mCube: Towards a Versatile Gesture Input Device for Ubiquitous Computing Environments

Doo Young Kwon, Stephan Würmlin, and Markus Gross

Computer Graphics Laboratory, ETH Zurich
8092 Zurich, Switzerland
{dkwon, wuermlin, grossm}@inf.ethz.ch,
WWW home page: graphics.ethz.ch

Abstract. We propose a novel versatile gesture input device called the mCube to support both desktop and hand-held interactions in ubiquitous computing environments. It allows for desktop interactions by moving the device on a planar surface, like a computer mouse. By lifting the device from the surface, users can seamlessly continue handheld interactions in the same application. Since mCube is a single completely wireless device, it can be carried and used for different display platforms. We explore the use of multiple sensors to support a wide range of tasks namely gesture commands, multi-dimensional manipulation and navigation, and tool selections on a pie-menu. This paper addresses the design and implementation of the device with a set of design principles, and demonstrates its exploratory interaction techniques. We also discuss the results of a user evaluation and future directions.

1 Introduction

Everyday objects and our environments themselves are getting enriched with computer systems [17]. Computer graphics and display technologies have transformed our environments to windows connecting the physical and the virtual world. In recent years many researchers have recognized the value of such ubiquitous computing environments and studied to develop more natural and intuitive alternatives for interaction techniques.

In this context, the use of gestures has emerged as an attractive solution for more intuitive communication between human and machine. For instance, gestures have been widely studied to control graphical objects in various displays namely desktop monitors [9, 12], tabletop displays [7], large wall displays [5], and immersive VR displays [22]. Gestures are also actively employed for eyes-free interactions in pervasive, mobile and wearable computing domains [3]. Using gestures, users control appliances and physical conditions of environments such as lights [19]. Gestures are added to commercial video game consoles [14, 1].

As the purpose of using gestures is getting diverse, a variety of devices have been proposed for gesture inputs. For instance, computer mice and tablet pens are adapted for stroke-based gesture inputs [9]. Glove based input devices are utilized to capture detailed hand and finger movements for hand gestures [22]. In addition to these commercial devices, several prototypes of input devices have been developed by investigating multiple sensor technologies [14, 19, 5] . However, while these devices have been

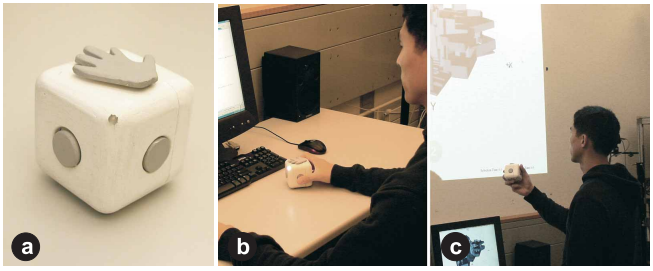

Fig. 1. (a) The mCube prototype device for combined desktop and hand-held interaction. (b) Desktop interaction on a planar surface. (c) Hand-held interaction in the free-space.

successfully tailored for special tasks and specific displays, it is still an open issue to make a gesture input device more versatile such that interactions can be more fluent in a ubiquitous manner in different applications and displays.

In this paper, we propose a versatile gesture input device called mCube which allows for combined desktop and hand-held interactions (Figure 1). We combine visual and embedded sensor technologies to acquire gesture information for robust recognition of a wide range of gestures as well as for multi-dimensional manipulation and navigation. The current prototype has a cube shape, and includes accelerometers, digital compasses, buttons, an IR sensor, and a potentiometer. Sensor data is collected and processed with a microcontroller and transferred via Bluetooth wireless communication. 3D positional information of the mCube is obtained by means of an attached LED, which is captured by multiple cameras and processed using computer vision algorithms.

Our ultimate goal is to minimize the use and learning of additional input devices, and to improve the work flow of interactions. Using the mCube, users can perform gesture commands, virtual object manipulation and navigation, and tool selections on e.g. a pie-menu. Particularly, users can switch between desktop positioning and hand-held positioning on the fly, yielding a high degree of freedom for appropriate interactions. The mCube is a single completely wireless device so that the user is not burdened by wires or additional devices and he can freely move between different display platforms.

In the following sections, we describe the mCube device with a set of design principles for a versatile gesture input device (§2). We explain the system implementation and its core algorithms for computing orientation and position of the device (§3). Then, interaction techniques of the device are demonstrated showing the capabilities of the mCube (§4). We conclude with a summary of a user evaluation (§5) and directions for future work (§6).

2 mCube: A Novel Versatile Gesture Input Device

In this section, we describe our approach to build a versatile gesture input device. First, we describe a set of design principles and solutions. Then we introduce our prototype device, called mCube with its hardware design and sensor configuration.

2.1 Design Principles and Solutions

Support for Combined Desktop and Hand-held Interactions. Ubiquitous computing environments can consist of computer controlled systems and multiple displays namely desktop monitors, table top displays, and wall displays. Considering potential spatial configurations of the user and displays, we categorize interactions into two groups (*desktop interaction* and *hand-held interaction*) based on whether the interaction takes place on a flat surface or in free space.

As one of the main principles, a versatile gesture input device should support both desktop and hand-held interactions. Moreover, the transition between them should be possible without additional operation requirements. Thus, users can instantly choose the best interaction for the current applications and displays. For this purpose, we use a cube form which has been widely used for the development of input devices in both interaction groups [11, 14, 15, 8]. A cube form affords users to intuitively grab, move, and rotate both on a flat surface and in free space [16]. For an automatic transition, the device exhibits a IR reflective light sensor on the bottom so that it can automatically reconfigure with respect to contact with a surface.

Support for Wireless Operation. A versatile gesture input device needs to be completely wireless so that the user can carry the device to another location and operate it freely in space. To achieve this goal, the main challenging problem is to track the 3D position of the device. We use cameras and detection algorithms from computer vision.

The visual sensor approach relies on the accurate detection of relevant features which is a challenging task under varying illumination and environmental conditions. Moreover, it is not a-priori clear if the required tasks for a versatile gesture input device can be accomplished given their inherent level of inaccuracy. Due to these reasons, we use a bright color LED which provides focal brightness on the captured images, thereby achieving easier and more robust tracking.

Support for Multifunctionality. A generic and versatile input device should support multi-functional operations so that the user does not have to change the input device between different tasks. We identify a set of required operations for a versatile gesture input device: gesture commands, navigation, manipulation of virtual objects, and tool selection.

The use of gesture commands has been motivated for various purposes such as appliance control in smart environments [19] and game control [14, 1]. A gesture input device is required to provide enough features to discriminate between different gesture commands. For this, we combine visual sensors and embedded sensors with a proper feature extraction methods.

We compute 6-DOF information combining the visual and embedded sensor data so that the device can be used to manipulate and navigate virtual objects. Manipulating virtual objects requires a certain degree of precision, which can be challenging to achieve with a vision-based approach. We facilitate this problem by using a bright color LED. Finally, for efficient tool selection, we allow the user to control e.g. a pie-menu by rotating a top handle attached to the cube.

Support for Design Variations and User Definable Ergonomics. These days, many commercial electronics such as MP3 players and mobile phones are designed consider-

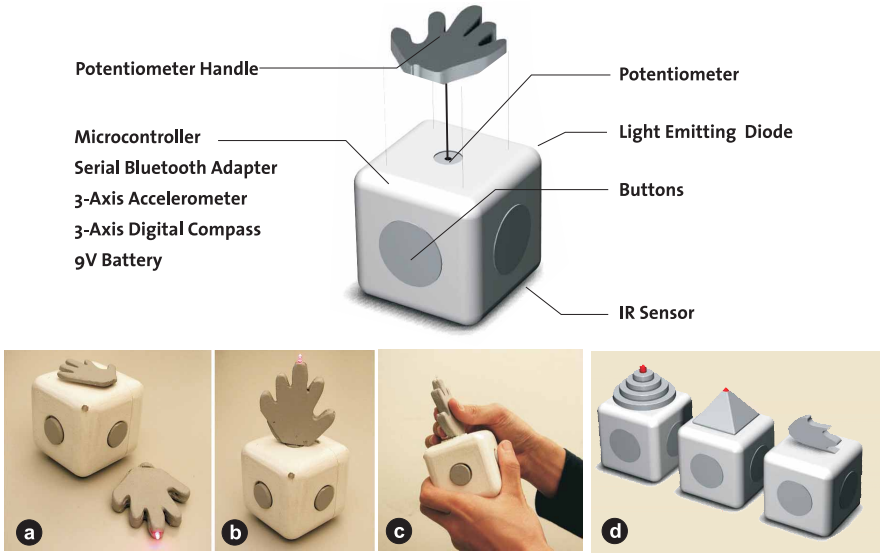

Fig. 2. (Top) Hardware configuration of the mCube. The device includes multiple sensors such as accelerometers, digital compasses, four buttons, a rotary potentiometer, an infrared (IR) distance sensor, and one color LED. In addition, it contains a micro-controller for acquisition, a Bluetooth transmitter, and a 9V battery power supply. (Bottom) The top handle can be easily replaced with other designs according to the user's preference. (a) A flat style top handle using the hand metaphor. (b) A vertical style top handle using hand metaphor. (c) The operation of the vertical style top handle with both hands. (d) Top handle design alternatives using different metaphors.

ing the user's desire to change the appearance and ergonomics of the device. We also consider these issues as a principle of a versatile gesture input device.

2.2 mCube Hardware Design and Sensor Configuration

Based on the proposed design principles and solutions, we developed the prototype device mCube. The mCube consists of two units: a *body* and a *top-handle*. The body is a cube with an edge length of 6 centimeters as shown in Figure 2. For the top handle, various designs can be provided so that users can choose according to preference for design, application, and ergonomics. For instance, by attaching another top handle in an upright position (Figure 2b and c), the mCube can be easily re-configured for two-handed interaction.

The mCube is designed to sense various events (button clicks, handle rotation, and surface contact) and the physical manipulation of the device (translation, rotation and tilt). We labeled the six sides of the body with top, bottom, front, back, left, and right.

We attach a button to each side face of the body such that the user can intuitively use them, mapping the button/handle direction to the user direction. A potentiometer is located under the top side, and senses the rotation angle of the top handle which is connected to the mCube body through a rotary shaft. On the bottom side, an IR reflective light sensor is located to sense the distance to the planar surface from 0 to

about 3 centimeters. It emits a small beam of invisible infrared light and measures the amount of light that is reflected back to the sensor. Since the sensor is sensitive to ambient light, the sensor is shielded. One color LED is attached to a corner of the mCube body to provide a bright color spot in the acquired camera images at a wide range. As illustrated in Figure 2, a LED can be installed to top-handles. In this case, the LED on the cube can be disabled using a LED switch located on the bottom side.

Acceleration is measured with two Memsic 2125 2-axis accelerometers attached in orthogonal configuration measuring dynamic acceleration (vibration) and static acceleration (gravity) with a range of $\pm 2g$ at a resolution higher than $0.001g$. Two Hitachi HM55B 2-axis digital compasses are integrated perpendicularly to provide information on the Earth's magnetic field in three dimensions at 6-bit (64-direction) resolution. A Javelin Stamp micro-controller with 32k of RAM/program memory is used to read sensor data using delta-sigma A/D conversion. The raw sensor values are then transmitted wirelessly from the device to the host computer using a F2M01C1 Bluetooth module offering a nominal range of approximately 100m. The complete system operates at 9V. Note that the current version is a prototype that can easily be miniaturized for production.

3 System Implementation

In this section, we give a brief overview of the mCube system setup and the core software modules for computing the rotational and positional information of the device. We also explain the extraction of gesture features from the combined sensor data.

3.1 Overview

The overall system consists of a pair of video cameras (USB or Firewire), the mCube device, a computer with active Bluetooth, and display devices (projectors and/or monitors) (Figure 3). In our current prototype setup, two firewire cameras are used to acquire the range of operation at 30 frames per second. The acquired images are processed to determine the 3D position of the mCube LED (§3.2). Synchronously, the mCube acquires the data of the embedded sensors at a sample rate of 60 Hz and transfers the data to the host computer over the Bluetooth network. On the host, the sensor data is directly read from a communication port using the *java.comm* library. In the preprocessing stage, the acquired data amplitude is scaled using linear min-max scaling. The visual features are derived from the sensor data with appropriate up-sampling in order to match the frequency of the embedded sensor data and low-pass filtering to remove noise.

3.2 Computing the Rotational and Positional Data

We acquire the rotational data of the device combining the values of the compasses and accelerometers. The main idea is to take the output of the accelerometers as pitch and roll, and use the output of the digital compasses to compute the yaw. To get appropriate yaw information from the digital compasses, we use the accelerometer data to compensate for the erroneous compass data. This technique is widely used in hand-held electronic compass applications. We refer to [10] for details on the computation.

Fig. 3. A prototype setup with a pair of video cameras, the mCube device, a desktop monitor, and a large display wall for wall projections.

As mentioned earlier, tracking the 3D position of the device using cameras is a challenging task because lighting and background conditions can change over time. We use a bright color LED for faster and more robust tracking. A focal brightness provides relatively robust tracking results even for small-scale movements in indoor environments. Moreover, using different color LEDs allows to track several devices at the same time for multiple users.

We developed a simple vision tracking algorithm to find the pixel positions within a certain color and brightness range. To compute the 3D position of the marker, we employ conventional triangulation from a pair of calibrated cameras [21], [13].

3.3 Extraction of Gesture Features and Rotational Invariance

In our context, gesture features should be robust and also invariant with regard to translations and rotations [4]. Such gesture features can be achieved by combining accelerometer data and visual features from the cameras [2].

For visual features, we tested a variety of invariant features derived from the visual sensor, including angular velocity, curvature, and velocity. However, all of these features have limited reliability and do only work for larger and longer gestures. The main reason for this lack of robustness is the differential nature of these features including computations of the first and second derivatives. This leaves us with the relative Cartesian positions (rx, ry, rz) computed with respect to the start position of the gesture. The relative position is translationally invariant, but cannot compensate for rotations. This type of invariance is achieved by adding data from the accelerometers to the extraction.

4 Interaction Techniques

In this section, we describe an exploratory set of techniques intended to investigate as thoroughly as possible the design space of mCube interactions. Some of these techniques can be selected and optimized for any application that seeks to use the mCube.

4.1 Switching Between Desktop and Hand-held Interaction

As described earlier, the mCube uses the affordances of cube shape which can be operated on a flat surface and in free space. To facilitate this operation, we program the

mCube to automatically recognize the current interaction mode based on the contact between the device and the surface of the table using the embedded IR sensor. If the value is lower than a desktop threshold (1 cm) for a certain amount time (2 sec), the interaction mode is changed to desktop interaction.

Using this feature, users can optimize the interaction for the required task in the current application and display setup. For instance, for manipulation tasks requiring more precise control, users can use desktop interactions minimizing the hand tremor which can be caused by the lack of fixed support in hand-held interaction. Alternatively, users can choose hand-held interactions to perform direct 3D inputs and operations in free space.

4.2 Top Handle-based Mode/Tool Selection

The mCube allows users to switch between different modes or select a tool on a pie-menu by simply rotating the top handle. The main idea is to use the highly intuitive operation of the pepper caster which we use almost every day to grind and scatter pepper on our dish. During this operation, we usually hold the bottom part and rotate the top part without any substantial previous learning. The main benefit is that users can use this handle during other operations. For instance, during navigation or manipulation of virtual objects, users can choose different tools such as rendering modes (wireframe or shaded rendering) or texture and color styles of the model without interfering with the positional control of the device.

We use the top-handle to define the device mode. We defined three necessary modes: an *idle* mode for power saving, a *recognition* mode for gesture recognition and a *pie-menu* mode for selecting different tools. Figure 4-b illustrates an example of a pie-menu with twelve icons. During rotation of the top handle, the widget is displayed and the designated sub menu is highlighted depending on the direction of the handle as shown in Figure 4. Different numbers of icons can be used by dividing the circular region of the top handle into a corresponding number of sections.

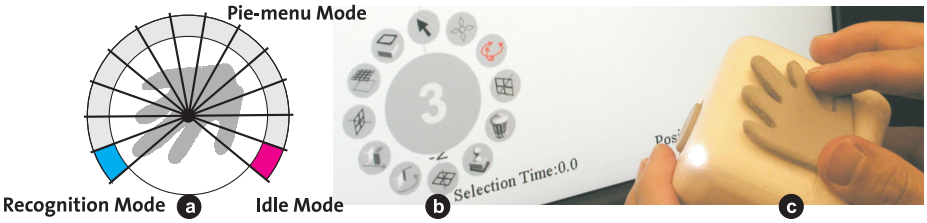

Fig. 4. (a) Three modes of the device controlled with the top handle: recognition mode, pie-menu mode, and idle mode. (b, c) In the pie-menu mode, users can select a tool with a modified pie menu by rotating the top-handle. The circular region of the pie-menu is divided into 12 regions for 12 icons of the pie-menu. The number 3 in the center of the pie menu indicates that hand-held interaction is currently activated.

4.3 Examples of mCube Gestures for Command Inputs

We designed a multipurpose set of gestures with the mCube exploring the intuitiveness and interoperability of the mCube in desktop and hand-held applications.

Gestures for Desktop Interactions. For desktop interactions, we designed a set of gestures which are suitable using a *glass metaphor*: pouring water in three directions (front, left, and right), twirling and rotating in CCW and CW as illustrated in Figure 5-a. These gestures can be easily learned because of their familiarity and intuitiveness from the operation of a real glass. To perform the glass gestures, users place the mCube on a surface and then perform the gestures by lifting the device from the surface. This simple initialization motion greatly disambiguates the recognition based on the IR-distance value for each gesture.

Gestures for Hand-held Interactions. One of the common tasks of hand-held interactions is to select a physical or virtual object using a pointing gesture and control it with subsequent gestures [19]. For this purpose, we designed a set of hand-held gestures targeting the commonly used actions on appliances, e.g. rotating the handle to turn on/off and change the volume (up/down).

In addition to the previous simple gestures, more complex 3D spatial gestures can be performed using the mCube. While simple gestures can be recognized with a heuristic approach which looks for simple trends or peaks in one or more of the sensor values [19], for complex gestures, we need advanced pattern matching techniques. Various recognition algorithms are proposed using statistical pattern matching techniques [18, 4], and multiple sensors are combined to provide better discriminating gesture features

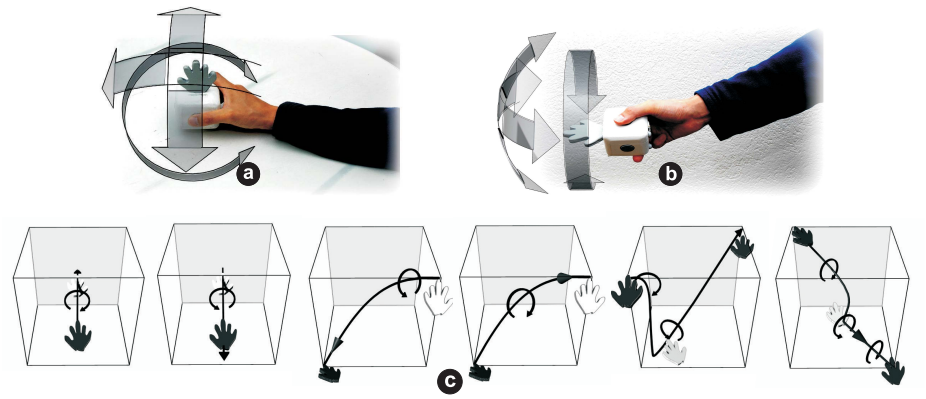

Fig. 5. (a) Examples of desktop gestures. A user is performing gestures on the desktop surface: pouring-left, -right, -front, and rotating-CW, -CCW. (b) Examples of hand-held gestures. A user is holding the device in the air and performing gestures: pointing-up, -down, -right, -left, and rotating-CW, -CCW. (c) The 3D spatial gesture examples with a box style 3D gesture volume. The line indicates the trajectory of the gesture and the end of the gesture is presented as an arrow. The hand symbol indicates the direction and rotation of the mCube using black for a palm-down position and white for a palm-up position. The 3D gestures are increasing in complexity from left to right.

[2]. Even though their explanation is considered outside the scope of this paper, it is assumed that the combined gesture features of mCube will improve the recognition and enable a larger gesture space. It is also important to consider human variability exhibited in 3D spatial gestures due to the difference in user performance [14]. During our developments, we found that the illustration of 3D spatial gestures plays an important role in minimizing the human variability and improving the recognition rates. Figure 5c illustrates examples of our 3D gesture diagrams for the mCube device.

4.4 Multi-dimensional Manipulation and Navigation

Virtual Object Manipulation. In desktop interaction, the object is controlled in the same way as a computer mouse. We implemented a similar implicit clutching using the IR distance sensor. For instance, when the user lifts up the device, the movements of the device do not affect manipulation. Users can also rotate the virtual object by physically rotating the device which is not possible using a conventional computer mouse as shown in Figure 6.

During hand-held interaction, the device is operated in space mapping the horizontal and vertical movement of the device to the corresponding object position. Therefore, users can directly move, rotate and tilt the object in any direction for 3D manipulation (Figure 6d). An explicit clutching technique is used with the right button of the mCube similar to other commercial 3D input devices.

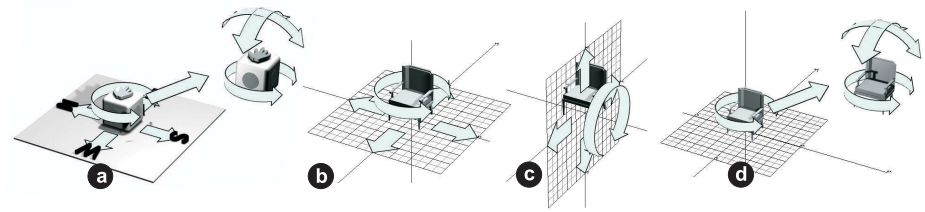

Fig. 6. Examples of virtual object manipulation: (a) The mCube is operated on the desktop surface and in space. (b, c) In desktop interaction, the object is translated and rotated on a working plane e.g., $x-y$ plane and $x-z$ plane. (d) In hand-held interaction, the object is operated with additional dimensions in space.

Virtual Space Navigation. To facilitate more efficient navigation, we use the four buttons as well as the rotational control of the device. The front, back, left and right buttons are used to control the movement of cameras. This idea is inspired by the *Cubic Mouse* which successfully utilizes a cube shape as an intuitive physical proxy of virtual objects in navigation purposes [8].

As illustrated in Figure 7, we implemented two navigation schemes: *examine viewer* to control a 3D virtual object like a 3D trackball and *walk viewer* to navigate through 3D virtual space with a walking metaphor. In the examine viewer, the rotational control of the mCube is used to rotate a virtual trackball. The virtual camera is located outside of the model pointing to the center of the model. The rotational movement of the device

is mapped to a virtual camera angle. The front and back buttons are used for zooming in and out, respectively. In the walk viewer, users can look in a certain direction while moving in another direction similar to the walking navigation in real world.

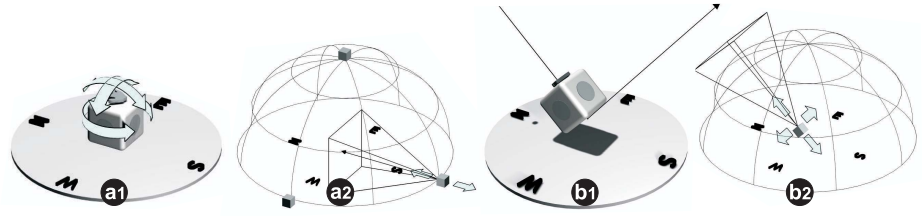

Fig. 7. Examples of virtual space navigation: examine viewer and walk viewer. (a) In the examine viewer, the virtual camera is rotated around the scene pointing to the center while manipulating the mCube. To zoom in and out, the front and back buttons are used respectively. (b) In the walk viewer, the virtual camera is moved and oriented based on button clicks and the orientation of the mCube.

5 User Evaluation

We conducted a user evaluation to determine perceived exertion and movement characteristics when the mCube is used in desktop and hand-held interaction. We invited six participants who have strong backgrounds in computer science and computer graphics. Each participant was given a demo of how the system and interaction techniques work. Then they practiced all implemented functions. All users got used to the system after short practice (about 15 minutes).

The top handle was easily understood and performed well. All participants felt comfortable when rotating the top handle in both desktop and hand-held interactions. As we intended, they could select a tool on the pie-menu during other operations such as navigation and manipulation. Specially in a large screen display, using the pie-menu with the top handle, users can minimize the use of cursors which might be difficult to precisely control therein. Some users stated that they could remember the item locations on the pie-menu and rotate the top handle even without looking at the widget. We expect that this can be a powerful mechanism minimizing the cognitive loads during interaction and may be an interesting topic for a more extensive user experiment.

To explicitly test the user capabilities of desktop and hand-held interactions and switching between them, we performed a simple 3D docking task. Each subject was asked to perform several recurrences of the same task to locate one cube to another cube positioned at different positions in 3D. Between the recurrences, users were asked to perform the same tasks alternately in both interaction modes. During this task, users can easily return to desktop interaction by putting the mCube onto the table and lifting it up for further hand-held interaction. As we expected, in the hand-held interaction, we clearly noticed that fine positioning was hard to accomplish even with the intuitive 3D operation in the air. Almost 50% of the task completion time was taken for the final

placement. These problems are mainly due to trembling of the user's hand and further caused by the well-known fatigue problem in hand-held input devices [20].

Some users felt uncomfortable in their grip when operating the mCube with only one hand. This feedback shows that careful ergonomic studies are required for designing the shape of the next prototype. In addition, the weight (140 g) of the device was considered still a bit heavy especially when users performed long hand-held interactions only.

6 Future Directions

The mCube is a promising prototype to understand the capabilities and limitations of a versatile gesture input device. We intend to perform a profound usability study to assess potential user acceptance of a versatile gesture input device with various target applications.

We also investigate the design variations to gain deeper insights into the perceptual issues in interacting with this class of input devices. Figure 8 shows a ring style mCube called *cubeRing* designed to be wearable on the finger. The cubeRing is designed with a highly unobtrusive form which minimizes a fatigue problem while keeping the required design principles for a versatile gesture input device. For instance, the flat bottom surface of the cubeRing provides a stable operation on the desktop surface, and the unique combination of a ring with a cube provides comfortable grip during the use in both the desktop and hand-held interactions.

To realize the small size, we use a coin size MICA dot module which is designed around a stacked configurable circuit board architecture, with a high-bandwidth transceiver and a general microprocessor, and a sensor board [6]. The cubeRing is equipped with two color LEDs one on the top side and another on the bottom side to enable robust tracking even with a hand rotation.

Fig. 8. (a) The cubeRing design with an adjustable finger ring on the top, and a small cube on the bottom. The cubeRing can be worn in the index finger, and operated on the desktop surface (b) and in hand-held position (c).

7 Conclusion

In this paper, we proposed a novel versatile gesture input device called mCube which supports various types of gesture inputs in desktop and hand-held interactions. We defined a set of design principles and solutions for the developments of a versatile gesture input device, and introduced our first prototype which integrates all necessary sensors.

Combining visual and embedded sensor data has been proven to be useful for developing a versatile gesture input device supporting robust features for gesture recognition. We demonstrated a variety of interaction techniques namely tool selection, gesture commands, navigation, and manipulation which can be useful in a wide range of applications.

Acknowledgments. This work was carried out in the context of the blue-c-II project, funded by ETH grant No. 0-21020-04 as an internal poly-project.

References

1. *Wii*. http://wii.nintendo.com/.
2. Helene Brashear, Thad Starner, Paul Lukowicz, and Holger Junker. Using multiple sensors for mobile sign language recognition. In *Proceedings of ISWC '03*.
3. S. Brewster, J. Lumsden, M. B., M. Hall, and S. Tasker. Multimodal 'eyes-free' interaction techniques for wearable devices. In *Proceedings of CHI '03*, pages 473–480, 2003.
4. Lee W. Campbell and David A. Becker. Invariant features for 3-d gesture recognition. *Second International Workshop on Face and Gesture Recognition*, 1996.
5. X. Cao and R. Balakrishnan. Visionwand: interaction techniques for large displays using a passive wand tracked in 3d. In *Proceedings of UIST '03*, pages 173–182, 2003.
6. Crossbow Technology., http://www.xbow.com/.
7. Julien Epps, Serge Lichman, and Mike Wu. A study of hand shape use in tabletop gesture interaction. In *CHI '06 extended abstracts*, pages 748–753. ACM Press, 2006.
8. Bernd Frohlich and John Plate. The cubic mouse: a new device for three-dimensional input. In *Proceedings of CHI '00*, pages 526–531. ACM Press, 2000.
9. L. Jr. Quill: a gesture design tool for penbased user interfaces, 2001.
10. Kionix. *Handheld Electronic Compass Applications Using the Kionix KXM52 MEMS Tri-axis Accelerometer*. http://kionix.com/App-Notes/app-notes.htm.
11. K. V. Laerhoven, N. Villar, A. Schmidt, G. K., and H. Gellersen. Using an autonomous cube for basic navigation and input. In *Proceedings of ICMI '03*, pages 203–210, 2003.
12. S. Malik and J. Laszlo. Visual touchpad: a two-handed gestural input device. In R. Sharma, T. Darrell, M.P. Harper, G. Lazzari, and M. Turk, editors, *ICMI*, pages 289–296. ACM, 2004.
13. OpenSource Computer Vision Library. Intel Corp., http://www.intel.com.
14. J. Payne, P. Keir, J. Elgoyhen, M. McLundie, M. Naef, M. Horner, and P. Anderson. Gameplay issues in the design of spatial 3d gestures for video games. In *Extended abstracts in CHI '06*, pages 1217–1222, 2006.
15. J. Rekimoto and E. Sciammarella. Toolstone: Effective use of the physical manipulation vocabularies of input devices. In *Proceedings of UIST '00*, pages 109–117, 2000.
16. J. G. Sheridan, B. W. Short., K. Van Laerhoven, N. Villar, and G. Kortuem. Exploring cube affordance. In *Proceedings of Eurowearables 2003*, 2003.
17. Marc Weiser. The world is not a desktop. *interactions*, 1(1):7–8, 1994.
18. Tracy Westeyn, Helene Brashear, Amin Atrash, and Thad Starner. Georgia tech gesture toolkit: Supporting experiments in gesture recognition. In *Proceedings of ICMI'03*, 2003.
19. A. Wilson and S. Shafer. Xwand: Ui for intelligent spaces. In *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, pages 545–522, 2003.
20. S. Zhai. User performance in relation to 3d input device design. volume 32, pages 50–54, 1998.
21. Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of Computer Vision'99*, pages 662–673, 1999.
22. T. G. Zimmerman, J. Lanier, C. Blanchard, S. Bryson, and Y. Harvill. A hand gesture interface device. In *Proceedings of CHI '87*, pages 189–192, 1987.