Marking Menus for Eyes-Free Interaction Using Smart Phones and Tablets

Jens Bauer, Achim Ebert, Oliver Kreylos, Bernd Hamann

Marking Menus for Eyes-free Interaction Using Smart Phones and Tablets

Jens Bauer[1], Achim Ebert[1], Oliver Kreylos[2], and Bernd Hamann[2]

[1] Computer Graphics and HCI Lab, TU Kaiserslautern, 67663 Kaiserslautern, Germany,
j_bauer@cs.uni-kl.de
[2] Institute for Data Analysis and Visualization, Department of Computer Science, University of California, Davis, CA 95616, U.S.A.

Abstract. Large displays are helpful tools for knowledge discovery applications. The increased screen real estate allows for more data to be shown at once. In some cases using virtual reality visualizations helps in creating more useful visualizations. In such settings, traditional input devices are not well-suited. They also do not scale well to multiple users, effectively limiting collaborative knowledge discovery scenarios. Smart phones and tablet computers are becoming increasingly ubiquitous and powerful, even having multi-core CPUs and dedicated Graphic Processing Units (GPUs). Given their built-in sensors they can serve as replacements for currently-used input devices, and provide novel functionality not achieved with traditional devices. Furthermore, their ubiquity ensures that they scale well to multi-user environments, where users can use their own devices. We present an application-independent way to integrate smart phones and tablets into knowledge discovery applications as input devices with additional functionality. This approach is based on Marking Menus, but extends the basic idea by employing the special capabilities of current consumer-level smart phones and tablets.

Keywords: input devices, collaborative interaction, multimodal interaction, 3D Interaction

1 Introduction

Smart phones and tablet computers are becoming increasingly popular and powerful. Current consumer-level devices feature multi-core CPUs and dedicated GPUs. They contain multi-touch screens, GPS receivers, compasses, and accelerometers, and offer connectivity via WiFi, bluetooth, and 3G technologies. Given those capabilities, smart phones and tablets are attractive alternatives to "traditional" input devices. Smart phones and tablets offer several benefits, due to their ubiquity and added functionality compared to typical input devices.

As these devices are multi-functional and wide-spread, most users of knowledge discovery applications already own at least one, which precludes the need to buy often expensive single-purpose input devices. The same holds for multi-user

environments, where each user can use his/her own device to control the system. Due to the devices' portability, they could potentially be used as (limited) "mobile memory" to transfer data between environments or as a kind of token to identify users.

Non-trivial applications typically provide the user with a moderate to large number of functions, which need to be mapped to the set of available input devices. Normally, each function is mapped to one device button, or gesture, or user interface (UI) button displayed. Problems arise when the number of required functions exceeds the number of input device buttons or gestures, or when the number of displayed UI buttons clutters the display. The touch-capable screens of current mobile devices, on the other hand, provide enough screen real estate to offer a large number of buttons and mapped functions – significantly more than practical for traditional input devices. However, the naïve approach of using mobile devices disrupts users' workflows, as they have to shift their attention back and forth between the devices' small screens and the main display environment.

We propose to utilize *Marking Menus* [16], a radial menu structure, as a central element of a novel interaction method employing the multi-touch screen of mobile devices. The touch screen is used to show (hierarchical) radial menus as they pop up. This enables eyes-free interaction for experienced users who do not need the visual feedback from the mobile device, and leads to increased efficiency for those users, while at the same time keeping the menu structures visible should they be needed. This selection method is extended by the usage of tracking sensors, accelerometers, multi-touch and/or in-menu slider controls to provide a larger variety of interaction possibilities, which are explained in detail later in this paper.

The advantages of this design over existing interaction methods are:

- Eyes-free interaction makes complex user interaction possible without interrupting the workflow.
- Auditory or haptic feedback is given to support eyes-free interaction even more.
- The general approach allows application of the design to a wide array of new and existing applications.
- The ubiquity of smart phones and tablets and their usability for other tasks make this approach very cost-effective.
- As the system uses its own mobile screen it can replace large parts of the application's graphical user interface (GUI) up to the complete GUI in some cases.
- Support for multiple users and the system scales well to the number of users.

We present the design and implementation of our prototype. A formal user study is beyond the scope of this paper, but is planned as part of our future work. Studies by Kurtenbach et al. [17] are applicable to this interaction method and show its general usability.

2 Related Work

2.1 Remote Interaction Using Mobile Phones

The use of mobile or smart phones as input device has been an active area of research. Ballagas et al. [3] presented two interaction techniques for camera-equipped phones. One uses the phone as a replacement for an optical mouse, with the physical x-y-translation of the phone mapped to a traditional cursor on the screen. At the time, the method incurred a latency of about 200 ms, too much for a practical applications; even today, high latency seems to be an issue for such approaches. This approach also does not increase the number of functions that can be mapped to a device. The second technique in the same paper is a pure selection method, where the phone detects at which object on the screen it is pointed by tracking markers displayed on the screen. These markers only appear for brief moments while the phone's camera is active to reduce display clutter. Both approaches only employ camera phones as direct mouse replacements, missing the opportunity to further improve the available interaction.

Extending that work, Jeon et al. [13] proposed interaction techniques to replace the computer mouse with mobile phones. The movement of the phone in the air is mapped either to a cursor, or directly to an interactive object. Three approaches of calculating the relative movement of the phone are described: motion-flow, marker tracking with the marker on an selectable object, and marker tracking with a free-floating marker. While this is more refined than Ballagas' et al.'s approach, it still suffers from the same basic limitations.

A specialized approach is presented by Madhavapeddy et al. [19]. A camera-equipped phone is used to control a flight booking application, where the starting and destination airports are selected by pointing at them with the phone, again using on-screen tracking markers. A similar approach was used by Thelen et al. [23] to interact with a 3D representation of the human brain. Here, 2D markers are rendered as part of a 3D model of the human brain, and scanning one of those markers with a smart phone displays additional information associated with that marker. Both are examples of usable designs, but have the drawback of being very specific to a certain kind of task, i. e., selection from a small number of preset targets.

2.2 Marking Menus

Kurtenbach et al. [16, 17] proposed and evaluated *Marking Menus* as an improved version of radial menus. The main difference between Marking Menus and transitional radial menus is the absence of a completely bounded target area for each menu item in the former. Radial menus simply arrange their items in a circular pattern around a center point in one or more "rings." An item is selected when the selection cursor is within the bounds of the item, and selection is usually affirmed by a button press on the input device (or any other available confirmation gesture). If the selected menu item has subitems, this causes

a new radial menu to pop up at or near the current cursor position. Marking Menus only have a single ring of items, and their respective target areas expand infinitely outwards from the center of the menu in a wedge-like shape. Holding the cursor still in the selection area of an item with sub-items pops up another Marking Menu showing the sub-items, and a confirmation event (button press, or, in the original example, lifting the pen from the display surface) selects the current item. The main benefit of this menu layout and selection mechanism is eyes-free item selection, which was at the time proposed to address the high latency of pen-based direct interaction displays on then-current workstations. Using Marking Menus, users could either put the pen down on the screen, wait for the menu to appear, and select items and sub-items by drawing a stroke from the center into the (wedge-shaped) selection areas, while experienced users could just draw the chain of strokes used to reach a certain item without even waiting for the menu to pop up. (Kurtenbach called this "Expert Mode".) Since the selection areas are theoretically infinitely large, user accuracy is very high and Marking Menus are very easy to use. These assertions are backed up by Fitts' Law [5] and Steering Law [1]. While the original high-latency problem Marking Menus were designed to address no longer matters, their effectiveness still does, and Marking Menus have been incorporated into many modern applications.

Pook et al. [22] proposed *Control Menus*, a general improvement that also applies to Marking Menus: instead of using menus only for selections, the continued motion of an input device after an item has been selected is used to change a continuous value associated with that item. They use an example of a "zoom" menu item, where any continued input device motion after selecting that item directly alters the current zoom level. This method could also be applied to to related continuous values by using both display axes simultaneously.

Perlin's *QuikWrite* [21] is a stroke-based text entry method. It lays out the available character set along the edge of a rectangle (usually covering the entire screen), and divides them into eight zones (N, NE, E, etc.). A character is selected by first moving the input device from the rectangle's center into a zone, then optionally into a second zone, and finally back to the center. For example, the top-left (NW) character of the top-right (NE) zone is selected by first moving into the NE zone, then into the NW zone, and then back to the center. QuikWrite allows to enter entire phrases of text with a single, continuous stroke of the input device.

Other touch text input methods include *Cirrin* [20] and *T-Cube* [24]. The former selects characters by touching them with a stylus and allows multiple selections with one single stroke, while the latter is a radial menu where the characters are arranged in multiple circles, so the direction and the length of a stroke together are responsible for character selection. Among these methods, only Quikwriting can seamlessly be used in a Marking Menu. Holzinger et al.[11] propose to use a stylus to aid handwriting on touch surfaces. This could also be integrated into a Marking Menu.

Guimbretière and Winograd [9] presented *FlowMenus*, a menu system designed for pens on large touch-sensitive displays. FlowMenus work like normal

Marking Menus for the first selection step, but selection in submenus of any level is based on curved stroke gestures back to the submenu's center: sloped clockwise, sloped counter-clockwise, and straight, supporting only three items per submenu. The advantage of this method is that the pen always returns to the original selected position, and therefore allows direct interaction with an on-screen element at that position after an item has been selected. However, in Expert Mode, it can be difficult to return precisely to the menu center, and small deviations may add up to a point where users might accidentally select a wrong menu item. FlowMenus also allow text input directly chained into the menu using QuikWrite, and value selection similar to Control Menus. But since submenu selection strokes need to curve back to the submenu's center, values now need to be selected based on the length of the curved stroke, which is less intuitive. This also shows a subtle menu design limitation of FlowMenus: menu items for direct object interaction and for text input need to be on the center of the menu structure and thus on an even level of the submenu hierarchy, while value-selection items need to be on an odd level for the same reason.

More new menu designs based on Marking Menus emerged recently. Bailly and Lecolinet [2] allowed curved strokes for menu item selection, resulting in so-called "Flower Menus." Depending on the initial stroke direction, the curvature and the curve direction, i. e., clockwise or counter-clockwise, different items can be selected. While this allows a higher density of menu items per submenu level, the total time needed to draw the curved strokes is notably higher than drawing just straight lines. Using a similar, but unrelated, design to Marking Menus, *Wavelet Menus* [7, 6] are also controlled by straight strokes. The menu is radial and the menu grows in radius whenever a submenu is opened and placed on. New submenus are positioned innermost. The main difference to Marking Menus is that the user has to release the touch after each level of the submenu hierarchy and start each new stroke in the center of the screen.

A first attempt to improve Marking Menus with multi-touch was presented by Lepinski and Grossman [18]. Their design uses chording gestures to invoke up to 31 different menus, based on which fingers of a hand are touching the screen. However, due to technical limitations, this approach requires the user to put down all five fingers first, and then lift a subset of fingers afterwards to invoke a menu. This is awkward, causes unnecessary delay, leads to selection mistakes if users lift fingers faster than the system can detect their gestures, and prohibits other simultaneous uses of multi-touch. In the near future these technical limitations might no longer apply to large touch sensitive surfaces, but consumer level smart phones and tablets will probably lack the feature of tracking non-touching fingers for still quite some time.

Kin and Hartmann [14] studied user experience with two-handed Marking Menus. They proposed either to split the menus into parts for each hand, and let the user select items either simultaneously or sequentially, potentially increasing efficiency. Their method requires clutching with the fingers to select items deeper in the menu hierarchy as one stroke has to be completed by lifting the finger

from the surface in order to access the next level of menu items, and does not allow multi-finger gestures.

Heidrich et al. [10] used an approach similar to Marking Menus to control a Smart Home. The gestures are performed on a table monitored by a Kinect and the menu is projected on the table. While the basic approach is the same as described in this paper, Heidrichet al.'s method does not allow multi-touch or value selection.

3 Design

Smart phones and tablets are not only usable as input devices, but also have output capabilities. When designing a user interface, one has to decide whether, and how, to use those capabilities. A device's screen, typically the primary output channel, can be used in a variety of roles: as a primary screen for a focus-and-context displays [4]; as a full additional screen, as in Air Display[3]; to clone the environment's main display, as in remote access applications such as Remote Desktop Protocol (RDP) or Virtual Network Computing (VNC); or even not at all, as in mouse control applications such as Remote Mouse[4]. One reason not to use a device's screen is that having to shift focus between multiple unrelated screens might interrupt a user's workflow.

Our approach uses touch screens for interaction feedback via radial menus similar to Marking Menus, described in Section 2.2. Kurtenbach 2.2 categorizes the usage modes of Marking Menus as Novice Mode (waiting for menus to pop up) and Expert Mode (completing interaction before or while a menu pops up). In the remainder of this paper, we refer to Expert Mode as eyes-free mode, to emphasize the fact that it does not require shifting attention away from an environment's main display.

3.1 Menu Design

A radial menu is presented to the user when they touch the screen (see Fig. 1). Its design is similar to the original Marking Menu. The menu is centered around the initial finger position, and menu items are selected as the finger enters their selection area. If the selected item has sub-items, a submenu pops up on selection (see Fig. 2). Unlike regular radial menus, the submenus have an additional wedge-shaped *dead zone*, i. e., a zone where no selection is made, along the line from the center of the current submenu through the center of its parent menu. This dead zone improves usability in eyes-free mode: when not looking at the screen while interacting, users run the risk of making shorter or longer finger movements than intended. To account for that, submenus initially move with the user's finger until the movement changes direction. This ensures that stroke length does not influence item selection.

[3] http://www.avatron.com
[4] http://www.remotemouse.net/

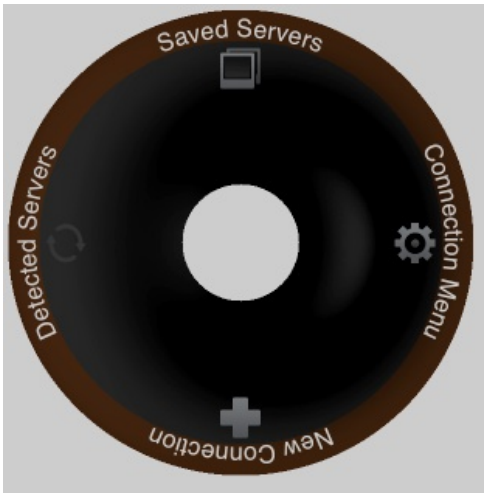

Fig. 1. The general design of the menu, in this example for four menu items, all with a descriptive name and an icon.

The wedge-shaped dead zone has the additional benefit of allowing to undo selections, as users can backtrack their strokes through multiple levels of the submenu hierarchy, including canceling menu invocation entirely. This is an extension of the original Marking Menu method. To support eyes-free interaction, the phone can provide auditory or haptic feedback whenever the user tracks back one submenu level.

To avoid unintentional selections caused by touch screen jitter, another circular dead zone has to be defined in the center of the menu. This dead zone should be as small as possible, as it defines the minimal length of a stroke to be recognized. The dead zone additionally improves selection accuracy, as the length of a stroke determines the accuracy of measuring that stroke's angle. Due to the rather coarse resolution of current-generation touch screens, a very short stroke may only be a few pixels long, resulting in inaccuracy when detecting the angle of that stroke. Thus, the size of the central dead zone is a device- and user-dependent configuration parameter.

To reduce display clutter in deep submenu hierarchies, where submenus are drawn on top of their respective parents, only the currently active submenu is drawn fully opaque, while parent submenus are drawn with increasing levels of transparency, i. e., the root menu is most transparent. Drawing the entire menu hierarchy in this way helps users to see their current position in the menu hierarchy and the direction of the most recent stroke, should they lose their place during eyes-free interaction. It also provides valuable feedback for the multi-touch interaction described in Section 3.3.

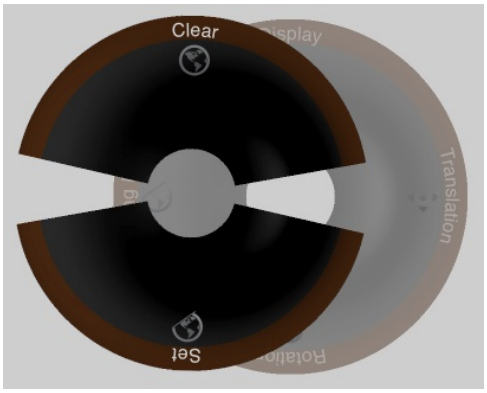

Fig. 2. The menu with a submenu opened up after the user moved their finger to the left. The submenu is on top of the half-transparent parent menu. Note the dead zone through in sub-menu to improve usability in eyes-free mode and to allow backtracking of selections.

3.2 Value Control

Our menu design also supports value selection similarly to Control Menus, as described in Section 2.2. Menu items with associated values are indicated by arrows (see Fig. 3). But instead of only allowing simple slider-like control over one or two dimensions, our design allows two modes of operation. In absolute mode it works like a regular slider control, reporting the absolute position of the touch(es) to the application. In relative mode, the actual location of the touch(es) are irrelevant. Instead only the changes in position are reported to the application. Absolute mode is useful for any fixed range of continues values, i. e., whenever a traditional slider control is useful. Relative mode is for situations where the absolute value of a variable is of less interest than the actual change to it. A common example for this is the position of an object. The user is normally more interested in moving the object a certain amount of units, instead of setting it to an absolute value. This relieves the application of keeping track of an absolute value (i. e., the position of the slider) that is of little interest.

For many visualization applications 1D or even 2D positioning is not enough. Employing the well-known multi-touch gestures of pinch and rotate, four degrees-of-freedom (DOF) can be controlled at once (x-Direction, y-Direction, Pinch and Rotation). This allows for simultaneous two-DOF-Movement, Zoom and one-DOF-Rotation, for example, to allow the user to drill down into a part of the visualized data without having to apply a new menu selection. Alternatively multiple quantifications can be made at the same time, with each finger touching the screen functioning as a separate one-DOF slider control, theoretically allowing for a 10-DOF control. The practical limit depends on actual touch-screen size, the size of the user's fingers and the user's dexterity. Most people should be able to use four sliders simultaneously without serious problems. The control provided might be too coarse or too fine. Therefore the control itself can

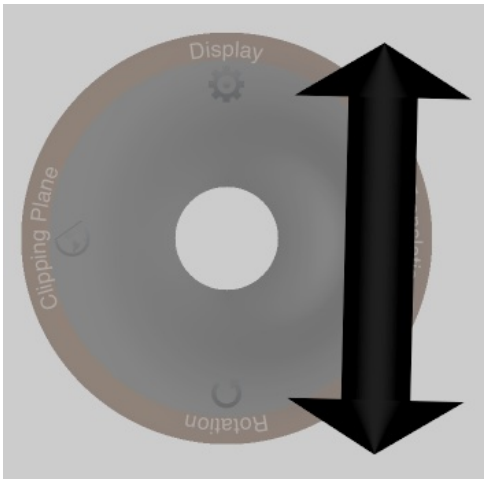

Fig. 3. An arrow shows up when the selected submenu supports direct value selection, just like a regular slider.

be scaled using the pinch gesture with a second finger (one-DOF or two-DOF controls) or a third finger (three-DOF and four-DOF). The scaling is visualized by scaling the control on the touchscreen accordingly.

3.3 Multi-touch Capabilitiy

In our method, multi-touch is not only used for value control, but also for normal menu navigation. As shown in Fig. 4, putting down another finger on the touch screen while a menu item with sub-items is selected causes the submenu for that item to detach from its parent menu, to move to the new finger's position, and subsequently to be controlled by the new finger. The parent menu sticks to the original finger, and move with it, but is otherwise be locked as long as a detached submenu is active. If, on the other hand, the currently active menu item is a value selection item, then the new finger will invoke multi-DOF value selection as described in Section 3.2. If the currently selected item is a regular menu item, then each additional finger will cause a selection event for that menu item, enabling rapid multiple selection of the same menu item by repeated tapping with an additional finger.

Detachable submenus are useful when stroke navigation through a deep menu hierarchy reaches the edge of the touch screen at any point. In that case, the user can place a second finger on the opposite side of the touch screen, and continue with menu selection normally. To prevent awkward positions, the initial finger can be released once the second finger starts interacting with the menu hierarchy. If, however, the first finger is kept held down, then only the detached part of the menu structure, and not the entire hierarchy, is dismissed after a successful selection. This enables a shortcut for multiple subsequent selections that have

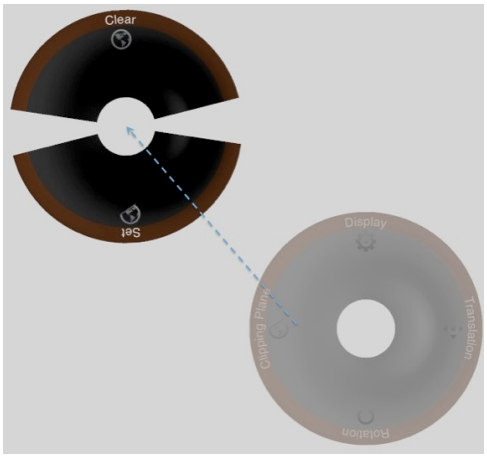

Fig. 4. Multi-Touch enabled navigation allows the user to detach a submenu and control it with another finger while the original menu still sticks to the old touch.

the same prefix in the menu hierarchy. For example, if the user wants to execute *left*, *up*, *left* followed by *left*, *up*, *right*, it will be possible to stroke *left*, *up* first, then perform *left* using a second finger, causing the menu at the first two parts of the stroke to be still open after the selection. With either the original or an additional touch the *right* stroke can be performed. A similar example is depicted in Fig. 5. Such shortcuts becomes more useful with deeper menu hierarchies. Very experienced "power users" can push the paradigm by using more than two fingers, detaching multiple submenus at the same time. With properly designed deep menu hierarchies, such expert shortcuts, while having a relatively steep learning curve, can lead to highly efficient interaction sequences.

The methods described up to this point support selection, quantification, an approximation of positioning and orientation via sequences of quantification, path via sequences of approximated position and orientation, and text when combined with QuikWrite or a comparable approach. True 3D interaction, however, requires at least 6 degrees of freedom, and single- or multi-touch gestures are not an intuitive replacement. While it is possible to chain two individual three-DOF interactions together to achieve six-DOF, this is difficult to handle for the user. Also it is only possible for two-handed interactions, and users should be able to choose one-handed vs. two-handed usage freely for themselves. Another way is to employ the smartphones build-in accelerometers to deliver the current orientation of the phone as additional information for each menu. As orientation provides three DOF (pitch, yaw and roll), seven DOF can be achieved through the combination of multi-touch quantification and orientation. Since 4 of these 7 DOFs concern rotation, a more practical limit is six DOF though. The quantification control already described can transmit the current orientation of the device as an additional 3 quantifications. This can be combined into free six-DOF-movement, for example, by letting the position of the user's touches

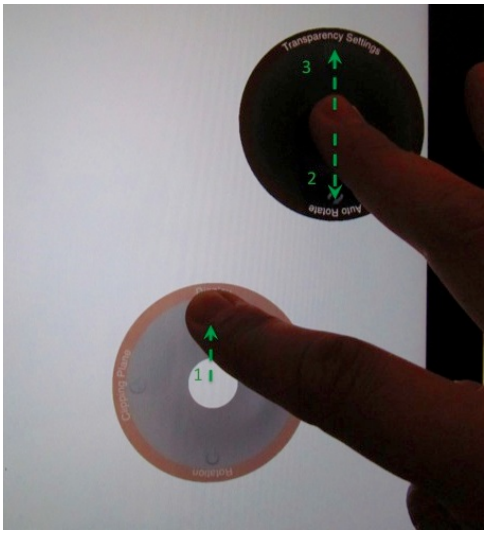

Fig. 5. Using detached menus as shortcut: The index finger touches the surface and is swept up (1). The middle finger swipes down causing the submenu to detach and toggle the auto-rotate function (2). The middle finger touches the surface another time, swiping up to activate the transparency settings (3).

control the x and y position, pinching of the fingers controls the z position (or zoom) and the device rotation can be transferred to the object being moved.

To support clutching (i.e., repeated swipes on the screen to control the same value) and to offer quick access to the last used function, the last invoked action can be part of the menu or can be activated by the use of multiple touches. For example a menu can be designed in a way that a single touch with a swipe to the left and then up lets the user control the position of some object. If the user is not satisfied with the object's position afterwards, a simple touch with two fingers will allow one to refine the positioning without having to swipe left and up again.

3.4 Hardware Requirements

Our design can be implemented on most current consumer-level smart phones and tablets. More specifically, target devices need to support multi-touch, need to have at least accelerometers for six-DOF interactions, and WiFi for communication. Other factors such as screen size or weight influence usability, but are not technically limiting.

4 Implementation

Unlike most other approaches, our proposed interaction method is neither specifically tailored towards, nor implemented in, a single application. It is designed

as a toolset for application or interaction designers wanting to include smart phone or tablet-based interactions in their applications. It can be included into a program as a library or plug-in. In our test implementation it has been added to the Vrui toolkit[5] [15] as a third party plug-in for and it can easily be included in any Vrui-based VR application.

4.1 Mobile Device

The prototype device-side application was developed in Objective-C for an Apple iPhone 4S and an iPad 2 (See [12] for potential problems when porting this to other platforms).

The device-side application implements the client component of the distributed architecture. On startup, it connects to an application-side server, found either via Service Discovery, or a manually entered host name. Upon connection, the client receives the application's menu structure and builds the menu's visual representation. If the application side requests orientation measurements, the client sends streaming orientation data at the maximal rate of 30 Hz to minimze lag. User interface events such as selection or quantification are reported asynchronously as they happen.

4.2 Extension Possibilities

The communication protocol is extensible and can be tailored toward a specific application should the need arise. Extensions are simple to implement on both sides and can make use of the already implemented features.

5 Conclusions

We have presented our eyes-free interaction method using consumer level smart phones and tablets. This method can be used effectively with applications for knowledge discovery and employs multi-touch and the phones sensors to allow full 3D interaction. Being eyes-free users can interact without interrupting their workflow to look at the input device. With its simple consistent design it can replace traditional menus and even whole dialog boxes. Our method is application-independent and can be used in other environments (e.g., Desktop) as well. Applications do not have to be altered to make use of this approach, but the prototype can be customized to include application-specific behavior.

A possible extension is device-based authentication. Devices can send public keys to the server on connection. Instead of a passphrase, the menu can be used to enter a combination of strokes serving as a password (similar to YAGP [8]). To prevent password sniffing, the communication can be encrypted using Transport Layer Security (TLS). The whole concept is also transferable to other application domains. When a multi-touch trackpad is available (as with most Macintosh

[5] http://idav.ucdavis.edu/ okreylos/ResDev/Vrui

computers) the menu can be controlled the same way. Unfortunately, the use of sensors to control rotation is then no longer possible.

The paradigm of our design is usable for a number of other application areas as well, such as desktop computers, tv sets, etc.

We plan do devise and perform a user study for our system, considering multiple applications and users from various disciplinary backgrounds (i.e., users applying out system to their domain-specific problems).

References

1. J. Accot and S. Zhai. Beyond Fitts' law: models for trajectory-based HCI tasks. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 295–302. ACM, 1997.
2. G. Bailly and E. Lecolinet. Flower menus: a new type of marking menu with large menu breadth, within groups and efficient expert mode memorization. *AVI '08 Proceedings of the working conference on Advanced visual interfaces*, pages 15–22, 2008.
3. R. Ballagas, M. Rohs, and J. Sheridan. Sweep and Point and Shoot: Phonecam-based Interactions for Large Public Displays. In *CHI'05 extended abstracts on Human factors in computing systems*, pages 1200–1203. ACM, 2005.
4. P. Baudisch, N. Good, and P. Stewart. Focus plus context screens: combining display technology with visualization techniques. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01, pages 31–40, New York, NY, USA, 2001. ACM.
5. P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47(6):381–391, 1954.
6. J. Francone, G. Bailly, E. Lecolinet, N. Mandran, and L. Nigay. Wavelet menus on handheld devices: stacking metaphor for novice mode and eyes-free selection for expert mode. In *Proceedings of the International Conference on Advanced Visual Interfaces*, AVI '10, pages 173–180, New York, NY, USA, 2010. ACM.
7. J. Francone, G. Bailly, L. Nigay, and E. Lecolinet. Wavelet Menus: A Stacking Metaphor for Adapting Marking Menus to Mobile Devices. *Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services*, pages 2–5, 2009.
8. H. Gao, X. Guo, X. Chen, L. Wang, and X. Liu. YAGP: Yet Another Graphical Password Strategy. *Computer Security Applications Conference, Annual*, 0:121–129, 2008.
9. F. Guimbretiere and T. Winograd. FlowMenu: Combining Command, Text, and Data Entry. In *Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 213–216. ACM, 2000.
10. F. Heidrich, I. Golod, P. Russell, and M. Ziefle. Device-free interaction in smart domestic environments. In *Proceedings of the 4th Augmented Human International Conference*, AH '13, pages 65–68, New York, NY, USA, 2013. ACM.
11. A. Holzinger, G. Searle, B. Peischl, and M. Debevc. An answer to who needs a stylus? on handwriting recognition on mobile devices. In M. Obaidat, J. Sevillano, and J. Filipe, editors, *E-Business and Telecommunications*, volume 314 of *Communications in Computer and Information Science*, pages 156–167. Springer Berlin Heidelberg, 2012.

12. A. Holzinger, P. Treitler, and W. Slany. Making apps useable on multiple different mobile platforms: On interoperability for business application development on smartphones. In G. Quirchmayr, J. Basl, I. You, L. Xu, and E. Weippl, editors, *Multidisciplinary Research and Practice for Information Systems*, volume 7465 of *Lecture Notes in Computer Science*, pages 176–189. Springer Berlin Heidelberg, 2012.
13. S. Jeon, J. Hwang, G. J. Kim, and M. Billinghurst. Interaction techniques in large display environments using hand-held devices. In *Proceedings of the ACM symposium on Virtual reality software and technology*, VRST '06, pages 100–103, New York, NY, USA, 2006. ACM.
14. K. Kin and B. Hartmann. Two-handed marking menus for multitouch devices. *ACM Transactions on Computer-Human Interaction (TOCHI) TOCHI Homepage archive*, 18(2):16:1–16:23, 2011.
15. O. Kreylos. Environment-Independent VR Development. In G. Bebis, R. Boyle, B. Parvin, D. Koracin, P. Remagnino, F. Porikli, J. Peters, J. Klosowski, L. Arns, Y. Chun, T.-M. Rhyne, and L. Monroe, editors, *Advances in Visual Computing*, volume 5358 of *Lecture Notes in Computer Science*, pages 901–912. Springer Berlin / Heidelberg, 2008.
16. G. Kurtenbach and W. Buxton. The limits of expert performance using hierarchic marking menus. In *Proceedings of the INTERACT '93 and CHI '93 conference on Human factors in computing systems*, CHI '93, pages 482–487, New York, NY, USA, 1993. ACM.
17. G. Kurtenbach and W. Buxton. User learning and performance with marking menus. In *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence*, CHI '94, pages 258–264, New York, NY, USA, 1994. ACM.
18. G. Lepinski and T. Grossman. The design and evaluation of multitouch marking menus. *Proceedings of the 28th international conference on Human factors in computing systems*, pages 2233–2242, 2010.
19. A. Madhavapeddy, D. Scott, and R. Sharp. Using Camera-Phones to Enhance Human-Computer Interaction. *Proceedings of Ubiquitous*, 2004.
20. J. Mankoff and G. D. Abowd. Cirrin: a word-level unistroke keyboard for pen input. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*, UIST '98, pages 213–214, New York, NY, USA, 1998. ACM.
21. K. Perlin. Quikwriting: continuous stylus-based text entry. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*, UIST '98, pages 215–216, New York, NY, USA, 1998. ACM.
22. S. Pook, E. Lecolinet, G. Vaysseix, E. C. Ura, and M. Bp. Control Menus : Execution and Control in a Single Interactor. *CHI '00 extended abstracts on Human factors in computing systems*, (April):263–264, 2000.
23. S. Thelen, J. Meyer, A. Ebert, and H. Hagen. A 3D Human Brain Atlas. *Modelling the Physiological Human*, pages 173–186, 2009.
24. D. Venolia and F. Neiberg. T-Cube: a fast, self-disclosing pen-based alphabet. In *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence*, CHI '94, pages 265–270, New York, NY, USA, 1994. ACM.