

# Orientation as an additional User Interface in Mixed-Reality Environments

Mike Eißele

Simon Stegmaier

Daniel Weiskopf

Thomas Ertl

Institute of Visualization and Interactive Systems  
University of Stuttgart, Germany  
{eissele | stegmaier | weiskopf | ertl}@vis.uni-stuttgart.de

**Abstract:** This paper presents an augmented reality system that makes use of a consumer-level mobile device equipped with an inertial orientation sensor. The device maps orientational information to user interactions. Furthermore, we utilize orientation to determine portions of the operator's context. The system makes use of the location- and context-aware platform Nexus [HKL<sup>+</sup>99] to further refine the user's context information. To evaluate the acceptance of the presented system a user study was performed.

**Keywords:** Inertial Orientation Sensing, User Interface, Context-Aware, Augmented Reality

## 1 Introduction

Most augmented reality (AR) systems make use of advanced display and tracking technology. Optical or video see-through head-mounted displays and high precision external tracking systems are used to achieve optimal results in terms of precision, update rate, and quality. But on the other hand, the user acceptance of these systems is low as many of them are very expensive and uncomfortable to wear. Additionally, an inadequate user interface often hinders an easy operation.

Mobile devices keep getting smaller, faster, and cheaper, and additionally have more functionality embedded than any previous generation. This enables and encourages more and more people to use these devices. Therefore, our system is build on a consumer-level Tablet PC that was equipped with an orientation sensor shown in Figure 1. Another example where alternative consumer-level devices are used to build an AR system is the AR-PDA project [AP01].

With the use of an orientation sensor the system is able to provide an additional possibility to interact with the device in an intuitive way. It is shown how changes in orientation can be mapped to user interfaces of existing applications in a generic way. Furthermore, the system uses the Nexus platform [HKL<sup>+</sup>99, CKL00] to retrieve additional information of the current user's context. Nexus is a middleware that connects providers and clients of context-aware applications. The platform provides a detailed model of the real world to location-aware and context-aware applications that can be used

Figure 1: Tablet PC with a mounted inertial sensor.

in in-door and out-door scenarios [HKL<sup>+</sup>99]. Context-aware applications can easily query the state of real-world and virtual objects via a well-defined protocol. To find adequate mappings and parameters of the user's context to system interactions, several user interviews were evaluated during the implementation of the prototype. Furthermore, two additional example applications are shown where tilt operations are used as UI. The acceptance of the presented prototypes is summarized in a short overview of the feedback given by the interviewed users.

The remaining paper is organized as follows: First, we will give a summary of previous work done in the area of user interfaces in AR environments. Afterwards a short survey on inertial sensors is given. In the following two sections elaborate on the proposed concepts. The next section describes the prototype implementations, followed by the results and a conclusion.

## 2 Previous Work

The exploration of user-friendly and intuitive input interfaces has been going on since the invention of electronic devices, especially computers. Concepts that are easy to understand and handle found their way to nowadays common computer systems. User interfaces in augmented reality systems are mainly based on simple buttons of VR based input devices. Only few research is done in adapting concepts of ubiquitous computing interface techniques to augmented reality systems. Alternative input interfaces to navigate through a menu on a hand-held device is presented by J. Rekimoto [Re96] who uses orientation to select different menu options. B. L. Harrison et al. [HFG<sup>+</sup>98] show the benefit of utilizing different sensors to capture gestures of the operator. A similar setup is described by K. Hinckley et al. [HPSH00]. J. F. Bartlett [Ba00] shows an alternative input technique where the orientation of a hand-held device is used to scroll and navigate within an electronic photo album. Using device tilt gestures as a text input method is presented by K. Partridge et al. [PCS<sup>+</sup>02]. D.

Tan et al. [TPB<sup>+</sup>01] use some interaction techniques based on changing the orientation of augmented objects. Verplaetse presents a good survey on inertial proprioceptive devices [Ve96]. Schmidt et al. show a simple hand-held prototype equipped with two mercury switches to recognize the orientation of the device [SBG99]. A survey of context-aware research is presented by G. Chen and D. Kotz [CK00].

### 3 Inertial Orientation Sensing

The presented concepts for a user interface are based on the device's orientation. An orientation can be measured relative to the previous orientation or absolute to a given coordinate system. Relative orientation can be captured by inertial sensors that are based on, e.g. gyroscopes. For measuring the absolute orientation a coupling must be established between the device and the reference coordinate system. Using a high-precision tracking system calibrated in reference coordinates can directly provide the absolute orientation including the absolute position. These systems require massive external installation and are therefore only functional within a limited area. Compared to inertial sensor systems this is a major deficit. A more elegant method to capture orientation absolute to world coordinates is to combine an inertial orientation sensor with additional sensors. An accelerometer that can detect the steady 1G earth gravity vector describes the orientation relative to the earth surface. Combining an electronic compass with an orientation sensor helps to describe the direction in cardinal points. Although a compass itself would provide absolute directional information in most applications it is only used to reduce drifting and stabilize a differential orientation sensor. This combination helps to overcome the inertia of a compass and on the other hand reduces drifting of the inertial orientation sensor during motionless periods.

Most commercially available off-the-shelf inertial orientation sensors are designed for virtual reality (VR) applications and differ greatly in precision and price. We equipped our prototype with an InertiaCube<sup>2</sup> from InterSense [II03] which is a combined device of three gyroscope, three accelerometer, and three magnetic field sensors. The sensor is designed for high-precision head tracking in VR and has, therefore, a very small form factor—roughly 30mm<sup>3</sup>—and weights only 25g. It has a price of about \$1700. We also built a cheaper prototype orientation sensor with less precision but equal form factor, based on two accelerometers and a microcontroller with an investment of about \$25.

### 4 Mapping Orientation to User Interaction

The change of the device's orientation can be mapped directly to some common user interactions. The choice of this mapping greatly affects the acceptance of the system, because the operations performed when the orientation of the device is changed should always be obvious to the user. In general, there are three possibilities to map orientation: a discrete, a partially constant, and a continuous mapping. In each case the angle to the user's reference operating orientation, referred to as *tilt angle*, must be known as illustrated in Figure 2. The reference orientation, also referred to as reference

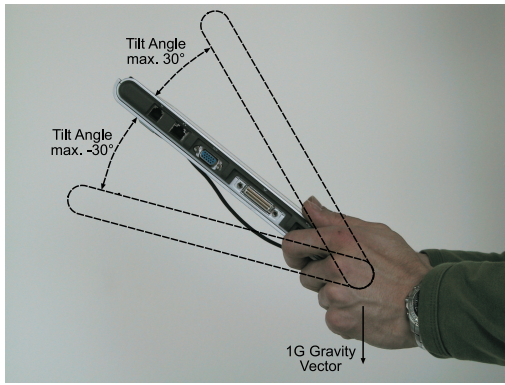


Figure 2: Maximum range of tilt operations.

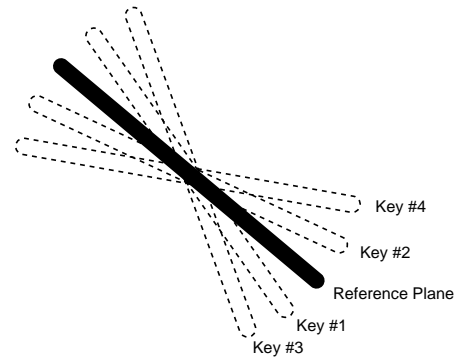


Figure 3: Defining different key events for different tilt angles.

plane, represents the default orientation of the device when the user does not apply any orientation changes for interaction. The setup of the reference plane should be accomplished by the user since it changes dependent on the user, his working position, and environmental lighting conditions. It is obvious that a reconfiguration becomes necessary very frequently as users tend to change their working position from time to time. Therefore, an adequate solution is to setup a single button to capture the current orientation as new reference plane. Another challenge for the system is to recognize when the user tilts the device but does not want to perform any user interaction. This happens in cases when the user stops working with the system and sets it down on a table or when the user changes his working position. The system will misinterpret this new orientation and will execute interaction events. To prevent this behavior the system somehow must recognize if the user tilts the device explicitly for user interaction. To solve this problem Rekimoto suggests to use a *clutch button* to activate the tilt functionality [Re96] which allows to use tilting for user interaction only during the button is pressed (*clutch mode*). Secondly, a *lock mode* can be integrated where the mapping of orientation to interaction can be turned on or off. The clutch mode offers the additional benefit that each time the button is pressed the reference plane could be readjusted to the current device orientation. Therefore, the reference plane adjustment is embedded within the activation process of the user interaction via tilt gestures. During our prototype development it turned out that users did not tilt the device for more than about 30 degrees, thus interaction techniques that require more extreme changes in orientation were neglected. On the other hand as users cannot hold the device without slight changes in orientation a *dead zone* is used to prevent execution of events at orientations only slightly different than the reference plane as can be seen in the Figure 4.

**Discrete mapping** uses a threshold on the tilt angle to recognize user interaction events. A typical example function to map the tilt angle to discrete user interaction events is shown in Figure 4, left. This technique can, e.g., be used to simulate single key-press events or toggling of application states.

**Partially constant mapping** can assign different events based on different tilt angles. To decide when to measure the tilt angle, a timer can be used to measure the time when the orientation does not change significantly. If this timer exceeds a user defined threshold it is assumed that the user positioned the device to its desired orientation and the system measures the tilt angle. Another possibility is to store the angle where the user stopped the tilt gesture and starts to change the orientation back to the reference plane. This tilt angle is then used to decide which action has to be executed by using

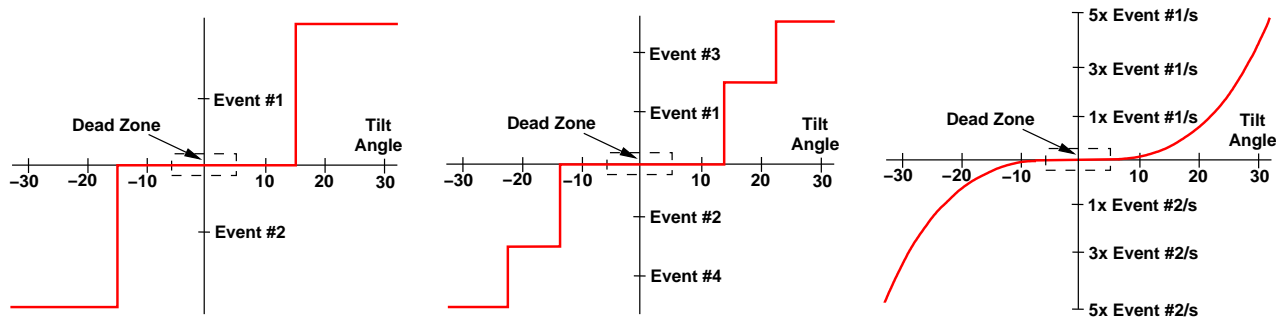


Figure 4: left: function to map discrete events, middle: function to map different events, right: function to map multiple events.

a mapping function as shown in Figure 4, middle. Figure 3 illustrates the device held at diverse tilt angles to trigger different events. This method can, e.g., generate a cursor down key-press event for slight tilt gestures and a page down key-press event for more serious changes in orientation.

**Continuous mapping** allows to generate multiple events within a time period. The number of events generated depends on the tilt angle. The more the device is tilted the more events are generated. The typical mapping function used in this scenario is illustrated in Figure 4, right. This mapping is useful to, e.g., choose elements within a list where the users can step through list elements or scroll window content, at varying speeds.

Application-dependent mappings of orientation to user interaction allows more sophisticated mapping techniques, especially if more information about the users current context is available. In particular, AR applications can benefit from more intuitive and supportive user interfaces as the users have to interact with the real and virtual world concurrently. But also the entertainment sector may use this kind of user input to enable games like the dexterity puzzle or the well-known computer game Marble Madness by Atari.

## 5 Select the Mapping of User Interaction via Context Information

The tasks that a user wants to perform are often context sensitive. Many work in this area contributes to the topic of location-aware or, in general, context-aware systems [BD03, CK00, SBG99, HKL<sup>+</sup>99]. As our proposed concept is orientation-aware with an inertial sensor and generally context-aware through the Nexus platform, context situations can be estimated based on many information, e.g., the application status, user interaction, device orientation, status of the environment, or location of the user.

Most mobile devices are designed to be operated not just when holding still but also while in movement. However, the precision of many input devices—like, e.g., touch-panel pens—suffers significantly from movement. Picking small graphical user interface (GUI) elements while standing still seems to be no problem for most users, but becomes a big challenge when walking. Sensing the orientation of the device with high precision allows to recognize the device’s operational status, i.e. if it is operated while moving or not. With this information the user interface can be modified to, e.g.,

include less functionality and larger control elements that are easier to tap. Furthermore, if the device experiences no movements at all—less than the natural shiver of a human hand—it must have been laid down. A device such as a mobile phone can then suspend most functions to save energy since, in general, it is operated in-hand. In contrast, if the sensor recognizes huge, rapid changes in orientation it can be assumed that the device is currently transported and that the heavy movements make it impossible to work with it. It can therefore just as well shut down and save power. Providing the possibility to sense an absolute orientation by adding a gravity sensor as described in Section 3 allows to add further functionality to the system. This simple enhancement allows to retrieve context information and support the user by, e.g., automatically changing the screen orientation from portrait to landscape and vice versa which is particularly useful on small-screen devices as smartphones or PDAs as shown by [SBG99].

All of the previously mentioned interaction techniques are possible by just capturing inertial orientation changes. Exploiting the possibilities of the Nexus platform enables the system to acquire much more context information. E.g., the user position in combination with the status of environmental lighting conditions can be used to adapt the background lighting or contrast of the display. Events that occur in the real world can influence the application, e.g., a ringing phone can also pause the application or mute the volume. But additional information for real and virtual objects can be accessed with Nexus and, consequently, orientation-based user interaction can be used to navigate through the available data.

## **6 Prototype Implementation**

To verify the proposed concepts of user interaction we developed an AR application that implements several methods to perform user interaction by orientation. To show that the concept of using orientation is not only limited to AR applications two other simple prototypes are presented: a generic orientation driver to control an existing window system and a simple computer game.

### **6.1 Augmented Reality Explorer**

AR systems require the device location and orientation in order to augment the virtual and real information. While sensing the orientation can be handled with inertial sensors, the absolute location cannot be determined easily and often requires costly additional hardware. Therefore, we implemented an AR explorer using the following assumptions: the users explore only single objects and maintain a fixed distance to them. Therefore, orientational information is enough to evaluate the position of the display relative to the object. A Tablet PC was equipped with the off-the-shelf orientation sensor InertiaCube<sup>2</sup> from Intersense [II03](see Figure 1). The prototype implementation displays a virtual object at the same location and orientation as the real-world object. The user can explore the object from different views without learning how to navigate, just by moving the display around the object as shown in Figure 5. Therefore, the AR explorer can be seen as a mobile AR window as often

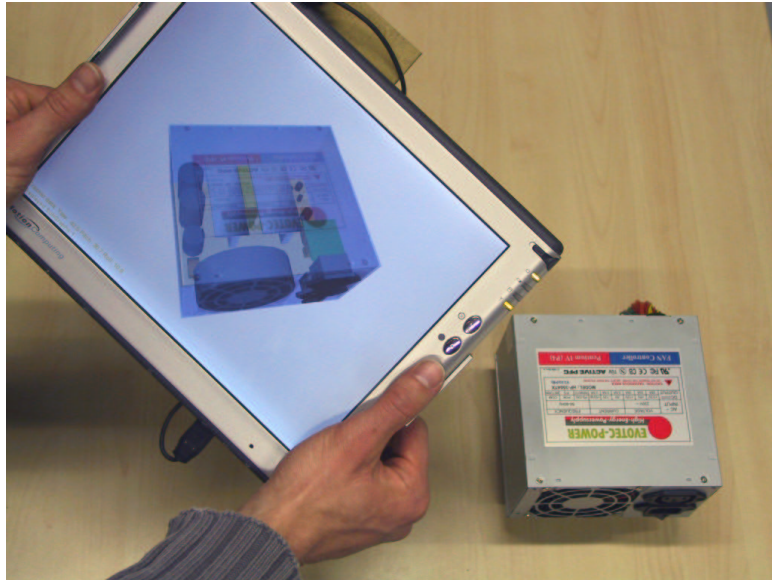


Figure 5: AR-Explorer to explore objects by orienting the display and switching irrelevant parts transparent.

used in medical applications [SSSW03].

The system has both operation modes implemented: the clutch mode and the lock mode. The user interface via orientation is divided into two parts: The orientation is permanently used to calculate the view direction to the exploration object and if the clutch button is pressed or the lock mode is activated, orientation is mapped to user interaction functions. The first part ensures that the orientation of the virtual object remains registered with the real-world object as can be seen in Figure 5. The mapping of tilt gestures to user interaction functionality allows the user to select different parts of the object with left or right tilt gestures and to modify them by tilting the device up or down. For the part selection we used a continuous mapping to allow the user a precise (*slow*) navigation with slight tilt operations and fast stepping through the parts using more extreme tilt gestures. To modify the selected part of the virtual object a partially constant mapping is used to apply four different functions to modify or interact with the selected part. Two events are used to switch parts to two different transparency levels. Furthermore, parts can be completely removed from the virtual object. At last, it is also possible to display additional data that is acquired from the Nexus platform. The application overlays an information window to visualize the data and enters the *information navigation mode* where changes of orientation only affect the navigation within the information view. This may contain, e.g., a technical manual, a user guide, or any other graphical/textual information. To store the information the hyper text transfer protocol (HTTP) is used. To navigate within HTTP documents with tilt operations we encountered the problem that the user wants to scroll the document and additionally wants to step through the included links. Naturally, both operations should be mapped to the same up or down tilt gesture which is, in general, not possible. An adequate solution is to scroll and step through the links simultaneously: When the user tilts the device the information display steps to the next link if and only if the link is contained in the currently visible area of the document. In all other cases the page is scrolled. This behavior can be further refined so that the system only steps to the next link if it is shown in a user defined area of the screen. The image sequence in Figure 6 shows successive tilt operations, the greenish area defines the region in which a link must be visible in order to be selected.

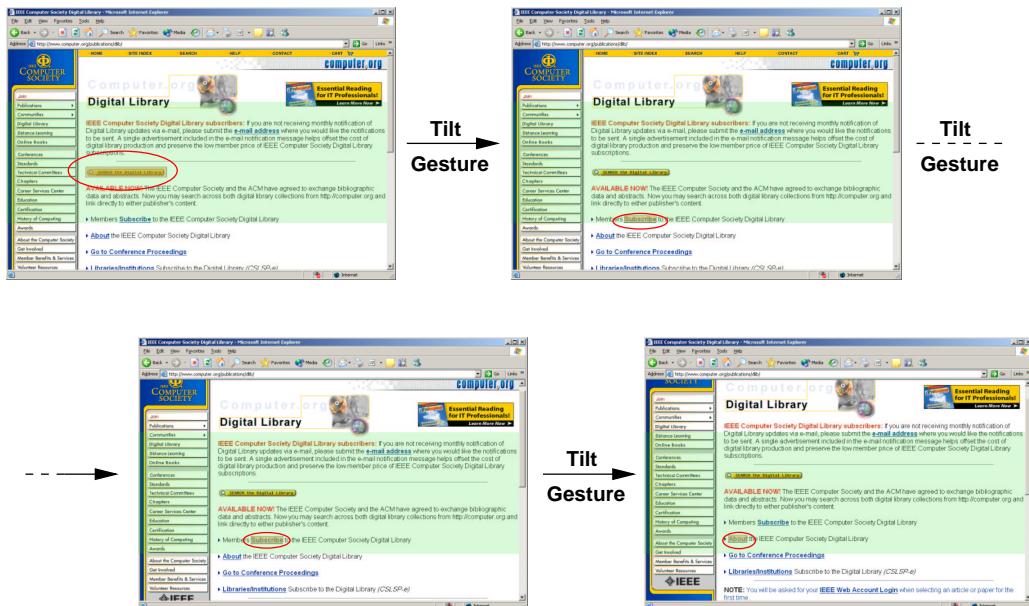


Figure 6: Tilt operations result in successive scroll and link-step actions. Initially, the first link is marked. After a tilt gesture the next link is selected. Tilting again scrolls since the following link is out of the greenish *link area*. A further tilt scrolls and additionally marks the last link.

Additionally, left tilting is discretely mapped to step back to the previous page of the displayed information and with right tilt operations it is possible to select links within the HTTP document. The information window is closed if the first information page is displayed and the user tries to go back to the previous page.

## 6.2 Generic Orientation Driver

For this prototype we used the same hardware setup as before (see Figure 1). The generic driver application has also both operation modes implemented. In the default configuration the orientation is mapped to the scrollbars using a continuous mapping to allow arbitrary scrolling speeds [Ba00, HFG<sup>+</sup>98]. Therefore, any application which uses scrollbars benefits from the new input technique. A schematic illustration of the algorithm is shown in Figure 7.

For applications not making use of scrollbars, input via tilt gestures is exploited by sending key-press events to the application. The prototype allows a per-application discrete key-mapping configuration. When an orientation change is recognized, the system searches the setup for the current window in focus. If no special configuration is found, the default behavior—mapping orientation to the scrollbars—is applied. This allows users to configure a wide range of applications to be operated via tilt operations.

The functionality to navigate within WWW pages via tilt operations—as described in Section 6.1—is also integrated and, therefore, the driver allows to browse the internet with just tilting operations.

The prototype can also detect the orientation relative to the 1G gravity vector which is used to implement an automatic screen configuration from landscape to portrait orientation and vice versa [SBG99].

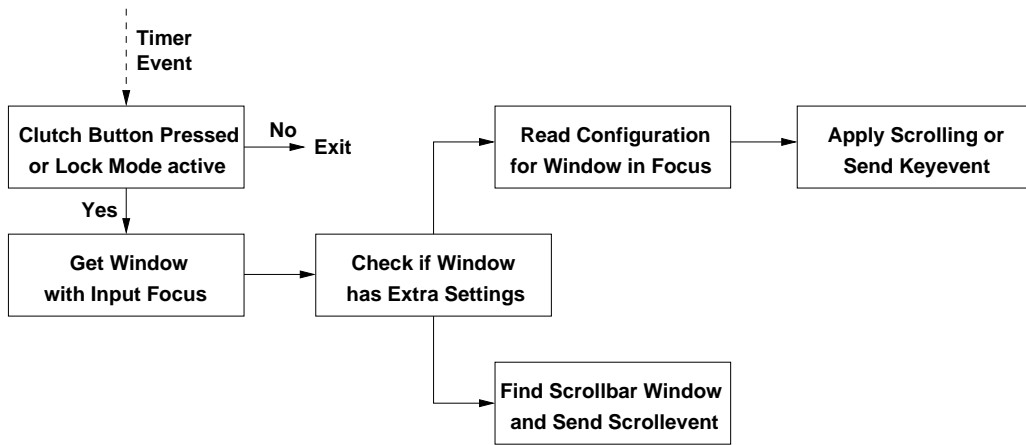


Figure 7: Algorithm to map orientation changes to scrolling or key press events.



Figure 8: Playing *Trackballs* (a *Marble Madness* clone) via tilting the device.

A threshold prevents screen reconfigurations if no significant tilt in either direction is recognized, e.g. if the device is lying on a table.

As a simple context-aware functionality the system detects if it is lying upside down, i.e. if the display is facing the ground. It is assumed that users do not operate the device in this position and, therefore, the system enters suspend mode.

### 6.3 Gaming Entertainment

Sensing the orientation of the device also gives rise to new applications and interaction techniques. In the entertainment area the orientation can also be used for games of many kinds as, e.g., racing games or simple dexterity-puzzle-like games. We adapted the open source game *Trackballs* [BRP<sup>+</sup>]<sup>+</sup>—a *Marble Madness* clone—using the device orientation for steering the ball (Figure 8).

	AR-Explorer View Orientation	AR-Explorer Part Selection	AR-Explorer Info. Browsing	Generic Driver Scrolling	Gaming Entertainment
Handling	+	+	−	+	+
Advantage	(++)	+	−	0	+
Precision	0	+	0	+	++
Intuitively Usable	++	++	−	++	++

Table 1: User Feedback on Prototypes.

## 7 Prototype Evaluation and Results

To evaluate the prototype applications a short user poll was performed. The users were given a short introduction to our orientation-aware system. Thereafter, the users operated the applications and had to rank the application’s user interface on several aspects. The scale of the rating at each aspect reached from minus two up to plus two. Seven users participated in our study, for all of them the concept of operating a user interface by changes in orientation was totally new. The user feedback on how comfortable the application can be operated is expressed in *Handling*. The *Advantage* item ranks the benefit of orientational input versus common input techniques. If the new input technique provides enough precision is expressed in *Precision*. Finally, in *Intuitively Usable* the users had to rate how intuitive the interface can be operated. The results are summarized in Table 1.

Some characteristics of the orientation-aware user interface could be seen in all interactions that were performed. Due to the device configuration the clutch mode can hardly be operated in portrait orientation as the button can no longer be used ergonomically (see Figure 1). But this problem is even more severe if common user interface techniques are used where more buttons have to be operated to perform the same interaction. Harrison et al. addressed this problem in [HFG<sup>+</sup>98]. Operating the device with the stylus while the users held the device with one hand was practically impossible. The weight of the device (approx. 1.4kg) is far too heavy to be handled ergonomically with only one hand. Additionally, the users found it quite hard to operate the stylus while moving.

**AR-Explorer View Orientation.** To evaluate the view manipulation of the AR-Explorer the users had to explore various parts of the examination object. All users intuitively used the Tablet PC as a window and could easily perform the task. Two users claimed the weight of the device restricts the handling. The advantage of using orientational information to determine the view direction versus defining it manually, e.g., via cursor keys, is huge. Nevertheless, compared to other AR window devices there is even a slight disadvantage, especially in terms of precision and drift of the inertial sensor.

**AR-Explorer Part Selection.** For selecting subparts of the exploration objects most users chose the clutch mode as they sometimes walked around the object to get a better view of the currently selected part and, therefore, had to deactivate the the user interaction. Four users preferred the user interface

using left/right cursor keys to select parts. During the selection of parts more extreme tilt gestures can be used to step through the list of parts very fast.

**AR-Explorer Information Browsing.** The users were able to scroll the information window content intuitively. But many users had problems selecting/following links and afterwards returning to the previous page. All of the users needed further explanation to understand the concept. Five users claimed that the interaction method is non-intuitive and too complicated to be operated easily.

**Generic Driver Scrolling.** To prove the presented mappings the scenario when reading a long (approx. 8 screens) web page was analyzed. Most users used the lock mode to read the web page, setting the reference plane in a way that the window content scrolled at a speed corresponding to their reading speed. Two participants used the clutch mode. They scrolled step-by-step each time by pressing the clutch button, scrolling, and releasing the button.

**Gaming Entertainment.** All users naturally interacted with the device and tried to keep the ball on its track. The users in our study were greatly amused by its simple and intuitive handling.

The study points out that a UI operated by orientation is in some scenarios a practical interaction technique. For some interaction tasks it is even preferred compared to common interfaces, e.g., a stylus. In addition, for more sophisticated user interaction there is still only one button to be pressed whereas using common user interfaces the user has to remember the functionality and placement of all used buttons. Another problem is that there is only limited space for buttons on a device. The study also showed that both operation modes have their advantages and disadvantages. The clutch mode is easy to handle and misinterpreted user interaction occurs seldom. The lock mode is preferred when the user continuously uses the user interface, e.g., while reading a text. In contrast, the poll also clearly states that interaction via tilt operations is not yet perfect in all applications. Navigating within HTTP documents using tilt gestures was found still not good enough to be used in practice by most users.

## 8 Conclusion

We have shown an alternative user interface for mobile devices in AR environments. Orientational information is mapped to several user interface functions to control the application via simple tilt gestures. The presented prototype is also able to recognize some states of the device's context to support the user. The performed user poll to evaluate our proposed method shows that some applications benefit from a context-aware user interface, especially applications running in AR environments. Nevertheless, the evaluation also showed that for some cases the presented user interaction techniques are inadequate and, therefore, additional concepts have to be found. In future we would like to add more sensor technologies to our prototype to develop more reliable context recognition techniques.

## 9 ACKNOWLEDGMENTS

This project is supported as a part of Special Research Field 627 (Nexus) by the German Research Foundation.

## 10 References

- [AP01] AR-PDA. The ar-pda digital mobile assistant for vr/ar-content project. <http://www.ar-pda.de/>. 2001.
- [Ba00] Bartlett, J. F.: Rock 'n' scroll is here to stay. *IEEE Computer Graphics and Application*. 20(3):40–45. 2000.
- [BD03] Barkhuus, L. und Dey, A.: Is Context-Aware Computing Taking Control away from the User? Three Levels of Interactivity Examined. In: *UbiComp 2003: Proceedings of the 5th International Conference on Ubiquitous Computing*. S. 149–156. Springer-Verlag. 2003.
- [BRP<sup>+</sup>] Broxvall, M., Radel, D., Perret, Y., Krueckel, P., Listopad, S., und Pollak, A. Trackballs. <http://trackballs.sourceforge.net/>.
- [CK00] Chen, G. und Kotz, D.: A Survey of Context-Aware Mobile Computing Research. Technical Report TR2000-381. Department of Computer Science, Dartmouth College. 2000.
- [CKL00] Coschurba, P., Kubach, U., und Leonhardi, A.: Research issues in developing a platform for spatial-aware applications. In: *Proceedings of the 9th workshop on ACM SIGOPS European workshop*. S. 153–158. ACM Press. 2000.
- [HFG<sup>+</sup>98] Harrison, B. L., Fishkin, K. P., Gujar, A., Mochon, C., und Want, R.: Squeeze Me, Hold Me, Tilt Me! An Exploration of Manipulative User Interfaces. In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. S. 17–24. ACM Press. 1998.
- [HKL<sup>+</sup>99] Hohl, F., Kubach, U., Leonhardi, A., Rothermel, K., und Schwehm, M.: Next century challenges: Nexus - an open global infrastructure for spatial-aware applications. In: *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom99)*. S. 249–255. Seattle, Washington. 1999. ACM Press.
- [HPSH00] Hinckley, K., Pierce, J., Sinclair, M., und Horvitz, E.: Sensing Techniques for Mobile Interaction. In: *UIST 2000: Proceedings of the 13th Annual ACM Symposium on User Interface Software and Technology*. S. 91–100. ACM Press. 2000.
- [II03] InterSense Inc. InterSense InertiaCube<sup>2</sup> Specifications. <http://www.intersense.com/products/prec/ic2/InertiaCube2.pdf>. 2003.
- [PCS<sup>+</sup>02] Partridge, K., Chatterjee, S., Sazawal, V., Borriello, G., und Want, R.: TiltType: Accelerometer-Supported Text Entry for Very Small Devices. In: *UIST 2002: Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology*. S. 201–204. ACM Press. 2002.
- [Re96] Rekimoto, J.: Tilting Operations for Small Screen Interfaces. In: *UIST 1996: Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*. S. 167–168. 1996.
- [SBG99] Schmidt, A., Beigl, M., und Gellersen, H.-W.: There is more to Context than Location. *Computers and Graphics*. 23(6):893–901. 1999.
- [SSSW03] Schnaider, M., Schwald, B., Seibert, H., und Weller, T.: Medarpa - a medical augmented reality system for minimal-invasive interventions. In: *Medicine Meets Virtual Reality 2003. Proceedings : NextMed: Health Horizon*. S. 312–314. Newport Beach, California. 2003.
- [TPB<sup>+</sup>01] Tan, D., Poupyrev, I., Billinghamurst, M., Kato, H., Regenbrecht, H., und Tetsutani, N.: On-demand, in-place help for augmented reality environments. In: *UbiComp 2001, Short Paper*. Atlanta, GA. 2001.
- [Ve96] Verplaetse, C.: Inertial proprioceptive devices: Self-motion-sensing toys and tools. *IBM Systems Journal*. 35(3-4):639–650. 1996.