Extending Touch: Towards Interaction with Large-Scale Surfaces

Alexander Schick[*] **Florian van de Camp**[*] **Joris Ijsselmuiden**[*] **Rainer Stiefelhagen**[*,#]

[*]Interactive Analysis and Diagnosis
Fraunhofer IITB Karlsruhe
{sci, ca, iss}@iitb.fraunhofer.de

[#]Institute for Anthropomatics
Universität Karlsruhe (TH)
stiefelhagen@ira.uka.de

ABSTRACT

Touch is a very intuitive modality for interacting with objects displayed on arbitrary surfaces. However, when using touch for large-scale surfaces, not every point is reachable. Therefore, an extension is required that keeps the intuitivity provided by touch: pointing. We will present our system that allows both input modalities in one single framework. Our method is based on 3D reconstruction, using standard RGB cameras only, and allows seamless switching between touch and pointing, even while interacting. Our approach scales very well with large surfaces without modifying them. We present a technical evaluation of the system's accuracy, as well as a user study. We found that users preferred our system to a touch-only system, because they had more freedom during interaction and could solve the presented task significantly faster.

Author Keywords

Touch and Pointing Interaction, Multitouch, Large Surfaces, Computer Vision, Visual Hull

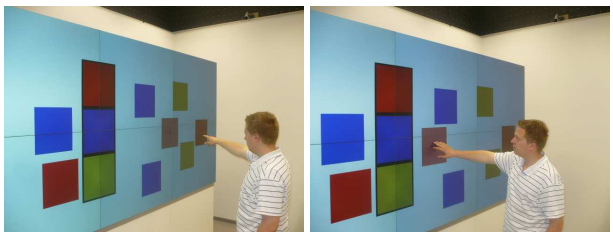
ACM Classification Keywords

H.5.2 Information Interfaces and Presentation: User Interfaces—*Input devices and strategies*

INTRODUCTION

During the last years, large displays became more widely available and increasingly received attention from the public. They are not only found in control rooms or action movies anymore, but also in public spaces like shopping malls or even some private homes as part of multimedia centers. In contrast to popular devices with smaller displays, like Apple's iPhone [1], large displays are often lacking the ability to naturally interact with them. However, table- and wall-like displays are catching up as projects like Microsoft Surface [2] and Perceptive Pixel [3] show.

ITS '09, November 23-25 2009, Banff, Alberta, Canada



(a) Pointing interaction (b) Touch interaction

Figure 1. Interaction examples.

But even if touch functionality is available, it does not entirely allow users to freely interact with objects on large-scale displays. Some of these objects might simply not be reachable by touch, because they are located too high up. Others might require the users to constantly walk around because they are far apart. In these cases, an extension to touch for interaction with large displays becomes apparent. We will show that pointing gestures are a natural extension to touch interaction and that using both touch and pointing allows highly flexible interaction. Our system handles both touch and pointing with the same method, thus allowing seamless switching between both modalities. We will show that this results in highly intuitive interaction.

Another advantage compared to existing touch solutions is that no modifications to the display are required. Our system is based on standard RGB cameras only that can be placed freely around the display. This allows us to equip arbitrary vertical surfaces with touch and pointing interaction. Most other touch technologies do not have this advantage, because they either require modifications to the screen itself or cameras need to be installed inside the display.

In order for an interaction technique to cope with arbitrary display sizes, good scaling properties are essential. We can expand the area of interaction and increase the accuracy by simply adding more cameras to our setup. The algorithm used scales linearly in the number of cameras and runs in real-time on off-the-shelf hardware.

The remainder of this paper is organized as follows: First, we discuss related work. Then, we introduce the technical details of our system. We then provide an accuracy evalua-

tion as well as a user study, before concluding our work.

RELATED WORK

In this section, we discuss existing approaches to touch-like interaction. Han developed a touch interface based on frustrated total internal reflection [4] that allows precise measurement of multiple points of interaction. Rekimoto and Matsushita implemented touch functionality in their *HoloWall* [5]. When an object touches the HoloWall, it reflects light emitted by an array of infrared LEDs behind the display. These reflections are captured by infrared cameras to detect the point of interaction on the display. Peltonen et al. used a similar approach for their *CityWall* [6] which can cope with changing light conditions. However, these approaches all require cameras to be installed behind the screen, thus limiting their applicability.

Resistance and capacitance based solutions to touch interaction rely on special, augmented surfaces. Again, augmentation of the display itself is required. Technical details to these are explained in Schöning et al. [7]. Augmented surfaces as well as the rear-projection based systems introduced above, all share one constraint: users have to physically touch the display. For large-scale surfaces, this implies that interaction with some areas is simply not possible because they are out of reach. We are developing a solution that allows users to interact regardless of their distance to the display. Our system allows interaction through touch as well as through pointing gestures.

It is difficult to create a system that allows precise and intuitive interaction when interaction events, e.g. touch events, cannot be directly measured, like with the approaches described above. This problem can be avoided using interaction tools that provide direct measurements. Parker and Inkpen magnetically track a stylus in six degrees of freedom [8], allowing both touch and pointing interaction. Xiaojun et al. developed uPen [9], a combination of a laserpointer and a contact-sensitive pen, that allows both pointing from a distance as well as direct interaction on the surface. However, these approaches require additional devices for interaction. We focus on intuitive interaction without the need for additional tools.

Benko and Wilson used special depth-sensing cameras behind the screen to measure the distance of the hand to the display [10], thus allowing hovering. Besides the need of modifying the screen, they reported that users required an introduction to the system because interaction was not intuitive.

Relatively few approaches exist that realize interaction using standard cameras in front of a display. Agarwal et al. implemented touch functionality on a tabletop surface with a top-mounted stereo-camera and custom filters to remove the display reflections [11]. They detected touch by measuring the distance of the fingertips to the display. However, this kind of camera setup is not possible for large vertical surfaces. Koike et al. introduced a similar system for a vertical display also based on one stereo camera [12], but they placed their camera on the ceiling. However, accuracy is limited. Because stereo cameras only produce 2D depth maps, only the depth of the hand's edge is measured. But, when touching the display with the whole hand, the center of the hand is the actual point of interaction, not the edge. Also, one stereo camera alone is not well suited for touch and pointing interaction with large-scale vertical displays for multiple users, because of severe difficulties with occlusions. Occlusions, however, are quite common if several people are interacting and using both hands simultaneously. We are interested in the full 3D reconstruction of the scene in front of the display to increase accuracy and to cope with occlusions.

A study similar in spirit, but different in approach is *Shadow Reaching* by Shoemaker et al. [13]. This system however, introduces constraints both on user movement and room setup due to the dedicated light source and it cannot detect touch interaction.

To the best of our knowledge, no system currently exists that allows people to interact freely with large-scale surfaces using both touch and pointing, in real-time, utilizing only standard RGB cameras. In the remainder of this paper, we will introduce such a system.

DESIGN AND IMPLEMENTATION

Figure 1 shows our system in action. A user is moving an object along our videowall, using both touch and pointing. To allow for a highly intuitive combination of touch and pointing interaction, we designed both to behave the same way. The current section presents the design and technical details of our system.

Touch and Pointing Interaction

In our interaction setup, we assume that touch is basically a binary action: either someone is touching or not. Here, we do not consider additional dimensions of touch like hovering, different pressure levels, and others. However, with this simple, binary action, a wide range of interactions is possible, starting from simple "clicking" to complex gestures with both hands [1, 2, 3]. Therefore, our goal was to implement a technical system that allows for this simple, binary action with both input modalities: touch and pointing. Building a complex user interface on top of this interaction mechanism is beyond the scope of this paper.

We want to allow users to seamlessly switch between both modalities without changing the way they interact with the system. To reach this goal and to maintain simplicity, we applied the basic properties of touch interaction to pointing interaction: we interpret pointing at an object and holding the arm still as touching it, withdrawing the arm as "untouching" it. This design consideration allows using both touch and pointing in a similar fashion.

We will now explain the technical details of our system that allow us to realize this design in a real-time system. Later, we present a technical evaluation as well as a user study that test the resulting interaction.

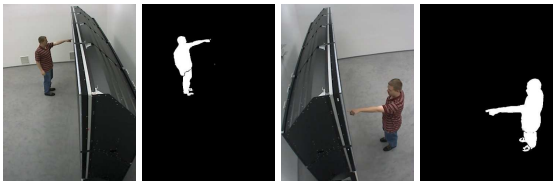


Figure 2. Camera images together with their foreground-background segmentations.

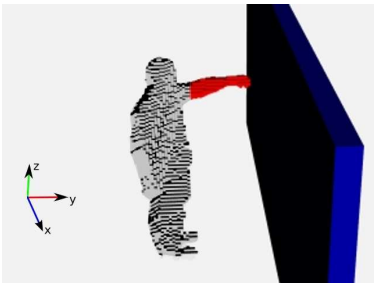


Figure 3. 3D reconstruction of the area in front of the display. The detected arm is highlighted. The display was added for visualization purposes.

3D Reconstruction

To allow interaction as described above, we need a method that does not distinguish between pointing and touching, but allows recognizing both at the same time with the same technique. For this purpose, we reconstruct the entire space in front of the display in 3D. This provides us with a 3D representation of the people acting and interacting in front of it, especially their hand and arm positions. In this 3D data, the only difference between touch and pointing is the distance to the display.

We used multiple cameras and voxel carving for 3D reconstruction. Here, we briefly introduce the concept of voxel carving. For in-depth details, we refer to [14].

Voxel carving is a way to compute the 3D structure or visual hull of foreground objects [14, 15]. To compute the 3D representation of any foreground object, we first assume that the whole space in front of the display is divided into a huge number of voxels that form a solid block together. A so-called voxel is a volumetric pixel or a 3D cube in real-world coordinates with constant side length $c \in \mathbb{R}$. The idea is to remove or *carve* every voxel that does not belong to the foreground. In order to do this, we need to compute the binary foreground-background segmentation for every camera image. Then, we project the 3D voxel into each 2D segmentation. If the voxel falls into the background in at least one image, it gets carved. Otherwise, it belongs to the foreground. The process is similar to the work of a sculptor removing tiny bits from a stone block until the statue remains. Figure 2 shows a typical camera view together with its segmentation image. Figure 3 shows the result of the voxel carving method. The visual hull can be described by the set of remaining voxels, represented by their center points:

$$H = \{x \in \mathbb{R}^3 |\ x\ belongs\ to\ the\ foreground\}. \quad (1)$$

To compute the voxel representation in real-time, we implemented our foreground-background segmentation as well as the voxel carving algorithm completely on the GPU using NVIDIA Cuda [14, 16]. We provide details on runtimes and configurations in the following section.

Using voxel carving as the foundation of our system yields several advantages. First, the time complexity of voxel carving scales linearly in the number of cameras. This allows us to vary their number and positions to cope with arbitrary display sizes and to increase the accuracy where necessary. In addition, all following processing steps solely rely on the 3D representation thus being independent of the number of cameras. Second, the huge number of voxels (usually tens of thousands) increases the robustness of our approach because each voxel can be seen as a sample point. Noise results in a relatively small number of corrupted voxels (or sample points) and therefore has a relatively small impact on the accuracy of the system. Third, using the 3D voxel data in combination with the known 3D world-coordinates of the display allows us not only to decide if users are interacting with the display, it also lets us determine directly where on the display the interaction takes place. All this is possible using one and the same framework. We will now explain how we detect user interaction with the display.

Arm Detection

After the background voxels have been carved, we want to analyze the remaining voxels to detect if and where the users touch the display or point at it. For both touch and pointing, it is sufficient to find the voxels that belong to the arms of the users (with the hand being a part of the arm). During interaction, the arm is always closer to the display than the rest of the body. Therefore, we first sort the voxels with respect to their distance to the display. Then, we "grow" clusters starting from the display and moving away from it. The clusters do not have to start at the display; they can begin anywhere in front of it.

Let $u, v \in H$ be two voxels and $d(u, v)$ be their Euclidean distance. Two voxels are direct neighbors if $d(u, v) = c$ (c is the voxel's side length). Two voxels are connected by a path $p(u, v)$ if and only if

$$d(u, v) = c \vee \exists w \in H : (p(u, w) \wedge p(w, v)). \quad (2)$$

Two voxels belong to the same cluster C_u if and only if there is a path between both:

$$C_u = \{x \in H | p(x, u)\}. \quad (3)$$

This allows very precise clustering, even to cluster crossed arms. Because we are interested in segmenting the arms only, we restrict the clusters to a certain length. Erroneous clusters, that do not resemble arms, can be detected by analyzing their length and thickness. Figure 3 shows examples of a 3D reconstruction with a recognized arm.

When we have computed the set of clusters, each cluster is analyzed to detect the direction it points to. In our setup, the

display lies in the x-z-plane and is orthogonal to the y-axis of the world coordinate system (Figure 3). As we stated before, the arms, and therefore the voxels, are always ordered towards the display along the y-axis. The voxel distribution along the x- (width) and z-axis (height) can be considered to only depend on the y-axis and to be linearly correlated to the y-values. (For example, when an arm points to the bottom-right corner, the x-values increase and the z-values decrease while moving along the y-axis towards the display.) Therefore, we can fit a straight line in 3D space to the cluster using double linear regression (x-y and z-y). The linear regression fits perfectly to the voxel approach because each voxel represents one sample point. In addition, linear regression belongs to the group of least square estimators and is therefore optimal. Combining both 2D straight lines from the linear regressions, $x = a_1y + b_1$ and $z = a_2y + b_2$, yields the pointing direction of the arm as a 3D straight line. Because all voxels of a cluster, i.e. sample points, contribute to this straight line, it is sufficient to use the line as a representation for the whole cluster:

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} b_1 \\ 0 \\ b_2 \end{pmatrix} + \begin{pmatrix} a_1 \\ 1 \\ a_2 \end{pmatrix} \times y. \quad (4)$$

The point of interaction at the display, either the touching or pointing location, is computed by intersecting the straight line from Equation 4 with the plane of the videowall. Due to the known display dimensions – both in 3D as well as pixel coordinates – the 3D point of intersection can be directly converted to 2D pixel coordinates.

With this approach, we are able to detect touch and pointing events in each frame no matter where the users are. We will now show, how these detections can be used for interaction by tracking them over time.

Tracking and Filtering

Tracking serves multiple purposes in our system. First, we need to know in what state of interaction the arms are (possibly both arms of multiple users). To do this, we need to keep track of the past trajectories. Second, tracking is essential to allow continuous actions like moving an object along the display. Third, tracking can help to stabilize the estimations of the arm movement by filtering. This is important because it is almost impossible to hold the arm perfectly still. The further away the users stand from the display, the higher is the impact of small movements on the pointing location on the display.

Tracking strongly benefits from the 3D clustering approach described above, because this 3D representation is very robust and mostly unambiguous. Therefore, we do not need a sophisticated tracking framework. Instead, the task of tracking is reduced to matching clusters from previous timesteps to clusters of the current timestep. We call a sequence of matched clusters over time a track. To measure the similarity between two clusters, we compute the Euclidean distance between the starting and end points of their fitted straight lines; this includes information about all voxels because all of them contributed to it in the linear regression analysis. The smaller the distance, the better is the match. Each cluster contributes to the best matching existing track. If a track gets no support from any cluster, it is removed. If a large cluster could not be assigned to a track, a new track is generated. A cluster can only support one single track.

After the clusters have been assigned to tracks, filtering is applied to update the tracks. Let $t_{\tau-1}$ be the fitted line of a track and c_t of a cluster; γ is the update factor. The new track t_τ then is

$$t_\tau = (1 - \gamma)t_{\tau-1} + \gamma c_\tau. \quad (5)$$

We used an adaptive update factor: if the movement was very large, we chose $\gamma = 1$. This allows the arm to move rapidly without the track dragging behind. If the arm moved only slightly, the track is updated slowly, causing filtering of noise and small unintended movements, and therefore more stable interaction. Let m be the magnitude of the movement, then $\gamma = 1 - \frac{1}{1+m}$.

After describing how we track arms over time, we will now conclude our system description by introducing our method to detect the state of interaction.

State of Interaction

Pointing gestures consist of three phases – preparation, peak, and retraction – as McNeill and Levy showed in [17]. Based on these phases, that also exist for touching, we defined three states for each track: approaching (A), holding (H), and withdrawing (W). Each state is assigned a confidence value C between 0 and 1 with the sum over all state confidences for one track being 1. The state with the highest confidence is the active state. A new track starts with $C(A) = 1$. In each frame, the confidence of one of the states is increased by α, whereas the others are decreased by $\frac{\alpha}{2}$. Therefore, α controls how fast it is possible to switch between the three states. The decision which confidence is increased depends on the distance of the track to the display. If the distance decreased, confidence $C(A)$ is increased; if the distance increased, then confidence $C(W)$ is increased; if the distance did not change, confidence $C(H)$ is increased. A state transition occurs when the confidence of one state surpasses the confidence of the currently active state. Using these three states, it is possible to model the two actions, touching and "untouching", simultaneously for both touch and pointing.

In this section, we introduced our system for touch and pointing interaction. We will now evaluate user interaction.

EXPERIMENTS

We conducted all experiments using the same setup. The videowall is $4 \times 1.5m$ with the highest point being at $2.3m$. The $4 \times 2 \times 2.3m$ space in front of our videowall is observed by two calibrated AXIS 210a network cameras positioned near the videowall's top corners (Figures 1 and 2). Both cameras have a resolution of 640×480 pixels, the videowall has 4096×1536 pixels. The cameras were calibrated using the *Camera Calibration Toolbox for Matlab* [18]. In this setup, the videowall's diagonal (4374.53 pixels) is projected onto 624 respectively 594 pixels in the two cameras ($7 : 1$ ra-

Table 1. Runtimes per frame.

Component	time $[ms]$
Foreground segmentation	6.50
Voxel carving	8.26
Touch and pointing detection	1.89
Wait for next camera images	15.80

tio; Figure 2). This means that a hand, with an average width of 15 pixels in a camera image, is mapped to 105 pixels on the videowall.

Table 1 shows the runtimes for all parts of our system on a standard workstation with a $3GHz$ Intel(R) Core(TM) 2 Duo CPU, $4GB$ of RAM, and an NVIDIA GTX280 graphic board. Both foreground segmentation and voxel carving run on the GPU. The space in front of the videowall was clustered into around 90^3 voxels. Capturing camera images runs in a separate thread with approximately $30fps$. As Table 1 shows, all computer vision components together run with $60fps$ and are actually slowed down by the cameras.

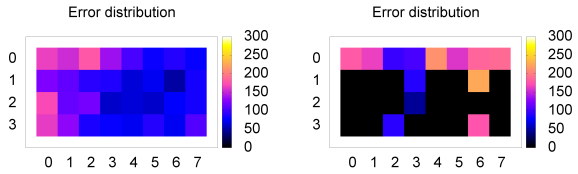
Technical Evaluation

The goal of this technical evaluation of our system is twofold: first, we want to evaluate the overall spatial precision of our system. Second, we want to analyze the correlation between the system's precision and the location of targets on the videowall.

To gather data that would allow this kind of evaluation, we displayed a cross on the videowall and asked users to touch the cross's center or point at it, whichever would feel more comfortable. The users did not get any kind of feedback when touching or pointing at the cross. We manually displayed a new cross whenever the users retracted the arm. In total, each of the five users in this experiment were presented 128 crosses evenly distributed over the whole videowall.

Table 2 shows an overview of the achieved precision results for touch, pointing, and both combined. The precision for touch interaction is high with regard to the above considerations. The precision for pointing is considerably lower. This is not surprising as there was no feedback to help the users adjust the pointing direction and pointing is less accurate in general. Since the ease with which users can reach different areas of the videowall differs, it was desirable to analyze the precision with respect to the target's locations on the wall. We divided the videowall into 32 areas and analyzed the mean error for each of these. Figure 4a shows a heat map for touch interaction. The best precision is achieved for areas in the videowall's center. Pointing gestures were especially used for the hard to reach upper targets. Figure 4b shows a heat map of these areas for pointing interaction. The black areas indicate that pointing gestures were not used for targets in this area.

Because our voxel-based system relies on the triangulation of foreground pixels, their precise extraction is crucial. Naturally, an object closer to the camera allows for a more precise extraction. Because of the triangulation, the system depends on a good extraction quality in both cameras. However, since the user is rather far away from one of the cameras when interacting with targets at the left or right end of the videowall, the optimal position of targets for both foreground pixel extraction and triangulation of the hand is in the videowall's center.



(a) Touch (b) Pointing

Figure 4. Accuracy for touch and pointing. Black areas in b) indicate that pointing was not used on the whole videowall in this accuracy evaluation.

Table 2. Accuracy with respect to the centers of targets on the videowall.

Modality	mean error		standard deviation	
	px	mm	px	mm
Touch	97.2	94.9	67.2	65.6
Pointing	170.1	166.1	93.8	91.6
Both combined	105.7	103.2	77.5	75.7

Our system achieves a high accuracy, considering the fact that, due to our setup, each hand projects to 105 pixels on the videowall and users did not receive any feedback. This accuracy can be increased by adding more cameras or using cameras with a higher resolution. Then more details of the hands would be visible and they could be reconstructed more accurately. Because our voxel reconstruction is independent of the resolution (except bandwidth), it could be applied directly. However, users were comfortable with the current accuracy as we will show now in our user study.

User Study

As a first evaluation of our interaction method, we conducted a user study with 19 participants. Four of them were female, 15 were male. The average age was 36, with the youngest participant being 22 years old and the oldest 66. The average height was 1.81m, ranging from 1.66 to 1.92m. Most participants had at least some experience with touch and little to no experience with pointing interaction (touch: $\mu = 2.05$, pointing: $\mu = 1.47$, where 1 = no experience, 2 = little experience, 3 = a lot of experience).

Experimental Setup. To investigate whether pointing interaction is a useful and natural extension to touch, we compared a setup with only touch to a setup with both pointing and touch. These setups are called *touch-only* and *touch-and-pointing* from now on. The task for the participants was to quickly move colored blocks of 300 by 300 pixels on the screen to the goal area of the same color (Figure 1). This size was chosen to allow comfortable interaction with respect to

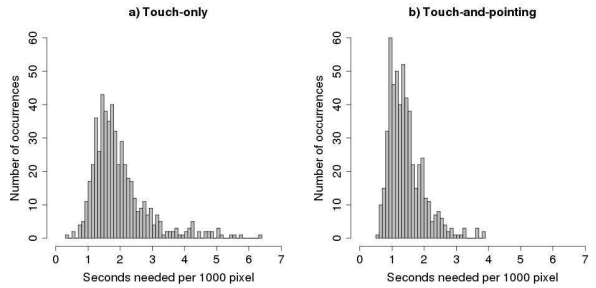


Figure 5. Time needed to move one block over a 1000 pixels during both experimental setups.

the accuracy of the system (see Table 2). One run of the experiment consisted of dispatching ten randomly positioned blocks, with each one having the colors red, blue, or yellow randomly assigned to it. How to carry out the task was up to the participant, but the goal was to clear the screen as fast as possible.

The study is conducted within-subject, with each participant carrying out three runs using the setup touch-only and three runs with the setup touch-and-pointing. Which setup came first was altered between subjects. After we explained the task and the system's capabilities, the participants could try out both interaction methods for as long as they wanted. Then, the actual experiments took place. Subtask completion times were recorded for each individual block. Afterwards, each participant filled out a questionnaire. The results are presented below.

Some detailed comments are in place here, concerning the system configuration during these experiments. To select a block, the participants had to touch it or point at it for $0.25s$. We chose this threshold empirically to avoid unintentional object selections. However, in future systems, this threshold could be omitted for touch interaction which usually does not have a delay. Distinguishing between touch and pointing can easily be done by looking at the distance of the hand to the screen given by their 3D positions. This would further improve the intuitivity of our system for touch interaction. The selected block then becomes partly transparent and arm movements will cause it to move. During touch-only, the hand must stay in contact with the screen, whereas during touch-and-pointing, the participants can stand up to two meters away from the screen and still interact using pointing gestures. Both hands can be used, but not simultaneously. Although the system allows for multi-user and multitouch interaction, we decided to keep things simple for this first experiment. During touch-and-pointing, it is possible to switch between touch and pointing at will, even while moving an object.

Subtask Completion Times. Instead of measuring the completion time of the entire task, we logged the time from each block's selection to its arrival in the goal area. This way, we isolated the time needed for interaction only, thereby eliminating the time in between subtasks. The isolated subtask completion times include walking from the selected object's origin to the goal; walking time in between subtasks is omitted. For one user, the subtask completion times were not processed due to technical difficulties, leaving us with 2×540 data points. These are normalized to get the number of seconds needed to displace an object 1000 pixels, thus paying respect to differing distances to the goal area. The results are sorted into $0.1s$ histogram bins and are shown in Figure 5.

Table 3. Comparison between the four quality ratings for touch-only and touch-and-pointing (1 = definitely disagree, 5 = definitely agree).

The interaction method ...	Touch-only		Touch-and-pointing		Touch-and-pointing was better?
	μ	σ	μ	σ	
was intuitive and easy to use	4.26	0.81	4.47	0.90	Borderline, $p = 0.14$
was reliable and accurate	3.47	0.77	3.79	0.86	Borderline, $p = 0.08$
allowed reaching all parts of the screen	2.47	1.26	4.42	0.84	Yes (better), $p \ll 0.01$
was *not* physically exhausting	2.74	1.24	3.58	1.02	Yes (better), $p = 0.02$

The data displayed in Figure 5 shows that the average speed for touch-only is significantly lower than the one for touch-and-pointing: $\mu = 1.98s$ versus $\mu = 1.43s$ (*Wilcoxon rank sum test*: $p \ll 0.01$). This could be partly due to the required walking time during touch interaction. Also, for touch-only, the data is more spread out with a standard deviation of $\sigma = 0.88$ as opposed to $\sigma = 0.54$ for touch-and-pointing. Thus, touch-and-pointing seems to be the faster and more reliable interaction method. We did not detect a learning effect over the three runs within each experiment, indicating that the interaction methods were easy to master.

Questionnaire Results. To evaluate and compare the perceived quality of both setups (touch-and-pointing and touch-only) we presented the participants with a questionnaire containing 12 statements with five-point Likert scales. The five-point Likert scale we used, was associated with the following ratings: 1 = strongly disagree, 2 = disagree, 3 = neither agree nor disagree, 4 = agree, 5 = strongly agree. The results for all 19 participants are presented and analyzed below. There were also three open questions on the questionnaire, regarding particularly strong aspects, particularly weak aspects, and any remarks or questions that the users might have.

For the 12 Likert statements, we interpreted the responses as the numbers 1 to 5 and then calculated the mean μ and standard deviation σ over all 19 participants. Then, the results of the four quality statements for touch-only were compared to the ones for touch-and-pointing (Table 3). The four remaining statements were compared to a normal distribution with $\mu = 3$ (Table 4), because they cannot be compared in a two-way fashion. All comparisons are tested for statistical significance using the *Wilcoxon rank sum test*, because

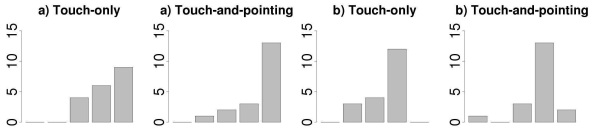


Figure 6. a) The interaction method was intuitive and easy to use. b) The interaction method was reliable and accurate.

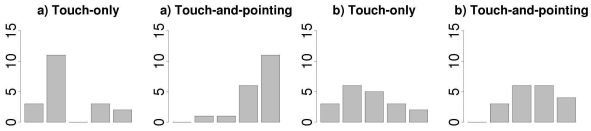


Figure 7. a) All parts of the videowall were easily reachable. b) The interaction method was *not* physically exhausting.

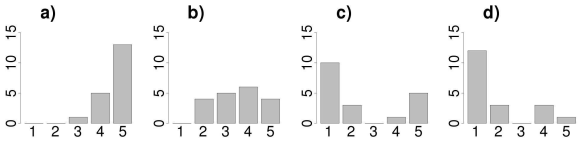


Figure 8. a) Pointing gestures were a useful addition. b) Touch and pointing worked according to the same principle. c) With touch-and-pointing, I switched repeatedly between touch and pointing interaction. d) I switched between touch and pointing while moving one single object.

some data cannot be considered to be normally distributed. However, t-test results on these data were similar.

Table 3 shows a comparison between the results of the four quality statements for touch-only and touch-and-pointing. The null hypothesis for each of them is that there is no difference in the Likert assessments between the two interaction setups. The first statement, "the interaction method was intuitive and easy to use", received a slightly higher rating for touch-and-pointing than for touch-only (both larger than 4 on average, Figure 6a). The second statement, "the interaction method was reliable and accurate", received an average rating between 3 and 4, with a slightly higher rating for touch-and-pointing (Figure 6b). From these high user ratings and the answers to the open questions, we deduce that for both experimental setups, intuitiveness, ease of use, reliability, and accuracy are satisfactory in an absolute way (not comparing the two setups).

The third statement says that "all parts of the videowall were easily reachable with this interaction method". Not surprisingly, Table 3 shows that touch-and-pointing received a considerably higher rating than touch-only. The former received a positive rating between 4 and 5, whereas the latter is slightly on the negative side of the Likert scale (Figure 7a). Note that in our touch-only setup, most people were still able to reach all areas of the videowall some way or another. But displays are becoming larger and larger, thus increasing the need for pointing interaction. For the fourth and last comparing statement, "the interaction method was *not* physically exhausting", touch-and-pointing also received a significantly higher rating (Figure 7b). Physical load is a known problem for vertical large-scale touch and gesture interfaces, apparently more so for touch than for pointing in our setup. This needs to be considered during interface design.

We acknowledge the advantage of touch-and-pointing over touch-only by rejecting all four null hypotheses (that there is no difference in the Likert assessments between the two interaction setups) in favor of the alternative that touch-and-pointing outperforms touch-only. Note that for the first two statements in Table 3 there is only a slight difference. For the third and fourth statement in Table 3, the difference is clear.

In Table 4, the four remaining Likert assessments are summarized. The null hypothesis for these is that $\mu = 3$. The first one, "the availability of pointing gestures was a useful addition", should be considered the most important statement of our questionnaire. Its average rating is definitely larger than 3; in fact, $\mu = 4.63$ (Figure 8a). The second statement in Table 4, "touch and pointing worked according to the same principle", received a rating lower than expected, but still significantly larger than 3 (Figure 8b).

Finally, we asked the participants whether they repeatedly switched between touch and pointing when both were available. The third statement in Table 4 presents the results for switching in general (between and during subtasks, Figure 8c), and the fourth one for switching during a subtask (while moving one single object, Figure 8d). Interestingly, for both statements, we observed peaks at the extremes of the Likert assessments. This implies that users solved the task with different strategies. Our system supported these different strategies due to its flexibility. However, we had to change our alternative hypothesis for these two to "this statement received a rating *smaller* than 3." For both statements, the null hypothesis was rejected in favor of the alternative ($\mu < 3$). However, we believe that many participants performed switches between touch and pointing unconsciously. The recorded videos showed that many users indeed switched, even while moving a single object. Often, they did not switch completely; instead, they used a hybrid state similar to hovering that they might have interpreted as touch. This would not be possible for a touch-only system.

Table 4. The ratings for the four remaining statements (1 = definitely disagree, 5 = definitely agree).

Statement	μ	σ	Wilcoxon rank sum test
Pointing was a useful addition	4.63	0.58	$\mu > 4$ with $p \ll 0.01$
Touch and pointing same principle	3.53	1.07	$\mu > 3$ with $p = 0.02$
Switching in general	2.37	1.77	$\mu < 3$ with $p = 0.08$
Switching during one object	1.84	1.34	$\mu < 3$ with $p < 0.01$

CONCLUSION

In this paper, we introduced our system that allows both touch and pointing interaction with large-scale surfaces. Our system is based on 3D reconstruction and does not require modifications of the display. Due to the nature of voxel carving, it scales easily in the number of cameras, thus allowing deployment for large surfaces. The system's accuracy is very good considering the technical constraints introduced by the camera resolution. We tested two setups, and both performed well in our user study. We found that users strongly preferred a system that allows both touch and pointing interaction to a system that allows only touch.

In future work, we will develop user interfaces for our system and evaluate these with more experiments. This will include a more detailed analysis of user behavior as well as a more sophisticated interface design, for example utilizing Fitt's law. Furthermore, we will apply full articulated body tracking to recognize more complex gestures and to distinguish between intentional and unintentional gestures. This would also allow us to use the head-hand line to determine the point of interaction which is more accurate for pointing gestures [19]. In addition, we will investigate what the best strategy is to increase the accuracy.

Acknowledgements

This work was supported by the FhG Internal Programs under Grant No. 692026. We also thank the reviewers for their valuable comments that helped to improve this paper.

REFERENCES

1. Apple iPhone, *http://www.apple.com/iphone/* (June 2009)

2. Microsoft Surface, *http://www.microsoft.com/surface/* (June 2009)

3. Perceptive Pixel, *http://www.perceptivepixel.com/* (June 2009)

4. Han, J.Y.: Low-Cost Multi-Touch Sensing through Frustrated Total Internal Reflection. ACM symposium on User Interface Software and Technology (2005) 115–118

5. Rekimoto, J., Matsushita, N.: Perceptual Surfaces: Towards a Human and Object Sensitive Interactive Display. Workshop on Perceptural User Interfaces (1997)

6. Peltonen, P., Kurvinen, E., Salovaara, A., Jacucci, G., Ilmonen, T., Evans, J., Oulasvirta, A., Saarikko, P.: It's Mine, Don't Touch!: Interactions at a Large Multi-Touch Display in a City Centre. SIGCHI conference on Human factors in computing systems (2008) 1285–1294

7. Schöning, J., Brandl, P., Daiber, F., Echtler, F., Hilliges, O., Hook, J., Löchtefeld, M., Motamedi, N., Muller, L., Olivier, P., Roth, T., von Zadow, U.: Multi-Touch Surfaces: A Technical Guide. Technical Report TUMI0833: Technical Reports of the Technical University of Munich (2008)

8. Parker, J.K., Mandryk, R.L., Inkpen, K.M.: Integrating Point and Touch for Interaction with Digital Tabletop Displays. IEEE Computer Graphics and Applications 26 (2006) 28–35

9. Xiaojun Bi, Yuanchun Shi, Xiaojie Chen: uPen: A Smart Pen-like Device for Facilitating Interaction on Large Displays. IEEE International Workshop on Horizontal Interactive Human-Computer Systems (2006)

10. Benko, H., Wilson, A.: DepthTouch: Using Depth-Sensing Camera to Enable Freehand Interactions On and Above the Interactive Surface. Microsoft Research Technical Report MSR-TR-2009-23 (2009)

11. Agarwal, A., Izadi, S., Chandraker, M., Blake, A.: High Precision Multi-touch Sensing on Surfaces using Overhead Cameras. IEEE International Workshop on Horizontal Interactive Human-Computer Systems (2007) 197–200

12. Koike, H., Toyoura, M., Oka, K., Sato, Y.: 3-D Interaction with Wall-Sized Display and Information Transportation using Mobile Phones. Workshop on designing multi-touch interaction techniques for coupled public and private displays (2008)

13. Shoemaker, G., Tang, A., Booth, K.S.: Shadow Reaching: A New Perspective on Interaction for Large Wall Displays. ACM symposium on User interface software and technology (2007) 53–56

14. Schick, A., Stiefelhagen, R.: Real-Time GPU-Based Voxel Carving with Systematic Occlusion Handling. Proceedings of the 31st Symposium of the German Association for Pattern Recognition (2009) 372-381

15. Laurentini, A.: The Visual Hull Concept for Silhouette-Based Image Understanding. IEEE Transactions on Pattern Analysis and Machine Intelligence 16 (1994) 150–162

16. NVIDIA Cuda, *http://www.nvidia.com/cuda* (June 2009)

17. McNeill, D., Levy, E.: Conceptual Representations in Language Activity and Gesture. In R. Jarvella, W. Klein (eds.): Speech, Place, and Action, John Wiley & Sons (1982)

18. Camera Calibration Toolbox for Matlab, *http://www.vision.caltech.edu/bouguetj/calib_doc/* (June 2009)

19. Nickel, K., Stiefelhagen, R.: Pointing Gesture Recognition based on 3D-Tracking of Face, Hands and Head-Orientation, International Conference on Multimodal Interfaces (2003) 140–146