Conference Paper

User-aware content orientation on interactive tabletop surfaces

Author(s):
Schlatter, Oliver; Migge, Bastian; Kunz, Andreas

Publication Date:
2012

Permanent Link:
https://doi.org/10.3929/ethz-a-007366609

Rights / License:
In Copyright - Non-Commercial Use Permitted
User-aware Content Orientation on Interactive Tabletop Surfaces

Oliver Schlatter, Bastian Migge, Andreas Kunz
Institute for Machine Tools and Manufacturing
ETH Zurich
Zurich, Switzerland
{olivschl,miggeb,kunz}@ethz.ch

Abstract—An increasing number of human computer interaction systems are employing interactive table surfaces. For these horizontally aligned screens, the orientation of text passages and any other ‘oriented’ graphical content is a common problem. A user will not be able to easily read the same text from different sides of such a table unless it adapts to his position.

To overcome this problem, we present an interactive system that extends the interaction space from measuring the direct manipulation on the interaction plane to observing the user in the space above the table.

Hence, the content of the graphical user interface can automatically be aligned to the position of the active user, which enables the ergonomic reading of a text.

We present a viewpoint tracking system, which utilizes the Microsoft Kinect depth sensor accessed with the OpenNI framework. This system does not need initial pose calibration and smoothens the vision based tracking data. In a next step, we show the benefit of extending the interaction space for a drawing application that allows multiple users to work on automatically oriented, digital notepads while still being able to freely move around the table.

Keywords-Human-computer interfaces, 3D user tracking, signal filtering

I. INTRODUCTION

Developers around the world are constantly in search of ways to make the interaction between a human and a computer more intuitive and natural. This aspiration towards an intuitive Human-Computer Interaction (HCI) led to the term Natural Interaction (NI) [1]. The objective of NI is that every user, independent of his level of experience, is able to interact with a system without any previous instruction. Therefore, the system has to understand the behavior of a human being in order to become intuitive to every user [1].

One big step in this area was the development of touchscreens. The advantage of this technology is the high level of intuitiveness provided to the user. Being able to directly interact on the displayed content makes the interaction intuitive and natural [1]. Interactive tables like the Microsoft Pixelsense [2], and digital whiteboards, e.g. the CollaBoard [3] and SMART Boards [4] are widely used in collaborative work.

Today’s common Graphical User Interfaces (GUI) presume a certain orientation towards the user, which is automatically given for large interactive vertical screens. On small devices like tablets or mobile phones, the digital screen is flipped relative to gravity, which is sufficient for a single user interface. However, if the device is horizontally oriented, even a small angle causes the display’s content to flip, which irritates the user.

This situation is even more critical for interactive tables. Due to the horizontal orientation of large screen tabletops, multiple users can work on the same content standing around the table. However, this makes the orientation of text no longer trivial.

To approach this problem, this work presents a system that automatically orients digital content on interactive tabletops to the user’s position, as shown in Figure 1. The figure shows a top view onto a tabletop system with two users working with sticky notes. This is realized by a user-aware table that tracks the user’s viewpoint and automatically adapts the digital content.

II. RELATED WORK

The most common way of HCI is still relying on two separate channels for in- and output. The user inputs data using a keyboard and a mouse, and the computer presents optical feedback on a display. This is the widest spread and cheapest way of HCI but it is neither intuitive nor natural and therefore not optimal in terms of NI.

In contrast, touch-sensitive interactive surfaces support a highly intuitive and simple usage. In- and output are closely aligned, since the touch panel is mounted directly on top of the display. Detecting multiple touch interactions is state of
Another improvement of HCI on interactive surfaces is to use tools that are more intuitive than the user's bare finger. Tangible User Interfaces (TUIs) [5] also allow direct manipulation or generation of digital content, since the shape of the interface - which the user already knows from daily life - provides information about the inherent functionality. Taking typical office applications as an example, a pen is used to sketch or write, while an eraser is used to delete content, etc. However, these interfaces only allow a so-called direct interaction, in which the TUI or the finger has direct contact to the object to be modified. Although this tabletop system is already much more intuitive to handle, there is other indirect input from the user like his position or deictic gestures, which are not consciously entered into the system but are nevertheless relevant for an intuitive operation.

On the other hand, the gaming industry works with a large TV screen and special gaming controllers that typically rely on such indirect input. They extend the interaction from simple button controllers to user movement or posture detection. The Nintendo Wii-Controller [6] enables to detect the user's movement in front of digital screens with a dedicated controller. The detection of natural user motions sensing orientation and velocity enables intuitive game control. To detect the user's body without additional markers or hardware, the Microsoft Kinect [7] or Asus' Wavi Xtion [8] utilize only visual feedback. They provide 3-dimensional location information of users. For gaming, they usually take the location information to control a virtual avatar.

There is a rapidly increasing number of systems that benefit from the Kinect's simplicity, also in other application fields. In rehabilitation for instance, there are many systems that use the capabilities of the Kinect [9], [10].

Recently, interactive surfaces and user tracking are combined, meaning that the integration of direct and indirect interaction makes a system even more intuitive to handle. This extends the interaction space from direct manipulation on the touch panel to the real world space in front of or above digital surfaces (see Figure 2).

Most of those systems address the idea of gesture recognition. Marquardt et al. [11] proposed a system that is able to fluently go from direct touch interaction to gesture recognition and therefore stated the term continuous interaction space. The system uses a glove with markers tracking the user's hand with a vision based outside-in system. Although the tracking is highly accurate, wearing a glove makes the interaction not applicable for daily application. Also commercial systems like Microsoft Pixelsense [2] nowadays allow markerless tracking of gestures not only directly on the screen, but also slightly above it.

Another system that also considers the alignment of text was developed within InfrActables [12]. The InfrActables

Figure 2. Interaction Space above a digital surface.

Handle is a TUI designed to be used on interactive tabletops. It is a little box that can directly be placed on the screen. It has a button on one side that allows the user to pop out a digital notepad underneath the TUI. This notepad can then freely be slid around the table and if necessary also rotated to another user.

We present a similar example that works without the TUI. Since the only use for this tool is the positioning of the notepad, the interaction becomes much more intuitive if the notepad is automatically aligned to the user's position.

III. CONTRIBUTION

We present an interactive tabletop system that extends the interaction space from direct manipulation on the digital surface to the 3-dimensional space above the tabletop, as shown in Figure 2. Tracking the user's viewpoint solves the problem of orienting the digital content, which is crucial for reading text or recognizing sketches. Our system supports multiple users to collaborate on large tabletop systems and enables detecting the active user.

In our application example, two users collaborate on a table using notepads. A notepad represents a limited area, i.e. a sticky note, on which a user can sketch or write. If he wants to share the content with the other user, he moves it towards him and rotates it. This seems trivial but today's interactive systems do not cover automatic content alignment. In this paper, we present an interactive system that tracks direct manipulation of digital content on a touch screen and also observes the user's viewpoint above the tabletop. Unlike most of the applications that use the Kinect camera, our viewpoint tracker has no need for an initial pose [13]. To provide stable tracking data from using a depth vision sensor, we further use a smoothing algorithm for which we elaborated the parameters experimentally. We implemented an example application that automatically orients the digital content, in terms of notes, according to the active user's viewpoint position in front of the screen.

A. 3-Dimensional User Tracking

To extend the interaction space from direct manipulation on the surface to the indirect interaction within the space
above the tabletop, the system must be capable of tracking the user’s location. This is realized by using the Microsoft Kinect camera [7], which was primarily released as a gaming controller in November 2010.

1) Sensor Hardware: Besides the depth sensor, the Kinect has also a RGB camera with a 640x480 resolution, an accelerometer, as well as integrated microphones, which were not used in our application.

The integrated depth sensor consists of two elements. On the one side of the Kinect, there is a laser projector that emits a pattern of infrared speckles into the room. In order to calculate the depth from this 2D information, the pattern has to be known at a reference plane. Therefore, the distances between the speckles when being projected on this reference plane are stored in the memory of the Kinect.

On the other side, there is an infrared camera. If an object will change its position with regard to the camera, then the distances between the speckles will change as well, which will then be used to calculate the depth information. The depth value can be calculated for every pixel from this disparity between reference- and real projection. The generated pixel map (resolution 640x480) is then handed over to the software framework. For further information about the mathematical model behind the depth sensor, please refer to [14].

2) Stabilized Viewpoint Tracker: We use the OpenNI framework [15], which was released by PrimeSense, the developers of the Kinect technology.

This framework provides a software driver for the hardware sensor, as well as advanced information. For instance, the orientation of the device can be detected using accelerometers, while a built-in user-generator tracks moving objects using a motion flow detector. If an object moves in front of the camera, the user generator will assign an ID to the particular pixels of that specific object.

The framework also has the possibility to track the 3-dimensional position of the user’s joints (skeleton tracker). To do so, the user has to perform a specified calibration pose prior to the actual application [13]. However, this is not a natural behavior for a user. In order to avoid this initial posture calibration, we use a simplified viewpoint tracking method, which does not rely on such an initial calibration pose. Since we only focus on the user’s viewpoint, we use the highest identified pixel in the depth map of the user generator. For defining the current viewpoint, we subtract 12 cm in gravitational direction from this highest detected pixel, while the gravitational direction is deduced from the accelerometers of the Kinect. The Kinect driver then transfers the tracking data to the user aware application.

One important aspect that has to be taken into account is the transformation from display to camera coordinates, since the tracking system captures the viewpoint in camera coordinates, while the user space above the tabletop application is defined in display coordinates. Therefore, the position and orientation from the camera relative to the screen is required in order to transform and rotate the coordinates and the pixel pitch of the display to scale the coordinates. To enable a precise tracking in display coordinates, an accurate calibration of the camera and the display position is crucial.

The measurements of the Kinect, and therefore the user’s position information as well, are highly noisy. In addition, the resolution of the measurements decreases with increasing distance to the object [16]. For these reasons, the use of filter methods is inevitable.

In a first attempt, we implemented an exponential smoothing filter. This is a very efficient and effective first order infinite impulse response filter (IIR filter), which is defined by Equation 1.

\[ y(k) = \alpha \cdot y(k-1) + (1 - \alpha) \cdot x(k) \]  

(1)

A value y is updated in time steps k as a weighted combination of the value at previous time step k-1 and the new measurement x at time k. The weighting factor is defined by the variable \( \alpha \in [0, 1] \). Within experimental studies, we figured out that 0.87 is a reasonable value for \( \alpha \); providing a good trade off between noise suppression and time delay.

One drawback of this exponential smoothing filter is its slow step response. If a user walks from one side of the table to the opposite one and interacts from there again, the system experiences a step in the viewpoint information during his next interaction, since the user is only tracked while he is interacting with the surface. For this case, we want the orientation of the notepad driven by the filtered viewpoint to quickly adapt to the new viewpoint. Thus, we modified the IIR filter with a signal depending alpha value, as shown in Equation 2 and 3.

\[ \alpha = \Theta(\Delta_{\text{max}} - x) \cdot (\alpha_{\text{max}} - \alpha_{\text{min}}) \cdot \frac{1}{1 - e^{b|\Delta_{\text{max}}| - c}} + \alpha_{\text{min}} \]  

(2)

\[ \Delta = \frac{x(k) - y(k-1)}{\sigma} \]  

(3)

Where \( \Theta \) is the Heaviside function and \( \Delta \) represents the discrepancy between the current filter output and the measured viewpoint position normalized to the standard deviation of the noise signal \( \sigma \).

\( \alpha_{\text{max}}, \alpha_{\text{min}}, b, c, \) and \( \Delta_{\text{max}} \) are the filter parameters. From equation 2 we derive the plot shown in Figure 3. It shows the relationship between \( \Delta \) and \( \alpha \).

If the system experiences a step input, an alpha value close to zero causes the rapid adaption of the measured, new position of the user \( x(k) \). The Heaviside function separates steps from steady input signals. If a step is given into the system which exceeds the filter parameter \( \Delta_{\text{max}} \), \( \alpha \) is set to zero. This causes an immediate step response.
If, in contrast, the user is moving slowly, \( \Delta \) is normal distributed and therefore within the range of a few standard deviations. In this case, the value of \( \alpha \) is almost one which ensures good noise suppression.

If the user is walking during the interaction, \( \Delta \) increases and we have an in-between of the previous two cases. Hence, a tradeoff between multiple objectives, i.e. time delay, noise suppression, and accuracy must be found. Those characteristics are defined by the filter parameters \( \alpha_{\text{min}}, b, \) and \( c \) which are shown in Figure 3.

Experimental studies showed that the extended filter gives a good performance if the parameters are set to the following values:

\[
\alpha_{\text{max}} = 0.994, \alpha_{\text{min}} = 0.6, b = 2.5, c = 11, \Delta_{\text{max}} = 10
\]  

(4)

Figure 4 shows the output of the extended and the IIR filter exemplarily. The measurement data is collected by tracking the viewpoint of a stagnant person, standing in one meter distance to the camera. Afterwards, the \( x \)-measurements were extracted and some normal distributed noise with a standard deviation of six millimeters was added to the measurement in order to test the effect of the noise suppression.

Since both filters are initialized at the origin, the beginning of this set of data is equivalent to a negative step input. The nonlinear filter extension has a much faster step response compared to the IIR filter, which takes approximately 0.6 seconds to adapt to the new position. Furthermore, the nonlinear extension has a better noise suppression than the IIR filter, although the IIR filter provides higher accuracy.

However, for this specific application the accuracy is less important than the noise suppression or the step response time, since the stability of the notepad alignment is more notable to the user than its precision. In this particular case, the nonlinear filter extension improves the viewpoint tracking information.

Finally, the three-dimensional location information of the user is transformed into display coordinates and send to the application.

**B. User aware application**

Our example user application, shown in Figure 6, is written in the programming language Python [17] and uses the GUI framework \( \text{Qt} \) [18] to visualize digital objects on the screen. We define the display coordinate system as shown in Figure 5. The horizontal (vertical) dimension of the screen is defined as \( x \) (\( y \)). The distance to the screen is defined as \( z \).

Whenever the application registers a pointing event, it takes the display coordinates of the interaction position and captures the active user’s viewpoint. The active user is determined as the one with the shortest distance to the interaction position.

As shown in Figure 5, the rotation angle \( a \) between viewpoint VP and virtual object (notepad) beneath the interaction point IP is calculated by trigonometry. This gives the angle between the \( y \)-axis of the monitor coordinate system and the line through the interaction position, and the projection
of the viewpoint onto this interaction plane. If no user can be tracked, the application creates an imaginary viewpoint, which is equal to the event position shifted by one pixel in positive y-direction. This always results in a rotational angle of zero degrees.

After acquiring and calculating all the necessary information, the application checks whether the event happened in or outside the notepad. If the event occurred outside, the notepad’s center point is moved to the event position and the coordinate system is rotated counterclockwise to the previously calculated rotational angle. Additionally, a mouse release event has to be generated in order to be able to draw again onto the notepad. Otherwise, one would unintentionally draw when the notepad is shifted to the click position. If the event occurred inside the notepad, it is not transformed in order to avoid jittering, but is interpreted as drawing an annotation on the notepad.

IV. SUMMARY AND FUTURE WORK

In this paper, we presented a tabletop system that is aware of the user’s position in front of the screen. The digital content (i.e. a notepad) is automatically oriented towards the active user, so that the user can read it from any position around the table. Hence, extending the interaction space from direct manipulation of digital objects on the interactive screen to the space above the tabletop tracking the user’s viewpoint enables multiple users to intuitively collaborate on horizontally oriented interactive surfaces.

Further, we determined a smoothing factor to stabilize the viewpoint tracking data without causing a noticeable time delay and defined how to determine the active user. Moreover, the alignment of camera and display is highly crucial for a precise mapping of the user’s viewpoint in the display coordinate system. Hence, an automatic calibration is necessary.

Future applications should also be capable of tracking the user’s hand location in order to identify the active user and detect hovering gestures.

REFERENCES


[18] [Online]. Available: http://qt-project.org/