_j relative to D_i (view matrix of D_j). In our prototype, we use the initial transformation of the host device in the marker coordinates as a reference point. Later on, every device, including the host, displays a view relative to this reference. In this way, there is no need to update the host transformation to all clients via network, so the network latency does not influence users' experience.

3.3 Demo Application

To demonstrate the dynamic tracking capability, we built an example application to show the potentials of the framework. In this application users can explore a virtual map via an interactive display surface built by multiple mobile devices arbitrarily placed on a table. In order to track mobile devices, a marker is setup on the ceiling above the table. Mobile screens, like movable windows, could be used to explore separate parts of the map. When placed next to each other, they show a bigger view of the map content, see Figure 5(a). Using this framework we can synchronize the visual content of a mobile screen with its location on a table, which could be used to create new interaction metaphors in multi-player games, e.g. table puzzle games, board games, etc.


(a) Map Exploration

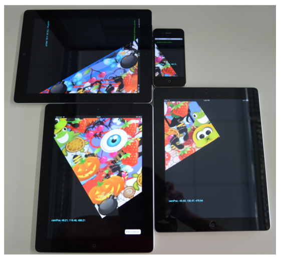

(b) Experiment Layout

Figure 5: Demo and Experiment. See more in the accompanying video.

4. EXPERIMENT AND RESULTS

In order to evaluate the efficiency and precision, we compare our calibration method FCC (Front Camera Calibration) to the approach AMC (Automatic and Manual Calibration) [8] . In this experiment, participants need to calibrate 3 iPad2 and 1 iPhone4 in the same layout using FCC and AMC separately, see Figure 5(b). In the FCC task, the user holds a marker cardboard over screens to calibrate. While in the AMC task, the user operates a host camera device to make automatic calibration and then manually calibrates the host device. A within subjects design is used for the experiment, i.e. each user needs to calibrate devices using two calibration methods (FCC and AMC). The order of tasks for different participants is balanced. The task completion time is measured and compared. Furthermore, since the calibration precision is important to the viewing experience, we also ask users to rate their perceived calibration precision of these two approaches on a 5-point Likert Scale (1 = very inaccurate, 5 = very accurate). We choose the user-perceived precision rather than the objective error distance, because borders of devices occlude visual content. Users have to complete the image in their mind. Even if there is a misalignment of several millimeters, it could be unnoticeable.

10 users participated the experiment, aging from 21 to 30. They were all university students studying computer science. A tutorial about the corresponding approach was given before each task. Participants were allowed to practice several times until they were familiar with the methods.

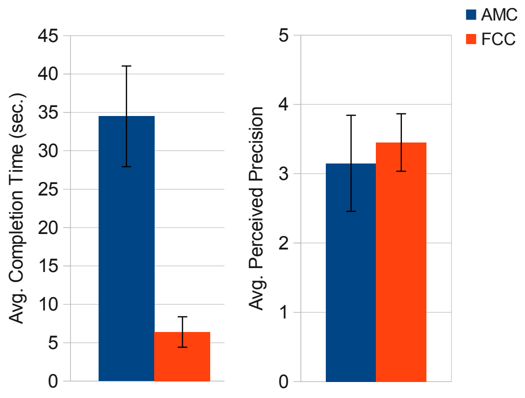


Figure 6: Average completion time and user perceived calibration quality (with 95% confidence interval)

The results are shown in Figure 6. One-way ANOVA was used to examine the effect of different approaches. Significant difference was found ($p < 0.0001$) on the average completion time: AMC 34.5 and FCC 6.4 seconds. While no significant difference was found on the average perceived precision ($p > 0.05$): AMC 3.15 and FCC 3.45. The experiment results indicate that FCC is more efficient than AMC (81.4% faster) which is because FCC is fully automatic, while AMC needs an extra step to manually align the host screen. Users' comments also show that FCC is easier to learn and use, since they do not need to care about any manual calibration patterns as in AMC. The user perceived calibration precision is comparable between AMC and FCC. The perceived alignment inaccuracy in AMC is mainly due to misalignment of the manual step, while the inaccuracy of FCC is from the marker detection algorithm, e.g. the orientation error could be noticeable when checking line segments crossing neighboring screens.

5. DISCUSSION AND FUTURE WORK

Scalability. In the current prototype, the marker has to be visible to every front facing camera simultaneously so that they can compute their relative views. Since cameras of mobile devices do not have a wide field of view (especially front facing cameras), if devices are placed in distance, we have to move the marker further above the table to make it visible to all cameras. The same issue happens when the number of devices on the table increases. Theoretically, our view computation method can solve this, see Figure 7. Although not all cameras can see the marker at that distance simultaneously, if we move the marker across their field of view, the marker is visible to neighboring cameras, e.g. C_i, C_j and C_j, C_k. Therefore we can compute the transformation T_{ij} between C_i, C_j and T_{jk} between C_j, C_k. The transformation T_{ik} can be computed by simply concatenating these two transformations:

$$T_{ik} = T_{ij}T_{jk}$$

Display surface. In the current prototype, all mobile devices are placed on a flat surface in order to build an interactive display. However, since the marker detection provides 3D information of devices, it is possible to expand the display to an unflat surface as well.

Tracking methods. The current prototype uses a fiducial

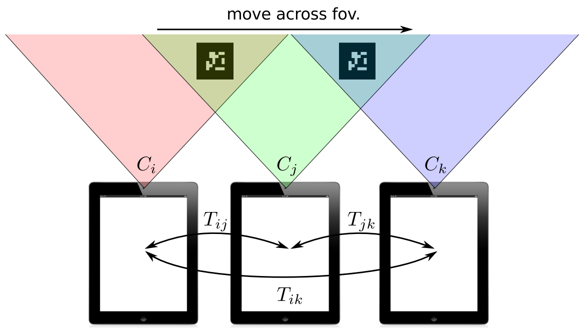


Figure 7: Scalability: If the number of devices increases, we can move the marker across the field of view of cameras to compute view.

marker to track devices, which is still not convenient for users. An alternative solution is to detect natural features in the environment, e.g. tracking features on the ceiling above the devices, or tracking hand hovering over screens.

6. REFERENCES

[1] Artoolkitplus. http://handheldar.icg.tugraz.at/artoolkitplus.php.

[2] R. Borovoy and B. Knep. Junkyard jumbotron. http://jumbotron.media.mit.edu.

[3] K. Hinckley, G. Ramos, F. Guimbretiere, P. Baudisch, and M. Smith. Stitching: pen gestures that span multiple displays. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 23–31, New York, NY, USA, 2004. ACM.

[4] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *IWAR '99: Proceedings of the 2nd IEEE and ACM International Workshop on Augmented Reality*, page 85, Washington, DC, USA, 1999. IEEE Computer Society.

[5] S. Lee-Delisle. PixelPhones. http://seb.ly/2011/09/pixelphones-a-huge-display-made-with-smart-phones/, 2011.

[6] D. Merrill, J. Kalanithi, and P. Maes. Siftables: towards sensor network user interfaces. In *TEI '07: Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 75–78, New York, NY, USA, 2007. ACM.

[7] Microsoft. Microsoft pixelsense. http://www.microsoft.com/en-us/pixelsense/, 2011.

[8] A. Schmitz, M. Li, V. Schönefeld, and L. Kobbelt. Ad-Hoc Multi-Displays for Mobile Interactive Applications. In *EG 2010 - Areas Papers*, pages 45–52, Norrköping, Sweden, 2010. Eurographics Association.

[9] J. Schwarz, D. Klionsky, C. Harrison, P. Dietz, and A. Wilson. Phone as a pixel: enabling ad-hoc, large-scale displays using mobile devices. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, CHI '12, pages 2235–2238, New York, NY, USA, 2012. ACM.

[10] I. Welikesmall. IPod Wall. http://vimeo.com/13404489, 2010.