Mobile Devices for Virtual Reality Interaction. A Survey of Techniques and Metaphors

Jens Bauer(✉) and Achim Ebert

Computer Graphics and HCI Lab, University of Kaiserslautern,
Kaiserslautern, Germany
{j_bauer,ebert}@cs.uni-kl.de

Abstract. Virtual Reality applications run in a lot of different environments using several different input devices. This can lead to incoherent input metaphors across different environments, even using the same application. Mobile smart devices, such as smart phones and tablets can be used as alternative input devices. They can provide an uniform way of interaction, independent of the actual setting of the VR environment. Additionally, their use does scale well to the number of users in multi-user environments. This chapter presents the current State-of-the-Art in using smart devices as input devices and assesses their applicability in the field of Virtual Reality.

Keywords: Human computer interaction · Input devices · Mobile devices · VR environments

1 Introduction

Virtual Reality applications run in a multitude of environments, such as CAVEs, Powerwalls, etc. or even on simple personal computers. Some applications are targeted at one special environment, others are more general (maybe through the use of VR libraries). Due to the cost involved in creating a virtual reality environment, it is often not possible to create the ideal environment for a specific application. In most cases, one or two general-purpose environments are created and the applications have to be created for (one of) these environments or existing applications have to be adapted. Another important challenge for virtual reality applications and environments is the trend towards collaboration. Traditional VR environments are tailored to single-user experiences. This simplifies the setup and simultaneously allows for a better immersive experience (e.g., due to 3D screens only allowing for a single "sweet spot" that is usually centered on the single user). In all cases however, input devices are tailored towards their specific usage scenario. While a feasible approach, this causes a number of problems as well. Application developers might need to implement the same functionality for multiple input devices, causing a higher workload. Users on the other hand need to adapt to different devices even for the same application if they use it in different environments. The emerging collaboration

G. Brunnett et al. (Eds.): Virtual Realities, LNCS 8844, pp. 91–107, 2015.
DOI: 10.1007/978-3-319-17043-5_6

aspect in VR poses a new requirement to input devices. Many of the traditional devices generally scale not well to the number of users. Reasons for this include the availability and pricing per device, spacial requirements for input devices (e.g., a computer mouse needs a stable surface to be operated on), etc. Thus many collaboration meetings become a kind of presentation, where one user has the only input device available and is therefore the only one able to interact [9]. One approach to counteract the generally low number of input devices, is to have users bring their own devices (referred to as BYOD in some literature). Since smart phones and tablets are in wide-spread use and their popularity is growing, they might be a good alternative to more traditional devices. They are also becoming more and more powerful. Current consumer-level products are featuring multi-core CPUs and dedicated GPUs. They incorporate multi-touch screens, GPS receivers, compasses and accelerometers, furthermore connectivity through WiFi, bluetooth, Near Field Communication (NFC) and 3G technologies. They offer several benefits, due to their added functionality compared to typical input devices. For example, they can be used as output devices, even on a per-user basis to provide individualized views. With their growing memory capacity they can also serve as a portable data storage. They can provide uniform means of interaction across different VR environments.

While research in Virtual Reality interaction is generally aiming more at specialized input devices, there is ongoing research about the possible application of smart phones and tablets as input devices in the field of Human-Computer-Interaction (HCI). This paper presents the most current and important results of this research and rate their applicability in Virtual Reality. While it is impossible to present all results in this field, we aim to present a selection of different approaches. It is unlikely that a more specialised overview will generally help designers of VR environments or applications, since there is a wide field of requirements in VR. In order to classify the results, we will first present taxonomies of input devices and metaphors in Sect. 2. In the later sections notable research results in this regard will be presented: techniques that base on trackpad-like interaction (Sect. 3), methods using the camera of the mobile device (Sect. 4), menu-based approaches (Sect. 5), gestures (Sect. 6) and application-tailored methods (Sect. 7) before concluding in the last section.

2 Taxonomy and Classification

There are some possibilities to classify the research results presented in the later sections. Quality dimensions of different input device approaches include the actual capability to effectively execute tasks in applications, the volume of control or number of different input vectors (e.g., degrees-of-freedom) and other small factors. In this section we introduce the taxonomy of tasks by Foley, Buxton's input device taxonomy and a few additional criteria. With Foley's taxonomy we can show the capabilities of input devices (and techniques) to execute different tasks. Buxton's taxonomy can be used to measure the amount of control vectors. A few special properties of input methods in the area of virtual reality interaction are not covered by these two taxonomies, which is why we added four additional factors.

2.1 Foley's Taxonomy of Tasks

An old, but still useful taxonomy is from Foley et al. [13] (as for example confirmed by [4,5]). It identifies six different tasks, that cover all uses of input devices. Complex interactions can be decomposed into these six basic tasks.

1. **Position:**
   Set the absolute or relative position of an object in 2D or 3D space.
2. **Orient:**
   Set the absolute or relative orientation of an object in 2D or 3D space.
3. **Select:**
   Select an item out of a list of several items.
4. **Ink:**
   Define a path consisting of one or more positions and orientations. The name Ink refers to the visual line represented by a path.
5. **Quantify:**
   Select a number from a continuous or discrete set.
6. **Text:**
   Enter arbitrary sequences of characters.

2.2 Buxton's Taxonomy of Input Devices

Another taxonomy of devices by Buxton [10] might seem appropriate at first. It classifies input devices by their physical properties, like the control agent (e.g., the hand), what is sensed by the input device and in how many dimensions (i.e., this is related to the number of Degrees-of-Freedom (DOF) provided by the device). This is also affected by the number of buttons or similar triggers and modifiers on a device.

This approach does not work well to classify different metaphors and techniques on similar input devices. But it still can be used in some circumstances, when a technique is deviating from the default multi-touch screen interaction schema. Additionally, as smart phones and tablets are not only input devices, but also feature output facilities (i.e., the screen, speakers, vibration, etc.), this taxonomy can be extended to the kind of output used (if any) by a certain method.

2.3 Other Traits

The taxonomies explained above do not cover all important characteristics. More interesting traits are:

- **Eyes-free interaction:**
  This is a characteristic, that is not very important for traditional input devices. Most of them are eyes-free by design. For example, a simple computer mouse can be operated without any trouble while looking at the screen instead of the mouse. This goes for almost all input devices in common use, with the notable exception of keyboard, at least for inexperienced users. But with the screen on smart phones and tablets and the resulting possibility of displaying content to the user via the input device, the discrimination between eyes-free techniques and those requiring the screen of the hand-held device is important.
- **Tracking:**
  While technically part of Buxton's taxonomy (as the number of dimensions provided), it is especially interesting for collaborative setups to know, if a certain method needs some form of tracking. This is due to the fact, that tracking does not scale really well to the number of users, as most tracking systems have a fixed maximum number of markers to track.
- **Secondary Device/Method:**
  Some methods and techniques might be usable only for a subset of features provided for the main input device. This can either help to provide at least a minimum number of interactivity to users who would otherwise be only spectators, or to have a specialised device, that is only used for a certain interaction and will do this in a better way than a standard input device.
- **Scalability:**
  It is important to measure for each technique, if it is actually applicable to more than one user and if there is a point, where it will perform worse when additional users are employing this technique concurrently. For example, all methods needing a mouse pointer will suffer from this, since it is becoming more and more difficult for the users to distinguish their mouse cursor from the others.

3 Trackpad-Based Techniques

The most straight-forward method of using a (small) touchscreen is probably to have it emulate a trackpad. This has the advantage of most users already knowing the trackpad metaphor and being able to easily handle it. Of course with this technique none of the advanced capabilities of smart phones and tablets are actually used. Several commercial apps doing this are already available, like *Remote Mouse*[1]. Such apps usually allow to send text to the connected computer. Extending from this, the *ArcPad* [20] can be used as a trackpad, but additionally tapping on any point of the pad will move the mouse cursor to the position on the screen according to the point tapped on the pad. E.g., a tap on the lower left corner will put the mouse cursor to the lower left corner of the screen. Respectively tapping the center of the pad will put the mouse cursor to the center of the screen. This method was created for large display environments, where movement of a mouse cursor might be cumbersome. Unfortunately, users

[1] http://www.remotemouse.net/.

can run into trouble with losing track of the cursor when tapping on the pad. This is especially an issue for users of computers with trackpads, where a tap normally is interpreted as a click instead of a cursor-relocation.

Since these methods are used to control a mouse cursor, they are basically able to complete all 6 of Foley's taxonomy (including text since text can be entered through the keyboard on the mobile device) through a level of indirection. Directly only position, select and text can be supported. Trackpads (including the *ArcPad*) provide 2DOF and one or two buttons. Output facilities of the smart phone or tablet are not used in this approach. It works eyes-free, as there is nothing to be displayed at the screen at all and does not need any tracker to be used. But it will not scale well to many users, as multiple mouse cursors will puzzle the users. Depending on the setup and the experience of the users this might work for a small group of users, with coloured mouse cursors or a similar technique to help differentiating the cursors.

For VR environments the use is actually limited due to the fact that it is only 2D-cursor-based. This creates a big problem when interacting in a 3D space, when requiring multiple DOF. Still this might be a viable solution for special setups, where 2D interaction is sufficient (e.g., selection of objects in a 2.5D scene on a powerwall).

4 Camera-Based Techniques

Some interaction methods rely on the camera of the mobile device. While all of the papers in this section use phones as devices, they also apply to tablets with cameras. They rely of markers that are available for the device to scan as a base.

Prior to the smart phone era, Madhavapeddy et al. [19] created a system where users can use a phone with a camera and bluetooth to interact with a world map application (Fig. 1). The application displayed a map of the world, augmented with Spot Codes, a circular bar code. The user(s) can take pictures of the application, containing a Spot Code. The phone will then query the application via the bluetooth connection using the Spot Code's content as an id, to get further information about the special spot on the map.

Thelen et al. [27] took this idea further by customizing the information transferred to the user's phone. This leverages on the idea of having a separate device for each user. The *3D Human Brain Atlas* (Fig. 2) [27] is basically a quiz about the human brain, featuring different levels of difficulty. By scanning a barcode prior to the beginning of the quiz, users select their own difficulty level. The phone remembers this throughout the game and the combination of a code scanned on the main screen together with the difficulty is sent to a server, which in turn sends the quiz question. Also the phone is used as a identification mechanism, where certain phones can be registered as instructor phones and will get the answers together with the question.

Point and Shoot by Ballagas et al. [6] uses the visual marker approach, but displays them only for short periods of time. The phone will send a notification to the main application to display the markers, then the picture is taken including

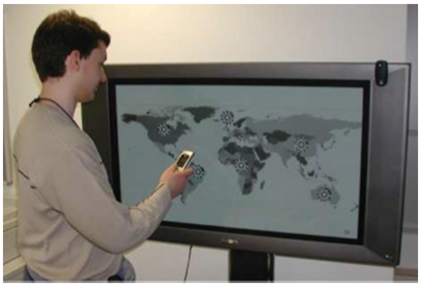
Fig. 1. Selection on the world map with spot codes [19].

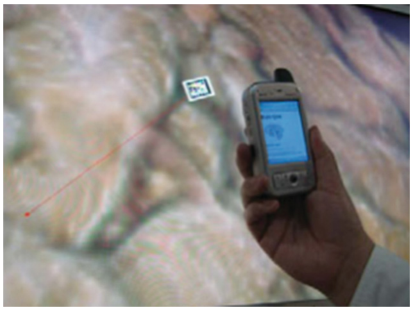
Fig. 2. A marker being selected on the *3D Human Brain Atlas* [27]

the markers, after which the markers disappear. The position where the phone was pointed to is then calculated based on the markers.

In the same paper another technique, called *Sweep* [6], was introduced. By optical flow calculations done directly on the phone, it is possible to detect the phones movement and use it similar to an optical mouse, even in mid-air. Unfortunately, due to technical limitations, this method incurred a high latency (about 200 ms) making its usage cumbersome. To our knowledge, there is no current study if the technical advances till today could lift this restriction, but it is very likely.

Rhos [25] proposed to use markers on physical objects to create Augmented Reality (AR) games. While this is not directly related, this can still be used with the approaches above, to create additional content and seamlessly add it into the basic content provided by the application to all users.

The following classification will exclude *Sweep*, since it is different from the other methods. It will be classified separately. The methods are basically only used to perform selection tasks (according to the Foley taxonomy). It can be noted that some of the techniques follow up with text input after the selection by using the phone's native keyboard. Combining several selections allows for the other tasks to be completed, too, but this might be cumbersome for the users.

All of these techniques feature input in 2 dimensions (as restricted by the camera). It is notable, too, that the input device is also used as an output device for most of the methods, making collaboration easier on a larger scale, since interaction is possible without interrupting the workflow of other users.

None of the methods are actually eyes-free, as it is necessary to point with the device at some position on the screen, which is of course only possible by looking at it. This might interrupt the user's workflow, but assuming this is used to present further information on the screen of the mobile device, this is not a problem. Tracking systems are also not needed for any of those methods, all of them are actually designed to do their own kind of tracking their position. Depending on the application, external tracking might be used to improve the accuracy of the chosen method. Transferring content to the smart phone or tablet makes it also a valuable secondary device. A program might provide the standard means of interaction (depending on the program and setup) and additionally can allow for mobile devices to be used for further information, or for additional users to get information. Regarding scalability, all the methods work well with many users, actual restrictions are posed by available screen space where the users can interact. *Point and Shoot* might also have some scalability issues because of the bar codes flashing up on the screen, whenever a user interacts. This might distract other users looking at the screen. The problem increases with more people using the system at the same time.

Depending on the actual setup of a VR environment, these method might work very well (e.g., a powerwall) or might not be a good choice at all (e.g., fully immersive environment).

Sweep can only be used for position tasks, adding a button functionality will also allow for selection and, through composition, all other tasks as defined by Foley. It tracks input in 2D, but it could be augmentable to actually support 3D. It is eyes-free, as it does not use any of the output facilities of smart phones and tablets and is used similar to a mouse. It is also not dependant on external tracking, since doing its own tracking is the core of *Sweep*. The technique in itself is highly scalable. The real problem is similar to trackpads: the number of mouse cursors that need to be displayed on the screen. The cursor is also the problem why this might not be a good choice of input method for VR applications. Similar to the trackpad solutions already described, this is only useful if a 2D pointer can be effectively used with the application.

5 Menus

Another approach of using smart phones and tablets as input devices is an external menu structure on the mobile device. This can be as simple as presenting all required functionality in the device's native UI and transferring all user input to the actual application. But this most likely causes a break in user interface design between main application and mobile device and is additionally not eyes-free in most cases, interrupting the workflow when used. For this reason several advanced menu versions have been proposed, most of them featuring an eyes-free interaction mode.

Many of those menus base on the idea of Marking Menus [18]. Marking Menus are basically a structure of radial menus. The user can use them eyes-free after learning the menu structure, by just remembering the path to draw (with finger or pen) to a certain menu item. When applied to mobile devices, the device can present the menu to the user who can then either progress to move to the desired menu item blindly, or look at the structure to find the menu item and thus help to remember the path next time. The strokes of the path can be drawn continuously or stroke-after-stroke, depending on the actual implementation.

As current smart phones and tablets allow for multi-touch interaction, standard Marking Menu interaction can be improved by utilizing this. Kin et al. [17]

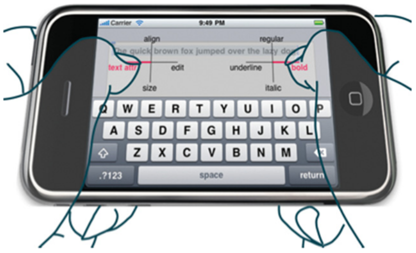

Fig. 3. Two-handed Marking Menus [17] employ multi-touch to draw parts of the Marking Menu strokes using alternating fingers.

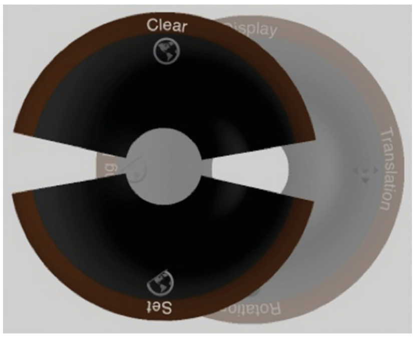

Fig. 4. Marking Menus with continuous strokes are the base for the multi-touch extensions by [7].

used multi-touch to enable faster input of the strokes, using two fingers at the same time (Fig. 3). They use the stroke-after-stroke version of Marking Menus, allowing to either draw the strokes simultaneously one with each finger, or one after another, alternating between fingers, a bit slower, but offering double the number of menu items, by changing the menu depending on the starting finger.

Another possibility of multi-touch extensions are proposed by Bauer et al. [7]. Using the continuous version of Marking Menus (Fig. 4), multi-touch can be used to continue strokes even when the user hits the edge of the screen by just using another finger to continue the strokes. Using the same technique, it is possible to have one finger on the screen, only to have the menu stick to the current state and have another finger select menu items from the same point. If a menu item *next* is accessible through a up-left-stroke sequence, one finger can do the "up" stroke, another one then can do the left stroke over and over again, without needing to add "up" in front of each stroke, as long as the first finger is on the touch screen. Also the number of items in the menu structure is increased by presenting different menus, depending on how many fingers are used to initiate the stroke-sequence. By including the idea of *Control Menus* [24], it is possible to not only select an item, but also to add value control as an item. Combined with multi-touch this can be used for higher dimensional value control and 3D interaction. Additonally, the accelerometer sensors on the smart phone or tablet can be used for further input variations.

Flower Menus by Bailly et al. [2] improve the number of items in a Marking Menu structure by not only allowing straight strokes, but instead take the curvature of a stroke into account. Using the stroke-after-stroke variant of Marking Menus, they have 12 items on each level. Each basic direction (i.e., Up, Down, Left, Right) can be combined with either a straight stroke, or a curved stroke to either the left or right side to get up to 12 items per level. Of course multiple levels of menu structure can be chained one after another (Fig. 5).

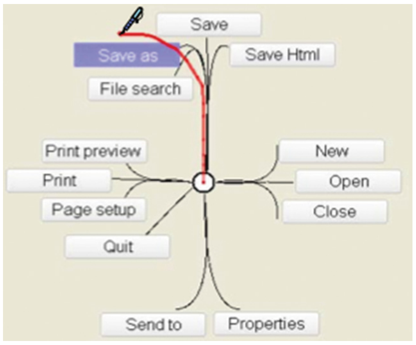

Fig. 5. The design of *Flower Menus* [2]. Each basic stroke direction (Up, Down, Left, Right) has 3 variations (straight, curved left/right).

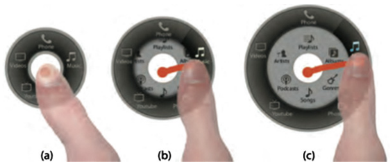

Fig. 6. *Wavelet Menus* [14,15] are a sequence of radial menus, that are conceptually hidden under each other and revealed as the user strokes from the center towards an menu item.

Basing on the idea of novice and expert mode interaction as used by Marking Menus, Francone et al. [14,15] proposed the *Wavelet Menu* (Fig. 6). It is derived from the Wave Menu by Bailly et al. [3], but better suited to the small screen estate on mobile devices. The first time the user touches the screen, the root level of the menu is revealed around the touch point. Then when the user starts moving the finger towards one menu item, the submenu is appearing from below the current menu, with the current menu expanding outwards. When the user releases the finger at the menu, the next level of menu (or actual functionality) gets selected. If the touch is released prior to this, the last menu stays on top instead. Like with Marking Menus, the user can remember the sequence of strokes needed to access a certain item and can then do the strokes without actually looking at the device.

Gebhardt et al. [16] use a HTML-based menu displayed on a mobile device to interact with VR spaces. It is targeted for use with configuration tasks of the system, but can also be used for other tasks. They created a custom set of widgets to be presented on the mobile device. An example of this can be seen in Fig. 7. The strong point of this method is the device-independence, since the only requirement for the mobile device used is HTML5-support. Also there are basically no requirements for the server-VR-system; it does not even need to be a VR system.

Menu selection only offers support for selection tasks. By including the Control Menu mechanic, quantify tasks can also be accomplished, and by composition everything else, even text (if paired by a technique like QuikWrite [23]). The input dimension of menus is defined by the number of items it contains, which can be a large number. With the Control Menu mechanic and common multi-touch gestures, like rotation and pinch-to-zoom, four dimensions of input can be achieved, enough to allow for 3D interaction, important for VR applications.

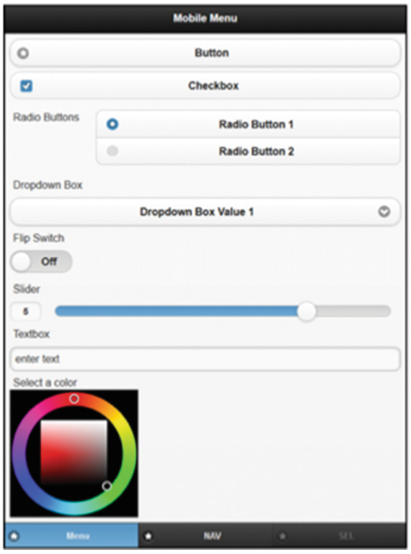

Fig. 7. A *HMTL5 Menu* [16] composed of several standard and custom widgets.

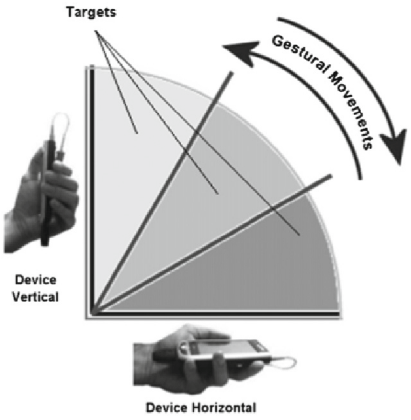

Fig. 8. The principle of *Motion Marking Menus* [22] is to divide the possible angle range into different parts, each corresponding to a menu item. By alternating the movement or adding small stops in between, a menu structure can be traversed.

All of the presented menus, with the exception of the HTML-based menu, can be used eyes-free after a training period. They do not need external tracking, but it could be used to provide a means of positioning and/or rotational input to the presented method. This will, of course, negatively affect the scalability of that method. Still any of the menu methods can be used as a secondary input method, especially to avoid menu structures in immersive environments, where navigating menus can be very cumbersome. Without tracking, menu interaction scales very well to the number of users, due to the minor communication overhead per device, the fact that users can practically interact from any position they can see the application screen/area and individual mobile devices do not interfere with each other during interaction.

For these reasons, all presented menu approaches are very well suited to be used in a VR environment, with the actual implementation being dependent on the parameters of the VR application.

6 Gestures

Using the built-in sensors of smart phones and tablets, especially accelerometers, gestures with the device itself can be recognised. Since these gestures do not use the touch screen or only for activation of the gesture sensing, it is easy to implement the same gesture recognition for other devices as well, as long as they feature the necessary sensors.

Device gestures can also be added to other input methods, such as the menus presented in the previous section. One has to keep in mind, that the gestures are more difficult to execute with larger and heavier devices, so large tablets are not a good input device to use the following methods with.

Motion Marking Menus (Fig. 8) as presented by Oakley and Park [22], base on the concept of Marking Menus, as shown in the last section. But instead of using swipes for the selection of menu items, selection is done via the angle it is held in. They use a button or a touch on the screen as an activation mechanic. The angle the device can be held is divided into a number of menu items. Two menu items, for example, each get a $45\,^\circ$ angle, meaning that holding the device horizontally or up to a $45\,^\circ$ deviation from the horizontal plane will invoke the first menu item, any other posture will activate the second item. By tilting the device up and down from the current posture while still holding the activation button, a menu sequence can be invoked, similar to touch menus.

Building from this general idea, *Jerk-Tilts* (Fig. 9) is a method by Baglioni et al. [1] for selection by gesture. Instead of having the user remembering a special posture or sequence of postures, they use a single tilt-and-back gesture in different directions to invoke functionality. For example, holding the device and executing a quick tilt to the right and then back to the original posture will invoke a selection of a certain kind. The same can be done in all four basic direction and combining two directions allow for a total of eight different possibilities. A big advantage of this method is, that it works without any activation button, allowing to use this together with other input methods using the touch screen.

Dachselt and Buchholz [11] suggest to use a throw gesture alongside tilt gestures (forming *Throw and Tilt* (Fig. 10)). Tilt can be used for selections, similar to the methods presented above in this section or to move a mouse cursor, while the throw gesture can be used to transfer data to the main application. Care has to be taken when the throw gesture is executed to have a firm grip on the input device, but it is intuitive to throw content towards the application.

All the techniques above provide means to accomplish selections tasks. Using tilt, *Throw and Tilt* can additionally provide means for position and/or rotation tasks. This can again be composed into all other possible tasks. The number of input dimensions is different for all methods. *Motion Marking Menus* have a dimensionality dependent on the number of menu items. *Jerk-Tilts* have a fixed dimensionality of eight, the number of possible gestures. With only three dimensions (two tilt dimensions and throw) *Tilt and Throw* has the least number of dimensions. All methods work eyes-free and need no external tracking and are also scalable to the number of users. Due to the greater motion of the throw gesture, *Tilt and Throw* might scale a little bit less than the other approaches, but this can be somewhat mitigated by providing an alternative to the throw gesture.

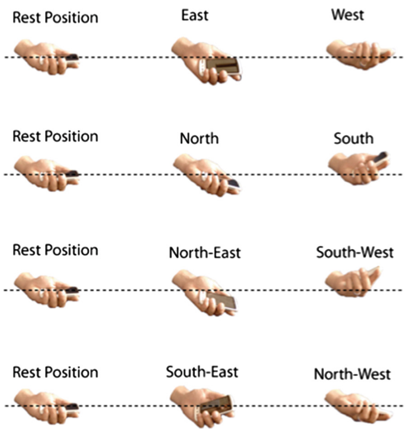

Fig. 9. *Jerk-Tilt* [1] are a tilt-and-back kind of gesture, that can be done in eight different directions, allowing for eight different functions to be invoked.

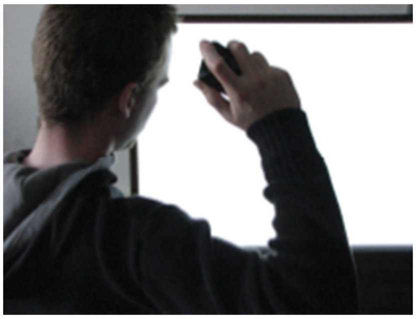

Fig. 10. The characteristic throw gesture from *Tilt and Throw* [11].

All these methods can well be used as a secondary input method to provide a subset of the application's main functionality to additional users. The methods can also be used with tracked input devices, that might already be present in a given VR setup, such as flysticks. This allows for a common input metaphor over multiple input devices.

7 Application-Tailored Techniques

This section contains several techniques, that are specifically tailored towards a certain application. While this limits their general applicability, it is still possible to use them in similar usage scenarios. Because these techniques do not have much in common, their classification will be written individually instead of grouping them together as in the previous sections.

The *Mod Control* (Fig. 11) system by Deller and Ebert [12] was designed for interaction with a map application. The system itself is modular and not directly tailored towards the application. But the reference implementation features several interaction modules to improve control of the map application through a smart phone. Among them is a combined touch/accelerometer interface for 3D navigation. The user can modify the current view as flying, controlling the forward and sideward speed using the touchpad while the devices rotation control yaw and heading of the view. This is a common problem, especially in VR environments. This module can be classified to accomplish position and orientation tasks, in 5 DOF, it can be used eyes-free and does not need external tracking, it is usable as a secondary device to other input devices which do have less DOF. Also it scales well to the number of users.

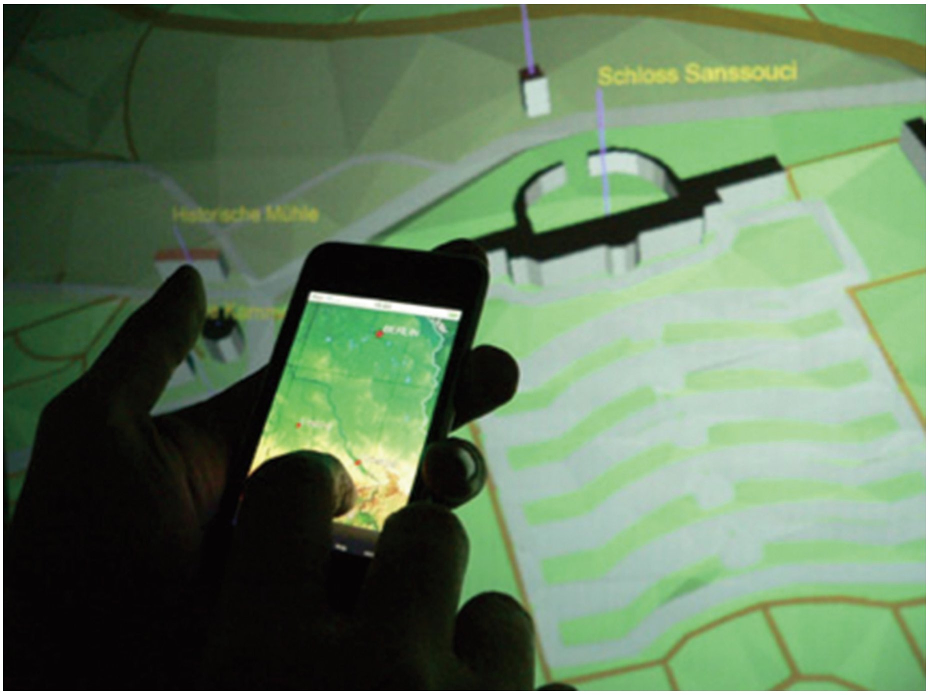

Fig. 11. *ModControl* [12] can be used to present an overview of the content to a mobile device and allows independent interaction.

Another interesting module is the image module, featuring a small version of the map shown by the main application. This allows users to explore the map independently from other users and send their current view back to the application if needed. It fulfils a position and select task, in two dimensions. Though it cannot be operated eyes-free, it scales very well since every user can interact totally independent of the others. For the same reason it also makes for a good secondary device for additional users. The applicability in the VR domain is dependent on the VR application. For terrain simulations, for example, it will work very well, while it is not useful for architectural applications.

Seewonauth et al. [26] propose two techniques, *Touch & Connect* and *Touch & Select* to initiate data transfer between a phone and a computer. Since today's smart phones and tablets have quite large memory capacity, they can be used as a mobile data storage. To avoid complicated file management, one of those two method can be used to easily copy a data set. *Touch & Connect* bases on NFC. The users select a file on the source device and simply touch a NFC tag on the target computer to copy the file. Note that the same approach can be used to get the current data set from the display or to initiate a connection (as input device) to the currently running application, avoiding the need to enter IP-Addresses or Host names.

Touch & Select uses several NFC tags to track the position of the mobile device. The user again selects a file on the smart phone or tablets and touches a point directly on the screen. This initiates the file transfer. The same principle can be used to interact with the screen directly for other effects.

Both methods allow to accomplish selection tasks, with *Touch & Select* having additional two dimensions of input. Since the user has to point with the device, it cannot be used eyes-free, but especially if used for data transfer, both methods are valid secondary input means. It does not scale as well as other methods to the number of users, since it is also dependent on the number of NFC tags provided and also it requires the users to stand near the tagged spots, limiting the concurrent use. External Tracking is only needed by the means of providing the NFC tags.

For VR applications, *Touch & Connect* is probably more useful than *Touch & Select*, since the latter selects a 2D screen position, which is only situational useful. But to initiate connections between a mobile device and a stationary computer system, *Touch & Connect* is very convenient. Please also note, that a similar behaviour is achievable by providing a visual tag, that can be scanned by a smart phone or tablet camera if NFC is unavailable.

Bauer et al. [8] provide another specialised approach for four different usage scenarios. Beside two scenarios where the content of the main screen is transferred to mobile devices, similar to the image view from *ModControl*, they also present two scenarios for interaction with 3D scenes. One design is using a joystick metaphor. The smart phone or tablet is used to control forward movement by tilting in the appropriate direction. Turning is done by tilting the device sidewards, not unlike a steering wheel.

In another scenario, users control 3D objects by using combined input from the touchscreen and accelerometers. Tilting the device causes the object to move in the X-Y-plane and using the touchscreen movement in the X-Z-plane is available. This makes it a 3DOF input method.

Both methods provide means to accomplish position tasks and with an added button or gesture it can also select. Through composition then all six tasks are possible. The first method provides only two dimensions of input, while the other one provides 3 dimensions (actually four, but since the X direction is mapped in two ways this reduces to three different dimensions). Both techniques can be used eyes-free, without external tracking and scale well to the number of users. They are useful for additional users to provide a reduced set of controls to interact with an application. And as both methods are designed for 3D interaction, they are in general useful for VR applications.

Semantic Snarfing describes a technique by Myers et al. [21] using a laser pointer to mark a spot on a large display. The semantic area (e.g., a dialog) this spot belongs to is then transferred to a mobile device's screen. Due to technical progress since the time the paper was published, the method can be used today without a laser pointer by utilizing the camera of a smart phone or tablet. This makes *Semantic Snarfing* a method similar to the camera methods described in their own section. What makes *Semantic Snarfing* special from the camera techniques is the semantics that needs to be provided in order to make the system work. The method has in general the same properties as the camera methods, as it allows selection (and then more tasks on the mobile device's screen), scales generally well to the amount of users, but does not support eyes-free interaction. Tracking is not required. Its usability in VR environments is dependent on the application and setup, quite similar to the camera methods. But as a secondary device, *Semantic Snarfing* is applicable to even more VR environments, as the primary device can be used to select the "snarfing point" for the mobile device.

Table 1. Overview of all presented methods. (★: methods using output capabilities.; c: achievable through composition; parenthesis depict simple improvements to the method.)

Method	Position	Orient	Select	Ink	Quantify	Text	Dimensions	Eyes-free	Tracking	Secondary	Scalability
Trackpad - Methods											
ArcPad	✓	(c)	(✓)	(c)	(c)	(✓)	2(4)	✓	✗	✗	-
Camera - Methods											
World Map			✓			✓	2★	✗	✗	✓	o
3D Humand Brain Atlas			✓			✓	2★	✗	✗	✓	o
Point and Shoot			✓			✓	2★	✗	✗	✓	o
Sweep	✓		(✓)				2 (3)	✓	✗		-
AR			✓			✓	2★	✗	✗	✓	o
Menu - Methods											
Marking Menus			✓				any	✓	✗	✓	+
Multi-touch Marking Menus			✓				any	✓	✗	✓	+
Improved Marking Menus	(c)	(c)	✓	(c)	✓	(c)	any+any	✓	✓	✓	+
Flower Menus			✓				any	✓	✗	✓	+
Wavelet Menus			✓				any	✓	✗	✓	+
HTML Menus	(c)	(c)	✓	(c)	✓	✓	any	✓	✗	✓	+
Gesture - Methods											
Motion Marking Menus			✓				any	✓	✓	✓	+
Jerk Tilts			✓				any	✓	✗	✓	+
Throw and Tilt	✓	✓	✓				any	✓	✗	✓	+
Application-Tailored - Methods											
Mod Control - Navigation	✓	✓					5	✓	✗	✗	+
Mod Control - Image	✓		✓				2	✗	✗	✗	+
Touch & Connect			✓				-	✗	✓	✓	o
Touch & Select			✓				2	✗	✓	✓	o
Joystick	✓						2	✓	✗	✓	+
Touch + Tilt	✓						3	✓	✗	✓	+
Semantic Snarfing	(✓)	(✓)	(✓)	✓		✓	2★	✗	✗	✓	+

8 Conclusion

Several interaction techniques are possible with smart phones and tablets. For applications where collaboration is expected, supporting interaction via mobile devices will add benefit to users and their collaboration. Even if not the full set of functionality can be accessed by smart phone or tablet interaction, it will still help users to be active users of an application instead of being only passive spectators.

However, it is important to choose the right form of interaction for the application in question in order to reach a high usability of the resulting system. The overview given by this paper should help in considering which methods are feasible for certain settings. Table 1 contains an overview of the properties reviewed in the previous chapter for easier comparison.Moreover, it can form a starting point for implementation of an own interaction metaphor. If possible, a metaphor for the mobile device should not deviate from the metaphor of the main device to avoid the necessity of learning to use two devices. In some application areas it is possible to avoid the usage of other devices completely, and using smart phones or tablets as the only input device.

References

1. Baglioni, M., Lecolinet, E., Guiard, Y.: JerkTilts: using accelerometers for eight-choice selection on mobile devices. In: Proceedings of the 13th International Conference on Multimodal Interfaces, ICMI 2011, pp. 121–128. ACM, New York (2011)
2. Bailly, G., Lecolinet, E.: Flower menus: a new type of marking menu with large menu breadth, within groups and efficient expert mode memorization. In: AVI 2008 Proceedings of the Working Conference on Advanced Visual Interfaces, pp. 15–22 (2008)
3. Bailly, G., Lecolinet, E., Nigay, L.: Wave menus: improving the novice mode of hierarchical marking menus. In: Baranauskas, C., Abascal, J., Barbosa, S.D.J. (eds.) INTERACT 2007. LNCS, vol. 4662, pp. 475–488. Springer, Heidelberg (2007)
4. Ballagas, R., Borchers, J., Rohs, M., Sheridan, J.G.: The smart phone: a ubiquitous input device. IEEE Pervasive Comput. **5**(1), 70–77 (2006)
5. Ballagas, R., Borchers, J., Rohs, M., Sheridan, J.G., Foley, J., Wallace, V., Chan, P.: The smart phone: a the input design space. Shoot (2006)
6. Ballagas, R., Rohs, M., Sheridan, J.: Sweep and point and shoot: phonecam-based interactions for large public displays. In: CHI 2005 Extended Abstracts on Human Factors in Computing Systems, pp. 1200–1203. ACM (2005)
7. Bauer, J., Ebert, A., Kreylos, O., Hamann, B.: Marking menus for eyes-free interaction using smart phones and tablets. In: Cuzzocrea, A., Kittl, C., Simos, D.E., Weippl, E., Xu, L. (eds.) CD-ARES 2013. LNCS, vol. 8127, pp. 481–494. Springer, Heidelberg (2013)
8. Bauer, J., Thelen, S., Ebert, A.: Using smart phones for large-display interaction. In: 2nd International Conference on User Science and Engineering (I-USEr) (2011)
9. Birnholtz, J.P., Grossman, T., Mak, C., Balakrishnan, R.: An exploratory study of input configuration and group process in a negotiation task using a large display. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI 2007, p. 91 (2007)

10. Buxton, W.: Lexical and pragmatic considerations of input structures. SIGGRAPH Comput. Graph. **17**(1), 31–37 (1983)
11. Dachselt, R., Buchholz, R.: Throw and tilt seamless interaction across devices using mobile phone gestures. In: Proceedings of the MEIS 2008, pp. 272–278 (2008)
12. Deller, M., Ebert, A.: Modcontrol – mobile phones as a versatile interaction device for large screen applications. In: Campos, P., Graham, N., Jorge, J., Nunes, N., Palanque, P., Winckler, M. (eds.) INTERACT 2011, Part II. LNCS, vol. 6947, pp. 289–296. Springer, Heidelberg (2011)
13. Foley, J., Wallace, V., Chan, P.: Human factors of computer graphics interaction techniques. IEEE Comput. Graph. Appl. **4**(11), 13–48 (1984)
14. Francone, J., Bailly, G., Lecolinet, E., Mandran, N., Nigay, L.: 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 2010, pp. 173–180. ACM, New York (2010)
15. Francone, J., Bailly, G., Nigay, L., Lecolinet, E.: Wavelet menus: a stacking metaphor for adapting marking menus to mobile devices. In: Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services, pp. 2–5 (2009)
16. Gebhardt, S., Pick, S., Oster, T., Hentschel, B., Kuhlen, T.: An evaluation of a smart-phone-based menu system for immersive virtual environments. In: 2014 IEEE Symposium on 3D User Interfaces (3DUI), pp. 31–34, March 2014
17. Kin, K., Hartmann, B.: Two-handed marking menus for multitouch devices. ACM Trans. Comput. Hum. Interact. (TOCHI) TOCHI Homepage Archive **18**(2), 16:1–16:23 (2011)
18. Kurtenbach, G.P.: The design and evaluation of marking menus. Ph.D. thesis, University of Toronto (1993)
19. Madhavapeddy, A., Scott, D., Sharp, R.: Using camera-phones to enhance human-computer interaction. In: Proceedings of Ubiquitous (2004)
20. McCallum, D., Irani, P.: Arc-pad: absolute + relative cursor positioning for large displays with a mobile touchscreen. In: Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology, pp. 153–156. ACM (2009)
21. Myers, B.A., Peck, C.H., Nichols, J., Kong, D., Miller, R.: Interacting at a distance using semantic snarfing. In: Abowd, G.D., Brumitt, B., Shafer, S. (eds.) UbiComp 2001. LNCS, vol. 2201, pp. 305–314. Springer, Heidelberg (2001)
22. Oakley, I., Park, J.: Motion marking menus: an eyes-free approach to motion input for handheld devices. Int. J. Hum. Comput. Stud. **67**(6), 515–532 (2009)
23. Perlin, K.: Quikwriting: continuous stylus-based text entry. In: Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology, UIST 1998, pp. 215–216. ACM, New York (1998)
24. Pook, S., Lecolinet, E., Vaysseix, G., Ura, E.C., Bp, M.: Control menus: execution and control in a single interactor. In: CHI 2000 Extended Abstracts on Human Factors in Computing Systems, pp. 263–264, April 2000
25. Rohs, M.: Marker-based embodied interaction for handheld augmented reality games. Virtual Reality **4**(5), 1–12 (2007)
26. Seewoonauth, K., Rukzio, E., Hardy, R., Holleis, P.: Touch & connect and touch & select: interacting with a computer by touching it with a mobile phone. In: Proceedings of the 11th International Conference on Human-Computer Interaction with Mobile Devices and Services, p. 36. ACM (2009)
27. Thelen, S., Meyer, J., Ebert, A., Hagen, H.: A 3D human brain atlas. In: Magnenat-Thalmann, N. (ed.) 3DPH 2009. LNCS, vol. 5903, pp. 173–186. Springer, Heidelberg (2009)