Physical Shortcuts for Media Remote Controls

Alois Ferscha, Simon Vogl
Studio Pervasive Computing Applications
Aubrunnerweg 1
A-4040 Linz, Austria
{alois.ferscha, vogl}@researchstudios.at

Bernadette Emsenhuber, Bernhard Wally
Department of Pervasive Computing
Altenberger Strasse 69
A-4040 Linz, Austria
{bem, wally}@pervasive.jku.at

ABSTRACT

The usability of remote controls for home entertainment systems like TV sets, set-top boxes, satellite receivers and home entertainment centers has reached overstraining complexity: about eight to ten remote controls with about sixty to eighty push-buttons each are typical for a home entertainment system setting today. To be able to harness the ever growing remote control interaction complexity, we propose *physical shortcuts* to express the most frequently used control commands. Embodied into an orientation aware artifact which serves as a tangible user interface, physical shortcuts are articulated as hand gestures by the user, and converted into control commands compatible with the built in infrared receivers of standard consumer electronics.

Starting with an analysis of the kinematics of the human hand, the types of grip and its correlation with the size and shape of an object which the hand grasps and holds, we study different tangible interface geometries with the potential to serve as a physical shortcut interface, and thus as a complementary remote control. Besides cubical and cylindrical artifact geometries of different sizes, also hybrid shapes are investigated with respect to their affordance, i.e. the action possibilities of an artifact readily perceivable by an actor. For the final cube like tangible interface design, ATMega168 microcontroller based electronics involving a three axis acceleration sensor and a gyroscope, together with low power IEEE 802.15.4 wireless communication components have been developed. A finite state machine based software architecture is deployed for artifact based hand gesture recognition, and table driven issuing of IR remote control commands. Finally, a fully functional cube remote control, the TA cube, is presented as a tangible remote control for an IPTV set-top box.

Keywords

Tangible User Interfaces, Media Control, Low Power, Wireless Communication

Categories and Subject Descriptors

H.5.2 [**Information Interfaces and Presentation**]: User Interfaces—*Haptic I/O, Interaction Styles, User-Centered Design, Prototyping*

General Terms

Design, Human Factors

1. INTRODUCTION

Remote controls for home entertainment systems, such as television sets, sound systems and set-top boxes, are designed according to a one button per function paradigm. Function overload of modern entertainment systems hence makes button based remote controls a rather confusing user interface. While some of the buttons are not used at all and some are used occasionally, there are usually a few functions that are used frequently: hopping channels/stations (TV or radio), controlling the volume and switching on/off. Recent television platforms like IPTV set-top boxes, additionally provide a graphical user interface in order to navigate through a hierarchical menu structure, demanding even more buttons or yet another remote control.

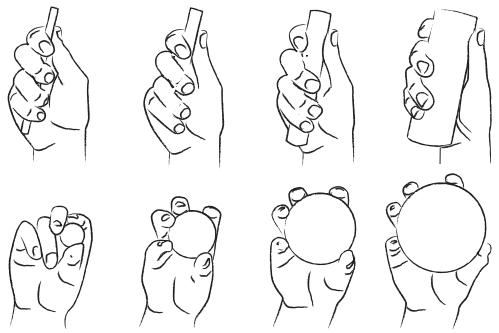

Figure 1: Grip-kinematics and capabilities of the human hand: power grip (above) and precision grip (below) [10].

Inspired by the observed inadequacy of the button-based remote control designs with respect to frequently invoked control commands, we have started to investigate on alternative control designs. As one such alternative we propose *physical shortcuts*, allowing to issue control commands with gestures natural to the human hand. These physical shortcuts are implemented as gestures for tangible artifacts, requiring a convincing affordance and a simple but sufficiently versatile

gesture set. Operating traditional remote controls follows a certain pattern, illustrated with the example of watching TV: (*i*) grabbing the remote control to switch on the TV, (*ii*) pushing some buttons while watching TV, (*iii*) putting away the remote control when done. Analyzing this informal interaction protocol reveals that the human hand already undertakes a lot of actions prior to the intended launching of a command, like grasping, holding or turning it to a face-up position. These hand gestures, are already expressing intent for a command launch, which could already be used to invoke the command itself. This observation and the design motivation of making command invocation easier and quicker, encourages a the use of tangible artifacts that can be manipulated using one's hands and that support a "grab-to-switch-on" functionality. An important essential in the design process for a tangible remote control is the functioning of the human hand when it comes to grip and control: a human hand can fundamentally execute two different kinds of grip [10]: a power grip and a precision grip (cf. Fig. 1).

This paper is organized as follows. We first devote a more rigorous analysis of the literature which looks at remote controls from a tangible interfaces point of view (section 2). Motivating and implementing our own design is covered in section 3, presenting usability and interaction related design issues, and section 4, presenting physical implementation details of the design in hard- and software. Finally, qualitative results are presented in section 5.

2. RELATED WORK

Tangible interfaces have a broad coverage in the human computer interaction literature, and various concepts as well as prototypes of tangible interfaces have been presented. While tangible objects such as augmented toys incorporate both form and function and therefore clarify the interaction style [6], it is much harder to design tangible artifacts for abstract tasks such as controlling a media center. In [3], for instance, so called "navigational blocks" (wooden cubes containing a microprocessor and labeled with a unique ID) are proposed to be flipped to either of their six sides in order to query information about certain elements in a virtual gallery. In [14] a commercial mobile phone is enhanced with near field communication capabilities and an accelerometer in order to control a personal computer using simple gestures. A cylindrical tangible user interface with embedded displays and sensors, TUISTER, is presented in [1]: upper and lower half of the cylinder can be twisted against each other, enabling interaction with respect to the absolute space orientation in order to infer which one of the two halfs was twisted (to differentiate between fine grained and coarse browsing in hierarchical structures). [4] gives an example on how to control a PC's media player using a tangible artifact incorporating accelerometers, magnetometers and gyroscopes, with respect to a pairing mechanism (the artifact allowed to sequentially control more than one actuator using RFID tags). A tangible media control system is also presented in [13], allowing to control objects (such as a cube that can be flipped to each of its sides) augmented with RFID tags and a tracking system to control e.g. a software midi synthesizer. [2] shows the results towards remote control in a living room through tangible user interfaces. One of the projects, Flip'n'Twist, uses a cube and a dial for media control by flipping the cube to its different sides and turning the dial. Each side of the cube sets the media control system in a distinct state (such as play, seek and volume control) and lets the user utilize the dial for fine grained operations. [15] classifies 13 different computational toys that can be considered tangible user interfaces. The five cube shaped artifacts allow various interaction styles, such as stacking, shaking, turning, flipping and touching, regarding adjacency, sequence and network topology of multiple devices if applicable. In [9], a cube comprising accelerometers and a proximity sensor is presented that can determine which side is facing up, whether one of three predefined gestures was performed and if the side facing up has changed (transition). Our prototype can distinguish more then these states and transitions as it incorporates an additional gyroscope in order to track rotations around the vertical axis. To keep power consumption low, the gyroscope is powered only when it is needed (see section 4.1). A similar approach regarding the underlying technology was taken in [16] where two accelerometers were used to distinguish which side of a cube shaped tangible user interface was facing up. The cube comprised a display on each side and a speaker so that it was well suited for applications such as quizzes, a math/vocabulary trainer and a letter matching game.

Observations of our previous prototypes for media control using (amongst others) a cube as the tangible artifact [5], we have realized that a system relying on a cube that can be flipped to its sides to distinguish among different modes of operation requires the user (a) to be very skilled in the usage of the cube or (b) to have a look at the cube each time interaction occurs in order to find out where to flip it, if the cube has corresponding annotations. It is easy to get confused and lose track of the current state and on which side to flip next. Additionally, it is hard to find a certain side of the cube if it is currently facing down or away from the user. By turning or flipping the cube to find the corresponding side, unintentional interaction could occur, leading to disaffection and desperation.

Thus it appears useful to use other gestures for cube interaction than flipping—we have therefore implemented a gesture library for media control using a cube with a labeled button on top. That way, the cube has a base orientation with the button facing up and the label being readable by the user. The possible gestures include tilting the cube to the front/back/left/right, pressing the button and turning the cube around the z-axis (the one heading upwards "through" the button). Each gesture leads back to the base orientation thus providing the user a known situation to start from.

In [8] an accelerometer based gesture control for a design environment is presented that allows users to map arbitrary gestures to certain functions (personalization). Besides controlling a VCR by supporting commands such as on, off, play, stop, forward and many more the gesture control system is also suited to navigate in a 3D design software. An enclosed user study shows that different users use different gestures for a certain command—for instance at least 20 gestures were mapped to the VCR record task by the test persons. While personalization is an important issue, our prototype defines a fixed set of gestures for simplicity's sake: users can execute the tasks they want to accomplish (controlling a TV set) instead of personalizing their tangible user

interface. Furthermore our cube is a remote control that is probably used by more than one person, thus it is easier to provide only one gesture set instead of forcing the users to identify themselves somehow to load their specific gesture sets. Nevertheless, personalization will be addressed in further projects, starting with personalizing the graphical user interfaces of IPTV set-top box menus according to users' behaviors.

3. DESIGN

Aside aesthetical and haptical issues, the design of a tangible remote control has to address the aspects of simplicity, affordance and focused functionality. As for the appearance related issue of a remote control, we follow the well established norm of product design proposed by Norman in [12].

3.1 Simplicity

Simplicity means that the usage of a product, an artifact, a tangible remote control should be easy and quick [12]. Every user should be able to handle the interface intuitively and without any training time. There shouldn't be any barriers taking an interface and using it. To avoid such barriers we observed people in public places and other familiar environments, focused on physical objects that people play with when they work, relax, talk or think.

We chose a cafe as such public place because it unifies the situations of working, relaxing, talking and thinking in one room. Our observations revealed that people tend to grasp and play with handy objects decorating their table. We observed that they perform different gestures with the can of their drink (cf. Fig. 2(a)). They turn it horizontally, vertically or shake it. Turning distinguishes itself as a smooth and endless movement corresponding to continuous input. But people also take cigarette packets or mobile phones and flip them on their different sides, rotate them or shake them (cf. Fig. 2(b)). These movements corresponds to discrete input which is typical for cuboids.

Figure 2: Experimental studies of handling well-known objects: (a) can, (b) cigarette packet.

Based on our observations we decided that users should handle our tangible user interface as intuitively as if it was a can or a pack of cigarettes—the user should be able to easily perform basic gestures such as rotation or flipping. To support this, the interface needs an affordance similar to a can or a cigarette packet.

3.2 Affordance

Affordance is a main concept in the industrial design process, as it is in many other disciplines. It means that an object which should be used by a human must invite to be taken and handled it in a purposeful way. The object design should force the user to employ it correctly [11].

To observe the affordance of different shapes, we developed a number of 3D printed prototypes (cf. Fig. 3). Experiments with adults and children demonstrated which shapes lead to which gestures: The cube has a pleasant size for a hand and encourages the user to grab, flip and rotate it. During our tests we also found that people pressed their fingertips into the holes at the cube's edges. The two red knobs animate the user to turn and drag them across the surface of a table. The two knobs distinguish only from how they were touched. Knob (b) was touched and moved with forefinger and thumb, knob (d) with the whole hand. The green cuboid forces test persons to press on the integrated touch screen, rotate it around the rounded vertical edges or flip it.

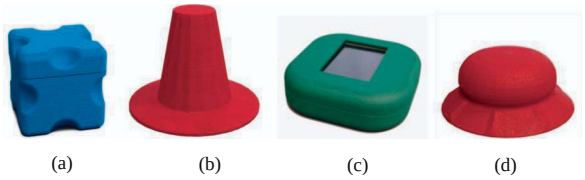

Figure 3: Prototypes for experimental affordance studies [4].

Our next step was defining the target group of our interface. We specified the typical remote control operator as an ordinary user of an entertainment system without any technical knowledge and motivation to handle a button glutted remote control. Also, the user is probably confronted with the challenge of finding the proper remote control (between all the others) for controlling the entertainment center. We thus decided to add another requirement: our artifact should grab the user's attention in order to be preferred over a standard remote control and it must have a size so that it can be easily grabbed with one hand to further increase the affordance of taking it into one's hand.

We studied appearances with different interface designs to determine how discrete and continuous motions could be mapped to 3D primitives. Fig. 4(a) shows an equilateral cube with rounded edges and corners. The cube can be flipped to every face as discrete motion and turned around every axis as continuous motion. Fig. 4(b) depicts a cylindric shape like a can which can be continuously rotated around the x- and the z-axis. It is also possible to flip it over the horizontal edge as a discrete movement. The hexagon, as illustrated in Fig. 4(c), has an arched bottom with six faces. The user topples the interface by tipping it on the top as discrete move limited to six states. A rotation about the z-axis is possible, but the edgy form forces a gradual rotation and not a continuous one.

Tests with the cube, the can and the hexagon determined that the cube is the best shape for our intentions. The circumcircle of the cube faces provides the constant radius for the turning motion, the rounded edges support the user to flip the object. In order to find out more about the affordance and ease of use of cubes we designed a set of cube versions. Fig. 5 (a) shows a cube which can be twisted in

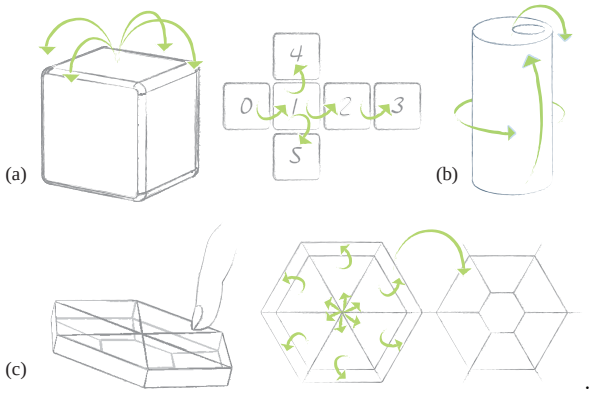

Figure 4: Gestures alphabets for tangible artifacts: (a) cube, (b) can, (c) hexagonal.

the middle. Version (b) is a cube with holes at the vertical edges for a safer grip and orientation. Model (c) has a flat corner which lets the cube be positioned in stable state, indicating the initial system state. As forth design concept we created a cube with a sash positioned at the golden section of the cube as orientation support (cf. Fig. 6) and a pedestal for bringing the cube to its "standby"-state (cf. Fig. 7).

After developing several virtual cube models we have produced physical models to analyze usability and technical realizability. The next step was excluding the counter-rotating cube design (fig. 5(a)) because it affords bi-manual handling which conflicts with the single-hand grip-kinematic of a human hand. The grasp supporting form in fig. 5(b) confused the users orientation, because it's hard to distinguish top and bottom. Finally the cube design in fig. 6 stood up to cube (c) because it was most adequate to the defined functionality (cf. Fig. 7), supported the single-hand grip-kinematic and is based on one orientation possibility.

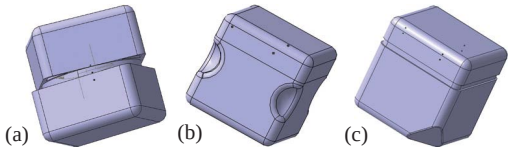

Figure 5: Cube designs highlighting individual design aspects: (*a*) counter rotation, (*b*) grasp support, (*c*) stable initial state.

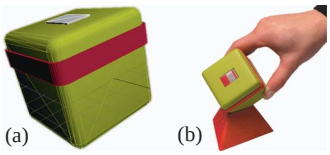

Figure 6: Final artifact design.

3.3 Focused Functionality

Having in mind a tangible remote control that is supplementary to a vendor provided remote control control, with the aim to quickly invoke the most frequently demanded commands, an analysis of a minimum meaningful set of commands revealed to focus on changing volume, switching the channel or navigating the menu, switching ON and OFF and bring the system to its origin state.

These functions were mapped to gestures which can be effected by the final cube design (cf. Fig. 7): flipping up and down, turning left and right, shaking and resting. As additional function and orientation support the cube was equipped with a push-button to switch between the set-up mode and the operation mode of the set-top box. Operating mode functions are volume and channel change which are activated by pressing the mode push-button. To change the volume the user has to rotate the cube horizontally. A clockwise rotation increases the volume, an anticlockwise reduces it. Switching the channel is caused by flipping the cube up and down. These gestures are also used in the set-up mode without pressing the push-button to navigate through the menu. To set the set-top box in standby mode the cube must be put on the pedestal. Shaking the cube left-right navigates to the menu home.

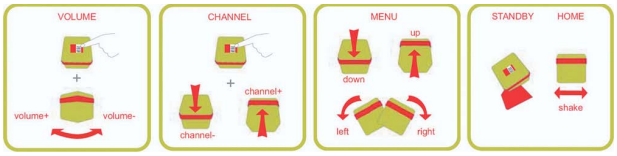

Figure 7: The operating alphabet of the TA cube.

4. HW/SW DESIGN

The cube platform consists of two different hardware components: The cube itself, which is responsible for gesture detection, and a converter box, which converts the gesture commands into infrared signals compatible with standard home entertainment equipment. We have designed a simple but flexible protocol to transmit gestures from the cube to the converter box, which maps these commands to IR codes using a lookup table. We use the protocol to transport data for other applications as well, like the pairing process between cube and converter, or for monitoring a stream of raw sensor data.

4.1 Cube Platform

After an analysis of the necessary gestures, we have assembled an initial prototype to test the set of sensors we have intended to use by extending a MicaZ-board, which we selected as our first platform after evaluating the power requirements and the radio subsystem. (cf. Fig. 8)

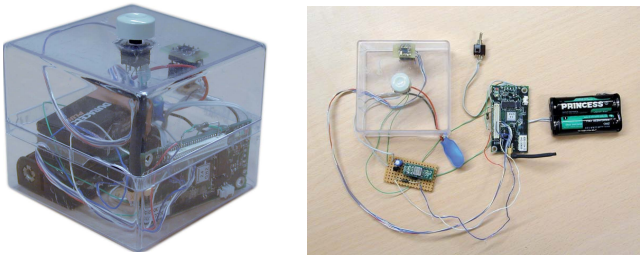

Figure 8: Initial Cube Hardware Prototype.

While we have found that this board is the right system in principle, it actually offers too many features along with

a too large board footprint. We have therefore selected a smaller Atmel CPU (an ATMega168 in a 32-pin package) as our system core. The final PCB we have produced needs less space, hosting all components on a single-side, dual-layer 38x48mm board (cf. Fig. 9).

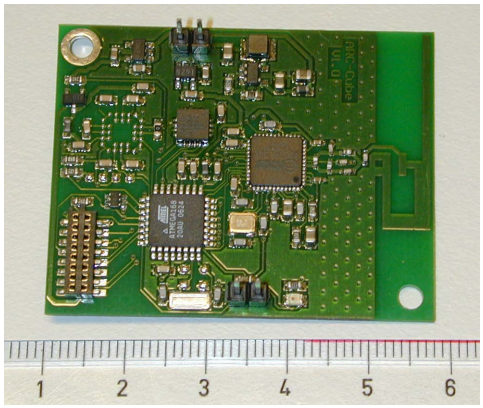

Figure 9: The final PCB—power supply and sensors are in the top left corner.

The board size has been determined by the cube design, which has provided us with enough slack to place all components on a single side. As can also be seen in the schematics (cf. Fig. 10), the board consists of four functional blocks: The microcontroller and expansion bus section, the sensors, the radio subsystem and the power supply.

4.1.1 Microcontroller Subsystem

We use an ATMega168 with an external 32kHz oscillator as the core, the firmware uses a little bit more than 9 of the available 16 kB of program memory. When running, the core operates on the internal RC-oscillator, at about 7.7 MHz. As this frequency changes widely with temperature, we have equipped the board with a mount option for different crystals, so it can be used securely when communication over a serial line is needed.

The 16-pin connector visible on the bottom left of Fig. 9 board provides an interface to the CPU for the in-system-programmer and features all supply voltages as well as a number of unused pins for future sensor- or interface boards. As the microcontroller has a limited pin-count, the connector shares some of its pins with the gyroscope, which may be omitted if the board is used for other applications.

4.1.2 Radio Subsystem

When we initially considered the requirements for a gesture-based control, it became clear very soon that integrating infrared emitters into the cube would not be a good solution: In order to reach the TV set, it would take at least one LED on every side of the cube, which results in considerable power-drain on the batteries on every command issued; furthermore, the LEDs would require holes in the cube surface, which would have a negative impact on the object design.

After an evaluation of available radio protocols and hardware (Bluetooth, ZigBee, wireless USB, telemetry solutions),

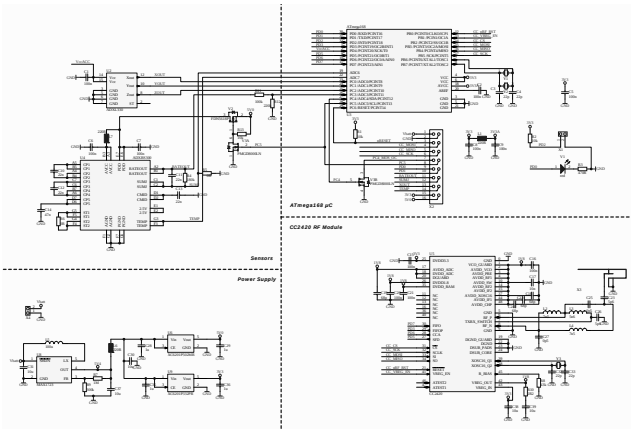

Figure 10: Cube platform layout.

we have decided to use a IEEE 802.15.4 compatible radio solution (a Chipcon CC2420) as they matched our system requirements best: Quick power-on times and low memory footprint for the radio stack keep system complexity down while pairing the requirements for an "interactive" feeling, we can send packets after a few milliseconds after powering up.

4.1.3 Sensors

In order to detect the gestures we have defined, and possibly others as well, we have decided to use a triple-axis accelerometer based on the Analog Devices ADXL330, which provides acceleration data on three analog outputs, which we connected directly to the ADC pins of the CPU. The data acquired lets us extract accelerations and the orientation of the cube with respect to gravity.

For the volume control commands, we use an additional gyroscope to detect rotations around the cube's z-axis which is heading from bottom to button (top). The ADXRS 300 that we use provides the angular rate as a voltage level similar to the accelerometer and includes a temperature sensor (for internal drift compensation) which could also be read out.

4.1.4 Power Subsystem

Minimizing power consumption was a major issue in the system design, which influenced hardware decisions as well as the software architecture, but we had to cope with some unknown parameters for this prototype, like the size and type of battery that would be available. Therefore, we decided to use a step-up converter that can cope with input voltages between 0.8 and 5.5V, which enables us to operate the board with a single cell or rechargeable packs alike. While the CPU and radio chip would work at 1.8V, the accelerometer needs at least 2V so we decided to use a more common supply voltage of 3.3V for these components; the gyroscope needs a 5V supply, therefore we had to introduce a dedicated converter for this sensor. In order to minimize standby power consumption, we switch on the components only when they are needed: The accelerometer consumes only 330uA at 3.3V and can be powered directly by an output pin of the CPU. The gyroscope consumes a considerable amount of power (over 2mA in operation) and is switched

on and off using a MOS-switch to cope with the different input voltage.

With a duty-cycle of 60ms between measurements, the board consumes 600 μA on average, peaking at 24 mA when the radio receiver is on and 48 mA when the radio chip is transmitting a packet. In a test-setup where we performed the usual axes-readout and monitored battery voltage additionally, we transmitted over 120.000 packets (sending every three seconds) using a single AAA cell. Performance could be optimized further by drastically reducing sleep-state power consumption via a redesign of the power path and by using the new nano-power class of Atmel processors that use around 0.6 μA in sleep mode. Unfortunately, these have not been available at project time.

4.2 Cube Firmware

The firmware that operates on the microcontroller consists of two main blocks: One part is responsible for reading out the sensor values, while the other part consists of the radio stack. The whole system is driven by a timer interrupt that wakes up the CPU from deep sleep every 60 milliseconds. The firmware then starts the sensor data acquisition, analyzes the data via a finite state machine, optionally sends a packet over radio and goes back to sleep again.

4.2.1 Gesture Data Acquisition Loop

When the device wakes up, it turns on the accelerometer supply and the ADC block and reads out the values on the axes before switching everything off again. Using a set of thresholds, we convert the raw acceleration readout of each axis to one of three values—*idle*, *low* and *high*—indicating that there is either no force on an axis (idle—equivalent to $V_{DD}/2$), or that it is either positively (high) or negatively (low) accelerated. The finite state machine (FSM) uses these values as its input to switch between detection states. We have mapped each state to one function that handles the temporal behavior of the system (the up superstate in Fig. 11 is implemented as one function) in favor of a table-based approach due to the (RAM) memory constraints of the processor. Adding new gestures can easily be done by adding a new state enumeration and an accompanying function.

4.2.2 Navigating—Cursor Mode

When the cube is in an upright position, it is in the *idle* state, which is the start of every gesture.

If a user tilts the cube around an axis over a certain angle ($\geq 40°$, again defined by the threshold), the corresponding axis reaches a threshold and one of the states *right*, *left*, *up* or *down* is entered. When the user returns the cube to its initial position, the corresponding command (right, left, up, down) is sent via radio and the state machine returns to its idle state.

Informal user tests have shown us that many people use this "tilt-and-back"-behavior, while others were confused—they tilted the cube and waited for a reaction, which eventually came when they gave up and returned the cube to an upright position. To cope with this user group, we have introduced an additional timeout, so a command is emitted if the cube is tilted longer than 500ms. We have also experimented with an auto-repeat function, but we found that our testers would need more time to get used to this additional functionality while they were getting accustomed to the gestures themselves.

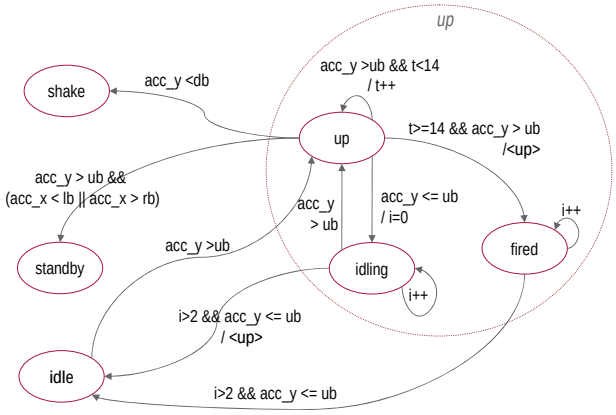

Figure 11: Gesture FSM: Detailed view of the *up* super state.

Furthermore, we try to minimize accidental triggering of commands by using time thresholds as well—a state needs to be active for at least three cycles before it is allowed to start the radio subsystem. Likewise, we ignore a return to idle for up to two time slots to eliminate unintended returns. Fig. 11 presents a detailed view of the resulting super state for the *up* gesture: The FSM enters the *up* state when the acceleration on the y-axis (*acc_y*) has exceeded the upper threshold *ub*. It stays there as long as this condition holds on, up to 14 iterations (equivalent to 800 ms), thereafter the <up> gesture is sent over radio and the state machine stays in *fired* until the user returns the cube into idle state. The FSM enters the *idling* sub-state when acceleration drops below the threshold, and fires a packet as well if this is detected at least three times. When an acceleration in the opposite direction is detected, *shake* state is entered, or when the user starts to tilt the cube on the other axis as well the state machine enters *standby*.

4.2.3 Button Use

The cube button serves a dual purpose. Obviously, it acts as a button: when it is pressed and released again, an "ok"-command is emitted—internally the state machine enters and leaves the *button* state. In addition, the button serves as a mode switch that overlays the navigation gestures described above. The *up* and *down* commands are replaced by *channel+* and *channel−* commands. If either of these two is detected, the state machine switches only between the channel+ and channel− states according to the y-axis readout to let people zap through channels until the button is released.

In addition, we turn on the gyroscope when entering the button state and analyze the current readings to detect angular movement. If the cube has not been tilted yet and we find rotation, the FSM enters a volume control mode: whenever the cube is turned to the left, *volume−* is entered,

analogously *volume+* to the right. In these two modes, the cube sends volume commands at regular intervals until the button is released. We have experimented with other approaches as well, like sending the volume commands as long as rotation is detected, but made our test users spin around themselves, which could be used for gaming, but not for casual TV interaction.

4.2.4 Special Commands

When the cube is set in its deposit, the thresholds of all three axes trigger. Whenever such a situation is detected, we wait for approximately two seconds until sending a power-off command and entering the *standby* state. When the cube is removed and held upright again, we instantly send a power-on command. During standby, we reduce the sensing duty cycle to 600ms to save more power.

Finally, we have implemented shaking gestures for seldom-used commands: When the cube is shaken horizontally, the state machine detects this by alternating between left and right states. After three direction changes a gesture is recognized and the *home*-command is sent. As this command forces the IPTV portal to jump to its main page no matter where the user has been navigating before, we have selected this longer interaction so that it cannot be triggered by accident.

4.2.5 Radio Subsystem

The software stack to control the radio subsystem has been built around the ChipCon driver library which consists of a small set of functions to initialize the radio chip, send a packet and to enter receiving mode, which triggers a callback function when data has been received.

For addressing, we use 16-bit addresses for the cube and converter to communicate as well as a manually chosen PAN id (0x2420). We have decided against implementing the PAN association procedures that are defined in the IEEE802.15.4 standard (cf. [7]), as the remote controls should mimic the usability of infrared remote controls: You can take any device compatible with your TV to control it, denying access would confuse users.

When the converter is powered on, it scans every channel to measure the current RF noise and switches to the most quiet channel for operation. When the cube is powered on, it looks for a converter box by sending a "ping" packet on every channel, which must be answered by the cube under 50ms. When an answer is received in time, this channel is marked active and used to send gesture commands. The channel search is entered again, if packets have been sent for more than three times without getting an acknowledge by the converter box.

4.3 IR Converter

The converter box serves as an interface to the home entertainment world. Unlike the cube, this box is always powered and waits for radio packets. As the hardware platform for the converter, we have extended MicaZ-boards with infrared LED transmitters connected to the supply voltage via power switches.

Whenever a packet matching the converter address is received, a callback function is triggered that examines the packet contents. If the software detects a correct command packet, it performs a lookup in a table where it retrieves information about the infrared protocol and the IR command value to emit. Using the timing data that we keep for each infrared device family, it sets up a system timer that triggers the sending function. The converter supports the protocols for Sony, Samsung, Philips (RC5), NEC and Grundig TV sets as well as a custom set-top box protocol (Amino).

We have hard-coded the lookup table for this prototype, but it could be easily overwritten with new values via a configuration program that lets users adapt it to their system environment. Alternatively, as we use only commands that are common to all protocols, every infrared protocol could be triggered when a gesture command is received, but this has a noticeable effect on overall reactivity, as each command takes several milliseconds.

4.4 Gesture Command Protocol

The communication between cube and converter is encapsulated in a versatile protocol to transmit a large variety of commands and protocol information. It makes use of lower-layer mechanisms like packet transmission numbers to keep track of lost or duplicate packets, so only the small payload defined in Tab. 1 remains.

Byte	0 hi	low	1	2	3..3+len
	flags	domain	cmd	len	param0

Table 1: RF protocol structure.

The first byte defines a set of flags in the upper nibble, indicating if the command is a button-press or -release event and if a keyboard-like or mouse-like pointing device is used. The lower nibble defines a *domain* of commands: We use 0 to indicate that we transmit gesture commands, $0xf$ is used for protocol-control messages like the converter-scan messages or to transmit debug information. Byte 1 indicates the command, which can optionally have up to 255 bytes of parameters (but limited by the IEEE 802.15.4 maximum packet size of 127). If the parameters are not used, the length byte is set to zero.

The protocol is flexible enough to transport all usual remote control commands, can operate in both directions for inducing state changes in the rc device and can be easily extended for other purposes: We have used it in development versions of the cube to transport battery level measurements or raw sensor data for gesture analysis by adding new command codes and defining their parameters.

5. CONCLUSION

We discussed the problem of cognitive overload induced by remote controls in daily life situations to design an alternative control interface with restricted functions and an affording design like a tangible artifact. With our interface design, we focused on the control of an IPTV set-top box. First we studied human hand kinematics and types of grips, gestures and forms which support the 3-step-usage of a remote control (grabbing, switching, putting away). Finally

a cube form with rounded edges has emerged as the perfect design for our tangible interface. This cube form allows a rotating motion for continuous input and a flipping motions for discrete input. So we reduced the functions of a standard remote control to the most often used like switching ON/OFF, volume-change, channel-change and menu navigation (up, down, left, right, ok and home) and mapped them to the possible cube gestures. As mode control and additional orientation help we integrated a push-button on the top of the cube. The implemented hardware for gesture recognition comprises an accelerometer and a gyroscope connected to a converter box by IEEE 802.15.4. The converter box transforms the radio signal to an infrared signal and transmits it to the set-top box.

6. ACKNOWLEDGMENTS

We wish to acknowledge the valuable contribution and guidance coming from Peter Bruck (Research Studios Austria), Helmut Leopold, Armin Sumesgutner and Helmut Rauscha (Telekom Austria), who provided critical input for the design process and requirements analysis. Peter Halbmayer and Christoph Lichtenberger (Studio Pervasive Computing Applications) have implemented major parts of the software architecture and gesture library.

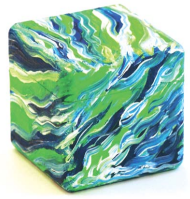

Figure 12: Artistic cube design.

Therese Wagenhofer has contributed to the artistic value and appearance of the prototype shown in fig. 12.

7. REFERENCES

[1] A. Butz, M. Groß, and A. Krüger. TUISTER: a tangible ui for hierarchical structures. In *IUI '04: Proceedings of the 9th International Conference on Intelligent User Interfaces*, pages 223–225, New York, NY, USA, 2004. ACM Press.

[2] A. Butz, M. Schmitz, A. Krüger, and H. Hullmann. Tangible uis for media control – probes into the design space. In *CHI '05: CHI '05 Extended Abstracts on Human Factors in Computing Systems*, pages 957–971, New York, NY, USA, 2005. ACM Press.

[3] K. Camarata, E. Y.-L. Do, B. R. Johnson, and M. D. Gross. Navigational blocks: Navigating information space with tangible media. In *IUI '02: Proceedings of the 7th International Conference on Intelligent User Interfaces*, pages 31–38, New York, NY, USA, 2002. ACM Press.

[4] A. Ferscha, C. Holzmann, and S. Resmerita. The key knob. In *ICDCSW '06: Proceedings of the 26th IEEE International ConferenceWorkshops on Distributed Computing Systems*, page 62, Washington, DC, USA, 2006. IEEE Computer Society.

[5] A. Ferscha and S. Resmerita. Gestural interaction in the pervasive computing landscape. *e & i Elektrotechnik und Informationstechnik*, 124:17–25, 2007.

[6] S. Hinske and M. Lampe. Semantic mapping of augmented toys between the physical and virtual world. In *Proceedings of Workshop on Tangible User Interfaces in Context and Theory held in association with ACM CHI 2007*, San Jose, CA, Apr. 2007.

[7] IEEE. *IEEE Standards 802.15.4, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)*. Number SS95127. IEEE Computer Society, October 2003.

[8] J. Kela, P. Korpipää, J. Mäntyjärvi, S. Kallio, G. Savino, L. Jozzo, and D. Marca. Accelerometer-based gesture control for a design environment. *Personal Ubiquitous Comput.*, 10(5):285–299, 2006.

[9] K. V. Laerhoven, N. Villar, A. Schmidt, G. Kortuem, and H. Gellersen. Using an autonomous cube for basic navigation and input. In *ICMI '03: Proceedings of the 5th International Conference on Multimodal Interfaces*, pages 203–210, New York, NY, USA, 2003. ACM.

[10] J. R. Napier. The prehensile movements of the human hand. *The Journal of Bone and Joint Surgery. British Volume*, 38-B(4):902–913, November 1956.

[11] D. A. Norman. Affordance, conventions, and design. *Interactions*, pages 38–43, May 1999.

[12] D. A. Norman. *The Invisible Comptuer*. MIT Press, Cambridge, Massachusetts, London, 1999.

[13] S. Oh and W. Woo. Manipulating multimedia contents with tangible media control system. In M. Rauterberg, editor, *ICEC 2004: Proceedings of the Third International Conference on Entertainment Computing*, Lecture Notes in Computer Science, pages 57–67, Berlin, Heidelberg, 2004. Springer.

[14] T. Pering, Y. Anokwa, and R. Want. Gesture connect: Facilitating tangible interaction with a flick of the wrist. In *TEI '07: Proceedings of the 1st International Conference on Tangible and Embedded Interaction*, pages 259–262, New York, NY, USA, 2007. ACM Press.

[15] E. Schweikardt and M. D. Gross. A brief survey of distributed computational toys. *Digitel*, 0:57–64, 2007.

[16] L. Terrenghi, M. Kranz, P. Holleis, and A. Schmidt. A cube to learn: a tangible user interface for the design of a learning appliance. *Personal Ubiquitous Comput.*, 10:153–158, 2006.