Bringing Haptic General-Purpose Controls to Interactive Tabletops

Von der Fakultät für Mathematik, Informatik und
Naturwissenschaften der RWTH Aachen University
zur Erlangung des akademischen Grades eines
Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von

Dipl.-Inform. Malte Hanno Weiß
aus Köln, Deutschland

Berichter: Univ.-Prof. Dr.rer.nat. Jan Borchers
Prof. James D. Hollan, Ph.D.

Tag der mündlichen Prüfung: 23. August 2012

Contents

List of Figures

List of Tables

Listings

Abstract

Interactive tabletops are large horizontal displays that allow multiple users to simultaneously and directly interact with digital content by touching the surface, manipulating physical objects, or conducting gestures above the surface. In the last decade, these devices have aroused much interest in the research community, and first commercial products have been released. Interactive tabletops combine a dynamic graphical user interface, a natural way of input, and a platform that is suitable for collocated collaboration. However, they provide only limited haptic feedback. In contrast to physical controls that guide the users' motion, visual on-screen controls cannot be felt and require visual focus when being operated. Also, due to the large contact area of fingers, input by touch is less precise than by conventional controls, such as mice or keyboards.

This thesis addresses the issue of limited haptic feedback on interactive tabletops, while focussing on precise input for productivity tasks. We introduce physical general-purpose controls that combine the benefits of haptic feedback with the dynamic nature of interactive tabletops. Our controls are passive, untethered, low-cost, and easy to prototype. Made of transparent materials and using the table's back projection, they can change their visual appearance on the fly and become versatile controls for various applications. Furthermore, we describe how to turn these controls into malleable user interfaces by employing an electromagnetic actuation mechanism. This allows to move a control or parts of it, to maintain the consistency between its physical and virtual state, and to change physical properties via software at run time. Finally, we present an output method that creates haptic feedback near the surface and only requires a minimal equipment worn by the user.

The thesis provides an introduction to the field of interactive tabletops and embeds our contributions into the context of related work. We explain our design and interaction concepts, the underlying hardware engineering, and software technologies. We evaluate our contributions in user studies and measurements and provide evidence for the usefulness of our techniques. We also describe potential applications with focus on productivity tasks. Finally, we illuminate implementation challenges, discuss limitations, and provide a perspective on recent developments as well as future trends in the field of haptic feedback on interactive tabletops.

Zusammenfassung

(Abstract in German)

Interaktive Tische sind große horizontale Bildschirme, die es mehreren Personen ermöglichen, gleichzeitig und direkt mit digitalen Inhalten zu interagieren, sei es durch Berührung der Oberfläche, durch Manipulation physikalischer Objekte oder durch Gesten über dem Tisch. Im letzten Jahrzehnt hat sich nicht nur die Forschungsgemeinschaft intensiv mit diesen Geräten beschäftigt, auch erste kommerzielle Produkte wurden bereits veröffentlicht. Interaktive Tische kombinieren eine dynamische grafische Benutzeroberfläche, eine natürliche Art der Eingabe und eine Plattform, die sich für gemeinsames Arbeiten eignet. Allerdings bieten sie nur ein begrenztes haptisches Feedback. Im Gegensatz zu physikalischen Eingabegeräten, die die Bewegung des Benutzers führen, können graphische Eingabeelemente auf einem Bildschirm nicht ertastet werden und erfordern einen visuellen Fokus während der Bedienung. Aufgrund der großen Kontaktfläche von Fingern ist eine Eingabe durch Berührung der Oberfläche außerdem unpräziser als durch konventionelle Eingabegeräte wie Maus und Tastatur.

Diese Doktorarbeit befasst sich mit dem Problem des eingeschränkten haptischen Feedbacks auf interaktiven Tischen, wobei wir das Hauptaugenmerk auf eine präsize Eingabe in Produktivitätsaufgaben richten. Wir führen physikalische Allzweck-Eingabegeräte ein, die die Vorteile von haptischem Feedback mit der dynamischen Natur von interaktiven Tischen verbinden. Unsere Eingabegeräte sind passiv, preiswert, leicht zu bauen und dabei nicht kabelgebunden. Da sie aus transparenten Materialien konstruiert sind, können sie die Rückprojektion von Tischen verwenden, um ihre visuelle Repräsentation dynamisch anzupassen. Dies macht sie zu wandelbaren Eingabegeräten, die in einer Vielzahl von Anwendungen eingesetzt werden können. Des Weiteren beschreiben wir, wie diese Eingabegeräte über einen elektromagnetischen Aktuierungsmechanismus zu formverändernden Benutzerschnittstellen erweitert werden können. Dies erlaubt es, auf Software-Ebene ein Eingabegerät oder Teile davon zu bewegen, die Konsistenz zwischen seinem physikalischen und visuellen Zustand aufrechtzuerhalten und physikalische Eigenschaften zur Laufzeit zu verändern. Schließlich präsentieren wir eine Ausgabemethode, die im Nahbereich der Tischoberfläche haptisches Feedback erzeugt und nur eine geringe Ausrüstung des Benutzers erfordert.

Diese Arbeit bietet eine Einführung in das Forschungsgebiet der interaktiven Tische und bettet unsere Beiträge in den Kontext bestehender Forschung ein. Wir beschreiben unsere Design- und Interaktionskonzepte sowie die dahinter stehende Hardware und Software-Technologie. Wir evaluieren unsere Beiträge mittels Benutzerstudien und Messungen und erbringen Nachweise für die Nützlichkeit unserer Techniken. Wir beschreiben außerdem potentielle Anwendungen mit einem Fokus auf Produktivitätsaufgaben. Schließlich beleuchten wir spezifische Herausforderungen bei der Implementierung, diskutieren Einschränkungen und bieten eine Perspektive auf jüngste Entwicklungen sowie zukünftige Trends im Gebiet des haptischen Feedbacks auf interaktiven Tischen.

Acknowledgements

Writing a PhD thesis is an immense amount of work, which would not have been feasible without the contributions, support, and inspiration of many people.

I thank my supervisor, Jan Borchers, for giving me invaluable advice for my PhD thesis and my projects, and for providing an environment that enables creative and keen research. He also established an extraordinary infrastructure for building novel systems. Projects like Madgets and FingerFlux would not have been feasible without his efforts to found the first FabLab in Germany.

I also want to thank James D. Hollan for supporting my thesis as co-advisor. He gave valuable feedback for the development of SLAP Widgets.

Thanks to all of the contributors to the SLAP Widgets project. Julie Wagner and Yvonne Jansen from RWTH Aachen University, and Roger Jennings and Ramsin Khoshabeh from University of California, San Diego, spent many days and nights to make SLAP Widgets a success. Yvonne also constructed the first version of the SLAP Table, with the help of Roger, who arguably cast the best FTIR compliant surface in the world. Stefan Hafeneger developed our first software framework to distribute touch events to tabletop applications. Thorsten Karrer gave valuable pointers to related work.

Madgets would not exist without the enormous support of Florian Schwarz. He developed crucial parts of the table hardware and implemented the actuation algorithm. He successfully mastered all administrative barriers to purchase custom-made electromagnets. In the last weeks before the deadline, we contributed nearly all of our time awake to this project. Working with him was an amazing experience.

I also want to thank everyone who helped to assemble the Madgets Table: Simon Jakubowski, Lucas Braun, Christian Remy, Simon Voelker, and Helga Weiß. Sticking thousands of fiber optical cables into ridiculously tiny holes was an inconvenient but highly important task for developing a novel tracking method.

My students, Simon Jakubowski and Lucas Braun, supported me in all of my research projects and considerably disburdened me. They built and iterated various prototypes. Simon helped to take many of the photos in this thesis. Lucas is the creative engineer behind the improved pressure point of the rigid SLAP Keyboard.

Simon Voelker helped to develop a new touch detection framework for multi-camera and multi-screen systems. He also built the new small-scale electromagnetic array prototype for our demo of FingerFlux at the UIST 2011 conference. Christian Remy supported the Madgets project and gave valuable feedback for the paper. He is also the leading author of our HCI design pattern language on interactive tabletops. Gero Herkenrath helped to design and to conduct the typing performance study of the SLAP Keyboard. I esteem Christian, Simon, and Gero not only as collaborators but also as friends that were always open for inspiring conversations.

Chat Wacharamanotham significantly contributed to FingerFlux. He also gave helpful advice for the statistical evaluation of the user studies in this thesis. I also want to thank René Bohne and David Peters for their help in the FabLab and for contributing their knowledge in the field of electrical engineering. Thanks to Ronny Seidel for conducting the user study for exploring the conflict between actuation and the users' desire for control. I also want to acknowledge all participants who joined the user studies and helped me to gain deeper insights into the design concepts.

Thanks to Patrick Baudisch for these five intense and eye-opening days at Hasso-Plattner-Institute. He tought me how important it is to get the story right.

I also thank all the proofreaders who provided indispensable feedback: Anja Schulz, Christian Remy, Florian Schwarz, Simon Voelker, and Gero Herkenrath.

Last but not least, I want to thank my wife, Helga Weiß. She not only helped to assemble the Madgets Table. During the entire time, she supported me emotionally and always gave me the strength and joy to keep up with my research.

Conventions

The following conventions will be used throughout this thesis:

The thesis follows the American standard for spelling.

We use plural "we" in the entire thesis instead of "I" even if the work was solely done by the author. Some of the materials in this thesis have been published previously. These will be stated in the thesis.

Occasionally, we use gender-specific names and pronouns to provide concrete usage scenarios and to improve readability. However, all references to fictional persons are not limited to a particular gender.

Names of concepts and physical prototypes are capitalized, e.g., "SLAP Widgets" or "Bell Madget".

We show box plots to visualize quantitative results. Whiskers expand to the lowest data point within 1.5 interquartile range of the lower quartile, and to the highest data point within 1.5 interquartile range of the upper quartile.

All code examples are written in Objective-C.

Chapter 1

Introduction

Human beings are endowed with manifold senses that allow them to perceive and process stimuli from the environment. Senses like sight, hearing, taste, smell, and touch provide an information channel that is steadily integrated into world knowledge and allows humans to react appropriately to external events. As all natural signals, these information channels are subject to noise, e.g., when listening to a person in a crowded room, and can lead to ambiguous interpretations, such as the various examples of optical illusions. Human beings naturally weigh different senses and integrate them into a clearer signal. For example, while talking, auditory and visual stimuli are combined, whereas lip reading (visual channel) plays a greater role in a crowded room with background noise. When walking in absolute darkness, the senses of touch, hearing, and kinaesthesia can partly compensate for the lack of visual input. The combination of senses leads to a higher security in the perception of the world and in the process of decision making.

Human beings combine multiple senses to receive a clear perception of the environment.

This principle of fusing senses plays an important role when designing interactive systems. An experienced industrial designer composes a product such that it addresses multiple senses in a coherent way. Let us have a look at a simple everyday example: washing clothes. Many washing machines have a mechanical rotary knob for selecting the washing program (Fig. 1.1). Three different roles are involved in the design of this control. The *user* follows the high-level goal to wash his clothes using a specific program. The *engineer* of the machine requires a number from 1 to n that maps to one of n possible programs stored in an internal microprocessor. Each program triggers the various pumps, motors, and heaters in the machine in a specific way. The *interaction designer* defines the interface between the user's goal and the required numeric input. Her task is to design a user experience that is natural and fits the user's mental model of the machine. She designs the knob such that it addresses multiple senses: The program knob exposes an imprinted arrow pointing towards the label of the current selection (visual sense). The knob also contains a palpable embossed arrow and snaps to the program positions when the user turns the knob (haptic sense). Finally,

Many successful everyday devices are designed to address multiple senses in a coherent way.

Example: A washing machine's knob addresses visual, haptic, and auditive sense.

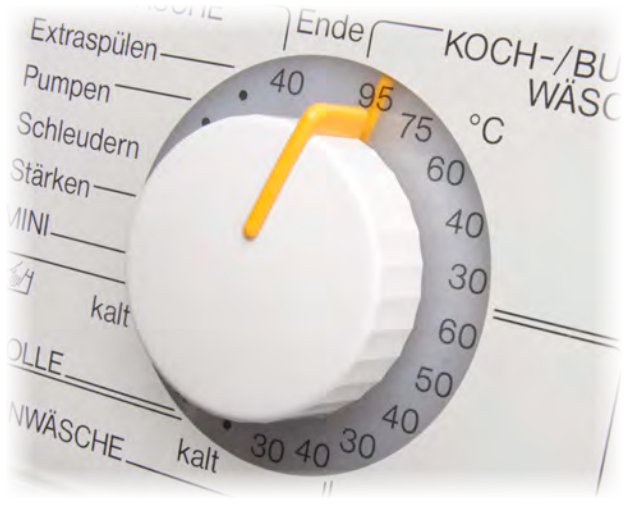


Figure 1.1: The knob of a washing machine addresses the visual, the haptic, and the auditive sense. Photo taken by the author.

the user hears a clear "click" sound every time he turns the knob to the next position (auditive sense).

Benefits of mechanical knob: 1. Matches user's mental mode. 2. User feels in control. 3. User employs multiple senses and associated memories.

In the age of microprocessors, the programs of washing machines are not controlled mechanically anymore. A solution involving two capacitive (non-mechanical) buttons for program selection in combination with an LCD display that shows the user's choice would be considerably cheaper, low-wear, and easier to produce. Nevertheless, mechanical knobs have strong benefits: First, they match the mental model of controlling a heavy mechanical machine. This model comes from a time when washing machines were indeed controlled by "mechanical programs" stored as bumps on a plastic disc that was connected to the program selection knob and processed by a motor. Second, the strong audio-haptic feedback gives the user a feeling of being in control. Third, the user can employ and fuse multiple senses. He can choose a program without looking by making use of auditive and muscle memory, and by just memorizing "two clicks to the right" for selecting his favorite program.

Designers iterate the "feel" of a product many times before release.

The way a product "feels" communicates its character. This is iterated many times before releasing it to the market. Today's car doors are another prominent example: They are designed in such a way that the strength required to close the door and the sound that it produces thereby communicates the brand and class of the car. Designers spend weeks to create a "feel" that matches the target users. While the importance of haptic feedback is well understood in industrial design, we are currently encountering a tendency of developers to remove the haptic feedback channel from our everyday computing devices.

Since the introduction of home computers in the 1980s, a computer screen in combination with a keyboard and a mouse has turned into the prevalent user interface for most computing tasks. While hardware and design was optimized over the years, and laptops and wireless internet nowadays allow users to work with a computer (nearly) everywhere, this general input and output concept has barely changed.

For over 30 years, keyboard and mouse were the main input devices for most computing tasks.

In his pioneer paper, Wellner [1993] presented an alternative to this setup in order to combine the benefits of physical and digital documents. The *DigitalDesk* was a conventional desk that was enhanced with a projector and a camera above the surface. It supported paper-based office work but also enabled users to interact with digital documents, which were projected onto the surface. In contrast to the "desktop metaphor", digital content could now be manipulated *directly* using pens or bare fingers. This publication marks the starting point of the research field on *interactive tabletops*. In the following, many papers have been published that deal with engineering, interaction techniques, and applications of interactive tabletops. We refer to Müller-Tomfelde and Fjeld [2010] for a historic survey on the field.

The DigitalDesk enhanced a conventional desk with digital documents. It was the starting point for research on interactive tabletops.

However, conventional setups involving keyboard and mouse remained the prevalent setup, until two recent events triggered the evolution of touch-devices for everyday tasks. First, Han [2005] published a tracking technique allowing practitioners to easily construct low-cost interactive tabletops. Since no deep electrical engineering skills were required anymore, building touch-based interfaces became feasible for everyone. The result was not only a prospering do-it-yourself community, the feasible technology also aroused the interest of many researchers. In 2006, the first Workshop on Horizontal Interactive Human-Computer Systems (TABLETOP) provided a platform for tabletop researchers to exchange ideas and research results. It eventually turned into an established international conference on Interactive Tabletops and Surfaces (ITS) in 2009. Papers on interactive tabletops are now being published on all major Human-Computer Interaction (HCI) conferences. Also, commercial tabletops, such as Microsoft Surface[1] or SMART Table[2], have been released.

The availability of a low-cost multi-touch tracking technology accelerated the research on interactive tabletops and the development of consumer products.

The second event was the release of the Apple iPhone in 2007. The iPhone is a phone and computing device with a multi-touch interface that lets users directly interact with a graphical user interface (GUI) using their fingers. From an interaction viewpoint, it represents a clear contrast to existing *smart phones* of that time, which relied on physical keyboards and pen or joystick input. Although touch-based smart phones have been released years before (e.g., the IBM Simon in 1992), the iPhone can be considered as the first touch-based phone that combines elaborate engineering with a well-working user experience. In combination with an established brand, it became the first successful touch phone and marked the commercial starting point for multi-touch in everyday tasks. Since then, multi-touch is an accepted input method, and many companies have released their own touch

The release of the Apple iPhone established multi-touch as a default input method on smart phones.

[1]http://www.microsoft.com/surface
[2]http://smarttech.com/table

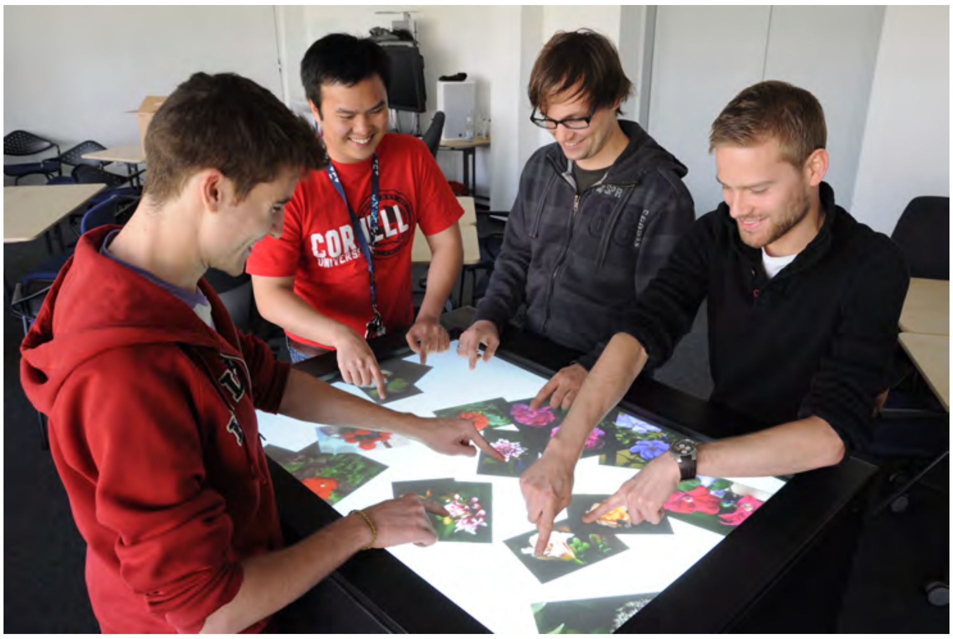


Figure 1.2: Interactive tabletops enable direct manipulation and collocated collaborative work.

phones. This was also the starting point for the success of tablet computers that were—until then—only niche products.

Interactive tabletops provide a large dynamic interactive surface, a natural input method with low learning curve, and a platform for collocated collaborative work.

Similar to tablet computers, interactive tabletops have the potential to replace traditional desktop computers for many tasks. They provide a large interactive GUI, and since they can be approached from multiple sides, they inherently afford collocated collaborative work (Fig. 1.2). Directly manipulating objects with fingers rather than using an indirect input device, such as a mouse, is a natural way of interaction and barely requires learning. Also, instead of desktop computers, which are *explicit* computing devices requiring their own dedicated place, a tabletop can be designed as an aesthetic piece of furniture that becomes an interactive surface in an ad hoc manner. As an *implicit* computing device, it can be integrated into the everyday living and working environment of users. All underlying technology is hidden from the user, which makes interactive tables more approachable for persons who usually do not work with computers at all. Interactive tabletops can be easily integrated into the users' ecology of devices. For example, instead of plugging a camera into a USB port and starting a photo editing software, users might just put their smart phones on the surface and "pour out" their digital photos onto the surface for presentation and sharing. The GUI interface on the table can be modified in real-time and react on external events.

Haptic feedback on interactive tabletops is limited.

However, in contrast to traditional computers, two issues impair the user experience on interactive tabletops. First, purely touch-based surfaces only provide the limited haptic feedback of touching a planar surface. Users cannot feel the shape of on-screen objects, and they do not receive haptic

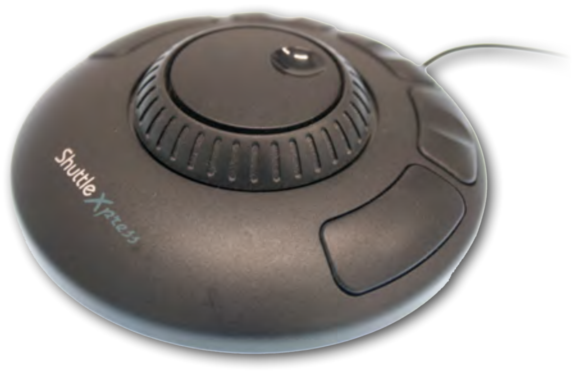


Figure 1.3: Input device for media navigation including two knobs and five buttons. A jog wheel in the center is surrounded by a spring-loaded knob that snaps back to its original position when released. Photo taken by the author.

Lack of general-purpose controls limits development of productivity tasks.

feedback when triggering actions. Second, there is a lack of general-purpose input tools. Current tabletop applications range from pure kiosk systems, over music synthesizers [Jorda et al., 2007], games [Antle et al., 2011], education, therapy applications [Giusti et al., 2011], to augmented laboratories [Tabard et al., 2011], and visualizations of scientific data [Sultanum et al., 2011]. Most of them employ specialized input techniques for their specific tasks. However, productivity applications, such as word processing, spreadsheet analyses, or graphical design, are rare. Most tasks in these applications require precise input of abstract parameters or extensive typing. For reasons of scalability, a small but efficient set of general-purpose controls is required.

Goal: Haptic general-purpose controls on interactive tabletops.

The goal of this thesis is to introduce general-purpose controls that encourage the use in productivity tasks, and to bring haptic feedback known from traditional physical controls, such as buttons, sliders, and knobs, back to interactive tabletops.

1.1 From Physical to Digital Controls

Benefits of physical controls: Strong affordance, guide users' motions, eyes-free interaction.

Physical controls are ubiquitous in our everyday life. We use buttons to trigger all kinds of actions, e.g., to toggle light switches, to play the next song on our music player, or to type the letters in a document. Knobs provide continuous and discrete input, e.g., to set the audio volume in a car, to step frame-by-frame through a video (Fig. 1.3), or to select a program on a washing machine. Sliders allow us to set an absolute value in a range, like the brightness of a lamp or the volume of a channel in an audio mixer. Physical controls are the result of hundreds of years of iterations and refinements, and all of them share the same benefits in terms of usability: They provide strong affordances, they guide the users' motions, and they can be operated without looking. The latter is especially important when

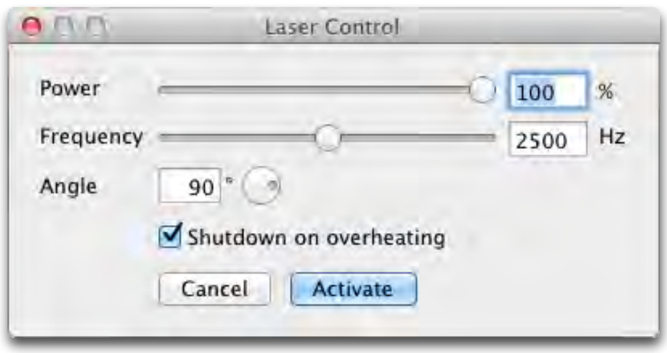


Figure 1.4: Screenshot of a fictional laser control application using standard GUI elements of Mac OS X 10.7. All GUI elements that are clickable or draggable have a 3D appearance using subtle color gradients, highlights, and shadows.

using controls while focussing on a different task, e.g., triggering the turn signal while driving a car. In the absence of visual perception, users can employ muscle memory and their full spectrum of motor skills. A crucial reason why physical controls support eyes-free interaction so well is the support of a *rest state*: Users can apply their hand to a control without triggering it involuntarily. Accordingly, they can focus on their main task and operate the control when needed without visually focussing on it.

GUI controls visually mimic properties of physical ones.

Since the introduction of GUIs in personal computers, designers tried to transfer most of the physical benefits to virtual on-screen controls. Physical affordances turned into visual ones: In modern desktop operating systems, on-screen controls visually mimic their physical counterparts. For example, active GUI widgets, such as buttons, checkboxes, or sliders, are designed in a 3D look. Some graphical rotary knobs contain visual notches, affording rotation (Fig. 1.4). Although these affordances are "false" on desktop applications—the controls cannot be pushed with fingers anymore—they successfully support the metaphor that the mouse cursor represents a finger interacting with objects on a "desktop". Nowadays, these affordances gain more meaning in the context of touch devices.

On-screen controls do not guide users' motion, but keyboard and mouse still provide haptic and auditive feedback.

However, with the conventional desktop setup involving computer mouse and keyboard, on-screen controls cannot guide the user's motion anymore. Although most controls impose software constraints (e.g., a scrollbar value cannot be set outside its range), a mouse device can theoretically be moved on a (infinitely large) planar surface and does not provide physical constraints that block the movement. Therefore, a user must monitor the cursor's trajectory on the screen when operating a virtual control. Nevertheless, it should be mentioned that users still can rely on their haptic sense when working with today's desktop GUIs. The keyboard and mouse buttons provide haptic feedback when being pushed; together with an important, albeit subtle, clicking sound. Every time a user clicks on a virtual button using the mouse, the haptic and auditive experience is similar to that of real pushbuttons.

If a computer mouse is involved, graphical buttons also provide a rest state: Users can move the cursor onto a button, visually focus on a different task, and click without looking when required. Just imagine a user fixating the countdown of an online auction and clicking the "confirm bid" button in the last second in order to buy a product for the best price. In these situations, users rely on their haptic sense and on the fact that the hand is stable when it rests on a physical device. Once the mouse cursor is placed on a virtual control, the user can be sure that it stays there until he moves it somewhere else. Some operating systems visually support this rest state by highlighting the control that is currently positioned beneath the mouse cursor.

Mouse provides rest state: Users can hover above on-screen controls without triggering them.

As mentioned above, touch screen devices are gaining much interest in both industry and research. Rather than moving a mouse cursor first and clicking, users can directly tap with a finger to trigger a virtual button. These screens do not need any external input device, such as mouse or keyboard, which makes them suitable for mobile devices. Furthermore, they are appropriate for use in public spaces, e.g., in ticket machines, because the touch detection technology can be secured behind glass. Bimanual input and multi-touch gestures further enrich the interaction and allow users, e.g., to freely zoom and rotate a digital map.

Touch-based devices allow to interact with on-screen objects directly without external input devices.

However, touch devices only provide a limited haptic experience. Besides the plain screen surface, users cannot feel on-screen controls when touching them. A user does not experience a "click" feeling when tapping a virtual button. Due to the large contact area of a finger, hitting tiny objects on the screen is difficult, which is often referred to as the "fat finger problem" [Mankoff et al., 2000; Siek et al., 2005]. The finger also occludes objects. There are several approaches to increase touch accuracy, e.g., by averaging the touch position of two fingers or by employing a virtual lens [Benko and Wigdor, 2010]. However, these techniques also imply indirect input. Holz and Baudisch [2010] show that a detection of the finger posture and a per-user model can increase direct touch accuracy, but training a user model is impractical in ad hoc usage scenarios.

A touch screen only provides a limited haptic experience.

When interacting with a touch screen, a user can barely employ muscle memory and only exploits a small subset of the hand's mechanical degrees of freedom. Furthermore, as opposed to physical objects or traditional GUIs, touch screens lack a rest state: By default, putting a finger down on the screen triggers an input event. There is not a simple implementation allowing users to rest their fingers on virtual controls and trigger them later while focussing on different tasks. This is also confirmed by Benko and Wigdor [2010], who declared the lack of a state that allows to preview an action before triggering it as a main source for input error and frustration on interactive surfaces. Yet, even if hovering above objects could be detected, it would not support eyes-free input well, because hovering over a longer period is exertive. To disambiguate a hand or finger resting on the surface from an intended touch input, other input modalities can be employed, such as double-tapping or, if technically available, a touch pressure threshold.

Touch screens do not provide a rest state, and muscle memory is barely available.

However, this complicates the interaction, makes it less intuitive, and also might demand per-user calibration.

Direct touch interaction requires visual focus. This causes input errors, drifting, and disadvantages for visually impaired users.

Direct touch is an interaction technique that requires users to visually focus on the controls they operate and not on the data they manipulate. Typing with a virtual keyboard is slower and more error-prone than writing on a conventional physical one [Barrett and Krueger, 1994]. Since users cannot feel the keys, they occasionally have to look at the keyboard to realign their hands after drifting. However, when looking at the keyboard, they lose the visual focus on the text output and potentially miss events, such as typing errors [Paek et al., 2010]. This lowers efficiency and limits the practicability of touch screens for applications that require much, precise, or fast input, such as most of our everyday productivity tasks. Besides the worse user experience for most users, the need for visual attention means that visually impaired users are considerably disadvantaged when using touch screens. Audio feedback has proven useful to provide blind users access to touch-based interfaces (e.g., [Kane et al., 2011]), but it can be disturbing in a multi-user context.

1.2 Contributions

This thesis investigates how to improve haptic feedback on interactive tabletops for productivity applications. We introduce a new class of physical controls that combine the haptic qualities of conventional general-purpose controls with the dynamic nature of graphical user interfaces. Specifically, our contributions are as follows:

SLAP Widgets, general-purpose controls that combine haptic feedback with the dynamic nature of interactive tabletops

1. We address the issue of limited haptic feedback on tabletops by introducing SLAP Widgets, a novel class of passive, tangible, general-purpose tabletop controls that combine haptic feedback with dynamic relabeling.
    (a) We explain the design of SLAP Widgets and how they are used to modify virtual objects on the surface.
    (b) We describe a tracking algorithm that senses the position and state of physical controls placed on the tabletop via passive markers.
    (c) We contribute a scalable software framework for developing applications on interactive tabletops that support direct manipulation and ad hoc use of SLAP Widgets.
    (d) We provide evidence that SLAP Widgets can outperform pure on-screen controls in eyes-free tasks.
    (e) We present a qualitative user study that reveals that the concept of SLAP Widgets is intuitive and easy to learn.
    (f) We present a study on typing that compares translucent physical keyboards based on the SLAP Widgets concept with pure on-screen and conventional keyboards.

2. SLAP Widgets cannot maintain the consistency between their physical and virtual states, which constraints their use in many productivity tasks. Therefore, we introduce Madgets, passive, magnetic widgets that employ magnetic fields to change their physical alignment and configuration.

   Madgets, physical controls that maintain physical-virtual consistency and enable new applications for tabletop tangibles

   (a) We present an interactive tabletop design that combines electromagnetic actuation, fiber optical tracking, and graphical output in a compact design without the use of external cameras and projectors.
   (b) We introduce an electromagnetic actuation mechanism that allows to move, align, and configure multi-element physical controls on our tabletop.
   (c) We explain a tracking algorithm that allows to detect objects using low-resolution input and gradient markers.
   (d) We describe novel applications for tangibles on tabletops that make use of electromagnetic actuation.

3. Designing physical controls requires many iterations to tune specific hardware parameters, and changing these parameters is barely possible after a control has been built. We show how electromagnetic actuation supports prototyping and dynamic modification of physical properties of tabletop controls.

   Application of electromagnetic actuation to simulate and dynamically adapt physical properties of controls

   (a) We contribute mechanisms to change physical effects in tangible controls on the fly, such as friction, weight, spring resistance, and the number of notches.
   (b) We present studies that give evidence that these effects can be created in a way such that user can reliably perceive them.

4. Finally, we leave the scope of tangibles and show how haptic feedback can be provided in cases where physical controls are difficult to apply. We contribute FingerFlux, a novel output method that generates haptic feedback above the surface requiring a minimum equipment worn by the user only.

   FingerFlux provides haptic feedback above tabletops with minimal equipment worn by the user only.

   (a) We explain how electromagnetic fields can apply haptic sensations to fingers with attached permanent magnets.
   (b) We prove that FingerFlux provides haptic feedback in the space up to 35 mm above the surface.
   (c) We show that FingerFlux can effectively reduce drifting when operating on-screen buttons in an eyes-free fashion.

1.3 Structure

The thesis is structured as follows:

- Chapter 2 gives an overview on interactive tabletops. It describes the general concept and introduces input methods as well as tracking techniques. Alternative form factors and open research questions are also discussed. Furthermore, we explain how to capture design knowledge required to build interactive tabletops and applications.
- Chapter 3 introduces the concept of SLAP Widgets. It describes related work, our hardware design, and the underlying software framework. It also presents a basic widget set and interaction techniques for associating SLAP Widgets with virtual objects. Finally, the chapter presents usage scenarios and user studies that evaluate the effectiveness and the user experience of our controls.
- Chapter 4 explains Madgets and shows how to solve inconsistencies between the physical state of controls and their inner virtual state. It demonstrates how to actuate multi-element controls on tabletops using electromagnetic actuation. It also explains our specific hardware and software design. Finally, the chapter shows how to transfer our actuation technique to novel tangible applications.
- Chapter 5 demonstrates how to apply electromagnetic actuation to simulate physical effects in tangible controls, such as perceived weight, friction, spring resistance, and dynamic notches. We describe measurements and proof of concept studies.
- Chapter 6 introduces the FingerFlux concept. It explains how electromagnetic fields can provide haptic feedback above the surface using a permanent magnet attached to the user's fingertip. We present various applications and report on two user studies that demonstrate the capabilities of this method.
- Chapter 7 concludes the thesis. We summarize our contributions and provide an outlook on future work.

Chapter 2

Interactive Tabletops

Before we address the issue of haptic feedback, we give a brief introduction to interactive tabletops. We describe the underlying concept and its inherent challenges. Furthermore, we discuss optical tracking methods that are used in many of today's tabletops. We will also provide a brief overview on established input methods and look at non-horizontal form factors. Finally, we will describe how to capture knowledge of designing interactive tabletops.

An interactive tabletop is a large horizontal display supporting input via fingers, hands, styli, or physical objects.

An interactive tabletop is a large horizontal display that is sensitive to direct input via fingers, hands, styli, or physical objects placed on the surface. The surface is usually positioned in a height that is convenient for either standing or sitting at the table. The input and output concept of interactive tabletops considerably differs from that of conventional desktop computers involving a vertical screen, a mouse, and a keyboard. The key characteristics of interactive tabletops are:

Direct input and output occur on the same surface. Users directly interact with virtual or physical objects.

- *Direct manipulation:* The entire surface is interactive. Input and output are performed in the same area. Users can *directly* interact with virtual or physical objects on the surface using their fingers. For example, virtual buttons are triggered by tapping on them. Direct manipulation [Hutchins et al., 1985] is a natural input technique that is easy to comprehend with a low learning curve. Finger input has also proven to be faster for target selection tasks than mouse input [Kin et al., 2009].

Interface is accessible from multiple sides.

- *Omnidirectional interface:* Following the natural affordance of tables, users can approach interactive tabletops from multiple sides and interact with digital objects from all directions.

Interactive tables are suitable for collocated collaborative work.

- *Collocated collaboration:* Interactive tabletops are suitable for collocated collaborative work. The interactive surface can be surrounded by multiple users. As objects are directly manipulated, collaborators are aware of each other's actions. Also, social protocols are inherently in place, e.g., for distributing the limited interactive space among users.

Underlying technology is hidden from the user.

- *Calm technology:* In contrast to conventional desktop computers involving keyboard and mouse, which are explicit computing devices, interactive tabletops are pieces of furniture with an implicit capability to interact with digital content. Following the spirit of Ubiquitous Computing [Weiser, 1991], the underlying technology is hidden.

Example: Photo sorting

A typical interactive tabletop is shown in Fig. 1.2 on page 4. In the shown “photo sorting” application, multiple users can simultaneously drag, rotate, and scale virtual photos using their fingers. Besides the active surface, no further input devices are visible.

The nature of interactive tabletops imposes many design challenges on application developers:

Interface elements on a tabletop must be readable and operable from multiple sides.

- *Rotatability:* Since the table can be operated from different sides, interface elements must be readable and operable from all directions that users can approach the surface from. That is, all elements must be rotatable, either manually or (semi-)automatically (e.g., [Dragicevic and Shi, 2009]). This also implies that desktop GUIs with their fixed orthogonal alignment of objects cannot be simply transferred to interactive tabletops.

Simultaneous input from multiple users must be disambiguated. Global states are inappropriate.

- *Multi-focus input:* Unlike desktop GUIs, interactive applications do not have a single focus policy. Multiple users can interact with multiple fingers on the same or different objects simultaneously. As long as objects are only manipulated by direct touch (e.g., photo sorting), this is not an issue. However, if on-screen controls are used to change other objects, e.g., when using a virtual knob to change the brightness of a photo, or an on-screen keyboard to enter text, a policy must be implemented that defines concurrent associations between input controls and target objects. Furthermore, a designer cannot assume global application modes, because multiple users might perform different tasks.

Direct touch input lacks precision.

- *Input precision:* When using finger-based interaction, precision is an issue in cases where the relatively large contact area of a finger tip must be mapped to a single point of action, e.g., when pushing a button.

Haptic feedback is limited when interacting with virtual objects.

- *Haptic feedback:* As mentioned before, the haptic feedback provided by interactive tabletops is usually limited to the sensation of a planar surface. Virtual elements cannot be felt, and no “rest state” is available. This makes eyes-free interaction difficult.

Tasks demanding personal input or output require user identification.

- *Personalization and privacy:* Thanks to the input awareness, interactive tabletops enable a democratic interaction in which all users are equitable. However, there are applications where privacy is necessary. For example, imagine users entering passwords and accessing personal information. In this situation, they require an exclusive output method, which hides their data from other users, and an exclusive input technique to avoid others to manipulate their data. The latter necessitates user identification for all input events.

- *Ecology of objects:* Besides input devices, users may also put everyday objects like coffee mugs, keys, or papers on the table that are not intended to trigger input. An interactive tabletop must regard the users' ecology of objects.

Users' ecology of everyday objects must be regarded.

This thesis focusses on the issues of haptic feedback, input precision, and multi-focus input, with the ultimate goal to provide general-purpose controls for productivity tasks on interactive tabletops.

2.1 Optical Tracking

Precise detection of fingers touching the surface or objects placed on top of it is essential for a fluent user experience. In the following, we describe the most relevant optical tracking methods that have been implemented on interactive surfaces. Other tracking technologies, such as resistance-based or capacitive sensing, are not described, but we refer to Schöning et al. [2010] for an overview on those methods.

2.1.1 Frustrated Total Internal Reflection (FTIR)

Frustrated Total Internal Reflection, or *FTIR*, was originally patented by White [1965] for use in fingerprint scanners. Han [2005] applied this technique for touch detection on interactive tabletops.

If a light ray passes through the intersection of two transparent mediums, it is refracted or reflected according to Snell's Law (Fig. 2.1):

Snell's Law describes refraction and total reflection of a light ray passing through the intersection of two transparent mediums.

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{v_1}{v_2} = \frac{n_2}{n_1}$$

That is, the ratio between the incident angle θ_1 and exiting angle θ_2 equals the ratio of phase velocities of the two materials and the inverse ratio of their indices of refraction.

Setting $\theta_2 = 90°$ ($\sin(\theta_2) = 1$) yields the *critical angle*

$$\theta_c \quad = \quad \arcsin\left(\frac{n_2}{n_1}\right).$$

Note that the equation is only solvable if $n_2 < n_1$ (assuming $n_1, n_2 > 0$). At the *critical angle*, the incident ray is orthogonally reflected alongside the intersection. If the incident angle is smaller ($\theta_1 < \theta_c$), the ray is refracted

Light rays exceeding a critical angle are totally reflected.

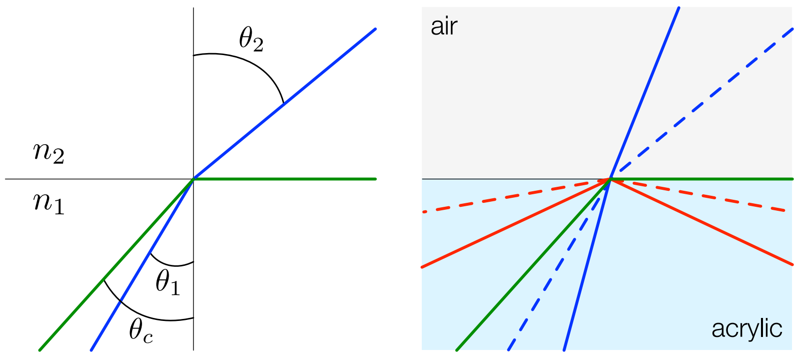


Figure 2.1: Snell's Law. The images show the dependence of exiting from incident angles of rays passing the intersection between two transparent materials. At the critical angle θ_c, rays are reflected in parallel (green). Below θ_c rays are refracted (blue). All light rays with an angle beyond θ_c are totally reflected (red). The right image shows example light rays assuming an index of refraction of 1.0 for air and 1.5 for acrylic, respectively.

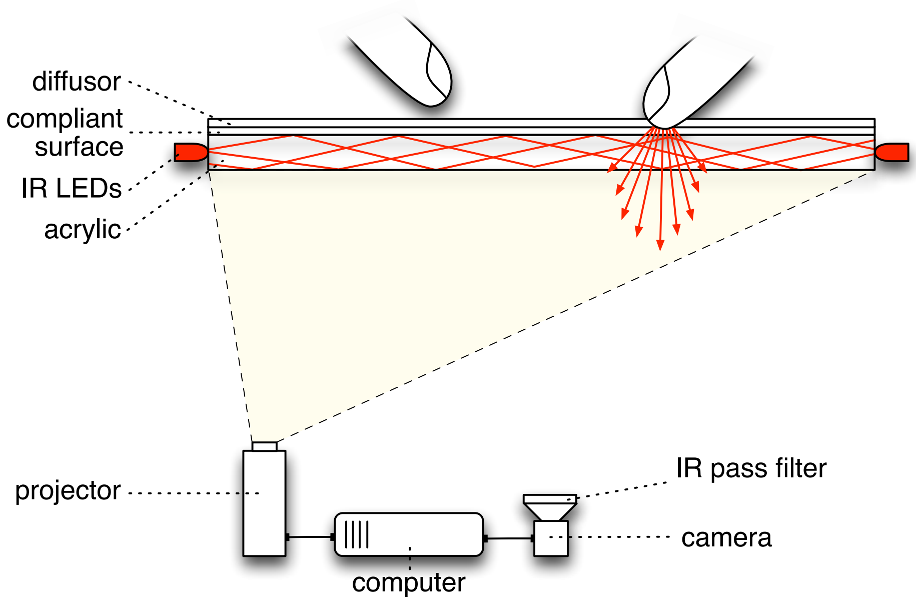


Figure 2.2: Typical FTIR tracking setup. FTIR allows to detect binary touches. Objects hovering above the surface do not reflect (much) IR light to the camera.

into the other material. If it is larger than the critical angle ($\theta_1 > \theta_c$), the ray is reflected, a *total internal reflection* within the material occurs.

FTIR tracking: Infrared light bouncing within the surface is scattered on surface touch.

Based on this effect, Han proposes to build a touch-sensitive tabletop as shown in Fig. 2.2. The tabletop mainly consists of a transparent acrylic plate, a thin *compliant surface* placed above, and a thin diffusor layer on the very top. The acrylic is edge-lit by infrared (IR) LEDs surrounding the surface. The light, which is invisible to the user, enters the acrylic and, due to total internal reflection at steep angles, "bounces" inside the

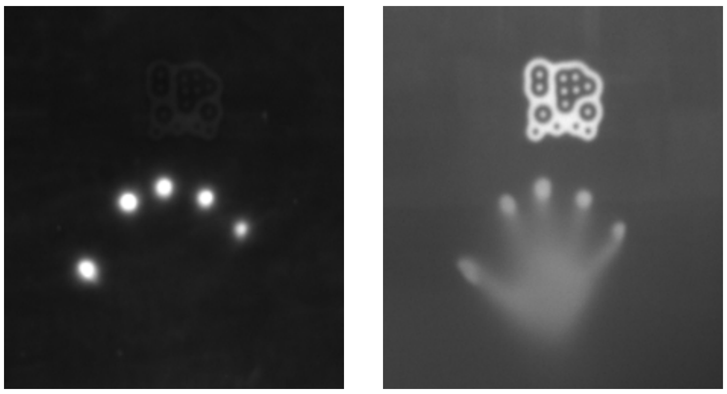


Figure 2.3: Sample raw camera images of FTIR and DI tracking. The user touches the surface with five fingers, and a visual marker is placed on the tabletop. Left: When using FTIR, finger touches are distinctly visible. Everything else is nearly invisible. The minor appearance of the marker results from scattered light inside the table. Right: The camera image of DI captures the visual marker, finger touches, and the hovering palm. However, differentiating touch from minor hover is difficult.

plate without leaving it. However, if a user touches the surface, the light "frustrates" at the contact area, leaves the acrylic, and hits the finger. The fingertip then diffusively reflects the IR light downwards into the table.

A camera beneath the surface captures these events as bright spots in the camera image. An algorithm then interprets these spots as input and updates the GUI accordingly. In section 3.5, we will explain the underlying tracking algorithms in detail.

Computer interprets spots in camera image as input events.

This technique only works robustly as long as the fingers are slightly wet, due to natural oils or sweat. In order to detect touches from dry fingers or physical objects, a compliant layer, usually made of silicone, is added (see Han [2005] for are detailed explanation). The quality of this layer is crucial for the quality of touch detection. For example, if the material is too sticky, spots remain even if the user has released his finger from the surface. In our prototypes, foamed silicone yielded the best results.

Compliant surface improves detection quality.

A projector displays the GUI on a diffusor layer placed on top of the surface. The projector can be either placed above (top projection) or inside (back projection) the table, while the latter is preferable to avoid shadows and occlusion issues. To avoid interferences between the touch detection mechanism and the GUI, an IR pass filter is attached to the camera, and, optionally, an IR block filter to the projector. This makes sure that the camera only sees spots caused by surface contact.

Projector displays GUIs. Filters avoid interferences between infrared and visible light.

The main advantage of FTIR is that it produces binary touch events for fingers, hands, or any other objects that exert enough pressure onto the surface. If an object is only slightly hovering above the surface, it is nearly invisible to the camera (Fig. 2.3, left). Accordingly, FTIR generates a good

+ Binary touches
+ Inexpensive setup
+ Easy to build

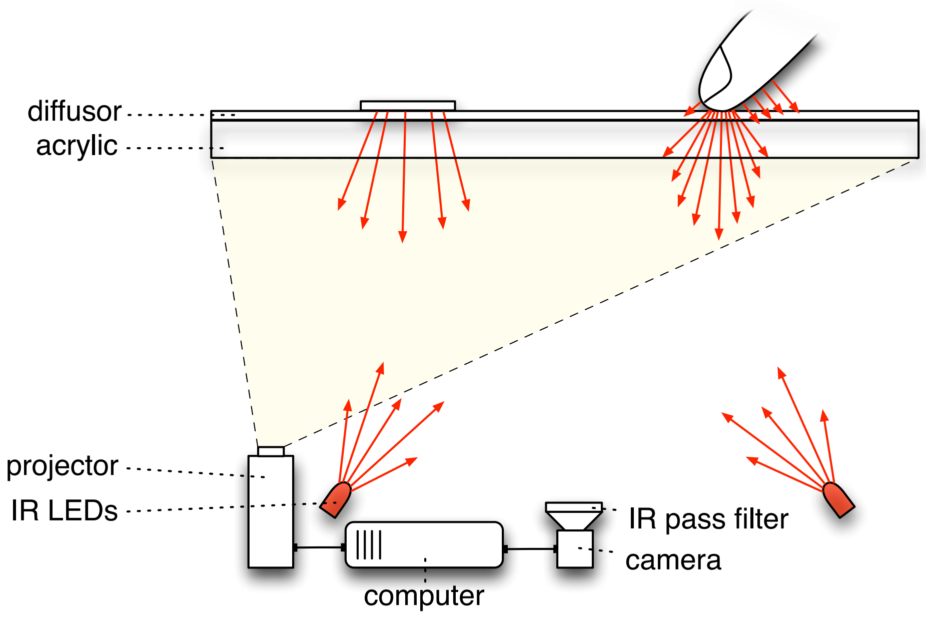


Figure 2.4: Typical DI tracking setup. DI allows to detect finger touches as well as objects lying on or hovering above the surface.

signal-to-noise ratio. Simple Computer Vision algorithms can be employed to convert a camera frame to a list of input events. To a certain degree, the brightness of spot can also be mapped to the applied pressure on the surface.

– No detection of visual markers

FTIR is optimal for finger touch tracking. It is also relatively inexpensive and easy to build, which is one reason for its success in the recent years. On the downside, FTIR can only detect physical pressure points. Recognizing visual patterns that are printed on paper is not possible.

FTIR can also be detected with a camera above the table.

As shown by Echtler et al. [2009], FTIR can also be implemented with a camera positioned above the table, as so-called *Inverted FTIR*. This allows to replace the projector with an LCD panel and makes the space beneath the surface available, e.g., for a user's legs when sitting at the table.

2.1.2 Diffused Illumination (DI)

DI tracking: LEDs inside the table emit IR light that is reflected by objects on and above the surface.

Diffused Illumination (DI), as introduced by Matsushita and Rekimoto [1997], employs a simpler technical construction (Fig. 2.4). IR LEDs are placed inside the table and emit light towards the surface. The light passes the transparent plate and is reflected by objects near the surface. A camera in the table then detects these reflections. Again, a projector renders the GUI onto a diffusor layer.

The main benefit of this technique is that it allows to detect visual markers. Attaching markers to physical objects allows to sense their position and

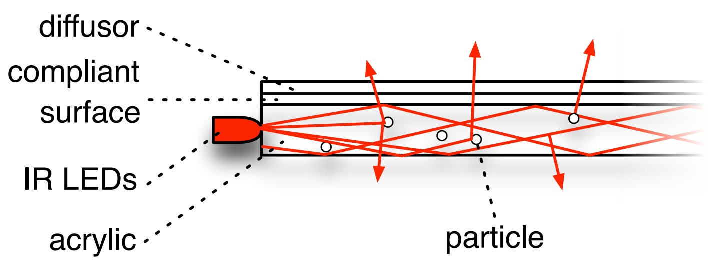


Figure 2.5: Principle of DSI tracking. Particles inside the acrylic diffusively reflect injected IR light.

orientation on the surface (e.g., Jorda et al. [2007]). Also, *hovering* of hands or objects can be detected and interpreted as further input. Note though that the more distant the reflections are from the surface, the stronger a diffusor layer blurs them. That is, hovering can only be detected within a limited range above the surface. A switchable diffusor, as presented by Izadi et al. [2008], can solve this issue.

+ Detection of visual markers
+ Inexpensive setup

DI is optimal for detecting visual markers on the surface. It also does not require a compliant layer. However, touch events are more difficult to track and demand a sophisticated visual differentiation between fingers close to the surface and those touching it (Fig. 2.3, right). Also, the exact construction of a DI setup is tricky. Specular reflections of IR light at the surface illuminate the components inside the table (projector, cables, etc.) that are then visible in the camera image by back reflection via the surface. Thus, the inside of the table should be isolated with light absorbing materials like velour. This, however, hampers a uniform light distribution on the surface via diffuse reflections. Usually, reflection artifacts are subtracted from the signal by acquiring a background frame beforehand, but DI generally causes a worse signal-to-noise ratio than FTIR when detecting finger touches only.

– Worse signal-to-noise ratio
– Uniform light distribution is difficult to achieve

2.1.3 Diffused Surface Illumination (DSI)

Diffused Surface Illumination (DSI) is a combination of FTIR and DI. The setup equals an FTIR setup, but the acrylic layer contains microscopic particles (Fig. 2.5). Some of the IR light rays that are fed into the surface bounce due to total internal reflection, some hit the microscopic particles and are diffusively scattered into all directions. DSI does not require IR light sources beneath the table surface. Note though that the particles scatter a substantial portion of the IR light into the table, causing a worse signal-to-noise ratio than FTIR.

DSI tracking: FTIR setup with particles inside acrylic. Part of the light is diffusively scattered like in a DI setup.

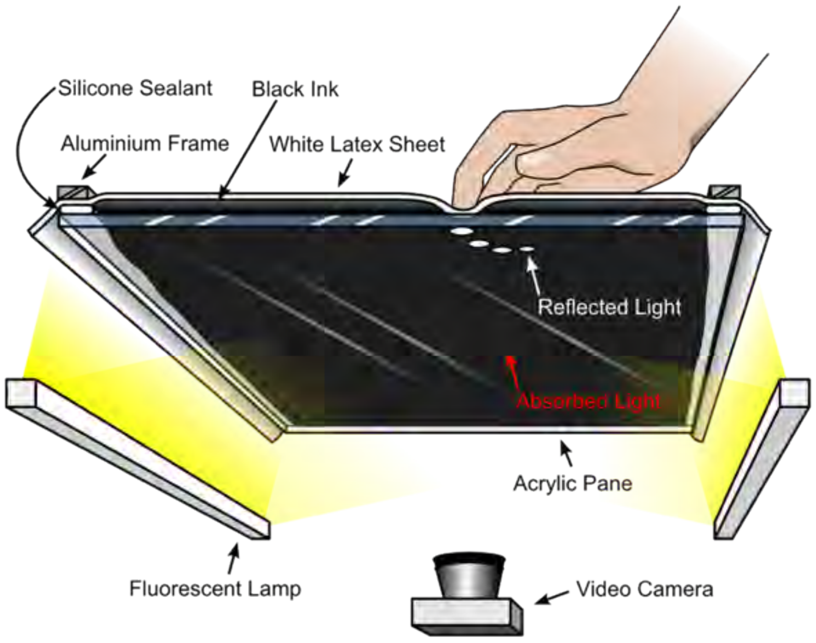


Figure 2.6: Liquid displacement tracking. Finger touches displace light absorbing ink and expose white spots to a camera. Image courtesy of Hilliges et al. [2008].

2.1.4 Liquid Displacement

Liquid displacement tracking: Touching the surface displaces light absorbing ink and reflects light at contact area.

Hilliges et al. [2008] introduce binary touch detection by sensing liquid displacements. The surface consists of black ink liquid on top of a transparent surface. White latex on top forms a pouch and encases the black ink so that the user touches the latex when interacting with the surface. As shown in Fig. 2.6, fluorescent lamps emit light onto the surface, while a camera in the table captures all reflections from the surface. Without any finger contacts, the camera only sees a black image. However, if a user touches the surface, the liquid is pushed aside, and the white latex touches the glass at the contact point. The latex reflects the light into the camera and creates a noticeable touch.

+ Very good signal-to-noise ratio
– Difficult construction
– Only works for horizontal surfaces

The technique provides a very good signal-to-noise ratio. On the downside, hovering cannot be detected, and top projection is required to display a GUI. Also, the construction is not simple and only works for horizontal, non-tilted surfaces.

2.1.5 Thin Form Factor Tracking

A common drawback of all previous approaches is the large volume of the device due to the induced distance of the camera and the projector. Hodges et al. [2007] addressed this issue with *ThinSight*. They mounted an array of

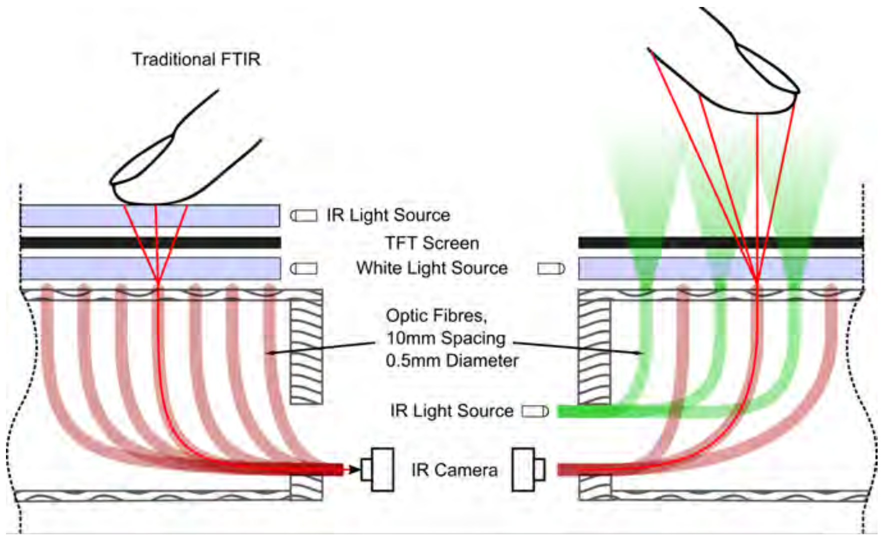


Figure 2.7: FiberBoard reduces the form factor of FTIR and DI tracking solutions using fiber optical cables. Image courtesy of Jackson et al. [2009].

infrared emitters and detectors behind the backlighting of an LCD panel. Similarly to DI, the IR LEDs emit IR light that is reflected by objects close to the surface. IR photodiodes detect these reflections. Although the LCD panel considerably attenuates IR light, the signal is still strong enough to detect finger touches and markers. *FLATIR* is a similar system that uses an FTIR surface instead of IR emitters [Hofer et al., 2009]. On the one hand, these techniques allow thin form factors; e.g., these devices could also be placed on a wall. On the other hand, the engineering efforts and costs for achieving a reasonable resolution are relatively high.

Photodiodes instead of a camera allow for a thinner form factor but require higher engineering effort.

FiberBoard by Jackson et al. [2009] combines FTIR or DI with an array of fiber optics mounted behind an LCD panel. The fiber optics divert the reflections from the surface to a nearby camera (Fig. 2.7) yielding a table thickness of only 8 cm. This method effectively reduces the thickness of an interactive tabletop but also involves a difficult construction process that is only feasible for a relatively low resolution. The paper placed 40 $\times$ 30 fiber optical cables behind a 19" LCD screen achieves a resolution of 2.63 dpi. Spots are detected after bicubic upscaling the low resolution camera image. According to the authors, this low resolution grid is sufficient to hit targets with sub-finger-width. We will look at this tracking technology in chapter 4 in more detail.

FiberBoard uses fiber optics to reduce volume beneath tabletop for FTIR and DI setup.

A promising upcoming thin form factor tracking technology are *sensor-in-pixel (SIP) displays*[1], which will be part of the second version of the commercial interactive tabletop *Microsoft Surface*. These displays contain pixel-sized infrared emitters and sensors embedded into a high resolution RGB LCD matrix. The underlying principle equals that of ThinSight but is considerably smaller. The about 10 cm thick display can even scan text on paper sheet.

Sensor-in-pixel displays integrate pixel-sized emitters and sensors into the display for touch detection.

[1]This technology is also referred to as *Microsoft PixelSense*$^{\text{TM}}$.

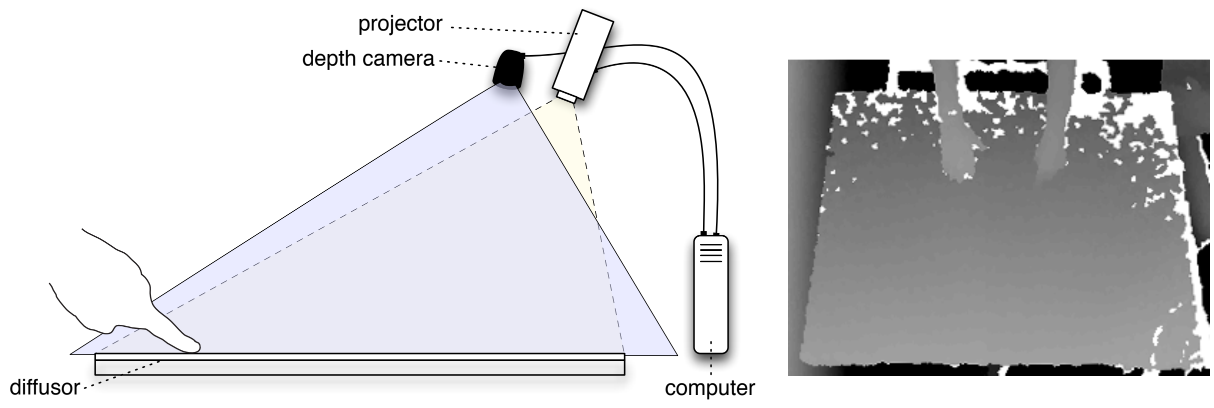


Figure 2.8: Tracking interactions on and above tabletops using depth cameras. Left: Setup involving top projection. Right: Example output of a depth camera placed above a table. Pixel intensity encodes depth.

2.1.6 Depth Cameras

Depth cameras allow to track hands and objects above the surface. However, exact detection of surface touches is difficult.

Wilson demonstrated that depth cameras, such as *Microsoft Kinect*[2], can also be used as touch detectors on interactive surfaces [Wilson, 2010]. The Kinect camera emits a random dot pattern via an IR laser projector. An internal IR camera generates a depth image by reprojecting local 2D patterns using the known camera intrinsics and extrinsics. A depth camera placed above a tabletop delivers a 2D depth map including the surface plane and all objects in between (Fig. 2.8). With simple thresholding against the surface, objects near the surface can be extracted and converted into touch events. The major benefit of this method is that it can also track interactions above the surface (e.g., [Hilliges et al., 2009; Marquardt et al., 2011]) or touches on non-planar surfaces. On the downside, the detection using depth cameras is not as accurate as previously mentioned approaches. Besides technical limitations like sensor noise and resolution, in the aforementioned setup, the camera only captures the *backside* of fingers and objects. This makes differentiation between hover and touch difficult.

2.1.7 Digital Pens

Digital pens retrieve their position by scanning imprinted micro patterns. They allow precise hand-drawn input.

An *Anoto Pen* is a commercial digital stylus allowing precise real-time input on interactive surfaces[3]. It is based on a micro random dot pattern printed on a sheet. The dots are so small that they can barely be noticed by users. The stylus contains a small camera beneath the tip that captures the local imprinted pattern when the tip touches the surface. The pattern is designed so that the local pattern can be mapped to a unique position. The stylus allows for both, actual writing with ink as well as digital capturing the input and streaming it in real-time. Digital pens like the Anoto Pen

[2]http://www.xbox.com/en-US/kinect
[3]http://www.anoto.com/

are especially helpful for precise hand-drawn input like annotations or note taking. We refer to Steimle [2012] for an overview.

2.1.8 Combining Techniques

Visual tracking techniques can be combined. Filters avoid interferences.

Most visual tracking techniques can be combined. For example, if exact binary touches and recognition of visual patterns are required, FTIR and DI can be used simultaneously by employing different wavelengths and specific band pass filters for each technique. FTIR and Anoto pens can be combined by printing the dot pattern onto the diffusor [Leitner et al., 2009]. As long as tracking signals do not interfere, combining a depth camera with an FTIR or DI technique is also imaginable.

2.2 Input Techniques

Using the previously mentioned tracking methods, interactive tabletops can provide a wide spectrum of input methods, from simple tapping, over complex gestures, to interaction with physical objects. We discuss the most common ones in this section.

2.2.1 Triggering and Arranging Virtual Objects

Virtual buttons are activated by tapping them.

The simplest way of interacting with a tabletop is activating on-screen objects by just tapping them. For example, virtual buttons, as they are wide spread on touch screens in public spaces, are triggered by tapping them with a finger.

Nearly all multi-touch enabled tabletops provide the free arrangement of virtual rectangular objects projected on the screen. This technique is often referred to as "photo sorting", because it is a popular demonstration for direct manipulation. Virtual objects, such as images, videos, or text boxes, are projected on the screen. They can be overlapped and virtually stacked.

Virtual objects can typically be arranged with the following basic operations:

Basic operations include bringing an object to the front by tapping, dragging it with a finger, or transforming it with multiple fingers.

1. *Bring to front*: Similar to clicking a window in a desktop GUI, tapping an object brings it to the front.
2. *Translation*: Putting a finger onto an object at position p, moving it to a new position p', and releasing it again, translates the object by $\Delta p = p' - p$ according to the finger's trajectory (Fig. 2.9, left).
3. *Fixed aspect ratio transform*: Dragging two points of an object allows to translate, rotate, or uniformly scale it. This operation is prevalent

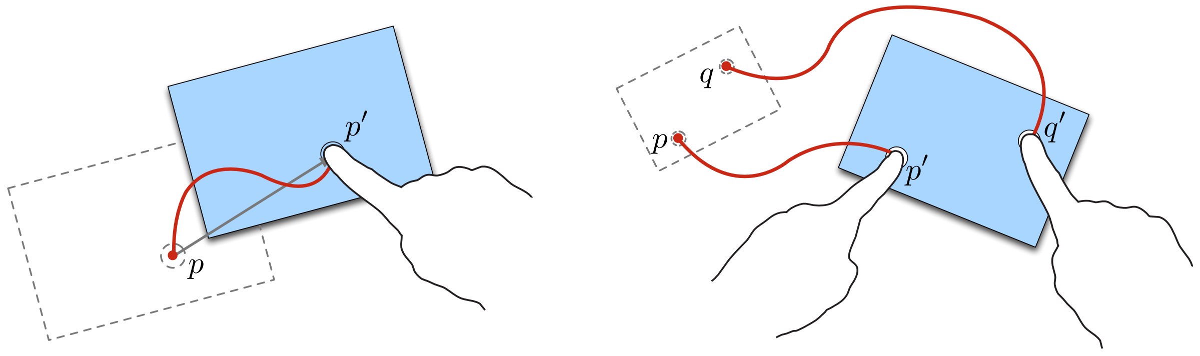


Figure 2.9: Common direct manipulation techniques. Left: Translation with single finger. Right: Translation, rotation, and scale using two fingers.

because it maintains the aspect ratio of objects, which is helpful when scaling photos or maps.

By default, this is implemented by computing an affine transform for the object so that start and end point of the dragging trajectories map to the same point on the object (Fig. 2.9, right). Thus, the matrix representing the affine transform in extended coordinates[4] reads

$$M \quad := \quad \begin{pmatrix} s \cdot \cos(\alpha) & -s \cdot \sin(\alpha) & t_x \\ s \cdot \sin(\alpha) & s \cdot \cos(\alpha) & t_y \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} a & -b & c \\ b & a & d \\ 0 & 0 & 1 \end{pmatrix}$$

where s is uniform scale factor for x and y dimension, α is the angle of rotation, and $t = (t_x\ t_y)^T$ is the 2D translation.

Assuming that p, q are the starting positions of both trajectories and p', q' are new positions, both in extended coordinates, the affine matrix is computed by solving the linear system

$$\begin{aligned} & p' &= M \cdot p \\ \wedge \quad & q' &= M \cdot q. \end{aligned}$$

Aspect ratio preserving transform is defined by two finger draggings. Least-squares fit is used for dragging with more than two fingers.

Given only four unknowns (a, b, c, d), the affine transform M can be uniquely determined by two different start and end points. Note that a general 2D affine transform matrix contains six parameters, but we exclude non-uniform scaling and shearing in this operation, which removes two degrees of freedom.

If more than two fingers are used, the linear system is overdetermined and can be solved using a least-squares fit. Let p_i be the starting and p'_i be the ending points of n trajectories, with $i \in \{0, 1, ..., n-1\}$. We then find M by solving

$$\sum_{i=0}^{n-1} \left\| p'_i - M \cdot p_i \right\|^2 \quad \rightarrow \quad \min .$$

[4] Extended coordinates include an additional component for each vector, where 1 indicates a point and 0 a direction. They allow to write affine transforms as matrices with one dimension higher than their contained linear map.

These basic operations are both easy to learn and easy to implement. Nearly all multi-touch-based mobile devices support them. However, the restriction to fingertip contact as only input considerably limits the expressiveness of the hand. *ShapeTouch* by Cao et al. [2008] addresses this issue. Instead of converting spots to events consisting of 2D position and radius, ShapeTouch interprets the entire contact shape as input and, e.g., allows to use the palm or thenar for input. The authors introduce a *virtual force metaphor* that maps the size of the contact shape to force exerted on objects. Using this metaphor, they created a pseudo 3D model that allows to stack virtual objects or, e.g., to peel an object back to insert another one beneath it. Hilliges et al. [2009] employ a 3D physics simulation to enable interactions with 3D objects rendered on the surface. Users perform input via gestures above the surface that are projected down to the tabletop. For example, 3D objects can be grasped and moved via a pinch gesture. A depth camera captures all interactions in the space above the table.

Use of fingertip input only limits expressiveness of hand. Alternative approaches employ full hand shape as input.

2.2.2 Drawing and Gestures

An interactive tabletop is an ideal platform for drawing digital images, handwriting texts, or annotating documents using fingers or styli. This kind of input is natural and must not be learned, because many real-world analogies exist.

Interactive tabletop is ideal for graphical drawing tasks.

A major challenge of interaction beyond direct manipulation of objects is the execution of *actions*. For example, imagine a function for deleting an image object. Dragging objects to a dedicated deletion area, like a “trash bin”, or using menu bars is not an option, because it might be out of reach for some users. Multiple trash bins waste space on the surface, just like attaching a delete icon to each image, which also impairs the visual design. *Gestures* have been proven useful to trigger actions on interactive tabletops.

A gesture is a sequence of input events on a tabletop that is mapped to a semantic meaning. In our example, an image could be deleted by just striking it out, or more formally, by creating a dragging trajectory that can be segmented into lines that are alternatively aligned into opposite directions (Fig. 2.10, left). Gestures can be designed within a continuum from abstract mappings, e.g., double-tap to copy object, down to those with a geometric meaning, such as drawing a shape around objects to select them. A common gesture that is implemented in many demonstrations is *flicking*: Quickly dragging an object and releasing it without decelerating, gives the object a certain amount of inertia; it keeps moving until it is stopped by virtual friction (Fig. 2.10, right). This is helpful to move virtual objects across large distances. Gesture input is not limited to the planar surface but has also been implemented within the space above the surface [Hilliges et al., 2009].

Gestures are helpful to trigger actions without the need for icons or menus.

Example: flicking for moving objects across long distance

A common challenge when using gestures is the disambiguation of inputs. A gesture can only be recognized as such *after* the input has been per-

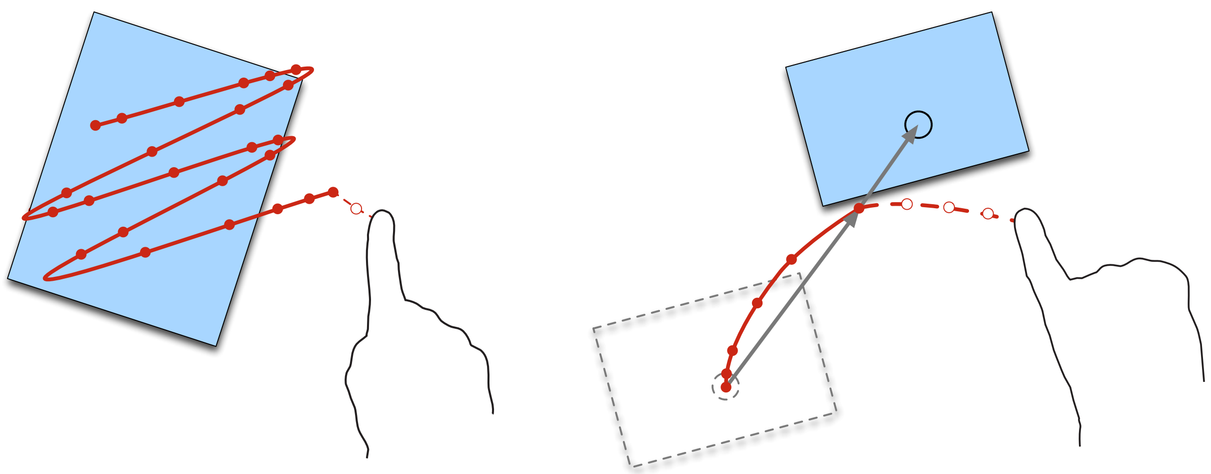


Figure 2.10: Examples for gestures on the surface. Left: Striking out an object. Right: Flicking an object. Continuous red lines represent touch trajectories. Dashed lines are projected finger positions after releasing the table. Red dots denote constant time steps and illustrate local velocity.

Challenges: disambiguation between gestures and direct manipulation, lack of visibility

formed. Until the input is terminated, the system can, e.g., not differentiate whether a user wants to quickly move an object or whether she is striking it out for deletion (cf. Fig. 2.10, left, again). It is not predictable whether an input belongs to a gesture or direct manipulation, an issue which can only be solved using (quasi-)modes. Also, gestures are invisible and must be learned by the users.

There is a whole body of research dealing with gesture input on interactive tabletops. We refer to Wobbrock et al. [2009] for an overview on related work and an exploration of tabletop gestures. In section 3.6.2.4, we will describe the implementation of a 2D gesture detection engine.

2.2.3 Tangible User Interfaces

In their seminal paper about *Graspable User Interfaces*, Fitzmaurice et al. [1995] propose to interact with digital data tabletops using physical objects. They introduce a first hardware prototype involving a back projected display and cubes that are tracked via an Ascension Flock of Birds magnetic tracking device. The authors present a drawing application, *GraspDraw*, that allows users to create and manipulate simple shapes using physical objects. Instead of using virtual GUI menus, users employed a physical gesture to select a drawing tool; they had to “dunk a brick in a compartment in [a] tray to select a particular tool” [Fitzmaurice et al., 1995, p. 446]. This paper marks the birth of the so-called *Tangible User Interfaces* [Ishii and Ullmer, 1997].

A Tangible User Interface (TUI), short *tangible*, is a physical object that represents and manipulates digital data [Ishii, 2008]. An early example is

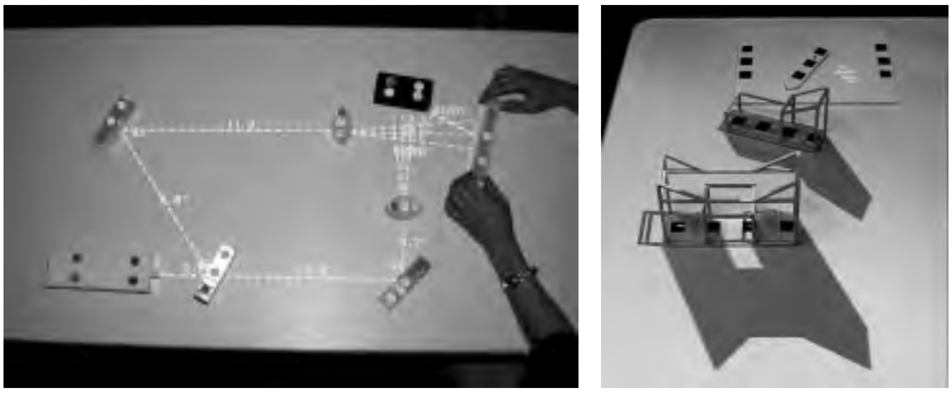


Figure 2.11: Early projects involving Tangible User Interfaces on interactive tabletops. Left: Illuminating Light provides tangibles that represent optical elements, such as light sources and lenses. Users can arrange them on the tabletop to conduct optical simulations. Right: Urp is an urban planning application. Tangibles represent buildings that can be aligned on the surface. Digital simulations of shadow, wind, and reflections enrich the urban model. Image courtesy of Underkoffler and Ishii [1998, 1999].

A tangible is a physical object representing and manipulating digital data.

Illuminating Light, which allows users to simulate optical effects for educational purposes [Underkoffler and Ishii, 1998]. Users can put physical objects representing optical artifacts on a table, such as a laser light source, a mirror, or a lens. The application then simulates a ray cast in the 2D plane of the table, starting from the light sources passing through all optical tangibles (Fig. 2.11, left). In a similar fashion, *Urp* allows architects to plan the layout of buildings and to investigate their environmental effects [Underkoffler and Ishii, 1999]. Users can place wooden models of buildings on an interactive table. A projector then renders, for example, the cast shadow for that 3D model according to a variable sun position (Fig. 2.11, right). This enables architects to physically assemble the layout of a city and check at which time of day a certain region, e.g, a park, receives sunlight. The same application can also simulate wind or the reflection of sun due to glass windows. Urp also provides some physical objects to perform measurements, such as the distance between two points.

Tangibles give data a physical form and enable intuitive haptic interaction.

Tangibles give virtual data a physical form and enable direct physical manipulation. In contrast to touch interaction with GUIs, they provide rich haptic feedback and allow users to employ their full spectrum of haptic skills:

> “Tangible User Interfaces (TUIs) aim to take advantage of these haptic interaction skills, which is a significantly different approach from GUI. The key idea of TUIs is to give physical forms to digital information. The physical forms serve as both representations and controls for their digital counterparts. TUI makes digital information directly manipulatable with our hands, and perceptible through our peripheral senses by physically embodying it.”
>
> [Ishii, 2008, p. xvi]

Challenge: Maintain physical-visual consistency of tangibles

Muscle memory is available, and objects can even be controlled eyes-free. A particular design challenge is to maintain the illusion that the physical object and the projected virtual data belong to the same object, i.e., to maintain the visual-physical consistency of all tangibles:

> "The success of a TUI often relies on a balance and strong perceptual coupling between the tangible and intangible representations. It is critical that both tangible and intangible representations be perceptually coupled to achieve a seamless interface that actively mediates interaction with the underlying digital information, and appropriately blurs the boundary between physical and digital. Coincidence of input and output spaces and realtime response are important requirements to accomplish this goal."
>
> [Ishii, 2008, p. xvii–xviii]

For example, in the Urp project, the rendering of shadows must be consistent with the positions of the physical objects. Otherwise, the illusion that the physical object and the visual projection belong to the same tangible breaks down.

Interactive tabletop is a suitable platform for tangible interaction.

Interactive tabletops are a well suited platform to maintain physical-visual consistency of tangibles: The positions and orientations of physical objects on the surface can be easily tracked via attached visual markers, e.g., using DI. The projector or display of the table can update the visual representation accordingly. However, real-time tracking and updates of the projection are crucial.

In the following chapters, Tangible User Interfaces will play a crucial role for providing haptic feedback on interface tabletops.

2.3 Non-Horizontal Surfaces

BendDesk is a non-horizontal, curved multi-touch desk that allows users to sit at the device.

Besides horizontal tabletops, interactive surfaces using other angles and shapes have been constructed. We created *BendDesk*, a curved multi-touch table [Weiss et al., 2010c]. It was envisioned as an interactive desk that is entirely touch-based and does not require conventional input devices, such as keyboards and mice. It merges a vertical and horizontal interactive surface with a curve (Fig. 2.12), using FTIR tracking on all surfaces. A major feature of this system is that it mimics the ergonomics of a regular work place. By using short throw projectors with aspheric mirrors and cameras at steep angles, users can sit at the table and reach the entire surface without occluding the projection or camera. While the underlying technology is similar to conventional table setups, special care has to be taken to compensate for the substantial geometric distortion induced by the overlapping projectors and cameras. Curve is a similar system with a vertical surface that is slightly tilted backwards [Wimmer et al., 2010].

FLUX is a tiltable multi-touch table.

FLUX is a tiltable table by Leitner et al. [2009] that can be transformed from a collaborative horizontal surface, over a single user tilted drafting

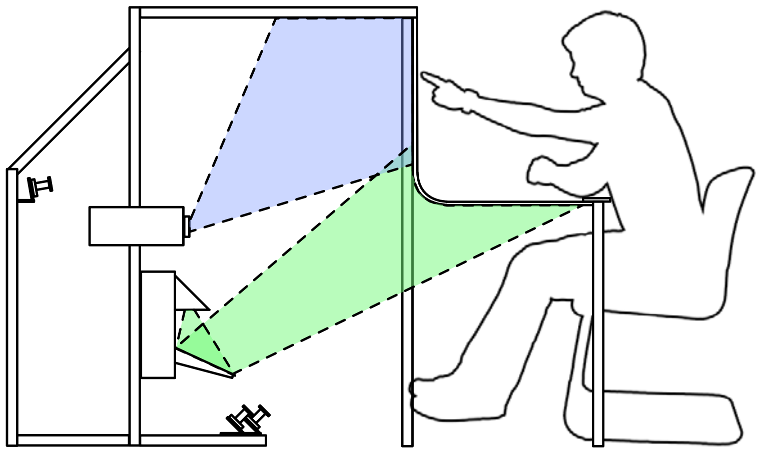


Figure 2.12: BendDesk is a curved multi-touch desk. Two short-throw projectors generate the graphical output. Three cameras capture touches on the surface. Image adopted from [Weiss et al., 2010c].

table, to a vertical presentation surface. It embeds a mirror into the construction beneath the tabletop to provide leg space.

Multi-touch input is also possible on spheres using DI.

Also pure organic shapes have been implemented. Benko et al. [2008] introduced an interactive multi-touch enabled sphere. It is based on DI and uses a wide angle lens for projection and touch detection. A cold mirror makes sure that both IR lights and camera share the same optical path.

2.4 Capturing Design Knowledge

Capturing tabletop design knowledge across disciplines avoids repetition of common mistakes.

Constructing interactive tabletops and developing applications for them is tricky and requires many iterations if started from scratch. Capturing design knowledge and experience into a form that can be efficiently read and extended, can help researchers, engineers, and designers to access the field of interactive tabletops, prevent them from “reinventing the wheel”, and avoid repetition of common beginner’s mistakes.

Although the research field is relatively young, there has already been publications that capture design knowledge. Scott et al. [2003] present a set of eight design guidelines for collocated collaboration on interactive tabletops:

Guidelines provide general advice for designing systems and interaction techniques.

1. Support Interpersonal Interaction
2. Support Fluid Transitions between Activities
3. Support Transitions between Personal and Group Work
4. Support Transitions between Tabletop Collaboration and External Work

5. Support the Use of Physical Objects
6. Provide Shared Access to Physical and Digital Objects
7. Consideration for the Appropriate Arrangements of Users
8. Support Simultaneous User Actions

Taxonomies categorize existing systems and input techniques. They help to detect "blank spots" in a field.

The paper, which is targeted on researchers, backs up these guidelines with examples and various references to research papers. Wallace and Scott [2008] describe contextual factors, such as social and cultural, activity, temporal, ecological, and motivational factors, and how they influence the design of the software interface, physical form, and connectedness. Another way to capture design knowledge are taxonomies, which provide a space to categorize existing systems and input technologies. Grossman and Wigdor [2007] introduced a taxonomy for 3D interaction on tabletops.

Drawback of guidelines and taxonomies: no concrete directions, and limitation to certain target groups.

Guidelines provide general abstract advice for designing systems and interaction techniques, while taxonomy gives an overview on a field and helps to find unexplored "blank spots". However, a common drawback of both approaches is that they do not provide *concrete directions* for specific problems that occur in a design process. They are also often limited to a certain target group, which hinders a knowledge exchange between designers, practitioners, and researchers.

HCI design patterns capture solutions for recurring problems in HCI.

We addressed this issue by developing an HCI design pattern language for interactive tabletops. An HCI design pattern is a document that captures the solution for a recurring problem in HCI. A pattern language is a set of interconnected patterns. Based on the concepts introduced by Alexander et al. [1977] and Borchers [2001], a pattern language is characterized by the following aspects:

Patterns are comprehensible for an interdisciplinary audience.

- *Lingua franca:* Patterns are targeted at wide interdisciplinary audience that involves researchers, designers, and practitioners. Thus, patterns are written in a language that everyone, also non-experts, understands.

Solutions are described in a specific context.

- *Solutions in context:* Instead of general advice, patterns provide solutions within a specific context. Each pattern contains concrete example scenarios in which a problem occurs and the solution is applicable. Also, patterns provide links to other related patterns.

Patterns are easy to read.

- *Readability:* Patterns are written in prose that is easy to understand for non-experts. Furthermore, patterns follow a clear consistent format, which allows to read them in an efficient way.

Pattern languages are organized on different abstraction levels.

- *Different abstraction levels:* Pattern languages can be arranged on different scales, starting with general patterns on the highest level down to very specific patterns. Using the links between patterns, a reader can navigate within these abstraction levels.

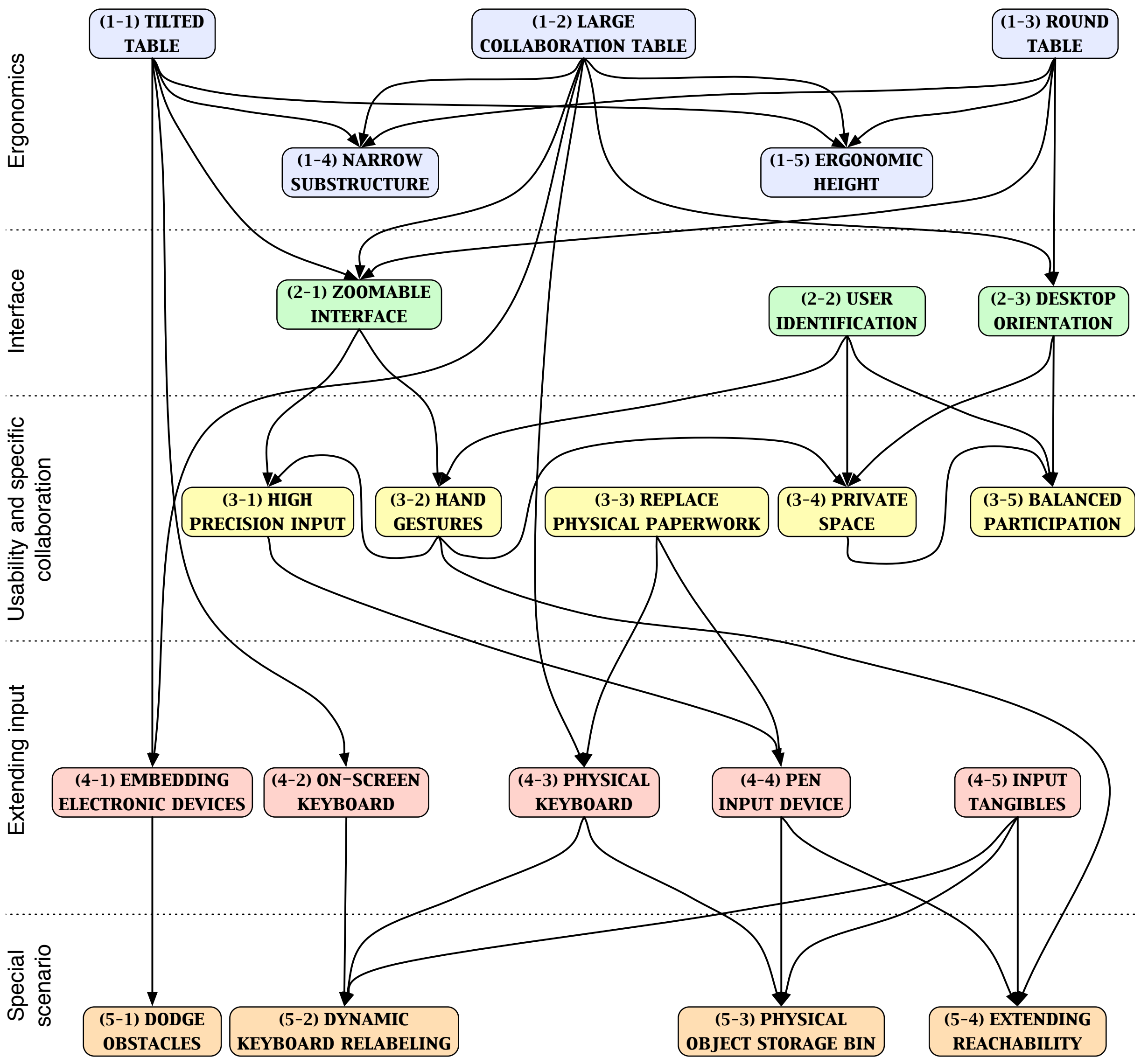

Ergonomics
Interface
Usability and specific collaboration
Extending input
Special scenario
(1-1) TILTED TABLE
(1-2) LARGE COLLABORATION TABLE
(1-3) ROUND TABLE
(1-4) NARROW SUBSTRUCTURE
(1-5) ERGONOMIC HEIGHT
(2-1) ZOOMABLE INTERFACE
(2-2) USER IDENTIFICATION
(2-3) DESKTOP ORIENTATION
(3-1) HIGH PRECISION INPUT
(3-2) HAND GESTURES
(3-3) REPLACE PHYSICAL PAPERWORK
(3-4) PRIVATE SPACE
(3-5) BALANCED PARTICIPATION
(4-1) EMBEDDING ELECTRONIC DEVICES
(4-2) ON-SCREEN KEYBOARD
(4-3) PHYSICAL KEYBOARD
(4-4) PEN INPUT DEVICE
(4-5) INPUT TANGIBLES
(5-1) DODGE OBSTACLES
(5-2) DYNAMIC KEYBOARD RELABELING
(5-3) PHYSICAL OBJECT STORAGE BIN
(5-4) EXTENDING REACHABILITY

Figure 2.13: Structure of our tabletop pattern language. This version contains patterns in five categories. Patterns are put into context to each other via links. Image adopted from [Remy et al., 2010].

- *Generativity:* Pattern languages provide a generative structure that inspires to create new designs from the patterns.

Patterns inspire new designs.

- *Teaching:* Pattern languages are helpful to introduce beginners to a novel field. Recent studies show that they outperform traditional guidelines in terms of learning success [Koukouletsos et al., 2009].

Patterns are helpful for teaching.

- *Examples:* All patterns contain concrete examples in which a solution has been successfully applied.

Every pattern contains concrete examples.

Fig. 2.13 shows the structure of our pattern language. It provides patterns for ergonomics, interface design, usability and collaboration, input, and special scenarios. It should be mentioned that a pattern language is never complete and always represents a snap-shot of current knowledge. This

Pattern languages are always in a process of iteration and refinement.

applies especially to the field of interactive tabletops that is relatively new and fast evolving due to new engineering and research results. Patterns are added, refined, and removed over time. Authors add a ranking to each pattern reflecting how confident they are in the maturity of the presented solution.

We published the pattern language at the European Conference on Pattern Languages of Programs (EuroPLoP 2010) [Remy et al., 2010] together with Christian Remy, Martina Ziefle, and Jan Borchers. The development of the language was also the topic of the diploma thesis by Remy [2010], supervised by the author of this thesis.

2.5 Closing Remarks

In this chapter, we provided an introduction to the field of interactive tabletops. We described key characteristics and challenges, optical tracking approaches, and input methods. We also looked at non-horizontal surfaces and HCI design patterns as knowledge repositories for tabletop design.

One of the key benefits of interactive tabletops, which is often named in literature, is direct manipulation. It is indeed a natural way for interacting with virtual objects, such as photos distributed on the surface. However, its usage is limited to spatial tasks. In productivity applications, text is typed, abstract parameters are edited, and precise input is required. For these tasks, general-purpose controls, such as buttons, knobs, sliders, and keyboards, are indispensable. These controls are well known from conventional desktop GUIs, but their transfer to tabletops is difficult. Operating on-screen controls is imprecise and requires visual attention during operation. We will deal with this topic in the next chapter and introduce haptic general-purpose controls that respect the specific nature of interactive tabletops.

Chapter 3

Translucent Controls on Tabletops: SLAP Widgets

In the last chapter, we introduced interactive tabletops. They are input and output devices that provide a large interactive surface and enable direct interaction with objects by multiple users. Direct finger input provides an intuitive and convenient way to interact with on-screen objects on interactive tabletops. Users can trigger buttons by just tapping on them, move and transform pictures by dragging them, or just draw something with their fingers. However, while direct manipulation is well suited for spatial arrangements and drawing tasks, its application is restricted in the domain of productivity tasks.

Direct manipulation and gestures are not appropriate to substitute general-purpose UI controls.

Productivity tasks on conventional desktop computers, e.g., composing texts, editing spreadsheets, or annotating videos, require a set of generic input controls, such as a physical keyboard and on-screen UI elements like sliders, knobs, and buttons that are controlled with a mouse. Direct manipulation is not expressive enough to substitute those general-purpose controls. Gestures enable more complex input, but they are invisible, hard to learn, and do not map well to abstract parameters. For example, finding two different intuitive gestures for setting the saturation of an image and the volume of a video is difficult.

Publications
SLAP Widgets was developed in collaboration with the Department of Cognitive Science, University of California, San Diego. This work has been first published as poster at the TABLETOP '08 workshop [Weiss et al., 2008] and then as full paper at the CHI '09 conference [Weiss et al., 2009b]. It has been demonstrated at the CHI '09 [Weiss et al., 2009c] and the TEI '09 conference [Weiss et al., 2009d]. SLAP Widgets was the topic of the diploma thesis of Julie Wagner [Wagner, 2009], supervised by the author of this thesis. Parts of the chapter, especially the related work, has been published as book chapter [Weiss et al., 2010a]. The user study on keyboard performance was accepted as contribution to a workshop on text entry at the CHI '12 conference [Weiss et al., 2012].

However, conventional desktop GUIs cannot be just transferred to interactive tabletops, because they do not follow the nature of these devices:

Tabletop UI elements must be moveable and rotatable.

- Unlike a desktop GUI, the UI configuration on a tabletop is in a *steady flow*. Since multiple users can access the same data from different sides, the arrangement of UI elements must be flexible. UI elements must be *moveable* and *rotatable*.

UI elements must be disposable.

- Space matters if multiple users are involved in a task. Accordingly, attaching a full set of editing buttons to every element on the table is not an option. UI elements must be disposable: They should be added to the interface when needed, and removed if not required anymore.

Tabletops require multi-focus policy.

- There is not a single focus policy, i.e., UI elements must be *associated* with the data they manipulate.

Productivity tasks require efficient and precise input.

- And most importantly: Users demand efficient and precise input methods in productivity applications. Due to the lack of haptic feedback, directly operating on-screen controls with the fingers is imprecise. Users cannot *feel* virtual controls when touching the surface. They always require users to focus on the control they operate (e.g., a jog wheel), instead of the object they manipulate (e.g., a video). Furthermore, the relatively large size of a fingertip limits input precision. These drawbacks lead to both higher error rates and higher task completion times.

In this chapter, we introduce a novel class of general-purpose tabletop controls: *SLAP Widgets*. They were envisioned and implemented as input methods that support productivity tasks in an efficient way, while respecting the nature of interactive tabletops. SLAP Widgets combine the flexibility of on-screen UI elements with the haptic benefits of physical controls.

3.1 Design Considerations

Design requirements:

The intention of our research was to develop general-purpose tangible controls for interactive tabletops that are suitable to augment productivity tasks. To support these tasks while making use of the specific nature of tabletops, our controls had to fulfill a set of design requirements:

- Physicality as conventional controls

- *Strong physicality:* Our tangibles should provide the haptic quality of conventional physical controls. They should give users rich haptic feedback and strong affordances. Furthermore, they should guide the users' motion for eyes-free interaction.

- Scalability

- *Scalability:* A control should be applicable and reusable for different productivity tasks. An extension of the complexity of an application should not imply the development of a new control.

- *Ad hoc use:* Interactive tabletops are dynamic working environments. The entire user interface can be changed quickly, especially when multiple users interact with the system simultaneously. An optimal control should be useable in an ad hoc manner: A user can put the control anywhere on the table when she needs it, operate it as long as she wants, and remove it from the surface afterwards. That is, tangible controls should be integrated into the fluent interaction on tabletops without consuming space when not needed. This also implies that they should be lightweight and easy to move on the table.

- Usable and removable during run-time

- *Unobtrusive:* A physical tabletop control should not interfere with any direct touch interaction. Therefore, it should not consume more space on a tabletop than required to operate the control. This implies that it should be self-contained without using cables that could potentially clutter the interface.

- Low space consumption, self-contained design

- *Consistent feedback:* The control should provide physical and visual feedback to the user about its current state. As tangible user interfaces depend on the illusion that their physical and virtual representation are merged, the physical state of the control should always be tightly coupled with its visual state.

- Consistent physical-visual feedback

- *Easy to prototype:* The underlying system should allow designers to rapidly create and iterate new tabletop devices. This implies that the process of building should not require an extensive engineering background. The underlying technology should be mostly hidden from both users and designers.

- Support of easy prototyping

- *Robust:* The control should be sufficiently stable for everyday's use.

- Robust construction

3.2 Related Work

As explained in the previous chapter, tangibles are a synthesis of physical objects and digital data. Users can perceive the state of tangibles from its visual-physical configuration, and they can directly change the data by physically manipulating the tangible. Tangibles were originally envisioned as interactive objects for specific purposes:

> "Tangible User Interface serves as a special purpose interface for a specific application using explicit physical forms, while GUI serves as a general purpose interface by emulating various tools using pixels on a screen."
>
> [Ishii, 2008, p. xvi]

Our research intends to develop *general-purpose* tangibles that can be employed in various productivity tasks. Many projects have been developed that combine the benefits of haptic feedback and a dynamic graphical representation. In the following, we describe crucial milestones in the continuum from pure special-purpose to general-purpose applications.

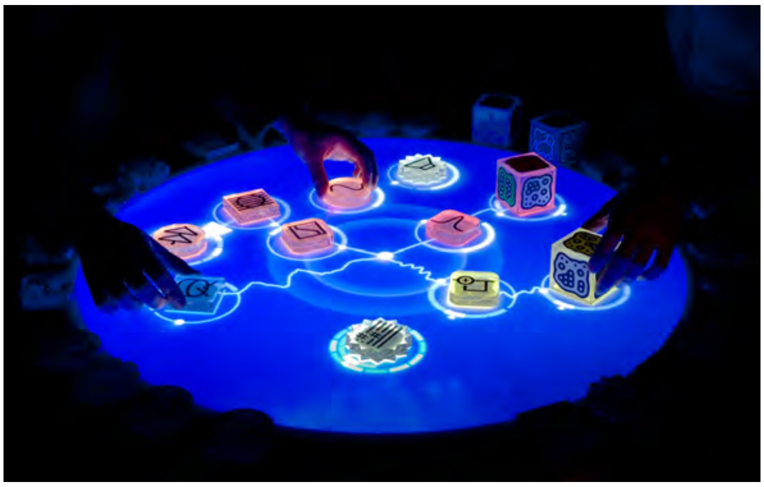


Figure 3.1: The reacTable is a musical instrument based on tangibles arranged on an interactive tabletop. Photo taken by Xavier Sivecas. Image courtesy of Jorda et al. [2007].

3.2.1 Special-Purpose Tangibles

Initial special-purpose tangibles were limited in terms of scalability.

Two early special-purpose tangibles have already been described in the previous chapter: Illuminating Light and Urp allow to place physical objects on the tabletop and to conduct physical simulations. The power of these applications is their intuitive use and the absence of any indirect input methods or modes. Again, this supports the idea of Ubiquitous Computing by hiding the input technology [Weiser, 1991]. However, their scalability is limited. Each tangible type in these applications is represented by exactly one physical and visual representation. The application input is determined by the placement of the tangibles. Enriching the application requires building new tangibles for every single entity.

reacTable allows to compose music by arranging tangibles. These represent entities but can also change values.

Later projects have addressed this issue by adding further degrees of freedom to the interaction. The *reacTable* is a table-based musical instrument allowing users to synthesize music by placing tangible blocks on a table [Jorda et al., 2007]. The system provides a variety of different blocks, e.g., for sound generation, effect generation, or sequencing of samples. Each block is made of semi-translucent acrylic; the shape, color, and label imprinted on top indicates the function of the tangible (Fig. 3.1). When a user places one of the tangibles on the table, its current state is projected next to it. Unlike the aforementioned projects, reacTable allows to configure each tangible individually. For example, after placing a global volume control onto the table, users can change the volume by rotating the tangible. They can also use finger gestures and operate virtual sliders and buttons around each object. Furthermore, reacTable lets users build connections between the elements on the table. A basic example is an oscillator tangible that transfers the generated sound to a band-pass filter. That band-pass filter then pipes the filtered result to the output channel. Unlike Illuminating Light and Urp, which incorporate tangibles as small-scale *representations*

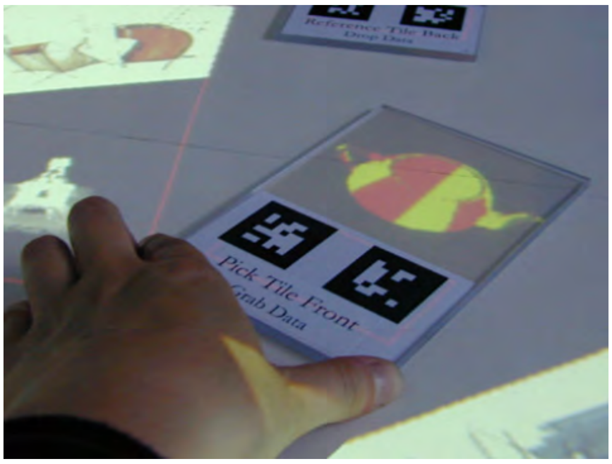


Figure 3.2: Tangible Tiles are physical general-purpose tiles for interaction with virtual objects. Image courtesy of Waldner et al. [2006].

of objects, reacTable assigns modular *functions* to physical objects. Users can also control them like *input devices* that change intangible properties. The modular concept and abstract design of the tangibles enables various ways to synthesize complex sounds and to extend the set of tangibles.

Modular concept and abstract design of tangibles increase scalability.

This idea of modular tangibles has also been applied to the field of programming. *Scratch* by Horn et al. [2009] is a tangible programming toolkit that intends to teach basic programming concepts. The system is deployed as a museum exhibit that lets users control a robot by assembling building blocks to simple programs. In a user study, the author compared a version with tangible blocks to a pure on-screen touch-based version. Although the programs that visitors created were not significantly different between both conditions, the study revealed that the tangible interface attracted more visitors than the digital version and encouraged a more active collaboration of users, especially children.

Scratch synthesizes programs by composing tangible building blocks.

3.2.2 General-Purpose Tiles

Projects as reacTable and Scratch are helpful to compose complex structures. However, as most first-generation tangible systems, they are special-purpose applications. A reacTable control is designed for use in musical applications, and due to its specific shape it could hardly be applied to a different context. Rekimoto et al. state in their paper about *DataTiles* that the “scalability challenge is one reason [special purpose] systems have not yet seriously competed with mainstream graphical user interfaces” [Rekimoto et al., 2001, p. 269]. Or with words of Shaer and Hornecker [2010]:

Lack of scalability is a major issue of special-purpose tangibles.

"One of the biggest challenges for TUIs is scalability. Successful applications for small problems or data sets often do not scale up to complex problems involving many parameters and large data sets. With any application that attempts to provide a tangible representation this will be an issue, as larger representations take up more space."

[Shaer and Hornecker, 2010, p. 106]

Productivity tasks require the input of abstract values. Special-purpose tangibles do not have the expressiveness to achieve this and require designers to extend the set of physical objects for every further application feature. Tangible tiles, as presented in the following, provide a better scalability.

DataTiles is a set of modular tangible tiles that combine haptic feedback with dynamic relabeling for general-purpose controls.

With DataTiles, Rekimoto et al. [2001] intended to design a system that combines the benefits of graphical and physical user interfaces. The authors introduce DataTiles, a set of semi-transparent tangible tiles for use on tabletops. Users can choose among a variety of graspable tiles, place them into a grid on the table, and bring them into relation to each other. Each tile is transparent and can display digital content beneath it. Users interact with these tiles using a digital pen. The system is designed for general-purpose applications and provides a variety of different tiles. An *Application tile* is a tangible that provides a specific application, such as a weather forecast or a video player. When put on the table, such a tile immediately starts and shows content. *Portal tiles* link to external entities and communicate their status, such as a printer, web-cam, or even a person. *Parameter tiles* allow to modify values of other tiles. These tiles have grooves to guide the motion of the user's pen. For example, a parameter tile with a circular groove can be used to continuously navigate through a video. Furthermore, the system provides *Container tiles* to store data and *Remote tiles* that provide a remote view to other tiles. The strength of this system is the ability to compose different tiles in an ad hoc manner. For example, if users want to control a video, they can place a parameter tile onto the table for navigation and remove it if not needed anymore. This enables dynamic general-purpose applications without the need for a keyboard or mouse.

Tangible Tiles are transparent tools to manipulate virtual images.

Tangible Tiles by Waldner et al. [2006] later extended the DataTiles concept. Virtual objects are displayed on the tabletop, and tiles can be used to modify them (Fig. 3.2). For example, a pick tile can be placed on top of a digital image to virtually "pick it up". When the physical tile is then put down at a different location, the digital image also moves there. The authors also introduce so-called *Functions tiles*, e.g., to magnify or erase objects. Unlike DataTiles, Tangible Tiles allow for an arbitrary alignment on the tabletop.

3.2.3 Physical Controls for General Purposes

However, although both projects provide graspable tangibles for general-purpose applications, their haptic feedback is still very limited. *VoodooIO* by Villar and Gellersen [2007] provides a toolkit that allows designers to

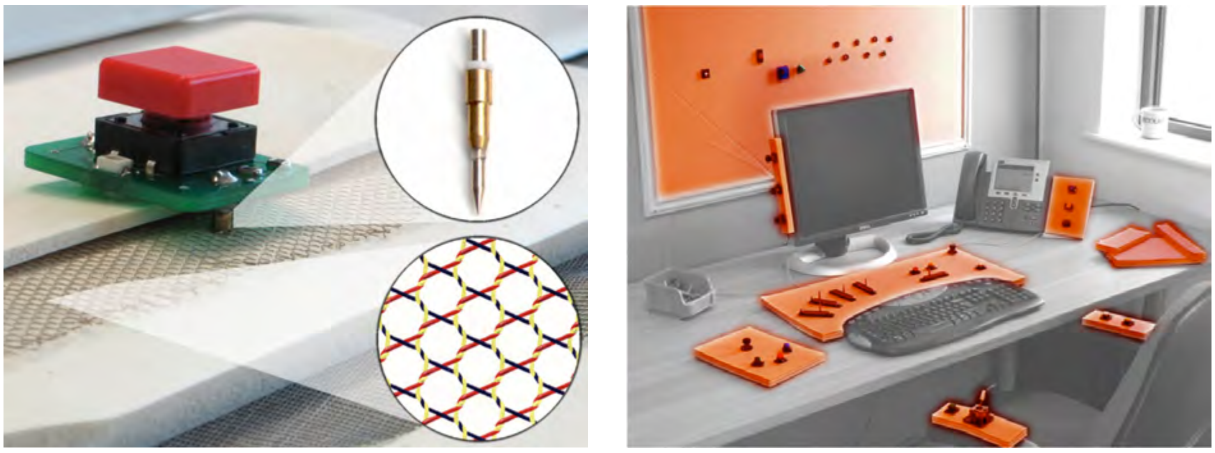


Figure 3.3: VoodooIO. Left: Technical principle. A control stuck into the surface closes a circuit and communicates its current state. Right: VoodooIO surfaces can extend a workspace in various ways. Image courtesy of Villar and Gellersen [2007].

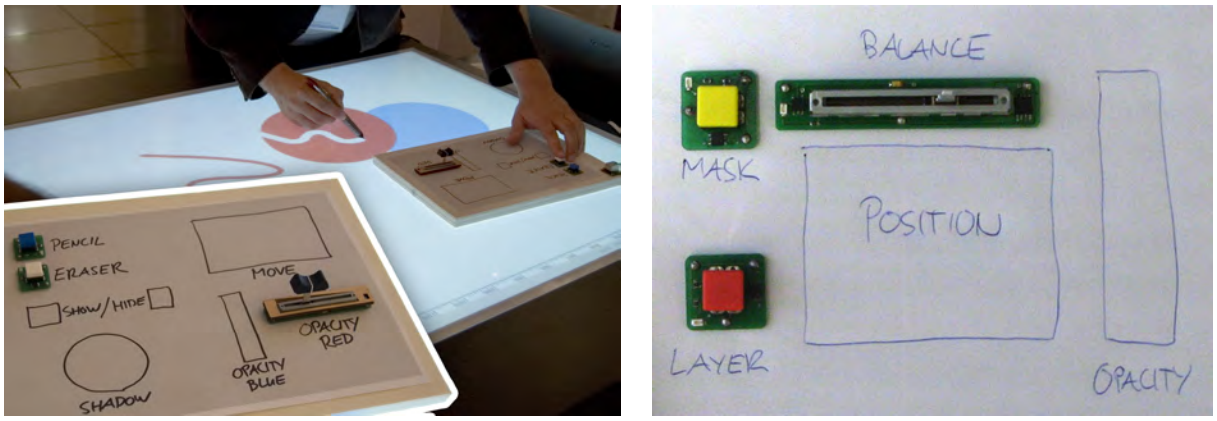


Figure 3.4: VoodooSketch provides flexible interactive palettes. Users can plug physical controls into these palettes or draw controls using a digital pen. Left: VoodooSketch in the context of an interactive tabletop application. Right: Sample palette combining physical and virtual controls. Image courtesy of Block et al. [2008].

freely place conventional physical controls on a "malleable control structure". The system is based on a substrate that contains two parallel conductive layers that are embedded between layers of laminate. Each VoodooIO control is mounted on a board with attached spikes. When sticking the control into the surface the spikes not only ensure physical support, they also close a circuit with the two conduction layers (Fig. 3.3, left). Via this circuit, the control is powered and communicates its current state to a network adapter. Users can choose from a set of standard controls (buttons, sliders, knobs), place them at any position on the surface, and immediately start using them. When not required anymore, controls can simply be removed from the surface; the use of laminate lets the remaining holes "heal" quickly. Furthermore, the substrate is bendable, robust, and can be cut into arbitrary shapes. Multiple surfaces can be combined and enrich arbitrary surfaces with physical controls (Fig. 3.3, right).

VoodooIO provides fully reconfigurable surfaces into which users can stick physical general-purpose controls.

VoodooSketch extends VoodooIO with pen input for operating hand-drawn controls.

Block et al. brought this concept to interactive tabletops and extended it with digital pen interaction. *VoodooSketch* is based on VoodooIO with an additional layer containing a tiny imprinted dot pattern on top [Block et al., 2008]. In addition to placing physical controls on the substrate, users can draw controls on this layer using an Anoto pen. A small camera in this digital pen tracks its current position on the paper by interpreting the unique dot pattern beneath this pen's tip. A recognition engine classifies the user's drawings and then instantly enables them as controls. The system enables a very intuitive interaction: For example, users can stick physical buttons into the substrate and write labels next to them, such as "Save file" or "Load file". The label's text is detected using text recognition and is automatically associated with a function in the current application. Alternatively, users can draw a rectangle and label it. It is then triggered by tapping the pen within the rectangle. Also, continuous controls can be sketched. A rectangle labeled with "opacity" yields a slider that is set by moving the pen tip within that rectangle (Fig. 3.4). In the same way as VoodooIO, the substrate can be cut. Furthermore, the layer with the dot pattern can be detached and reused, which can be considered as a physical save and load function. A disadvantage of the sketching technique is that the controls' shapes are filled with ink dots and strokes when they have been used for a while. VoodooSketch was designed as an augmenting palette to support tasks on interactive tabletops. However, the need for tethering and the static visual appearance of the (sketched) controls limits their flexibility in the context of interactive surfaces.

Substrate can be cut. Hand-drawn interface can be re-used. However, static visual appearance limits flexibility for interactive tabletops.

SenseSurface[1] is a concept that intends to bring the idea of VoodooIO to computer screens. Small magnetic physical controls can be stuck directly onto the screen to augment applications with haptic input. Position and status of these controls is communicated via a matrix that is attached behind the screen. SenseSurface knobs could, for example, be placed on top of the virtual knobs of a GUI based equalizer. A DJ could then control them precisely without looking while concentrating on his performance. However, the project has not been published in research or released as a commercial product.

Devices with fixed set of physical controls increase precision but are not suitable for general-purpose applications

Fiebrink et al. [2009] provide a tangible tabletop device to address the lack of precision and haptic feedback on multi-touch tables. The device contains four physical buttons and sliders that can be linked to actions and parameters in a musical tabletop application. When placed on the table, the device is detected, and a graphical UI is displayed around it. Besides the physical controls, the system provides virtual copies next to the device. Thus, users can employ both physical controls for eyes-free and precise operations, and on-screen controls. Using a set of touch gestures, every control can be dynamically mapped to a continuous value or discrete action in the tabletop application. The authors also provide methods to load, save, and share mappings. Although the system improves precision for specific tabletop tasks, its usefulness for general productivity tasks is limited: The device contains a fixed, unchangeable set of controls, needs a cable, and consumes a relatively large area on the tabletop. This makes

[1]http://www.girtonlabs.com/index.php/projects/sensesurface

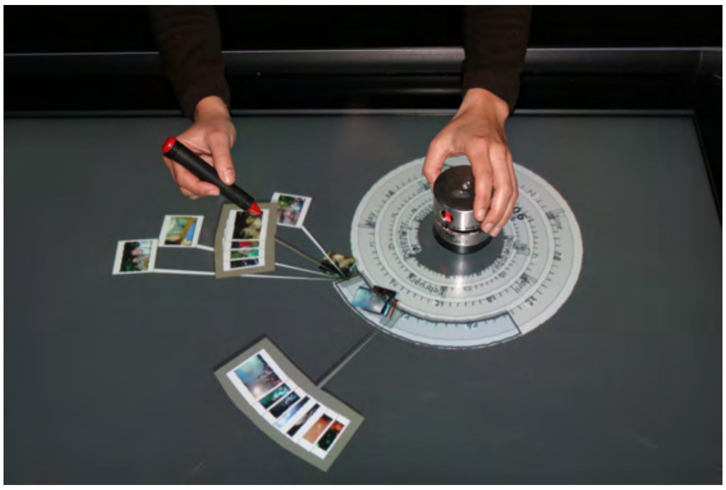


Figure 3.5: PhotoHelix allows to browse, sort, and share large collections of digital images using a rotatable tangible and a pen. Image courtesy of Hilliges et al. [2007].

it less flexible for use in general productivity applications where the UI is dynamically changed.

PhotoHelix is a physical knob in combination with a helix visualization for browsing through large photo collections.

PhotoHelix is a physical knob that allows for browsing through a large library of photos on an interactive tabletop [Hilliges et al., 2007]. When placed on the table, a helix of photos, ordered by time, is displayed around the control. A curved lens window at the outer part of the helix provides a detailed view of the underlying photos (Fig. 3.5). By turning the knob, the helix rotates, and the lens moves inward or outward. Using pen or finger as input device, images can be scaled, rearranged, or shared with other users. While the system was created for the special purpose of photo organization, it could be generalized to other large data collections. Furthermore, it is a good example for the synthesis between a rich haptic experience and interaction with digital data.

3.2.4 Typing

Typing efficiency is essential to perform productivity tasks on interactive tabletops.

Typing is essential for most productivity tasks. All GUI programs that deal with documents, e.g., word processors, spreadsheet programs, or e-mail clients, require users to enter text frequently. Design applications, such as drawing, 3D modeling, or video editing programs, often need the specification of exact numbers or labels. Web browsers at least require to enter URLs. The quality of *typing* on an interactive surface directly influences how efficiently users can enter data in tabletop applications and, thereby, how well productivity applications can be transferred to tabletops. If users have to "paint" letters with their bare fingers using direct touch, it's unlikely that a group of collaborators will create satisfying results in a reasonable amount of time. Ultimately, the convenience to type text determines whether productivity tasks will ever be transferred from the conventional PC to interactive surfaces.

Tabletops applications should provide simultaneous text entry for different objects.

Typing on interactive tabletops is still considered as an open problem. Interactive tabletops provide a highly dynamic user interface. Applications can be aligned anywhere on the surface, and multiple users can approach the table from any direction. Multiple typing devices are required that allow users to enter strings at different locations at the same time. Accordingly, application designers have to provide pairing techniques to associate each device with the particular text that a user wants to edit.

On-screen keyboards are easy to implement but slower and more error prone than conventional ones.

The standard solution for text input on touch screens are on-screen keyboards. Most smart phones, tablets, but also touch-based ticket machines overlay the interface with a QWERTY keyboard that can be operated by tapping keys with fingers. Those virtual keyboards are easy to implement, do not require any external devices, and can be dynamically relabeled. However, as mentioned before, typing text on a flat surfaces is slow and error prone. Users do not receive haptic feedback when pressing a key and cannot feel the boundaries of the keys. This requires a frequent switch of visual attention between the soft keyboard and the position where text is entered. Barrett and Krueger [1994] compared touch performance of casual users and expert typists on a conventional and a piezo-electric soft keyboard. In both subject groups, typing was significantly faster and less error prone on the conventional than on the soft version. The authors also report a significant learning effect over five sessions, but it was similar among subjects and keyboard condition. Findlater et al. [2011] conducted an in depth study in which expert typists had to enter text on a flat surface using an on-screen QWERTY keyboard. They measured the effect of keyboard visibility and visual feedback on typing performance and captured how well users hit the bounding boxes of keys. Even under an (simulated) ideal typing condition, in which every keystroke was presumably correct, typing on the flat surface was 31% slower than on a conventional keyboard. The authors recommend modifications to the default keyboard layout, such as changing the size of certain keys, extending the space bar area, and curving the keyboard shape. More importantly, the introduction of an individual classification model *per user* significantly improves typing accuracy. However, applying a typing model per user is difficult at an interactive tabletop: It requires a dedicated calibration procedure for every new user and an accurate assignment of touches to users. Furthermore, the reported typing speed was still very low; about 28 words per minute in the presence of visual typing feedback.

Modified keyboard layout and per-user model improve efficiency, but per-user models are difficult to realize on tabletops.

Difference between visual and physical typing is smaller when interacting with one hand.

Note that the difference between physical and virtual key entry is smaller when typing on small mobile phones [Lee and Zhai, 2009]. The authors also provide evidence that audio or haptic feedback can improve input performance on these devices. However, the study results are difficult to transfer to full on-screen keyboards on large surfaces. First, when typing with one hand, the performance gain of the other hand targeting the next keys simultaneously is lost. Second, on mobile devices, the location of touch input is close to the output location. Thus, the visual focus switch between typing a string and verifying the input might be insignificant. On interactive surfaces, input and output location can be distant.

Typing devices should be efficient, easy to learn, small, collapsible, rotatable, mobile, and similar to other input.

Hinrichs et al. [2007] estimate the usability of text-entry methods, which are normally employed in PCs or mobile devices, in the context of large interactive surfaces. Besides general criteria, like typing speed and ease of learning, they evaluate text input techniques according to tabletop specific requirement. A technique must not consume too much space and must be easy to remove from the table (collapsibility). As interactive tables can be operated from any side, an input device must be rotatable. Also mobility and the integration in the direct-touch interface play an important role.

Physical keyboards are efficient, but usage on tabletops is limited.

Physical keyboards are highly optimized and established input devices. Thanks to their labeling, they are approachable by novel users, and expert users can achieve high typing speeds. However, they consume much space on the table, and “switching back and forth between touch-typing on an external keyboard and direct-touch interaction within the virtual workspace can be disruptive” [Hinrichs et al., 2007, p. 107]. Also, to ensure collapsibility, storage space in or around the table is required. Using physical keyboards on additional small mobile devices could solve this issue. Furthermore, they can even provide privacy, e.g., to enter passwords, if required. However, entering longer texts on small physical keyboards is inconvenient. Speech recognition is an interesting alternative. Besides microphones, it does not need any external device. However, their use on tabletops is limited because they need calibration, and they are relatively slow and error-prone, especially when multiple users interact at the same time.

Speech recognition is error-prone, especially in a multi-user context.

On-screen methods are suitable for shorter, slow input.

The authors also evaluate on-screen methods. Soft, stylus, or gesture based keyboards can be easily created, removed, and relabeled. However, those methods tend to be slow and, as aforementioned, soft keyboards are more error prone than conventional ones. They are, therefore, only suitable for shorter amounts of text. All methods that use external devices or on-screen keyboards require an association with the position on the table, where the text is required. Hence, handwriting recognition is a promising alternative, because text can be written directly at the desired position. However, handwriting is a relatively slow technique and requires a high resolution input sensor.

No perfect input technique exists for tabletops. Choice of technique depends on task.

Hinrichs et al. conclude that there is no perfect technique for text input on tabletops yet, and that the choice of the method depends on the specific task. For an application that requires a lot of text input, a wireless keyboard as implemented by Hartmann et al. [2009] might be the right choice. For annotations on documents or photos, direct pen interaction and handwriting recognition might be suitable. On-screen keyboards are probably more appropriate for public exhibits where external devices tend to chafe quickly.

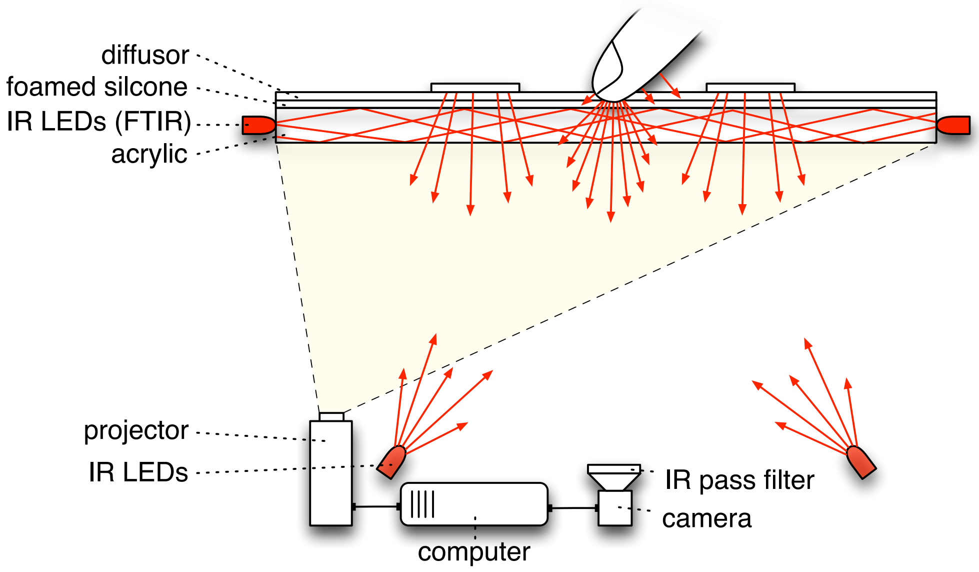


Figure 3.6: The SLAP Table is an interactive tabletop that combines FTIR and DI tracking. Modified image from [Weiss et al., 2010a].

3.3 System Design

The applicability of the aforementioned tangibles for productivity tasks on interactive tabletops is limited. They are either too specific, not providing input for abstract parameters and not scaling well, or they only offer restricted haptic feedback. Other projects contribute general-purpose controls, but these are difficult to apply to applications with quickly changing UIs.

In this section, we introduce haptic general-purpose controls for interactive tabletops. Based on our design considerations and the related work, we developed a novel class of physical controls that provide rich haptic feedback while regarding the dynamic nature of interactive tabletops. After describing our tabletop infrastructure, we will explain the design of our controls in detail.

3.3.1 Tabletop Infrastructure

We employ a vision-based interactive tabletop as infrastructure for our controls. We will refer to this setup, which is illustrated in Fig. 3.6, as the *SLAP Table*. A clear 88.5 × 112.5 cm acrylic plate forms the table surface and the support for further input and output layers.

SLAP Table combines FTIR and DI tracking.

Input In order to detect finger touches as well as lightweight objects on the tabletop, we combine FTIR and DI (section 2.1). A 0.63 mm layer of foamed silicone serves as compliant layer for touch detection. We added two LED light sources: For touch detection via FTIR, the acrylic layer is surrounded by a 355 cm ribbon of 860 nm infrared LEDs. The ribbon contains 277 LEDs with a space of 1.28 cm between each LED. To detect lightweight objects on the tabletop, 10 flood lights (4 containing 15 LEDs and 6 containing 7 LEDs) in the table produce a diffuse illumination of the surface. A Point Grey DragonFly camera with a 4-8 mm zoom lens ($f/1.4$) in the table captures all touches and objects on the surface. It delivers 8-bit gray scale images in 640 × 480 pixels and 120 fps.

Output An Optoma EX525ST short-throw projector beneath the table renders the user interface in 1024 × 768 pixels on a 0.127 mm thick Dura-Lar diffusor foil. The projected size of the GUI amounts to 92 cm × 68 cm, that is, a single pixel covers about 0.09 cm × 0.09 cm (28.42 dpi). To avoid interferences between output and input, we attached a B+W 093 IR pass filter to the table's camera. It ensures that all visible light from the projector is blocked. As some projectors also emit IR light, it can also be helpful to mount an IR block filter in front of the projector's lens.

The table design also contains a raised non-interactive edge that surrounds the table surface. This 8.5-13 cm wide strip provides space to put down tangibles or everyday objects, such as pens or coffee mugs.

Our software runs on a MacBook Pro with a 2.66 GHz Intel Core 2 Duo processor and 4 GB RAM.

3.3.2 Widget Design

We envisioned our controls as lightweight ad hoc input tools named SLAP Widgets. Users should be able to literally "slap" a control on the tabletop when needed, operate it for a while, and remove it as soon as it is not needed anymore. In allusion to desktop GUIs, we consider these tangibles as "widgets", because they provide general-purpose building blocks to setup a tabletop UI, as opposed to special-purpose tangibles.

SLAP Widgets were envisioned as lightweight, general-purpose controls for ad hoc use.

SLAP Widgets are general-purpose controls made of translucent acrylic and silicone (Fig. 3.7). Acrylic provides a rigid structure, while the soft silicone enables designers to incorporate deformable surfaces. Since all widgets are translucent, we can use the table's back projection to change their visual appearance on the fly. This allows us, e.g., to change the icon of a button or the range of a slider on the fly. Thus, although the *feel* of each physical control is somewhat fixed, we can change its *look* on the fly. This opens many interaction possibilities, as will be described below.

SLAP Widgets are general-purpose controls made of translucent acrylic and silicone. They combine haptic feedback with dynamic relabeling.

Every widget is mounted on a set of reflective rectangular markers. Each marker consists of bright white printing paper. These markers form a unique *footprint* that is detected by the camera when the control is placed on the table. The unique footprint is used to identify the widget and to track its current position and orientation on the table. The latter is especially important to register the back projected graphics with the physical

Position, orientation, and state of a widget are communicated via arrangement of reflective markers.

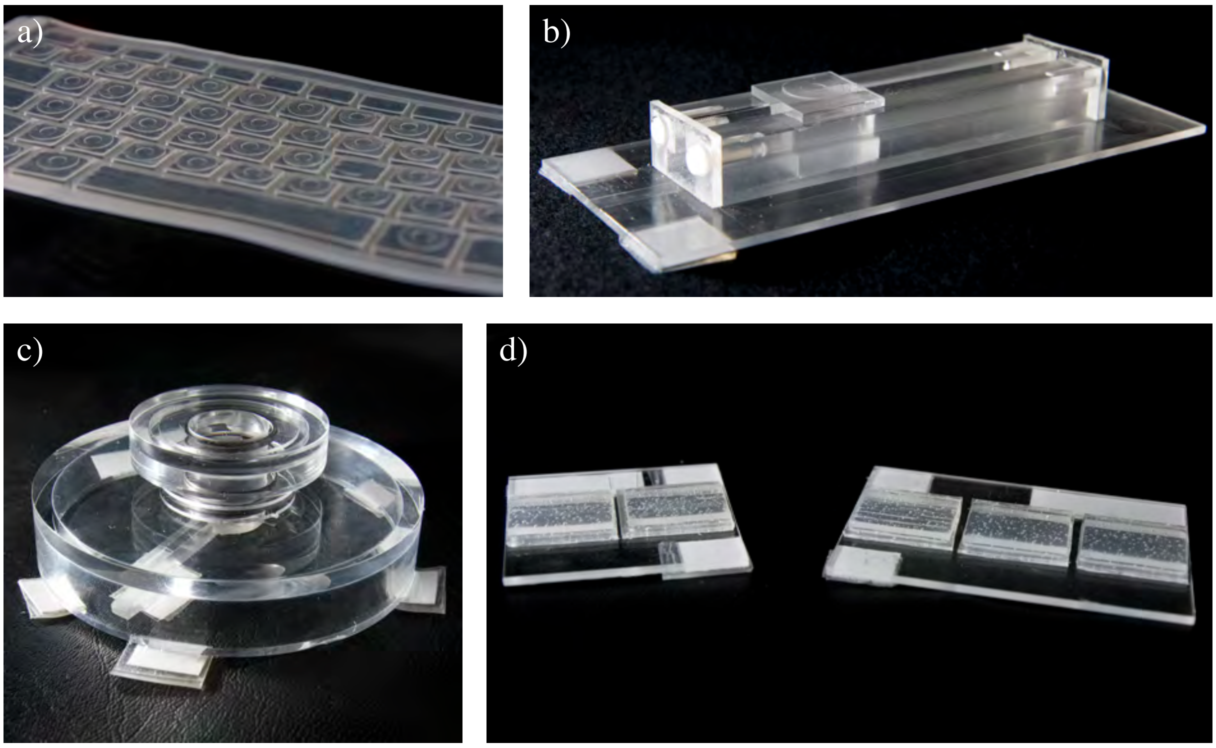


Figure 3.7: SLAP Widgets are general-purpose tangibles made of acrylic and silicone. Attached paper markers enable tracking. The basic widget set contains (a) keyboard, (b) slider, (c) knob, and (d) keypads with two and three buttons, respectively. Image adopted from [Weiss et al., 2010a].

objects. As soon as a widget is recognized by the camera, a local coordinate system is computed that acts a reference system for the widget's graphic representation. The footprint also contains a unique ID to distinguish multiple widgets of the same type, and *state markers* which represent the current state of the control, e.g., the position of the knob in a slider or the push state of a button. Fig. 3.8 illustrates the marker design of a SLAP Knob. Footprints are designed in such a way that they minimize graphical occlusion and create unambiguous patterns. Furthermore, every footprint contains a single long marker that avoids finger touches to be interpreted as widget markers. We also added translucent rubber pads beneath most widgets to avoid them from drifting away during operation.

SLAP Widgets are passive controls without electronic parts.

Note that SLAP Widgets are passive controls that are untethered and do not contain any electronics. Input and output are handled by the table hardware. If a user operates a widget, its physical state is represented by the 2D arrangement of markers. Our system reads and interprets this footprint using the internal camera and sends this information to the application controller (see section 3.6 for more details). The avoidance of electronic parts enables a low learning curve, because no knowledge about electrical engineering is required. Furthermore, as long as a widget's state is communicated via its footprint, there are no physical limitations in terms of widget design.

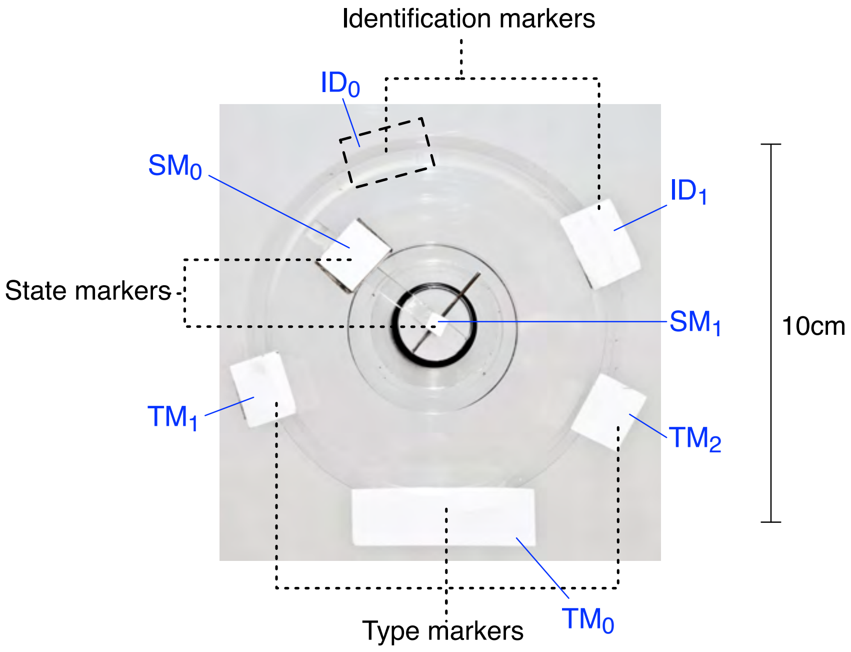


Figure 3.8: Footprint design of a SLAP Knob as seen from below. The arrangement of *type markers* (TM_i) determines the widget type. The presence or absence of *identification markers* (ID_j) encodes a unique ID among multiple widgets of the same type. The state of a widget is derived from its *state markers* (SM_k).

3.3.3 Basic Widget Set

As shown in Fig. 3.7, we developed a basic set of SLAP Widgets, including keyboards, button keypads, sliders, and knobs, to provide controls for a broad spectrum of productivity tasks. In this section, we describe the physical construction of each widget, its communication via the footprint, and potential ways to use it in a tabletop application.

Basic widget set contains keyboards, keypads, sliders, and knobs.

3.3.3.1 Keyboard

The SLAP Keyboard is a flexible physical keyboard for typing tasks on interactive tabletops (Fig. 3.9, left). It is based on an off-the-shelf iSkin[2] silicone keyboard protection cover (Fig. 3.7a). This cover is designed for QWERTY keyboards of MacBook Pro laptops. We glued thin 0.025 cm PVC plates onto each key to improve its stiffness. Additional plastic rings on top of these plates improve haptic feedback. As in physical keyboards, we also placed small bumps onto the F and the J key to provide an orientation for eyes-free typists. Two rigid acrylic bars are pasted to the short edges of the keyboard. They contain the footprint of the keyboard and stabilize the widget when being placed on the table.

The SLAP Keyboard is a flexible, silicone keyboard with flexible back projected key labeling.

[2]http://www.iskin.com

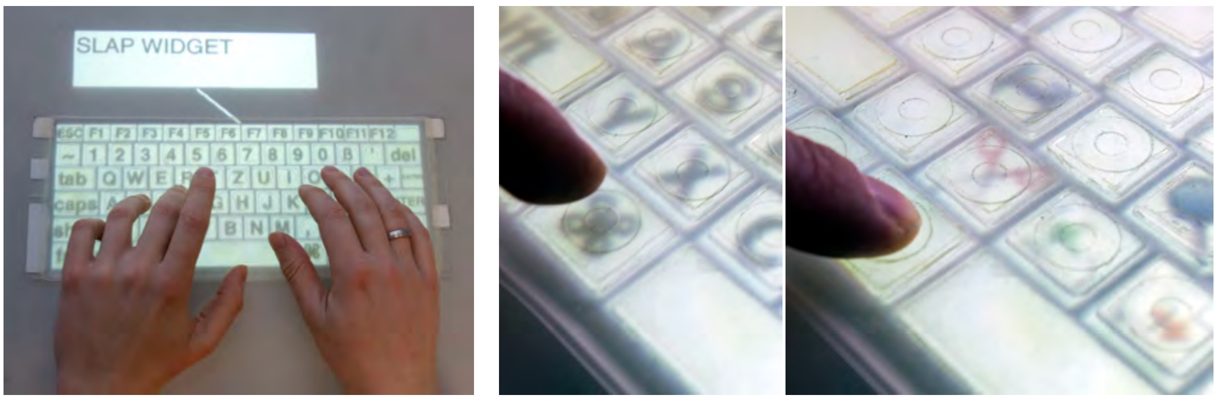


Figure 3.9: SLAP Keyboard in use. Left: A user types text into a text field. Right: Example relabeling. When holding the *Command* modifier key, certain keys show the icons of the functions associated with the key combination.

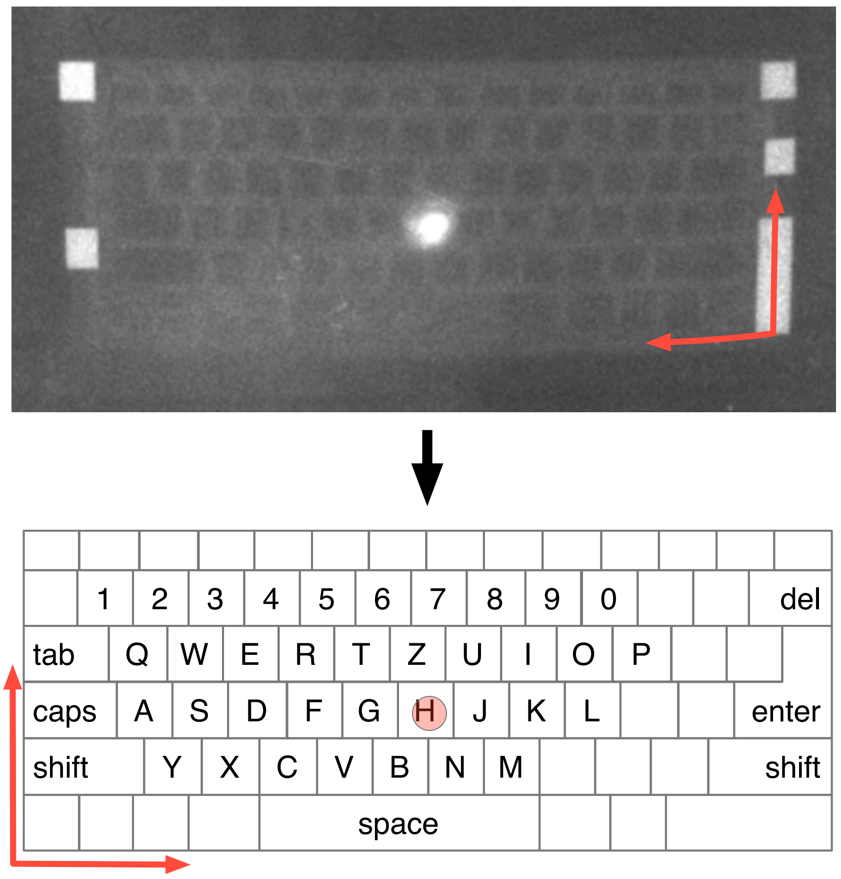


Figure 3.10: Keystroke detection of SLAP Keyboard. Top: Raw IR image with "H" key being pressed. The arrangement of markers on the acrylic bars determine the widget type, ID, and its local coordinate system (red arrows). Bottom: Keystrokes are detected by performing hit tests of spots within the keyboard area against the bounding boxes of keys in local coordinates. A keystroke event equals a spot (red circle) appearing and then disappearing in a bounding box.

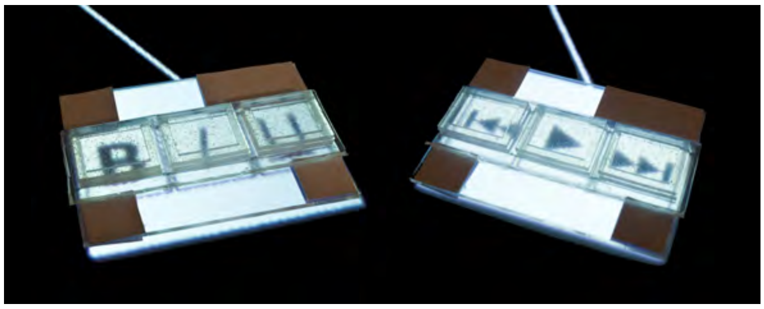


Figure 3.11: Example applications of SLAP Keypads for text editing and media control.

When placed on the table, the system detects the widget's footprint and projects a virtual keyboard beneath the widget. The SLAP Keyboard is so thin that the FTIR effect works through the material. That is, when a user presses a key, he pushes the key plate into the complaint layer, and a spot appears in the camera image. All spots within the rectangle of the keyboard which do not belong to the footprint are considered as potential key strokes. If a spot becomes visible within the rectangle of a key, we send a *key pressed* event to the application, using the key ID as parameter. Another event for the same key can only be sent if the spot disappears first. The keystroke detection is illustrated in Fig. 3.10. The IR image is mirror inverted as the camera is placed below the table. It is also radially distorted, which is compensated by our tracking algorithm. Note that reliable typing detection requires a high camera frame rate.

IR spots within the keyboard's area are interpreted as keystrokes.

The SLAP Keyboard fulfills many of the requirements stated by Hinrichs et al. [2007] in their survey paper on text-entry methods. First, it is *collapsable*. It can be easily slapped anywhere on the table without the risk of damaging electronics, and thanks to the use of silicone it can be rolled-up if not required anymore. Second, users can quickly rotate it or pass it to other users. Third, while providing haptic feedback, the SLAP Keyboard does not require significantly more space on the surface than its on-screen counterpart. It can be used for all tasks that require text input, such as writing or annotating documents. Leveraging the back projection, we can change the label of each single key on the fly. Thus, the keyboard layout can be easily remapped between different languages. Also, context-sensitive relabeling is possible. If a user hits the control or command key, keys could show the icons of the corresponding functions (Fig. 3.9, right), which makes them easier to learn for beginners.

The SLAP Keyboard is collapsable, rotatable, and does not require much space.

3.3.3.2 Keypads

SLAP Keypads are small groups of pushbuttons that are aligned in a row. We developed prototypes for two and three button keypads. They consist of a rigid acrylic mount that contains the reflective markers for registration. Buttons are made of cast soft silicone (Fig. 3.7d). Just as the SLAP

SLAP Keypads are groups of silicone pushbuttons.

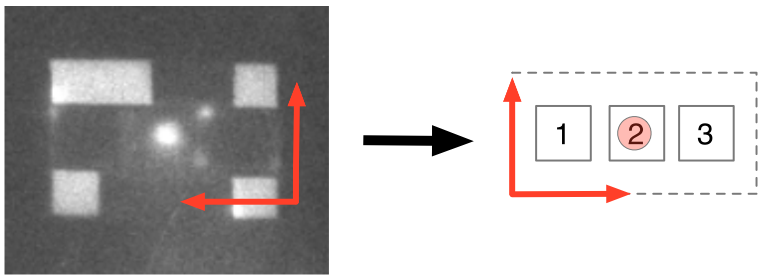


Figure 3.12: Button push detection of SLAP Keypad with three buttons. Left: Raw IR image with middle button being pressed. Right: Touch interpretation in local coordinates.

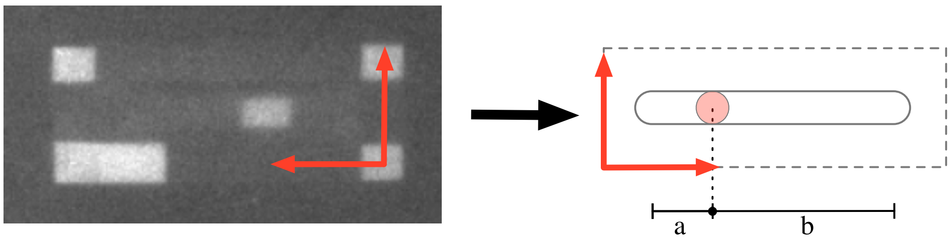


Figure 3.13: Position detection of SLAP Slider. Left: Raw IR image with identifying markers and center marker that determines slider position. Right: Interpretation of sliding knob's spot in local coordinates. The relative position of the slider equals $\frac{a}{a+b} \in [0, 1]$.

Keyboard, the keypads recognize a button push by interpreting spot events in the bounding boxes beneath the keys (Fig. 3.12).

Keypads can be used as tool palettes to trigger frequent actions.

Keypads can be used as small and quickly accessible tool palettes to trigger frequent actions for specific applications. For example, in a video application, a two button keypad could provide an interface to start or stop the video. Or in a word processor, a three button keyboard could contain functions to toggle bold, italic, or underlined text (Fig. 3.11). The ability to relabel widgets also allows users to quickly reconfigure the layout of each keypad. Moreover, multiple SLAP Keypads can be composed if more buttons are needed.

3.3.3.3 Slider

The SLAP Slider is an acrylic control for setting an absolute continuous value.

The SLAP Slider allows to set absolute continuous values. It consists of a rigid plate that contains the footprint and supports two acrylic rails. Users can linearly move a sliding knob within these rails to set a value (Fig. 3.7b). The sliding knob contains a reflective marker beneath its bottom that exposes a spot to the camera. Using a simple affine transform, the offset to the widget's local coordinate systems can be mapped to the position value of the slider (Fig. 3.13).

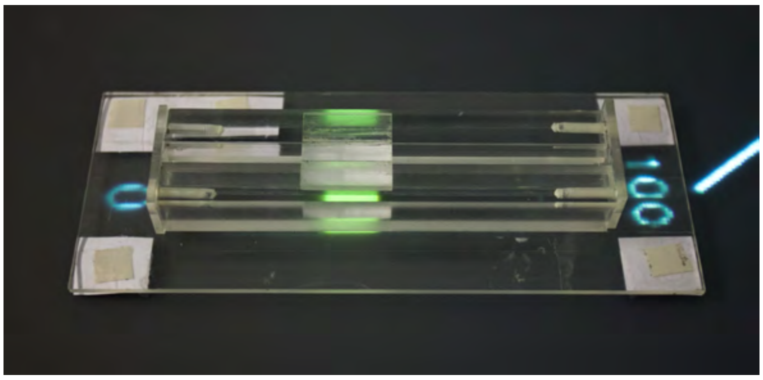


Figure 3.14: A SLAP Slider in use for setting a continuous value.

When a user places a SLAP Slider on the table, a virtual counterpart is projected onto the surface. Every time the user moves the sliding knob, and the position of the spot changes, the system sends the updated slider value to the application. However, to avoid jittering due to camera noise, we employ an offset threshold that must be exceeded before a "knob position changed" event is sent out.

IR spot position of sliding knob is mapped to slider value.

Range limits are displayed on the left and right side of the slider. These labels can be changed according to a particular application. For example, the slider can be used to navigate through a video; the table would then display "0:00" on the left and the total video length on the right side. If a user would use the slider to set a percentage parameter, such as the saturation of an image, the slider could display "0" and "100" at its sides (Fig. 3.14).

The input resolution of a continuous SLAP Widget depends on its size, the table's camera resolution and distortion, and on the distance between camera and surface. The knob in our slider prototype can be dragged over a distance of 8 cm. With the specific SLAP Table configuration, the tracking algorithm can distinguish about 45 different positions. A high resolution camera or the combination of multiple cameras would considerably increase this number.

Input resolution of continuous widget depends on its size and the table configuration.

3.3.3.4 Knob

A SLAP Knob is a pushable knob to set continuous relative values. It consists of a cylindric acrylic case with a spring-loaded turning knob on top (Fig. 3.7c). In the first generation, we employed a bearing to decrease friction. However, it turned out that the friction induced by the spring provides a convenient resistance during rotation; thus, a bearing was not used in later iterations. The footprint of the control is placed alongside the outer circle of acrylic base. The turning knob is mounted onto a rotational axis. This axis does not touch the surface but is connected to an arm inside the acrylic case. A reflective marker glued to the end of this arm lies on the

A SLAP Knob is a pushable knob made of acrylic to set a continuous relative value.

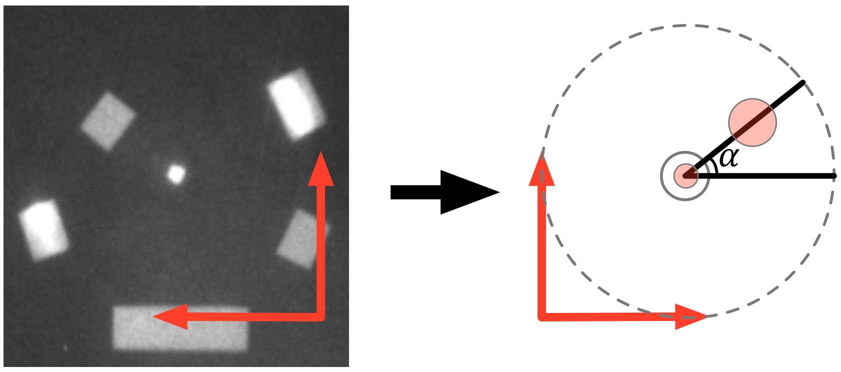


Figure 3.15: Angle and push state detection of SLAP Knob. Left: Raw IR image with knob being pressed and rotated by 38 degree counter-clockwise. Right: The absolute angle of the knob equals the counter-clockwise angle between the positive x-axis and the vector from the center to the spot of the arm's marker. If a spot appears in the middle circle, the knob is considered as pushed down.

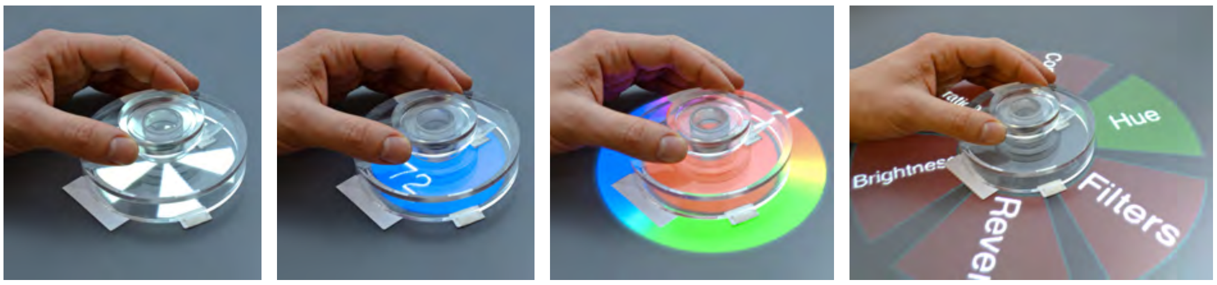


Figure 3.16: Modes of SLAP Knob. From left to right: jog wheel, value, hue, and menu mode.

Angle of arm's marker is mapped to relative angle.

surface and exposes a spot to the camera. The relative angle between the x-axis of the local coordinate system and the vector from this spot to the knob's center determines the angle of the control (Fig. 3.15). This angle is reported to the application on every turn. However, similar to the SLAP Slider, we employed a threshold to compensate for jittering.

Visibility of center marker is mapped to push state.

We can also detect whether a control is pushed. A reflective marker is mounted to the bottom of the rotation axis and slightly hovers over the surface. When the user pushes the turning knob, the reflective marker hits the surface, and the camera sees a spot in the center of the knob; like in the keyboard case, the system sends a *key pressed* event. Note that the spot is detected due to DI tracking. An alternative design is the use of a plastic pad beneath the rotational axis that produces an FTIR spot when pushed down. This, however, proved to be less reliable than a reflective marker.

The SLAP Knob supports four several modes for different tasks.

The SLAP Knob can be used in a variety of contexts. In our prototype system, we developed four different modes of the SLAP Knob (Fig. 3.16):

- *Jog wheel mode:* The knob can be used as a jog wheel to browse frame-wise through a video. In this case, we project a circular stripe pattern beneath the knob that gives visual feedback during navigation by following the rotation of the handle.

- *Value mode:* Like the slider, the knob can be used to set a continuous value within a specified range. The table displays this range as a circular scale around the control and highlights the current value. Note though that the knob does not provide physical constraints as the slider; it can be turned beyond the value ranges, and a software constraint is required to maintain valid values.

- *Hue mode:* In this mode, the knob shows a color wheel for all hue values in the hue-saturation-value (HSV) model. The currently selected color is highlighted by a white line. For example, users can use this mode to change the tone of an image.

- *Menu mode:* The knob can also be used to navigate through hierarchical menus. In this mode, the table displays a pie-menu beneath the knob, while the currently selected menu item is highlighted. By turning the knob, other items can be selected. When the knob is pushed, the item is activated. This can either trigger an action, load a sub-menu, or bring the knob into a different mode. For example, activating a "Change hue" item could switch the knob into the hue mode. When pushing again, the value is applied, and the menu is shown again.

3.4 Interaction Design

SLAP Widget retrieves specific function through pairing with a virtual object.

SLAP Widgets are general-purpose controls that gain their specific function first when they are associated, or *paired*, with a virtual object, such as an image, video, or text box. If a SLAP Widget is placed on the table for the first time, a blue halo around the object indicates that it has been recognized by the system, and that it is ready to be paired with an object (Fig. 3.17a). As the widget has not differentiated into a specific control yet, no graphics are displayed beneath the control.

Pairing gestures trigger an association between a SLAP Widget and a virtual object. Halos indicate detection and pairing state.

If a user performs a pairing gesture between the physical widget and an on-screen object, the system tries to associate these two. If the pairing was successful, a green halo flashes around both objects, indicating the established connection. As long as the objects are paired, they are connected with a white line (Fig. 3.17b). Furthermore, the graphical representation of the physical control is updated according to its specific task. For example, if a SLAP Knob is paired with a video, it is switched to jog wheel mode. When pairing it with an image, the menu mode offers several options to edit the picture, such as brightness, saturation, or hue. If a pairing fails, two red halos flash around the objects (Fig. 3.17d). This happens when two objects are incompatible or when a pairing is not defined, e.g., a SLAP Slider cannot be paired with a SLAP Knob. Users can release a pairing by repeating the pairing gesture on two linked objects. Two cyan halo flashes confirm the successful disconnection of the objects (Fig. 3.17c). The connecting line and task-specific graphical representations disappear.

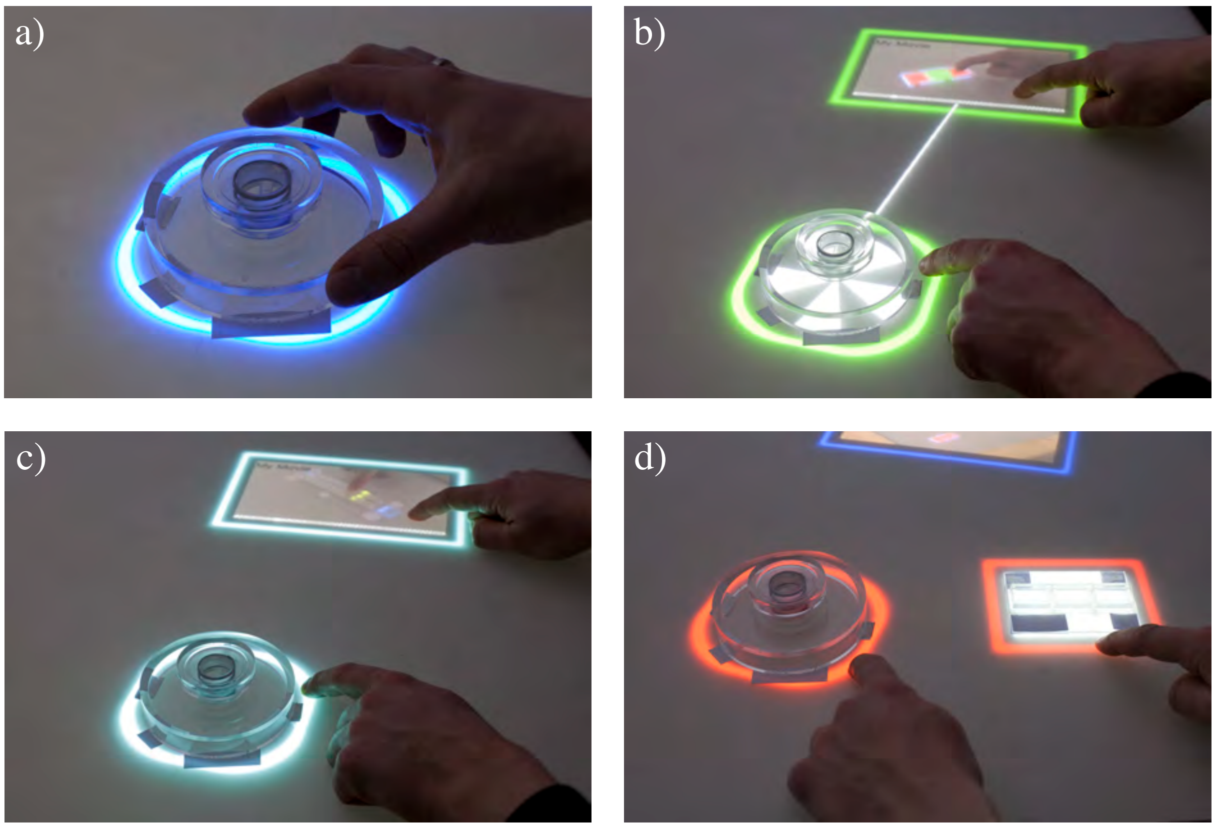


Figure 3.17: Detection feedback and pairing mechanism of SLAP Widgets. a) A user places a SLAP Knob on the surface. A pulsating blue halo around the control indicates that it is detected by the table. In this moment, no specific function is assigned to the knob. b) The user conducts a pairing gesture between the knob and a virtual movie object. For a short moment, a green halo flashes around both objects, confirming the successful pairing. The knob is now specialized to a jog wheel and updates its back projection. A line illustrates the connection between knob and movie. The knob is ready for frame-by-frame browsing. c) By repeating the gesture, both objects are "unpaired". The connecting line and the knob's back projection disappear. A cyan halo flash confirms this action. d) The user tries to pair a knob with a keypad. As this connection is not implemented, a red flash indicates the failed pairing.

Pairings between same object types or with multiple objects are also possible.

Besides simple 1 : 1 mappings, multiple widgets can be paired with a single object (n : 1), and a single widget can be mapped to multiple objects (1 : n). Also, pairings between virtual objects or between SLAP Widgets are imaginable. This could be helpful for prototyping. For example, a designer could create a virtual widget first and pair it with on-screen objects before crafting a physical object. Also, a SLAP Widget could be used to configure another one. For example, a two button keypad could be associated with a knob in order to switch between jog wheel mode for video navigation and menu mode for frame editing.

3.4.1 Pairing Gestures

Pairing gestures should be easy to conduct and reliable, without interferences with other tasks.

A suitable pairing gesture that interconnects physical widgets and on-screen objects must fulfill several requirements. Since pairing is a frequent action, it must be reliable and easy to conduct, but also unique enough that it does not interfer with other interactions on the table. Furthermore, collocated users could create multiple pairings simultaneously, which have to be disambiguated. We implemented two different pairing gestures:

- *Synchronous Double Tapping:* Users double-tap next to a widget and into the rectangle of the corresponding virtual object. Both taps must be synchronous. A tap event is defined as a stationary finger spot that lasts for a short time (< 0.2 s). Two taps are considered as synchronous if they occur within 0.1 s.
- *Synchronous Hold:* Users concurrently hold down two fingers next to a widget and within a virtual object. When both fingers are not released for one second, pairing is triggered. The two hold gestures are considered as synchronous if both touches begin within a time interval of 0.1 s.

Both methods establish pairings without interfering much with other interactions or pairings, because they require two synchronous gestures. However, during public demonstrations of the system, the Synchronous Hold gesture turned out to be easier to conduct by users.

There are other potential pairing gestures that we did not implement. For example:

- *Dragging handles:* All objects expose a set of pairing handles that can be pulled out by dragging. Two objects are paired by dragging a handle from one object to the other. The inverse gesture unpairs the objects again.
- *Pairing by proximity:* If a SLAP Widget is moved close to a virtual object, it is automatically paired. It remains paired until it is removed from the table.
- *Pie menu:* If a SLAP Widget is placed on the surface, a pie menu appears around the control, which shows all possible pairings with other objects. Each pie item is spatially aligned into the direction of the potential target object. Tapping a menu item establishes the corresponding pairing. Tapping the resulting pairing line disconnects the objects again.

Pairing and unpairing could be triggered by different gestures.

In our implementation, a connection between two objects was disbanded by just repeating the pairing gesture. Alternatively, a different gesture could be employed for unpairing, e.g., by simply striking out the pairing line (cf. Fig. 2.10 on page 24).

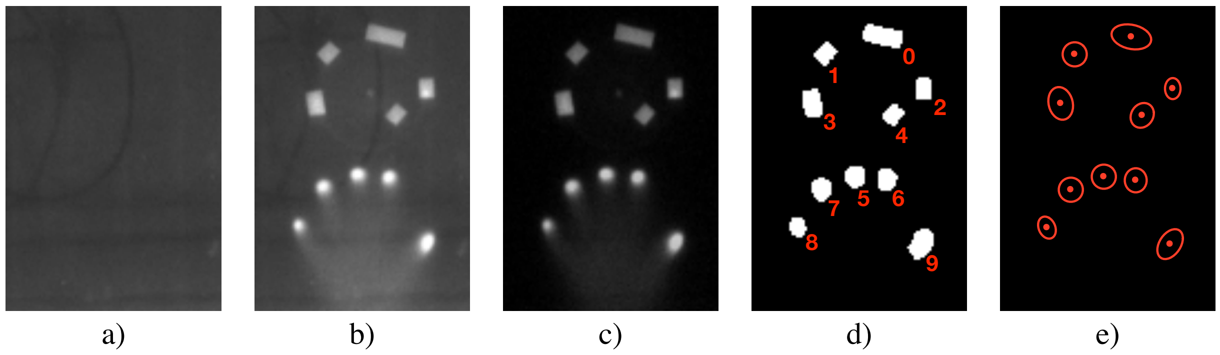


Figure 3.18: Main steps of image processing pipeline for detecting finger touches and markers. A user placed five fingers and a knob on the surface. a) Background image. b) Raw source image. c) Source image after background subtraction. d) Thresholding and connected components. e) Final spot events. Axes of ellipses denote principal components.

3.5 Input Sensing

Finger touches and markers expose bright spots to the table's camera. Video input is noisy and distorted.

The camera in the table captures an infrared image of the table's surface. Finger touches and reflective markers—appearing as bright spots—are interpreted as system input. The camera image is usually subject to noise, and ambient IR light, such as indirect sunlight or other light sources, lowers the contrast. Moveover, the camera's lens induces radial distortion. This section explains the tracking pipeline that extracts touch events from the noisy input image, and how widgets and finger touches are recognized using these events.

3.5.1 Tracking Pipeline

Background frame cancels out ambient light and reflections.

Our tracking pipeline is illustrated in Fig. 3.18. Before detecting objects on the table, we capture a *background frame* **Bg** (Fig. 3.18a). This is an image of the tabletop without any objects on the surface. It is used for canceling out ambient light and reflections inside the table.

Tracking pipeline:

Let $\mathbf{I}(\mathbf{Im}, \mathbf{x})$ be the intensity value of an image **Im** at pixel position $\mathbf{x} = (x, y)$. For each raw camera frame **F** (Fig. 3.18b), we detect spots by applying the following steps:

Subtract background

1. Subtract background to isolate input signal, yielding $\mathbf{F}'$ (Fig. 3.18c):

$$\mathbf{I}(\mathbf{F}', \mathbf{x}) \quad := \quad \mathbf{I}(\mathbf{F}, \mathbf{x}) - \mathbf{I}(\mathbf{Bg}, \mathbf{x}).$$

Create thresholded binary image

2. Create binary image $\mathbf{F}''$ by thresholding (Fig. 3.18d). All pixels brighter than a certain value are considered as foreground (spots), while all other pixels belong to the background:

$$\mathbf{I}(\mathbf{F}'', \mathbf{x}) \quad := \quad \begin{cases} 1 & \text{if } \mathbf{I}(\mathbf{F}', \mathbf{x}) \geq t_B, \\ 0 & \text{otherwise,} \end{cases}$$

where t_B is a threshold specific to the setup.

3. Create a list of connected components in $\mathbf{F}''$ (Fig. 3.18d). Two pixels belong to the same connected component if they are both foreground pixels and 4-connected. Two pixels at (x, y) and (x', y'), respectively, are 4-connected if and only if

Retrieve connected components

$$(x', y') \in \{(x-1, y), (x+1, y), (x, y-1), (x, y+1)\}.$$

4. Filter all connected components. If a component contains too few (noise) or too many (large objects on the table, like user's elbow) pixels, remove it from list. Each remaining component is considered as *spot*.

Filter connected components

5. For all spots, compute the two principal axes. Assuming that the spots are elliptical, this gives us the axes of these ellipses (Fig. 3.18e). We retrieve these by conducting a *Principal Component Analysis (PCA)* [Bishop, 2006, pp. 561–570] on all pixel positions in the respective connected component. We compute a minimum oriented bounding box along these axes that includes all pixels of the component. Finally, we determine the *axis length ratio*, the ratio between the length of the first and second principal axis. The oriented bounding box is not necessarily optimal, but the resulting axis length ratio helps to differentiate between spherical finger touches (≈ 1) and long rectangular widget markers ($\gg 1$). This is crucial for the widget detection (section 3.5.2).

Compute principle axes of all spots and minimum oriented bounding box

In order to detect fast movements on the surface, the implementation of this algorithm must be as efficient as possible. Using our CPU-based implementation, we can detect spots with the frame rate of the camera (120 fps). Yet, as the first steps of the algorithm are performed on a per-pixel level, a GPU implementation using OpenCL[3] or CUDA[4] could speed-up the detection and allow the incorporation of further filters that increase the robustness against noise. Note that if the ambient light changes drastically, a new background frame must be acquired.

GPU implementation could speed-up detection and increase robustness.

For each spot, we create a *spot event* that contains the following attributes:

- *Location*, equals the center of gravity (COG) of the corresponding connected component. The location is expressed in GUI coordinate space (see section 3.5.2).
- *Radius*, defined as half of the maximum diameter (x and y) of the axis-aligned bounding box around the connected component.
- *Main axis (x,y)*, the first principal axis resulting from the PCA. Note that the second principal axis is perpendicular and is scaled version of vector (-y, x).

[3]http://www.khronos.org/opencl/

[4]http://developer.nvidia.com/category/zone/cuda-zone

- *Axis length ratio*
- *Type*, if the axis length ratio is < 1.75, the spot is considered as "default", otherwise as "rectangle".

Spot events are tracked over time as touch events.

If a user drags a virtual object across the table, we must be able to follow the user's trajectory even if other users interact simultaneously (and create different spot events). Therefore, spots are converted to *touch events* that are tracked over time. In addition to spot attributes, touch events contain a timestamp, a phase type, and a unique ID that does not change until the object (finger or marker) is released from the table. This necessitates that we register spot events between two successive frames.

Predictive tracking improves matching of spots.

For a spot at position $\mathbf{x}_i$ in the last frame, we first compute velocity $\dot{\mathbf{x}}_i := \nabla(\mathbf{x}_i)$ and acceleration $\ddot{\mathbf{x}} := \nabla(\dot{\mathbf{x}}_i)$, where ∇ denotes the backward difference over two frames. We then use these values to extrapolate a position $\mathbf{x}_{i+1}$ for the current frame. If we find a spot event in this frame which is *close* to the predicted position, it is associated with the previous touch event. This is a very basic prediction. A predictive tracking filter using a Kalman filter [Welch and Bishop, 1995] could be incorporated in future iterations.

Touch events occur in a began, moved or ended phase.

In all, a touch event in the current frame $i + 1$ can exist in three different *phases*:

- touchesBegan: This is a new touch on the tabletop. No spot in frame i could be associated with this new spot.
- touchesMoved: The touch has been moved to a new position. There is a corresponding spot in frame i.
- touchesEnded: The finger or object has been released from the table. For a spot in the previous frame i, there is no corresponding spot in the current frame $i + 1$; the spot disappeared.

All non-footprint touches are considered as finger input.

Within the set of touches, SLAP Widgets footprints are detected and tracked. All touch events that are not associated with a widget are interpreted as finger touches that allow users to interact with virtual objects and to perform gestures.

3.5.1.1 Camera to GUI Mapping

Only subarea of distorted camera image is relevant.

The resolution that the camera captures the surface with (VGA, 640×480) differs from the resolution of the projected GUI (XGA, 1024×768). Furthermore, the camera lens induces radial distortion, and only a subarea captures the surface. Designers create applications with a GUI coordinate

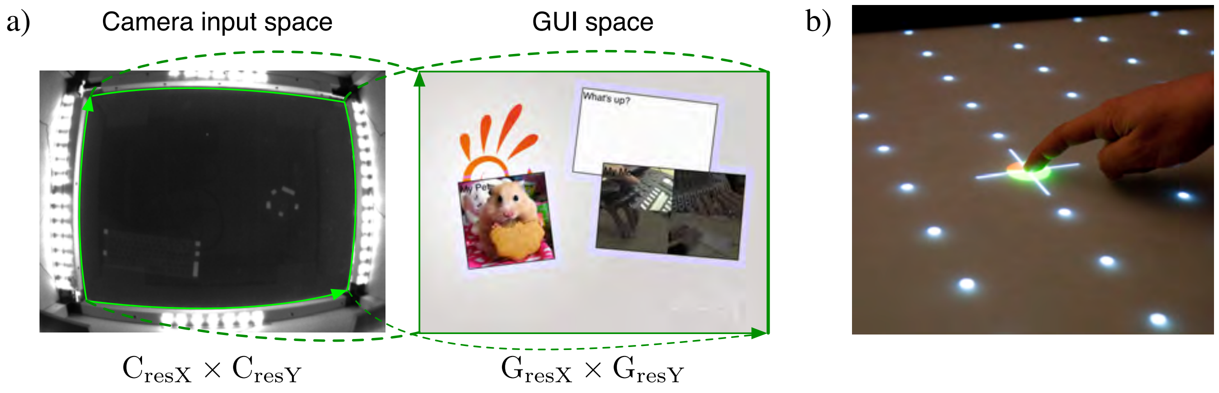

a)
Camera input space
GUI space
b)
What's up?
$\mathrm{C_{resX}} \times \mathrm{C_{resY}}$
$\mathrm{G_{resX}} \times \mathrm{G_{resY}}$

Figure 3.19: Mapping from camera to GUI space. a) The calibration process determines a mapping from the radially distorted region of interest in the camera image to the rectangular GUI. b) During calibration, the user subsequently touches points of a projected uniform grid.

system in mind, either in pixel or unified coordinates[5]. In order to map camera coordinates to GUI space and to compensate for radial distortion at the same time, we employ a homography (Fig. 3.19a). This requires the user to perform a one-time calibration process. Once calibrated, this process need not be repeated until the camera configuration changes.

We use homography to map from camera space to GUI space.

The calibration procedure is shown in Fig. 3.19b. The software projects a $M \times N$ uniform grid of dots onto the tabletop. In the beginning, the top left dot is highlighted, and the user is asked to hold down his finger onto this dot for 0.5 seconds. The software now stores the GUI position of the dot and the corresponding position of the finger spot in camera coordinates. This process is repeated for every dot in the grid. The result is a discrete map

Camera calibration: User touches projected points to define spline patch control points for GUI-to-camera mapping.

$$\begin{aligned}\Gamma_{\text{discrete}} : \{&\frac{0}{M-1}\cdot(\mathrm{G_{resX}}-1), \frac{1}{M-1}\cdot(\mathrm{G_{resX}}-1), ..., \frac{M-1}{M-1}\cdot(\mathrm{G_{resX}}-1)\}\\ \times\{&\frac{0}{N-1}\cdot(\mathrm{G_{resY}}-1), \frac{1}{N-1}\cdot(\mathrm{G_{resY}}-1), ..., \frac{N-1}{N-1}\cdot(\mathrm{G_{resY}}-1)\}\\ &\rightarrow [0, \mathrm{C_{resX}}-1]\times[0, \mathrm{C_{resY}}-1]\end{aligned}$$

that maps a uniform set of GUI coordinates to camera coordinates, where $\mathrm{C_{resX}} \times \mathrm{C_{resY}}$ is the camera resolution and $\mathrm{G_{resX}} \times \mathrm{G_{resY}}$ denotes the GUI resolution. To compute camera coordinates for GUI pixels *between* grid points, we use a bicubic spline patch that interpolates $\Gamma_{\text{discrete}}(i,j)\ \forall i \in \{0,1,...,M-1\}, j \in \{0,1,...,N-1\}$. The resulting *continuous* function that evaluates the spline patch is

$$\Gamma : [0, \mathrm{G_{resX}}-1]\times[0, \mathrm{G_{resY}}-1] \quad\rightarrow\quad [0, \mathrm{C_{resX}}-1]\times[0, \mathrm{C_{resY}}-1].$$

For every GUI pixel, we can now compute the corresponding position in camera space. However, for mapping camera spots to GUI coordinates, we need the *inverse* mapping

Inverse mapping, from camera to GUI, is pre-computed and stored in a look-up map.

$$\Gamma^{-1} : [0, \mathrm{C_{resX}}-1]\times[0, \mathrm{C_{resY}}-1] \quad\rightarrow\quad [0, \mathrm{G_{resX}}-1]\times[0, \mathrm{G_{resY}}-1].$$

[5]In unified coordinates, the 2D area of a GUI is addressed in a resolution-independent coordinate space $[0,1]\times[0,1]$. During rendering, these coordinates are mapped to actual pixel positions.

We create a discrete 2D look-up map that enables a fast mapping from camera to GUI space:

$$\Gamma^{-1}_{\text{discrete}} : \{0, 1, ..., \mathrm{C}_{\text{resX}} - 1\} \times \{0, 1, ..., \mathrm{C}_{\text{resY}} - 1\} \\ \rightarrow [0, \mathrm{G}_{\text{resX}} - 1] \times [0, \mathrm{G}_{\text{resY}} - 1].$$

This map is created as follows:

1. Initialize all pixels with $(-1, -1)$.
2. Uniformly super-sample the spline patch Γ. For a single sample $\Gamma(x, y) = (x', y')$, store the original GUI coordinate (x, y) in the discrete map at $(\lfloor x' \rfloor, \lfloor y' \rfloor)$.

The sampling density must be high enough so that no holes remain in the inverse map. Note that this map is neither surjective nor injective. The camera normally sees pixels outside the projection area; these pixels are marked as $(-1, -1)$ in the look-up table. Furthermore, multiple GUI coordinates can map to a single camera pixel; especially, if the GUI resolution is higher than the camera resolution.

Bilinear interpolation in the look-up map improves accuracy.

After calibration, we can quickly map spot locations from camera to GUI coordinates by a simple look-up in the $\Gamma^{-1}_{\text{discrete}}$ map. As touch locations are computed as center of gravity of connected components, they can be placed within pixels. Therefore, we employ bilinear interpolation when looking up a GUI coordinate.

3.5.1.2 Camera Parameters

Camera parameters must be chosen carefully.

The camera parameters must be set to appropriate values that match the specific table setup, ambient lighting, and application. For example, a long exposure yields brighter and clearer spots, but it also increases motion blur and decreases the maximum frame rate. Dim spots under short exposure can be brightened with a higher gain, but this significantly increases camera noise.

3.5.1.3 Receiving Touch Events

MultiTouch-Framework provides programming interface for multi-touch input.

Our touch detection is compiled into a framework, named *MultiTouch-Framework*. Programmers who intend to process touch events from the table hardware just have to include the framework in their project and implement the `MTTouching` protocol[6]. According to the three touch phases, the protocol contains three methods that the framework calls: When new touches occur, when touches are moved, or when they disappear from the

[6]A protocol defines an interface to communicate with an unrelated class. In Java, this concept is called *interface*.

```
@protocol MTTouching
@optional
-(void)touchesBegan:(NSSet*)touches withEvent:(MTEvent*)event;
-(void)touchesMoved:(NSSet*)touches withEvent:(MTEvent*)event;
-(void)touchesEnded:(NSSet*)touches withEvent:(MTEvent*)event;
@end
```

Listing 3.1: `MTTouching` protocol.

surface (Listing 3.1). A set of touch events and an event object, containing the timestamp, are assigned to each method call. Note that all methods in the protocol are optional; programmers can choose which methods they implement.

3.5.2 Widget Detection

Every footprint contains three kinds of markers (Fig. 3.8 on page 45):

Three different kinds of markers encode type, numeric ID, and state of each SLAP Widget.

- *Type markers* $\mathsf{TM}_0, ..., \mathsf{TM}_{\alpha-1}$ identify the kind of widget, e.g., whether it is a knob or slider. Per definition, TM_0 is always a "rectangle" marker (axis length ratio > 1.75). Furthermore, the distances between the first two markers is the largest of all type marker distances in the footprint, i.e.,

  $$\|\mathsf{TM}_0 - \mathsf{TM}_1\| \geq \|\mathsf{TM}_r - \mathsf{TM}_s\| \qquad \forall r, s \in \{0, ...\alpha - 1\}.$$

  This requirement makes sure that the local coordinate system of the footprint is as stable as possible (see below).
- *Identification markers* $\mathsf{ID}_0, ..., \mathsf{ID}_{\beta-1}$ encode a unique numeric ID.
- *State markers* $\mathsf{SM}_0, ..., \mathsf{SM}_{\gamma-1}$ represent the current state, e.g., to which degree a knob is turned or whether a button is pushed.

Type and ID markers are static and stored in file. State markers are dynamic and interpreted in software.

Type and ID markers are static and are specified when the physical widget is constructed. They are stored in a configuration file which is loaded at start-up of the tabletop application. State markers are dynamic; they can move and appear or disappear at run-time. Their interpretation is specified in software, i.e., each widget employs an own class to retrieve its state from its dynamic markers (see section 3.6.2). All markers contain the same properties as spot events (location, radius, main axis, axis length ratio, and type). They are stored in a *footprint coordinate system* that the graphical output refers to.

In the following, function $\mathbf{p}(\cdot)$ denotes the position of a marker or touch in its respective coordinate system. $\mathbf{p}(\cdot)_x$ and $\mathbf{p}(\cdot)_y$ represent the x and y component, respectively.

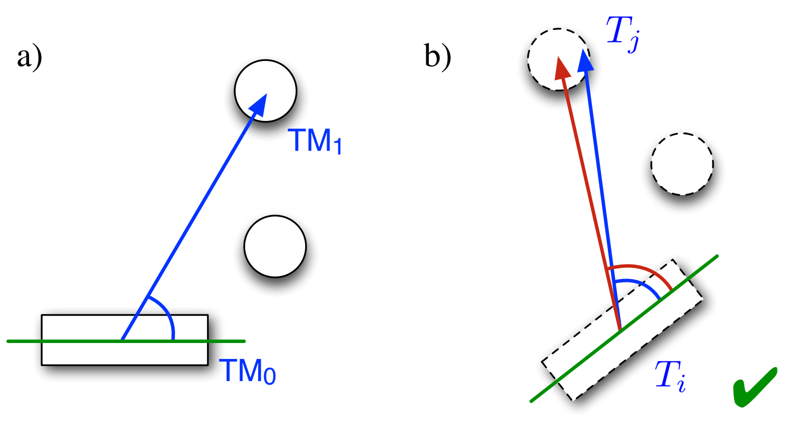


Figure 3.20: Principal axis angle deviation check. a) Sample footprint of a widget. The angle between the principal axis (green) of the first type marker TM_0 and the vector $\mathbf{p}(\mathsf{TM}_1) - \mathbf{p}(\mathsf{TM}_0)$ (blue) is tested. b-c) Sample touch clouds that are checked against footprint. Again, the vector between the assumed first two type markers $\mathbf{p}(T_j) - \mathbf{p}(T_i)$ (red) form an angle with the principle axis of T_i (green). If the difference between this angle and the footprint's angle is below a certain threshold, the test is passed (b). Otherwise, the touch cloud is not part of the widget and is rejected by the detection algorithm (c).

Detection The identification of widgets that have been placed on the tabletop equals the problem of finding n point clouds, the alignment of type markers of all n widget footprints, in the point cloud of current touch events $T_0, ..., T_\delta$. For every footprint, our algorithm proceeds as follows:

For each footprint, project all marker positions to local coordinate system.

1. The static type markers are projected into a local coordinate system, in which $\mathbf{p}(\mathsf{TM}_0)$ is the origin, $\mathbf{p}(\mathsf{TM}_1) - \mathbf{p}(\mathsf{TM}_0)$ the x-axis, and the y-axis is perpendicular. The affine transform matrix that maps from local to footprint coordinates reads

$$A \quad := \quad \begin{pmatrix} \mathbf{p}(\mathsf{TM}_1)_x - \mathbf{p}(\mathsf{TM}_0)_x & -\mathbf{p}(\mathsf{TM}_1)_y + \mathbf{p}(\mathsf{TM}_0)_y & \mathbf{p}(\mathsf{TM}_0)_x \\ \mathbf{p}(\mathsf{TM}_1)_y - \mathbf{p}(\mathsf{TM}_0)_y & \mathbf{p}(\mathsf{TM}_1)_x - \mathbf{p}(\mathsf{TM}_0)_x & \mathbf{p}(\mathsf{TM}_0)_y \\ 0 & 0 & 1 \end{pmatrix}.$$

   Each static marker position is then transformed to local coordinates by

$$A^{-1} \cdot \mathbf{p}(\mathsf{TM}_i) \qquad \forall i \in \{0, ..., \alpha - 1\}.$$

   Note that this projection has to be performed only on start-up after the static footprint for each widget has been loaded.

Find two touches matching the first two static markers of footprint.

2. Within the point cloud of touches, the algorithm searches for a pair (T_i, T_j) where T_i is a "rectangle" touch, and the distance between both touches is close to the distance between TM_0 and TM_1, i.e.,

$$\Big| \|\mathbf{p}(T_i) - \mathbf{p}(T_j)\| - \|\mathbf{p}(\mathsf{TM}_0) - \mathbf{p}(\mathsf{TM}_1)\| \Big| \quad < \quad \epsilon_D$$

   where ϵ_D is a distance threshold.

To avoid recognition of incorrect widgets at an early stage, we perform a quick check on the two pairs: The angle between the principle axis of the "rectangle" marker TM_0 and the vector $\mathbf{p}(\mathsf{TM}_1) - \mathbf{p}(\mathsf{TM}_0)$ must be close to the corresponding angle defined by marker T_i and the vector $\mathbf{p}(T_j) - \mathbf{p}(T_i)$ (Fig. 3.20).

3. Analogously to step 1, the two touches T_i and T_j span a local coordinate system, where $\mathbf{p}(T_i)$ is the origin and $\mathbf{p}(T_j) - \mathbf{p}(T_i)$ denotes the x-axis, and the y-axis is orthogonal to the x-axis. This resulting transform matrix reads

Define local coordinate system from found touches.

$$B \; := \; \begin{pmatrix} \mathbf{p}(T_j)_x - \mathbf{p}(T_i)_x & -\mathbf{p}(T_j)_y + \mathbf{p}(T_i)_y & \mathbf{p}(T_i)_x \\ \mathbf{p}(T_j)_y - \mathbf{p}(T_i)_y & \mathbf{p}(T_j)_x - \mathbf{p}(T_i)_x & \mathbf{p}(T_i)_y \\ 0 & 0 & 1 \end{pmatrix}.$$

4. For every type marker $\mathsf{TM}_k, \forall k \geq 2$ in the footprint, the algorithm now tries to find a corresponding touch $T_l \neq T_i, T_j$ that matches the marker's position in local coordinates:

Find corresponding touches for remaining static markers.

$$\|A^{-1} \cdot \mathbf{p}(\mathsf{TM}_k) - B^{-1} \cdot \mathbf{p}(T_l)\| \; < \; \epsilon_D.$$

We reject every pair (TM_k, T_l) if type of marker and touch differ or if their radii vary too much.

If a touch is found for each marker, the widget is detected. Otherwise, the footprint does not match the set of touch events, and the algorithm proceeds with the next footprint.

5. If the widget is detected, we compute the affine transform G that maps from footprint coordinates to global tabletop coordinates. Every graphical output of the widget is transformed using this map. We determine G by computing an affine matrix that registers the ordered point cloud of type markers $(\mathbf{p}(\mathsf{TM}_0), \mathbf{p}(\mathsf{TM}_1), ..., \mathbf{p}(\mathsf{TM}_{\alpha-1})))$ with the point cloud of corresponding touches $(\mathbf{p}(T_{k_0}), \mathbf{p}(T_{k_1}), ..., \mathbf{p}(T_{k_{\alpha-1}}))$ by minimizing the sum of squared distances

Compute (graphical) mapping from local footprint coordinates to global coordinates.

$$\sum_{l=0}^{\alpha-1} \left\| G \cdot \mathbf{p}(\mathsf{TM}_l) - \mathbf{p}(T_{k_l}) \right\|^2 \; \rightarrow \; \min$$

where k_l denotes the index of the touch that corresponds to marker TM_l with $k_0 := i$ (first touch) and $k_1 := j$ (second touch).

Note that, due to sensor noise and sampling issues, we never check for equality of values like position or radius, but use a difference threshold instead. Furthermore, it is unlikely that all markers of a widget become visible in a single frame. Our algorithm, therefore, considers all touch events for widget detection which were created within the last second.

Equality tests are relaxed spatially and temporally to cope with noise.

The detection algorithm has a complexity of $\mathcal{O}(f(t^2 + mt))$, where f is the number of widgets or footprints, m denotes the maximum number of markers in every footprint, and t is the number of detected touches. Early culling of non-"rectangle" touches and those that are older than one second

Early culling of irrelevant marks improves run-time.

Spatial search structure can reduce complexity.

speeds up the algorithm in every frame. The complexity of the widget detection can also be reduced to $\mathcal{O}(f(t \cdot \log(t) + m \cdot \log(t)))$ by employing spatial search structures, such as quad trees or k-d trees. These would speed up the search for surrounding marker correspondences (step 2 and 4). However, maintaining the search structure in every frame requires additional memory and time.

Presence and absence of ID markers maps to bits in a widget ID.

Identification When the widget is detected, we derive a numeric ID from the identification markers $\mathsf{ID}_0, ..., \mathsf{ID}_{\beta-1}$. In same way as step 4 of the widget detection, we search for touch events that match the position of $\mathsf{ID}_0, ..., \mathsf{ID}_{\beta-1}$ in local coordinates. The identification number then reads

$$\text{WidgetID} := \sum_{i=0}^{\beta-1} \text{exist}(\mathsf{ID}_i) \cdot 2^i$$

with

$$\text{exist}(\mathsf{ID}_i) = \begin{cases} 1 & \text{if touch close to } \mathsf{ID}_i \text{ exists,} \\ 0 & \text{if no corresponding touch event is found.} \end{cases}$$

That is, each significant bit in the widget ID encodes whether the corresponding identification marker was attached to the physical widget or not. An example encoding the ID 2 (0 + 2) is shown in Fig. 3.8 on page 45. Note that n identification markers allow to distinguish 2^n widgets of the same type.

Since it is possible that a widget type is recognized before all identification markers have appeared in the camera, we slightly delay the widget identification by a few frames. Once a widget is detected and identified, all touches involved in this process are blocked; they will neither be used for widget detection, nor can they be interpreted as finger input anymore. The widget is now ready to use for interaction.

Every widget uses an individual strategy to interpret state.

State All touch events that occur *within* the area of the SLAP Widget are used to determine the internal *state* of the control. As described in section 3.3.3, every widget applies its own strategy to map touch positions in local coordinates to a specific event or value. Therefore, this detection step is implemented in individual classes for each widget (see section 3.6).

When moving a widget, coordinate mappings are updated.

Moving If a user moves a detected SLAP Widget across the table, the touches corresponding to type markers change their locations accordingly. To keep the graphics of the widget aligned, our software computes an updated affine transform by repeating step 5 of the detection algorithm. However, in order to avoid jittering, the transform G is only updated if the position or rotation of the touches have changed significantly.

Upon removal, widget graphics are hidden. If detected later, previous state is restored.

Removing In case the user removes a widget from the table, at least one touch event associated with a type markers is converted into a touchesEnded event. In this case, the widget and all connected pairing lines are hidden. Furthermore, all touch events that are linked to the control are unblocked. When the widget is detected again later, its internal state and graphics are

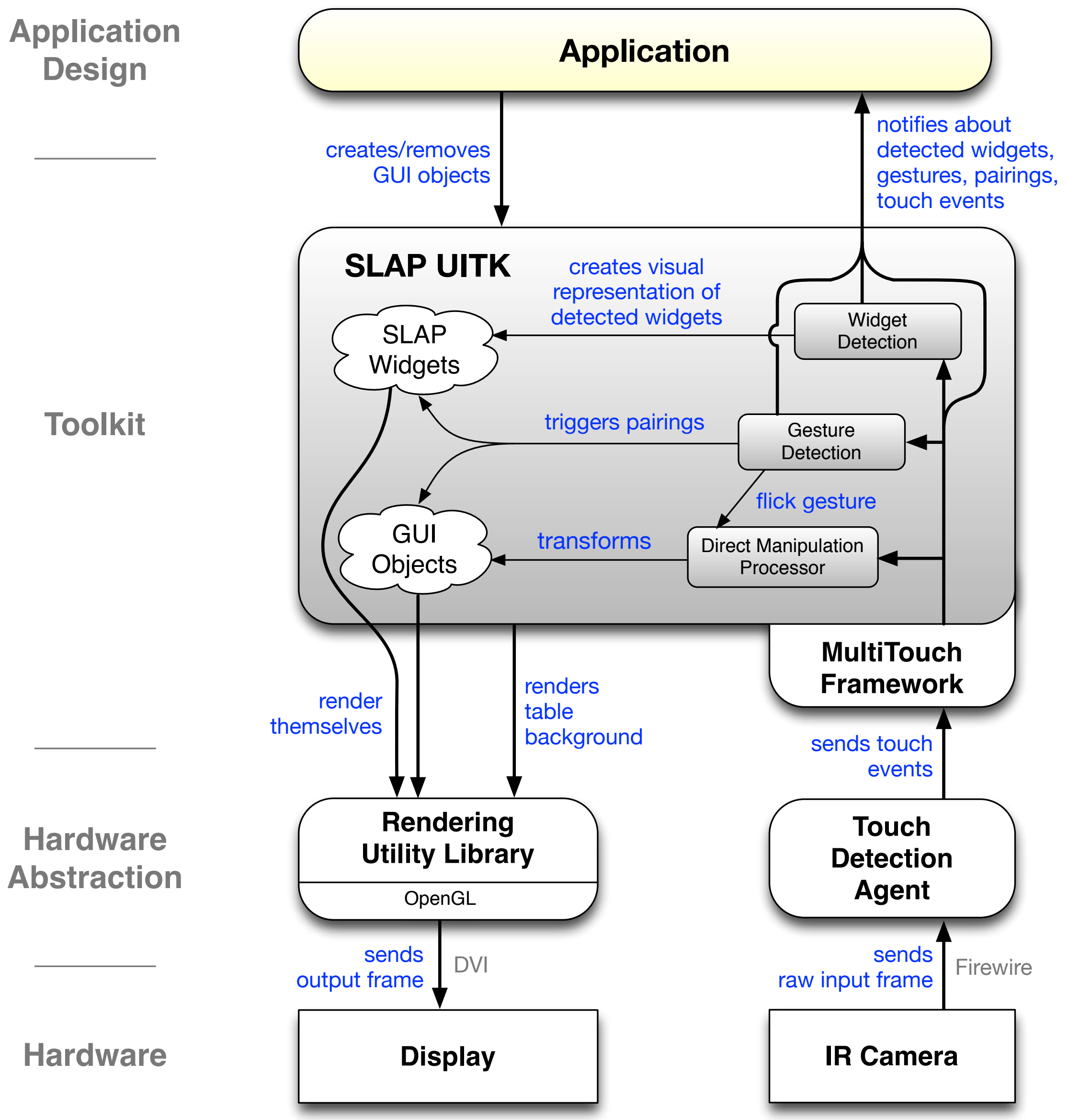


Figure 3.21: Software architecture of SLAP Framework.

restored. Note that the widget's configuration, including all pairings, is maintained even when the widget is removed. This allows users to quickly remove a control if the space is temporarily required for a different task. Afterwards, the control can be placed on the table again, and the previous task can be continued without further effort.

3.6 Software Architecture

Our tabletop applications and the interaction with SLAP Widgets is handled by the *SLAP Framework*. It provides a layered software architecture that hides the underlying table hardware from the program designer.

The SLAP Framework is written in *Objective-C* with support of the *Cocoa* framework[7]. We employ OpenGL to render graphics on the tabletop.

SLAP Framework provides layered software architecture that hides underlying hardware.

Fig. 3.21 shows the different abstraction layers of the SLAP Framework. The input and output hardware represents the lowest level (section 3.3.1). The touch detection agent (section 3.5.1) converts the camera's video stream to touch events that are sent via the MultiTouchFramework to the core of our framework, the *SLAP User Interface Toolkit (SLAP UITK)*. This toolkit provides a basic set of tabletop objects, such as images, videos, and text boxes, as well as basic operations to manipulate them. Users can translate, scale, move, or flick objects as described in section 2.2.

The SLAP UITK detects widgets, handles table interaction, and provides a programming interface.

The SLAP UITK also recognizes SLAP Widgets (section 3.5.2) and gestures (section 3.6.2.4). It generates and maintains visual representations of all detected SLAP Widgets. It also handles the pairing mechanism as well as the communication between widgets and virtual objects. Finally, it is responsible for rendering the graphical output. A *Rendering Utility Library* provides a set of graphical operations, such as drawing shapes and images. These operations are translated into OpenGL commands that are, in the end, send to the graphics card and shown on the surface. An application layer on top of the SLAP UITK provides a simple and extensible interface for programmers to develop SLAP applications. The application creates and controls the virtual objects that are displayed on the table. It can also react on various notifications from the UITK, e.g., when widgets are detected or objects are paired.

While a full documentation of the framework is beyond the scope of this thesis, we will now describe the basic concepts that enable programmers to write SLAP applications and to add new functionality.

3.6.1 Writing SLAP Applications

The SLAP Framework is compiled into a single framework that programmers can include in their project. Listing 3.2 shows a simple "Hello world!" application.

Writing a SLAP program basically requires three steps:

- Include the framework header (line 1).
- Initialize the framework (line 6). `createFullscreenInterface` creates a fullscreen view, reserves memory for all data structures, and initializes the detection algorithms.

[7]http://developer.apple.com/technologies/mac/cocoa.html

```
 1  #import <SLAPFrameworkGL/SLAPUITK.h>
 2
 3  - (void) awakeFromNib    // Called on start-up
 4  {
 5    // Create full-screen interface
 6    SLAPUITK* uitk = [SLAPUITK createFullscreenInterface];
 7
 8    // Create text box
 9    NSRect rect = NSMakeRect(100,100,400,200);
10    SGOTextBox* textBox =
11      [uitk addTextWithRect:rect rotation:45 inParent:nil];
12    [textBox setString:@"Hello world!"];
13  }
```

Listing 3.2: SLAP “Hello world” program.

- Create all required virtual on-screen objects, such as the text-box in our sample (lines 9–12). For the creation of a virtual object, a target rectangle, an initial rotation, and its parent object is required. The latter parameter allows programmers to arrange virtual objects in a hierarchy.

Programmer specifies objects, SLAP Framework takes care of table interaction.

Our sample program is already fully operational: It renders the graphics, detects finger touches and widgets, recognizes gestures, and handles pairings between objects. Users can place any of the standard widgets on the tabletop and edit the text box.

Delegate methods allow reaction to events.

If developers want to react on user input on the table or influence the graphical output, we allow them to implement *delegate methods*[8]. An example is shown in Listing 3.3.

In lines 10 and 11, the programmer specifies that the application class implements delegate methods which receive events from the UITK. An example is the method `object:hasBeenPairedWith:` (lines 25–29). Every time two objects are paired, the UITK calls this method, which, in this case, only outputs this event to the console. Programmers can also receive events from the view (line 10). In our sample, the method `drawBackground` is implemented. It is called *before* the SLAP UITK renders objects and, e.g., allows programmers to draw a “wallpaper” texture behind all other graphics (lines 15–22).

Tables 3.1 and 3.2 list the methods of two protocols that programmers can implement to react on events or to perform drawing, respectively. All delegate methods are optional; the UITK verifies that a delegate method is implemented before calling it. Note that a SLAP application can also process raw touch events by using methods from the `MTTouching` protocol.

[8] An Objective-C delegate is an object containing methods that are called from different objects to perform specific tasks. A delegate method can also be considered as the object-oriented version of a callback.

```
#import <SLAPFrameworkGL/SLAPUITK.h>

- (void) awakeFromNib
{
  SLAPUITK* uitk = [SLAPUITK createFullscreenInterface];

  // Create some virtual objects ...

  // Set delegate
  [uitk setDelegate:self];
  [uitk.view setDelegate:self];
}

// Draw background
- (void) drawBackground
{
  NSRect rect = [[SLAPUITK sharedUITK] canvasRect];

  duSetColorRGB(1, 1, 1);   // white color tone
  duDrawRect(0, 0, rect.size.width, rect.size.height, Stretch, Stretch,
    &backgroundTexture);
}

// React on pairing
- (void) object:(SLAPAlignableObject*) obj1
  hasBeenPairedWith:(SLAPAlignableObject*) obj2
{
  NSLog(@"Object %@ paired with object %@.", obj1, obj2);
}
```

Listing 3.3: Sample SLAP program that uses delegates to react on events and to render additional graphics.

Delegate	Description
`widgetPutOnTable:`	Widget detected on table
`widgetMoved:`	Widget moved on table
`widgetRemovedFromTable:`	Widget removed from table
`widgetsUpdated`	Status of widgets on table changed
`object:willBePairedWith:`	One object will be paired with another object
`object:willBeUnpairedFrom:`	One object will be unpaired from another object
`object:hasBeenPairedWith:`	One object was paired with another object
`object:hasBeenUnpairedFrom:`	Object was unpaired from another one
`pairingDeniedBetweenObject:andObject:`	Pairing between two objects is denied
`touchesBegan:withEvent:`	New set of touches on tabletop
`touchesMoved:withEvent:`	Set of touches moved
`touchesEnded:withEvent:`	Set of touch trajectories ended on tabletop

Table 3.1: Event delegates of `SLAPUITKDelegate` protocol.

Delegate	Description
`drawBackground`	Called *before* tabletop objects are drawn.
`drawForeground`	Called *after* tabletop objects are drawn.

Table 3.2: Rendering delegates of `SLAPViewDelegate` protocol.

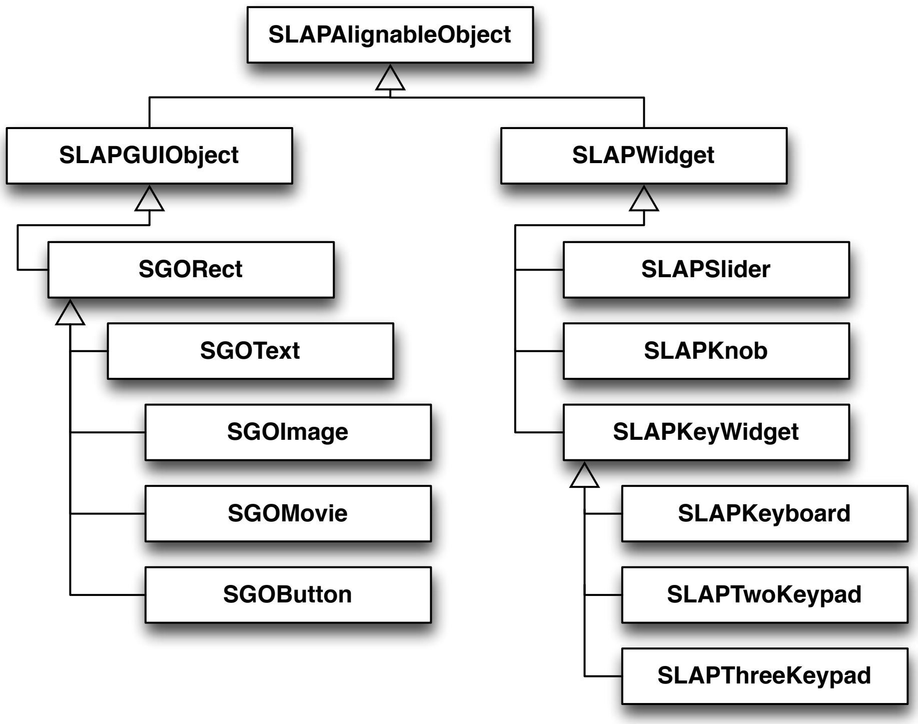


Figure 3.22: Class hierarchy of table objects in SLAP Framework.

3.6.2 Extending the Framework

We implemented a basic set of SLAP Widgets and virtual objects to prove our concept and to conduct user studies. However, more complex tasks might require additional UI elements. We constructed the SLAP Framework in a modular way so that it enables designers to create new controls and on-screen objects.

Fig. 3.22 shows the class hierarchy of basic objects in the SLAP UITK. The class `SLAPAlignableObject` provides functions to show 2D objects on the tabletop. It contains basic attributes like position, size, and affine transform, as well as function stubs for rendering and pairing. In the following, we describe how to add new virtual objects and SLAP Widgets. Both are represented by specializations of the class `SLAPAlignableObject`.

3.6.2.1 Virtual Objects

The SLAP UITK provides basic objects for table applications: Simple rectangles (`SGORect`), text fields (`SGOText`), images (`SGOImage`), movie players (`SGOMovie`), and on-screen buttons (`SGOButton`). Every virtual object is implemented in a sub-class from `SLAPGUIObject`. Thereby, the object provides basic finger interaction; users can move, scale or rotate it, and even flick it

New virtual objects are sub-classed from SLAPGUIObject.

```
 1 // Init the object
 2 -(id) initWithFrame:(NSRect)frame
 3         onGUIObject:(SLAPGUIObject*) parent;
 4 {
 5   self = [super initWithFrame:frame onGUIObject:parent];
 6   if(self)
 7     self.color = [NSColor redColor];  // Red is default color
 8
 9   return self;
10 }
11
12 // Draw filled rectangle
13 - (void) draw
14 {
15   duSetColor(self.color);
16   duDrawRect(0, 0, self.size.width, self.size.height, Fill);
17 }
```

Listing 3.4: Example implementation of a virtual object.

around. To create an own object, `SLAPGUIObject` (or one of its sub-classes) must be derived, and only few methods must be overwritten:

Initialization, drawing, and hit tests (for non-rectangular objects) must be overwritten.

- `initWithFrame:onGUIObject:` initializes the object and custom attributes. Listing 3.4 shows the initialization of a simple rectangular object (lines 2-10).
- `draw` renders the content of the object (lines 13-17). Programmers can employ OpenGL commands or functions from our Rendering Utility Library (prefix "`du`") to render simple shapes and textures. Note that the object is rendered in *local* coordinates. The programmer does not care how the object is transformed on the table surface, because this is handled by framework.
- If the object does not have a rectangular shape, the methods `isHit:` and `isInPairingArea:` should be overwritten. For a given global coordinate, they determine whether or not this point is within the area of the object and within the aura that is considered for pairing gestures, respectively. This makes sure that touch events are only sent to an object if they belong to its area.

`SLAPGUIObject` complies to the `MTTouching` protocol, which allows programmers to interpret touch events that occur inside the virtual object. The SLAP UITK sends every touch, whose location passes the hit test (`isHit:`) of an object, to the respective touch method of that object.

Listing 3.5 shows an example of a simple horizontal slider. Every time a spot appears in the rectangle of the control (line 7), its location is mapped to local coordinates, and a relative knob position between 0 and 1 is computed (line 13). The method also *captures* all touch events that it receives (line

```
 1 // Touches appear
 2 -(void) touchesBegan:(NSMutableSet*)touches
 3           withEvent:(MTEvent *)event;
 4 {
 5   [self processTouchesByChildren:touches withEvent:event];
 6 
 7   if([touches count] >= 1)
 8   {
 9     MTTouch* touch = [touches objectAtIndex:0];
10     NSPoint ptLocal =
11       [self pointInLocalCoordinates:touch.location];
12 
13     self.sliderPos = ptLocal.x / self.size.width;
14 
15     [[SLAPUITK sharedUITK] captureTouch:touch
16                             forGUIObject:self];
17     [touches removeObject:touch];
18   }
19 }
20 
21 // Touches disappear
22 -(void) touchesEnded:(NSMutableSet*)touches
23           withEvent:(MTEvent *)event;
24 {
25   [self processTouchesByChildren:touches withEvent:event];
26 
27   for(MTTouch* touch in touches)
28     [[SLAPUITK sharedUITK] uncaptureTouch:touch];
29   [touches removeAllObjects];
30 }
```

Listing 3.5: Touch event interpretation of a virtual slider.

15–16). This makes sure that the button receives the touchesEnded event, even if the user drags a touch outside the slider's area.

`SLAPGUIObject` also supports nested hierarchies of virtual objects. By calling method `processTouchesByChildren:withEvent:` (lines 5 and 25), touch events are first sent to the topmost child object under the touch position. If a child object uses a touch, it deletes it from the `touches` set (lines 17 and 29). Thus, only the remaining touches that belong to the parent object are actually processed by it.

Virtual objects support nested hierarchies.

3.6.2.2 SLAP Widgets

The development of a new SLAP Widget usually begins in with a construction in a vector-based graphics program. When the design is complete, all parts are cut out from an acrylic plate using a laser cutter. Soft components like the key pads' buttons are cast in silicone, using molds made of acrylic. Other parts like bearings or springs are bought separately. Then, the control can be assembled, and footprint markers are glued to the bottom.

Parts of SLAP Widget are created by laser cutting acrylic or by casting silicone.

```
-(void) touchesUpdated
{
  if(CFArrayGetCount(self.touches) > 0)
  {
    // Retrieve first touch
    MTTouch* touch = (MTTouch*) CFArrayGetValueAtIndex
                                        (self.touches, 0);

    // Compute slider position from first touch
    NSPoint ptLocal =
      [self pointInLocalCoordinates:touch.location];
    self.sliderPos = ptLocal.x / self.size.width;
  }
}
```

Listing 3.6: Touch event interpretation of a physical slider.

When specifying the footprint, special consideration is needed to generate an unambiguous pattern that does not interfere with existing widgets. After that, the widget is placed on the tabletop, and a special application, *SLAPFootprintGenerator*, captures the footprint.

New widget classes are sub-classed from SLAPWidget.

Once a control is built and its footprint is acquired, a new class must be derived from `SLAPWidget`. The class is responsible for rendering the back projected graphics, to interpret and store the internal state from the footprint, and to communicate with paired objects on the table. In a similar way as virtual objects, only few methods have to be overwritten:

Initialization, drawing, and hit tests (for widgets with non-rectangular bottom) must be overwritten.

- `initWithPosition:` initializes the object, the size of its graphical representation, and custom attributes.
- `draw` renders the back projection of the object in local coordinates.
- Analogously to virtual objects, the two hit tests `isHit:` and `isInPairingArea:` must be overwritten for non-rectangle widgets, such as the knob.

A widget's state depends on the arrangement of touches within the area it covers. The SLAP UITK keeps track of all touches that belong to a certain widget. As soon as touches appear, move, or disappear within the area of the widget, the UITK updates the list of touches for that widget and calls the method `touchesUpdated`. Each widget overwrites this function to determine its current state. An example of a simple physical slider is shown in Listing 3.6. We deliberately reduced a widget's touch interpretation to a single method, because widget designers are usually not interested in the phase of touch events but only in its positions. Of course, the phase can be accessed using the `phase` attribute of the touch event.

New widgets are registered in UITK.

To ensure that a widget is detected by the UITK, it must be registered using `registerWidgetClass`. For example,

```
[[SLAPUITK sharedUITK] registerWidgetClass:[MySLAPSlider Class]]
```

registers the class `MySLAPSlider`. From now on, the UITK will detect the widget when it is placed on the table.

If a widget contains keys only, like the keyboard or the keypads, it can be derived from class `SLAPKeyWidget`. It provides basic functions to create and update key layouts and to detect key strokes.

3.6.2.3 Inter-Object Communication

After pairing gesture, each object is requested whether it is pairable with the other one.

Every time a user tries to pair two objects on the surface, the UITK "asks" each object whether it can be linked to the other one. It calls the method `canPairWith:`, which is inherited from `SLAPAlignableObject`, for both objects with the respective other object as parameter. Only if both objects return "yes", the objects are connected.

After agreement to pair, connected objects configure each other.

After pairing, the UITK calls method `pairedWith:` of both objects. Objects can now configure each other. Most commonly, virtual objects configure the SLAP Widgets that they are connected to. For example, remember that a SLAP Knob can switch between multiple modes depending on the object it is connected to. If two objects are unpaired, their methods `unpairedFrom:` are called.

A simple example of a movie player is shown in Listing 3.7. The player can be paired with sliders and knobs. If paired with a knob, the knob is switched to jog wheel mode. When pairing it with a slider, the slider is initialized to create values between 0 and 1. An object must always consider that the target object might be paired with other ones already. For simplicity, our example movie player only agrees to pair with objects that do not have any associations yet (line 6).

When a SLAP Widget is operated, it sends events to all objects it is paired with. Two event types are possible:

A SLAP Widget can either send command or value events.

- *Command events* are produced when a button is pushed or an item in a knob menu is selected. They usually trigger actions. Each event contains a command ID.
- *Value events* set continuous values. SLAP Sliders and SLAP Knobs in value mode send these kinds of events when they are moved or turned, respectively. Value events contain an ID and a real number, which is usually mapped from the control's state to a programmer-defined range.

A virtual object class must implement protocol `SLAPCommandReceiver` or `SLAPValueReceiver` to be able to receive command or value events, respec-

```
- (BOOL) canPairWith:(SLAPAlignableObject*) object
{
  // Movie player can be paired with sliders and knobs
  return ([object isKindOfClass:[SLAPSlider class]] ||
          [object isKindOfClass:[SLAPKnob class]]) &&
          [object.targets count] == 0;
}

- (void) pairedWith:(SLAPAlignableObject*) object
{
  // If we pair with a knob, switch it to jog wheel mode
  if([object isKindOfClass:[SLAPKnob class]])
  {
    SLAPKnob* knob = (SLAPKnob*) object;
    knob.mode = [SLAPKnobModeJogWheel jogWheelMode];
  }
  // If it's a alider, set its range
  else if([object isKindOfClass:[SLAPSlider class]])
  {
    SLAPSlider* slider =  (SLAPSlider*)object;
    slider.sliderId = ID_SLIDER_MOVIE_POSITION;
    slider.minValue = 0;
    slider.maxValue = [self.movie length];
    slider.value    = 0;
  }
}
```

Listing 3.7: Pairing protocol of simple movie object.

```
@protocol SLAPCommandReceiver
-(void) executeCommand:(int) commandName
            fromSource:(id) source;
@end

@protocol SLAPValueReceiver
-(void) setValue:(float) value
          withId:(int) id
      fromSource:(SLAPWidget*) source;
@end
```

Listing 3.8: Protocols for receiving events from widgets.

tively. These protocols are shown in Listing 3.8. Every event method also contains a parameter that points to the sending widget.

An example interaction protocol between a movie object and a SLAP Keypad is illustrated in Fig. 3.23.

3.6.2.4 Gesture Detection

Gestures provide a rich input channel that enables users to trigger actions without the need for menu structures (cf. section 2.2.2 on page 23). The

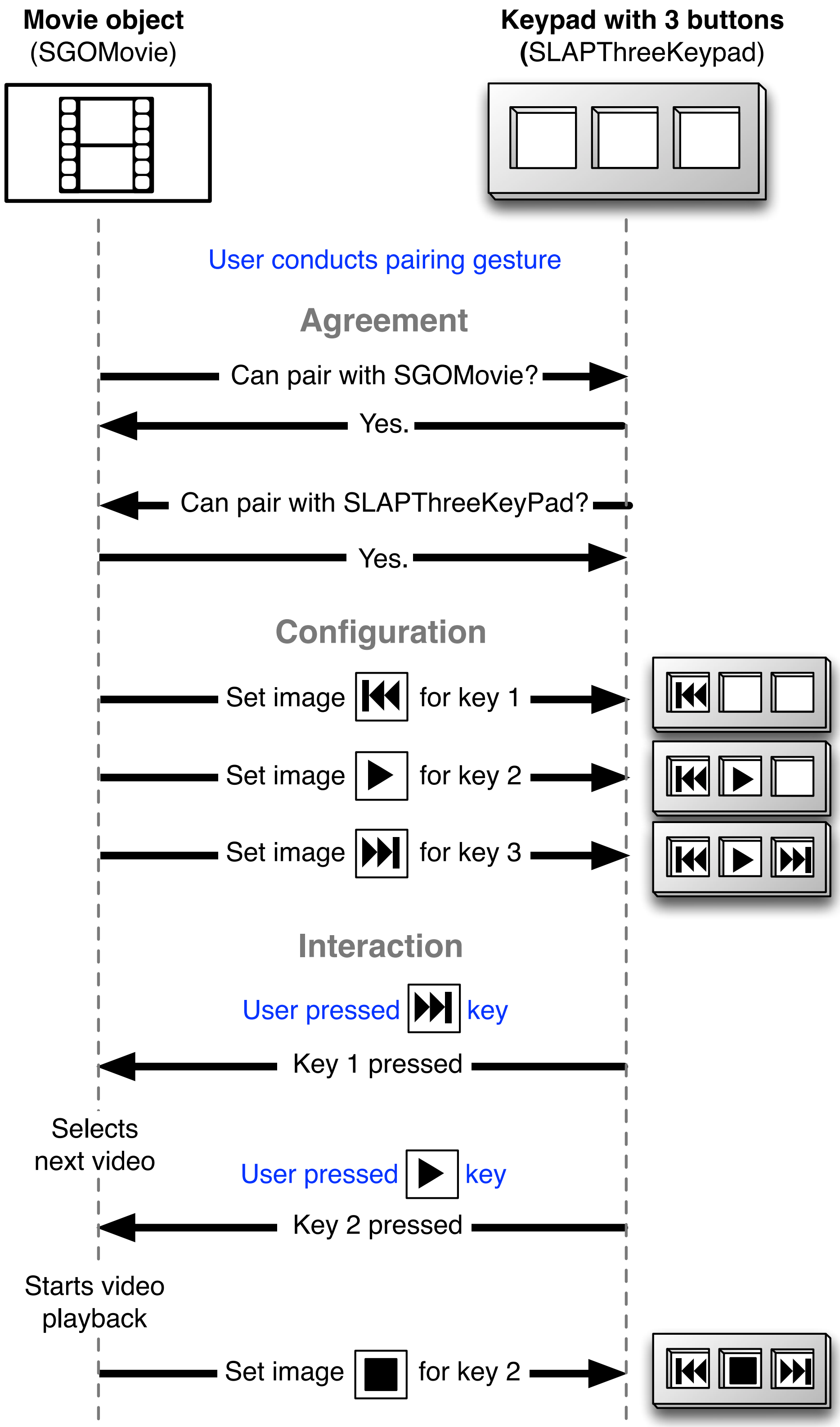


Figure 3.23: Example interaction protocol between a virtual movie object and a SLAP Keypad. Note that the movie dynamically updates the labels of the keypad when playback has started.

SLAP Framework contains extendable gesture detection engine.

SLAP Framework contains a gesture detection engine that processes touch events and matches them with a list of registered gestures. By default, the framework supports the aforementioned pairing gestures (section 3.4.1) and a *flicking* gesture for quickly moving objects to distant locations on the table.

Gestures are derived from Gesture class and can receive touch, path, or time events.

To implement a new gesture, a programmer creates a class derived from class `Gesture`. He then has to implement at least one of three handler protocols:

- `GestureTouchHandler` receives regular touch events of the `MTTouching` protocol (Listing 3.1).
- `GesturePathHandler` receives touch trajectories, i.e., the list of touch positions between a touchesBegan event and its corresponding touchesEnded event. All gestures that involve strokes and paths use this protocol.
- `GestureTimerHandler` receives a timing event every 10 ms. Gestures that rely on timings, like the pairing gestures, implement this protocol.

Gesture handling is separated into three protocols due to performance reasons.

The gesture detection engine distributes touch events to each gesture according to the protocols it implements. We intentionally chose to employ three different protocols in order to optimize for time and space consumption. For example, touch trajectories are only captured if at least a single gesture conforms to the `GesturePathHandler` protocol. Furthermore, they are only stored *once* for all gestures that use them. Within these handlers, the gesture is detected according to certain heuristics. For example, a "closed path" gesture is defined by a trajectory whose start and end point are close to each other.

Detected gestures call delegate method.

To activate a new gesture, it is added to the observer list of the gesture detector. Furthermore, the developer must specify a delegate that implements the `GestureDelegate` protocol. When a gesture is detected, it sends an event to this delegate. Listing 3.9 shows an example. The program first creates a gesture object that is able to detect circular strokes drawn on the table (line 4). It is then added to the gesture detection engine (line 7). All gesture events are processed by the handler `gestureExecuted:withInfo:` (lines 12-24). It receives a pointer to the gesture object that was detected as well as a dictionary with further information, such as start and end point of the trajectory. Note that the handler pipes default events, e.g., pairing gestures, to the SLAP UITK (lines 22).

3.7 Usage Scenarios

SLAP Widgets are helpful for all tabletop tasks that require eyes-free triggering of actions or precise setting of continuous parameters. Thanks to

```
- (void) initGestures
{
  GestureDetector* gd = [[SLAPUITK sharedUITK] gestureDetector];
  Gesture* circleGesture = [[CircleGesture alloc] init];

  [gd setDelegate:self];
  [gd observeGesture:circleGesture];

  [circleGesture release];
}

- (void) gestureExecuted:(Gesture*) gesture withInfo:(NSDictionary*) info
{
  if([gesture isKindOfClass:[CircleGesture class]])
  {
    // Circle gesture detected
    NSLog(@"Circle drawn.")
  }
  else
  {
    // This seems to be a default gesture
    [[SLAPUITK sharedUITK] handleDefaultGesture:gesture withInfo:info];
  }
}
```

Listing 3.9: Example program that reacts on a gesture.

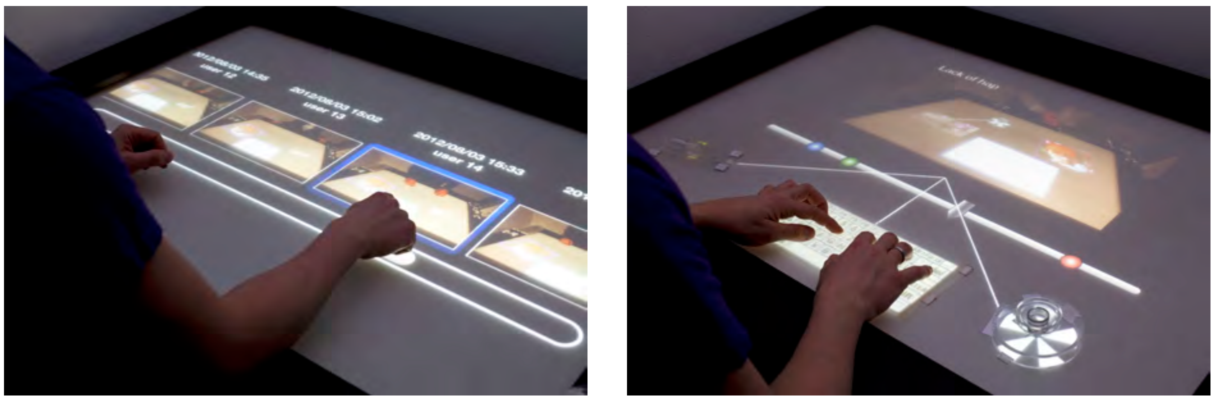


Figure 3.24: Video annotation scenario. Left: User browses and selects video via direct manipulation. Right: User annotates a frame in a user test video using SLAP Widgets.

the use of back projection, the controls are versatile, and allow users to apply our basic set of widgets to various applications. In the following, we describe usage scenarios that combine direct manipulation of on-screen objects with the ad hoc use of SLAP Widgets.

3.7.1 Video Ethnography

This scenario shows the use of SLAP Widgets in a video ethnography scenario where a user annotates points of interest in a video (Fig. 3.24).

In the video ethnography scenario, SLAP Widgets support video navigation and frame annotation.

After conducting a user test, Alice wants to analyze and annotate the video footage that she has captured about participants interacting with a novel mobile application. She has captured the videos using a digital camera that saved all files on an SD card. Alice approaches the table and plugs the SD card into an embedded reader device. The table now loads all video files from the card and displays previews of them in a horizontal dock. Alice browses to the video of the latest user test by dragging the dock to the left. She then double-taps the video she wants to annotate. The video browser fades out, and the preview of the selected video scales up to a video player. The table also shows a time-line that displays user-defined bookmarks. The application is now in annotation mode.

Alice takes a SLAP Slider and a SLAP Knob from the rim of the table and pairs them with a video object. She also pairs a two button keypad with the video. The keypad shows buttons to start/pause and stop the video. Alice remembers that the participant made a relevant observation in the middle of the test. She drags the slider to the middle and presses "Play" on the keypad to start the video. She watches the video until the participant says: "It's frustrating that I do not get haptic feedback here". Alice presses "Stop" on the keypad and uses the SLAP Knob in jog wheel mode to navigate back to the moment the user started his statement. She then grabs a SLAP Keyboard, pairs it with the video, and enters: "Participant: Lack of haptic feedback frustating". This annotation is now associated with the frame: The text appears at the top of the frame, and a bookmark is shown in the timeline. Also, an on-screen button appears next to the statement that allows deleting the bookmark.

Alice continues her work until all relevant points in the video are annotated. She later closes the annotation mode by double-tapping the video again and processes other user test videos. When she is done, she removes all widgets and unplugs the SD cards. The table's surface turns black.

3.7.2 Collaborative Image Selection and Editing

In the collaborative image selection and editing scenario, SLAP Widgets augment parallel subtasks.

Selecting and editing images for a daily newspaper is a crucial process that must be conducted in very limited time. In the digital age, the selection and cropping process is often still entirely paper-based. An employee usually prints out all pictures that are related to a story and aligns them on a table. In a quick meeting, an experienced editor culls the images, chooses the ones that will be printed, and *manually* crops them by folding. After that, an employee digitally crops the images and places them into the design. A *digital* scenario could look as follows (Fig. 3.25).

Chris, editor of the "Daily HCI", enters the room and approaches the SLAP Table. Doreen, the graphical designer, stands at the opposite side of the table. The five stories of the current issue are listed on the surface. Chris clicks on "SLAP Widgets: The Future of Surface Computing?". At his side of the table, the application now shows all images related to the topic in a

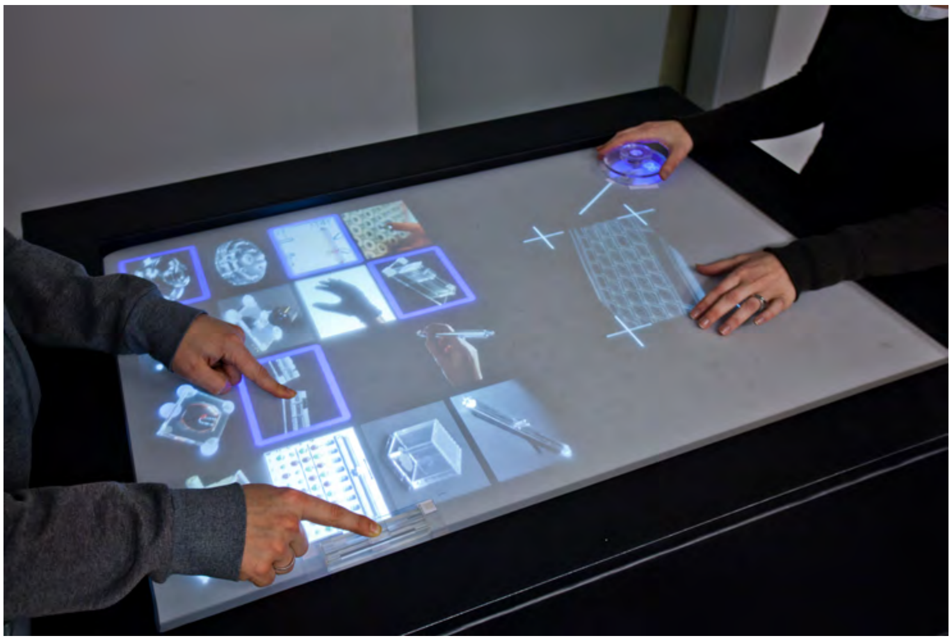


Figure 3.25: Collaborative image selection and editing by combining direct manipulation and SLAP Widgets.

grid. As there are more photos than could be shown at once, the application distributes the photos on multiple pages. Chris pairs a SLAP Knob with the photo grid for scrolling the viewport of the photo grid. He browses through the pictures and decides that a photo containing a user operating a knob while smiling should be part of the front page. He taps on the photo. A green frame appears around it, indicating that it is selected for further editing. Then, it scales up and moves over to Doreen's side. While Chris continues searching for interesting photos, Doreen starts to edit the selected one.

The selected photo is framed within a crop rectangle at Doreen's side. Using direct manipulation, she can now scale, shift, and rotate the photo. The crop rectangle is stationary; it denotes the area of the photo after cropping. Doreen decides that the colors need some improvements. She pairs a three button keypad with the photo and clicks on the icon for "Auto color", triggering a pre-adjustment of the colors. She then associates a SLAP Knob with the image. The knob is in "menu mode" and shows various options to change image properties. Doreen selects "Saturation" and pushes the knob. She then adjusts this property with the knob, which is now in value mode. She pushes the knob again to return to the menu. In the following, she changes further parameters until she is contented with the image quality. Doreen performs a flick gesture on the image, pointing into Chris' direction. This indicates that the editing is done. The cropped and modified image moves back to its position in the photo grid. The next photo that Chris has selected appears in her area.

In the meantime, Chris continues to select other photos. He also deletes those that will not be part of the issue by using a strike out gesture. When he is done, he tabs the "Layout" button. While Doreen edits further photos, Chris starts to embed the finalized photos into the layout of the newspaper.

3.8 User Studies

We conducted three user studies to evaluate the efficiency and usability of SLAP Widgets. These are described in the following.

3.8.1 Widget Performance

Hypothesis: SLAP Knob outperforms virtual knob in terms of task completion time and accuracy.

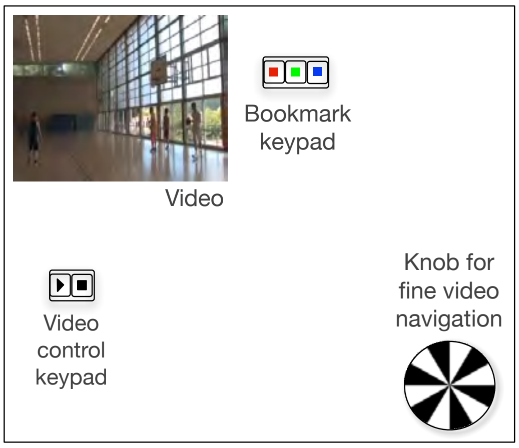
In our first study, we tested whether SLAP Widgets outperform on-screen controls in tasks that require eyes-free interaction. We chose a video navigation and annotation task as basis for a quantitative user test. In conventional video navigation, users look at a video screen while operating a jog wheel with their hand. We anticipated that a SLAP Knob outperforms its virtual counterpart in terms of task completion time and accuracy. We assumed that users can rely on their haptic sense, and—opposed to on-screen controls—they do not have to look at the control to realign their hands during operation.

3.8.1.1 Task

Participants' task: Find and mark unicolored frames in a video.

We instructed participants to find and mark specific frames in a video as fast as possible. We modified a set of videos by inserting three tinted *target frames* in each of them. The first target frame was uniformly filled with red, the second with green, and the last one with blue color. In the beginning of each trial, a video player showed the first frame of the video. The participant then started the video using a "play" key and watched the footage in the player until he recognized the tinted frame. As the videos were captured with 25 frames per second, the target frame could be perceived as a short flash during playback. Accordingly, the participant pressed the "stop" key and used a jog wheel to navigate back to the target frame. He finished the operation by pushing a button with the same color as the target frame. Users had to perform all operations with the right hand.

We implemented the task on the SLAP Table. Fig. 3.26 shows the test setup as seen from the participant who standed at one of the long sides of the table. In the far left corner, we showed a video player. Below that, a keypad contained buttons for starting and stopping the video.



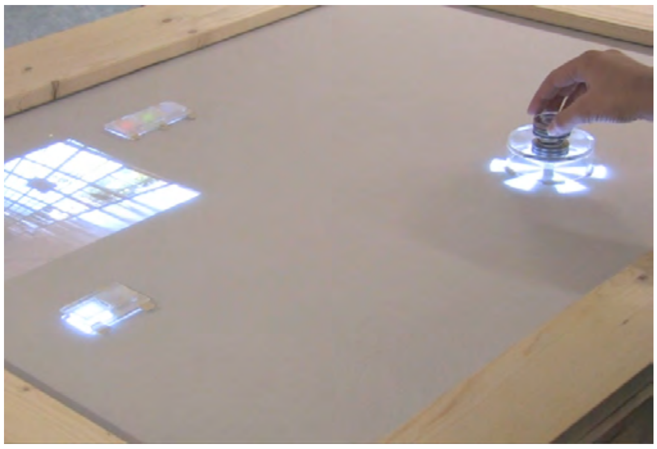


Figure 3.26: Setup of user study on widget performance. Left: GUI design. Modified image from [Weiss et al., 2009b]. Right: Photo from user study.

A jog wheel at the near right side allowed for frame-wise navigation through the video: Turning the knob clockwise moved forward in the video, counter-clockwise turns moved backward. A full 360° turn of the jog wheel skipped about one second (24 frames) of video footage. Note that in this setting the participant could not focus on the video and the knob simultaneously.

Jog wheel was used for frame-wise navigation.

Right to the video, the participant found a keypad containing a red, green, and blue button. These are buttons used to "annotate" the currently selected target frame.

3.8.1.2 Experimental Design

We tested two conditions (independent variable) that differ only by the presence or absence of SLAP Widgets:

Two conditions: SLAP Widgets vs. virtual on-screen controls.

- Condition "SLAP": The user interacts with physical SLAP Widgets. The keypads are represented by SLAP Keypads that involve two (start/stop) or three (annotation) buttons. A SLAP Knob acts as jog wheel for frame wise navigation through the video.
- Condition "Virtual": The physical controls are replaced by their virtual counterparts. While the graphics of the control equals the back projection of the SLAP Widgets, all widgets are now controlled by direct finger manipulation. That is, keypad buttons are triggered by tapping. The SLAP Knob is implemented like conventional GUI knobs: Users touch the control and turn it by dragging. The dragging continues until the finger is released, even if it is moved outside the knob's area (Fig. 3.27).

Each participant conducted four trials per condition, in which each condition contained three instances of a video frame that had to be annotated.

Within-subject design

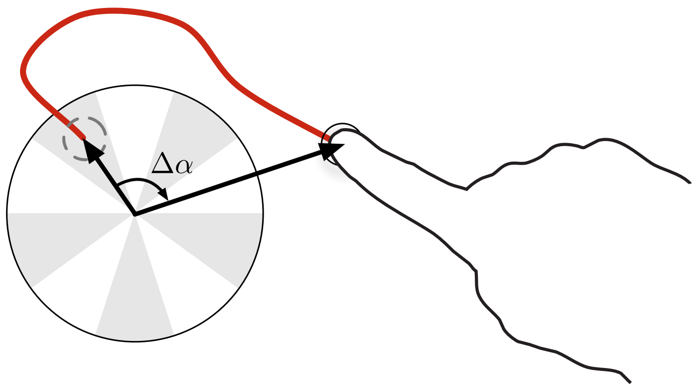


Figure 3.27: Implementation of virtual knob in widget performance user test. The signed angle between the vectors from the knob's center to the starting and ending point of the trajectory, respectively, yields the turn offset $\Delta\alpha$. The knob also works if the finger is dragged outside the control's area.

	Video content	**Length** [s]	**Target frames** [s]		
			red	green	blue
1	Basketball game	47	9	24	33
2	Billard tricks	29	8	15	25
3	City traffic scene	23	6	9	14
4	Person demonstrating on street	21	3	10	16
5	Person sorting letter blocks	22	4	11	17
6	City intersection scene 1	41	4	17	33
7	City intersection scene 2	23	4	10	18
8	Person climbing	30	6	13	22

Table 3.3: Test videos with time stamps in seconds of inserted target frames.

Therefore, 24 video frames were annotated per participant in total. Every participant performed both conditions (within-subject study), but we randomized the order of conditions to address potential learning effects. We randomly distributed eight videos to all trials, four for each condition. Table 3.3 lists the test videos and the time stamps of the target frames. Note that we varied the positions of the target frames to reduce learning effects.

In each instance, we measured three dependent variables:

- *Knob interaction time:* The time interval between the first knob turn after pausing the video to the last knob turn before correctly annotating the frame.
- *Time to push annotation button:* The time interval between the last knob interaction for finding the frame and pushing the annotation button.

	SLAP		Virtual		
Dependent variable	**Mean**	**SD**	**Mean**	**SD**	**N**
Knob interaction time [s]	4.463	1.480	6.697	3.076	22
Time to push annotation button [s]	1.538	0.224	1.419	0.179	22
Number of overshoots	2.094	0.463	3.117	0.831	22

Table 3.4: Results of widget performance study.

- *Number of overshoots:* The number of times the target frame was selected but left again, between pausing the video and annotating the frame. This value is 0 if the target was selected directly.

Regarding our dependent variables, we hypothesized:

- *H1:* Using the SLAP Knob for video navigation is faster than using the virtual counterpart.
- *H2:* The time to push the annotation buttons is shorter with a SLAP Keypad than with a virtual version.
- *H3:* The SLAP Knob produces less navigational overshoots than on-screen controls.

Before the first trial, participants were introduced to the multi-touch table and instructed about the task.

3.8.1.3 Participants

We tested 22 volunteer participants (2 female), between 22 and 36 years old ($M = 25.7$, $SD = 3.4$). 19 participants were right-handed, three left-handed. Nobody reported a color vision deficiency. We recruited all participants from our university campus using a posting at the cafeteria, and during public demonstrations of our multi-touch table.

3.8.1.4 Results

Time values were transformed with logarithm to achieve normality.

The results are shown in Table 3.4 and Fig. 3.28. The results of our time measurements where found to violate normality according to Shapiro-Wilk test but were sufficiently normal after applying the logarithmic transformation $y' = \log_e(y)$ (Table 3.5). We conducted two-tailed paired t-tests to test our hypotheses (Table 3.6).

Knob interaction time & overshoots: SLAP < Virtual

Fine navigation to the target frame via the knob was on average 2.234 s faster in the "SLAP" condition than in "Virtual" condition (mean 4.463 s vs. 6.697 s). Also, participants made about one overshoot less when navigating

Dependent variable	SLAP W	SLAP p	Virtual W	Virtual p
Knob interaction time	0.97773	.87707	0.84518	.00279*
Time to push annotation button	0.95563	.40626	0.89514	.02380*
Number of overshoots	0.98788	.99168	0.92695	.10598
$\log_e$(Knob interaction time)	0.97690	.86129	0.91744	.06731
$\log_e$(Time to push annotation button)	0.96602	.61946	0.93044	.12536

Table 3.5: Results of Shapiro-Wilk tests on measured data and logarithmic transformations. Asterisks denote significant results.

Dependent variable	Transformation	T	df	p
Knob interaction time	$\log_e(y)$	-4.2588	21	$.00035 < .001$
Time to push annotation button	$\log_e(y)$	2.4266	21	$.02433 < .05$
Number of overshoots	-	-4.6296	21	$.00014 < .001$

Table 3.6: Results of two-tailed paired t-tests of widget performance study.

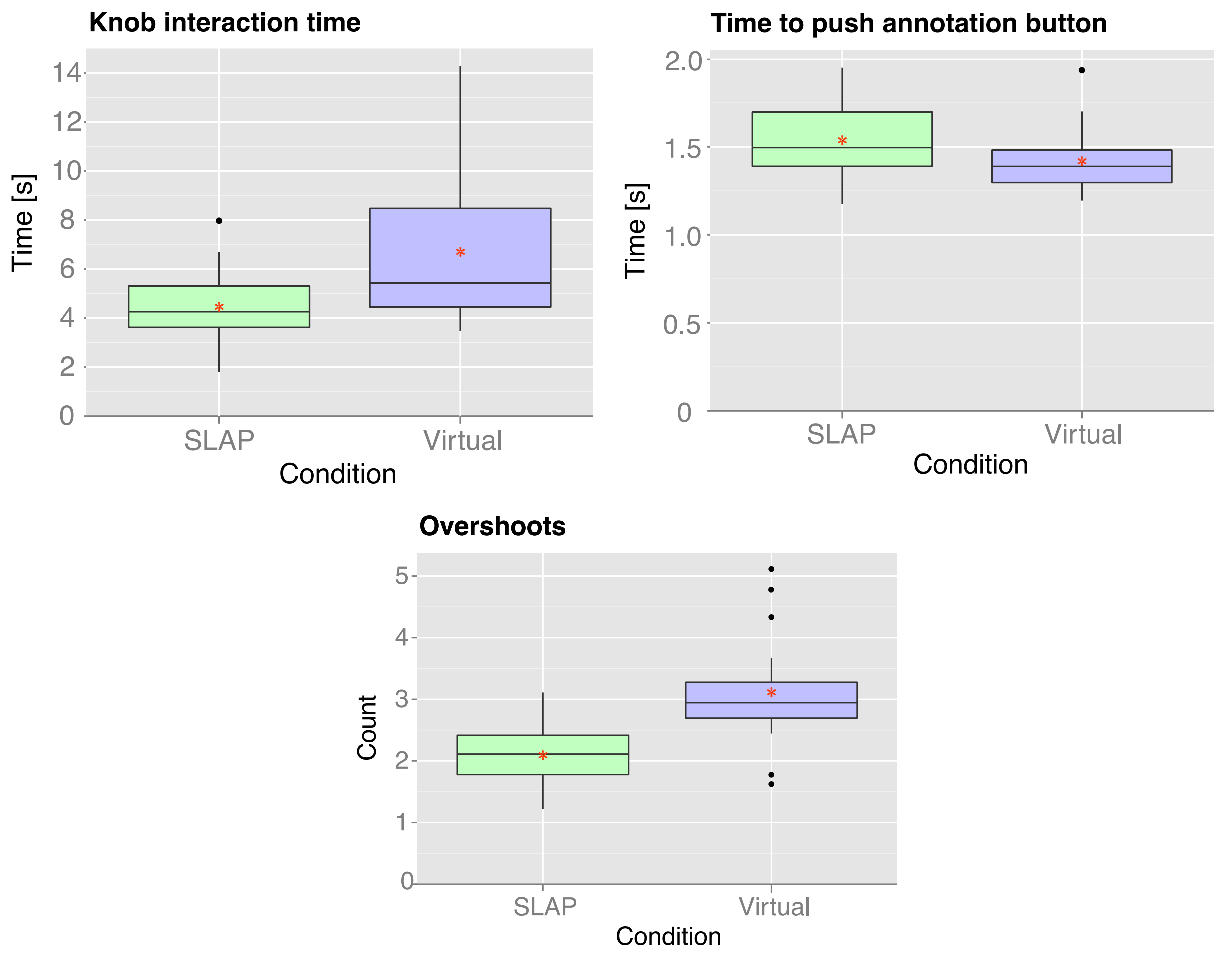


Figure 3.28: Results of widget performance study. In addition to the box plots, red asterisks denote mean values.

to the target frame when using a SLAP Knob instead of the virtual control (mean 2.094 vs. 3.117). Both results are significant, supporting $H1$ and $H3$. However, participants spent significantly more time for pushing the annotation button after fine navigation with the SLAP Keypad than with pure on-screen controls (mean 1.538 s vs 1.419 s). This rejects $H2$.

Time to push annotation button: SLAP > Virtual

3.8.1.5 Discussion

Our results show that SLAP Widgets can decrease interaction times and improve input accuracy. We believe that the reason for this effect is that SLAP Widgets provide haptic feedback and, thereby, a rest state. Once users have placed their hands on the SLAP Knob, they can focus on the video player and navigate through the footage without looking at the control. The physical knob guides the users' motion while they turn it. The visual and haptic channel are completely separated in the "SLAP" condition. In the "Virtual" condition, however, users easily drift away from the on-screen knob when operating it without looking. The only haptic feedback provided is the touch of one finger on the planar surface and the shearing forces during dragging. The virtual knob cannot physically guide the users' motion. This occasionally requires them to look at the control and to realign their fingers, which yields a longer interaction time. Also, finding an on-screen control that is out of sight demands visual attention. Contrariwise, a physical device can be located using the haptic sense only. When using on-screen controls, the visual channel is busy with two tasks at different locations, interpreting the current state of the video and maintaining the interaction with the knob.

SLAP Widget can decrease interaction times and improve input accuracy in eyes-free tasks.

Pure on-screen controls can yield drifting if not visually focussed.

Homing on SLAP Widgets is possible without looking.

Albeit the difference of 0.119 s is small, the time interval between knob interaction to pressing the annotation button was significantly longer with the physical SLAP Keypad. There are two potential reasons for this: First, in the SLAP condition, participants have to change the input modality from turning a physical knob to pushing down a silicone button. In the "Virtual" condition, both actions are triggered by *directly* touching the surface, which probably leads to a shorter movement trajectory. Second, there is no concurrent visual task in this time interval. That is, users can trigger the buttons while looking at them. For this simple task acquisition task, the haptic feedback of SLAP Widget does not provide any benefit in terms of accuracy or task completion time.

Haptics of SLAP Widgets like buttons do not provide performance benefits when visually focussed.

Some participants reported that interaction with the virtual knob became inconvenient after some time. The reason is that the natural lubrication of the finger diminishes while sliding on the surface. After a while, the fingertip becomes warmer due to the additional friction and even might create a squeaky sound.

Sliding on surface is inconvenient over time.

We found that users tend to drift when using a virtual knob in an eyes-free manner. This drifting could be reduced by applying the technique by

Schraefel et al. [2005], which lets the knob's center follow the user's drifting direction.

3.8.2 Qualitative Evaluation

With our second study, we intended to gather qualitative feedback on the interaction with SLAP Widgets, the pairing mechanism, and the overall user experience.

3.8.2.1 Procedure

The user study was conducted at the same multi-touch table as the previous study. In the beginning of the session, the instructor presented the SLAP Widgets to the participants. Then, every participant was asked to play with the controls and to reflect about possible uses of SLAP Widgets. The widget detection was disabled, and no further explanation was given. We intended to elicit existing associations with real-world controls that could be transferred to SLAP applications.

Participants performed series of pairing and interaction tasks and gave qualitative feedback.

After that, the instructor briefly demonstrated how a SLAP Widget is detected by the table and how it is paired with a virtual object using synchronous double-tapping. After that, the tabletop showed three virtual objects: an image, a movie object, and a text box. Then the participant, standing at the long side of the table, was asked to execute a series of tasks that involved interaction with SLAP Widgets and pairing of controls for different purposes:

1. *Play/stop video:* Place a SLAP Keypad on the tabletop, pair it with video player, and start and stop the video using the respective keypad's buttons.
2. *Navigate within video:* Take a SLAP Slider and a SLAP Knob and pair it with the video player. Use the slider to set the player to the middle of the video. Then, use the knob (in jog wheel mode) to navigate to a specific point in time (e.g., "now navigate to 1:24").
3. *Image editing:* Unpair SLAP Slider and SLAP Knob from video player and associate it with the image object. Use the slider to change the brightness of an image to 50% and adjust the saturation to 50% using the knob, which is now in menu mode.
4. *Labeling:* Pair a SLAP Keyboard with the text field and type your name. After that, reassociate the knob with the text field and select a new color for the text from the menu.

All SLAP Widgets that were required for this test were placed within reach of the participants on the border of the tabletop. Participants could freely report their thoughts during the test. A video camera captured the participants and the tabletop for later evaluation. The test ended with an interview.

3.8.2.2 Participants

10 participants, who were experienced with GUIs

10 participants (3 female) between 21 and 28 years old ($M = 23.1$, $SD = 3.3$) volunteered for our user study. We recruited them from our department and the university campus. All participants stated to use computers every day and to be experienced with desktop GUIs.

3.8.2.3 Results

Participants reported that interaction with SLAP Widgets was intuitive and self-evident. Pairings were easily understood.

Most participants (9) of our study found the interaction with SLAP Widgets intuitive and self-evident. The association between widgets and on-screen objects was easily understood and could be applied by all participants. Some of them (4) proposed alternative pairing gestures, such as placing a widget on the target virtual object and then sliding it to a position where it does not cause occlusion (“grasp and drag”). However, synchronous double-tapping was reported as especially suitable for the SLAP Keyboard. One person stated that SLAP Widgets provide a good mapping from real-world physical controls to their on-screen representation, which could help users who are not familiar with conventional GUI controls. Another participant emphasized that, in contrast to virtual controls, SLAP Widgets provide a rest state; users could rest their hands on the controls without operating them (cf. section 1.1). Some participants (4) stated that SLAP Widgets are quiet, and the addition of auditive feedback could be helpful, especially for keystrokes on the SLAP Keyboard. Feedback on the SLAP Keyboard was mixed. While the idea of a flexible, back projected keyboard was appreciated, most participants felt more comfortable with the on-screen keyboard.

3.8.2.4 Discussion

The participants’ feedback on SLAP Widgets was generally positive. They quickly understood the concept and were able to complete basic tasks. The qualitative feedback was helpful for all further iterations of our physical widgets.

Limitation of study: Participants were inexperienced with multi-touch interfaces.

One limitation of this study was that all participants were not experienced with multi-touch interfaces. At the time of the study, in September 2008, multi-touch smart phones or tablets were not prevalent yet. Many users had never experienced direct finger manipulation on interactive surfaces. Hence, many users were fascinated by this feature and commented that they did not understand why further physical widgets are required for interaction. In a

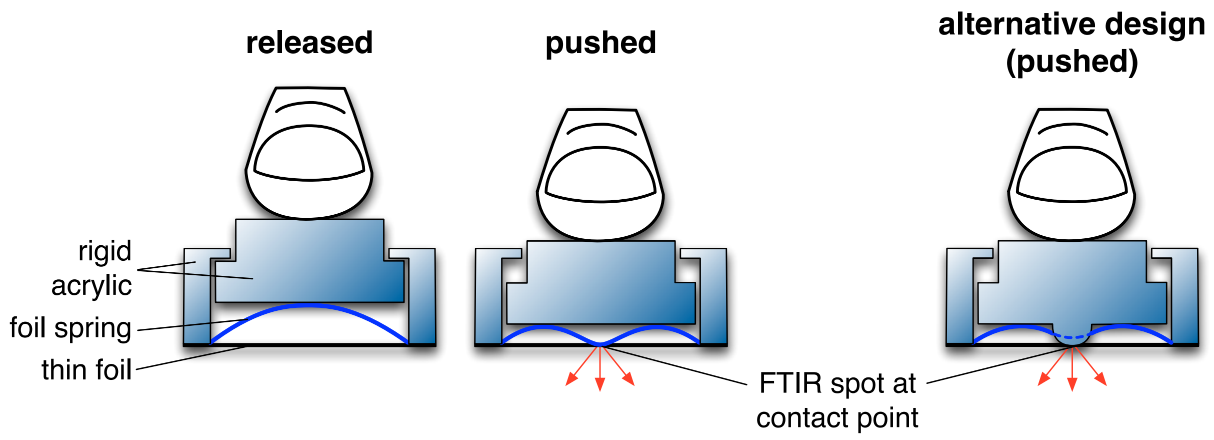


Figure 3.29: Key design of rigid SLAP Keyboard. A spring foil holds each key up. When the key is pushed, the foil collapses and creates an FTIR spot in the camera image. An alternative design contains a knob beneath each key that penetrates the foil and hits the surface at every keystroke.

future study, the qualitative feedback should be retrieved in two conditions, involving only pure virtual or physical controls, respectively.

Mixed feedback on SLAP Keyboard due to insufficient pressure point and detection errors

The idea of having a keyboard that can just be "slapped" on the table for ad hoc use was appreciated by many participants. However, most participants preferred the on-screen keyboard over the physical SLAP Keyboard. One reason was that our detection based on DI occasionally produced false positives and repetitive key strokes, leading to input errors. Furthermore, the haptic feedback of the SLAP Keyboard was still not good enough to compete with the conventional physical keyboard. Some participants stated that the edges of keys and the keycaps could be improved. Also the pressure point of keys was not clear enough; some participants reported that it was difficult to determine how keys had to be pressed. We will deal with this issue in the next section.

3.8.3 Typing

In our third user study, we have a closer look at the SLAP Keyboard. We first present a new keyboard design that incorporates previous qualitative feedback. In a controlled experiment, we then evaluate how efficient participants can type using SLAP Keyboards in comparison to conventional ones and pure on-screen versions.

3.8.3.1 Rigid SLAP Keyboard with Pressure Point

Rigid SLAP Keyboard is made of acrylic and transparent foils.

Participants of our qualitative study reported that the pressure point of the flexible SLAP Keyboard was unclear. To address this problem, we developed a second SLAP Keyboard. The keyboard is rigid and entirely made of transparent acrylic. Each key is spring-loaded on a thin foil. The

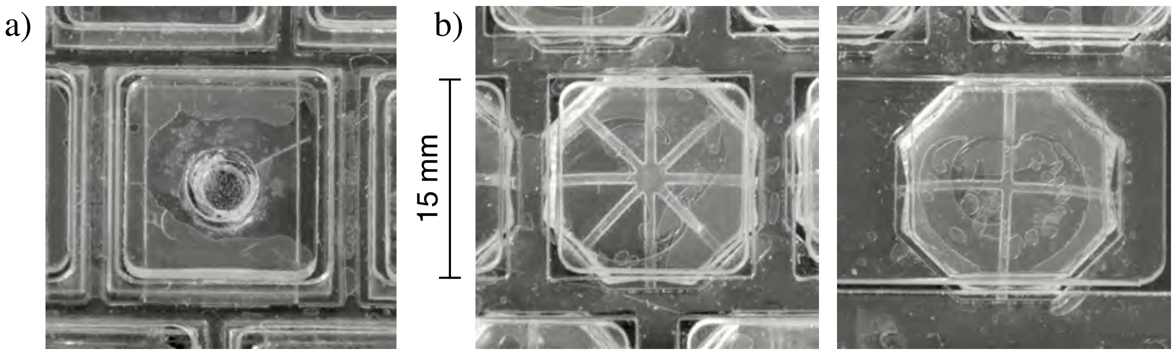


Figure 3.30: Different prototypes of foil springs beneath keys. a) Prototype with leaf spring that is penetrated by knob. When pressed, the knob touches the surface and creates an FTIR spot. b) Prototypes with stellar and cross spring, respectively. When pressed, the spring foil collapses and creates a spot. The stellar form of the spring ensures that a spot appears even if a key is pushed at its boundary. Multiple cross springs are combined for larger keys like Space.

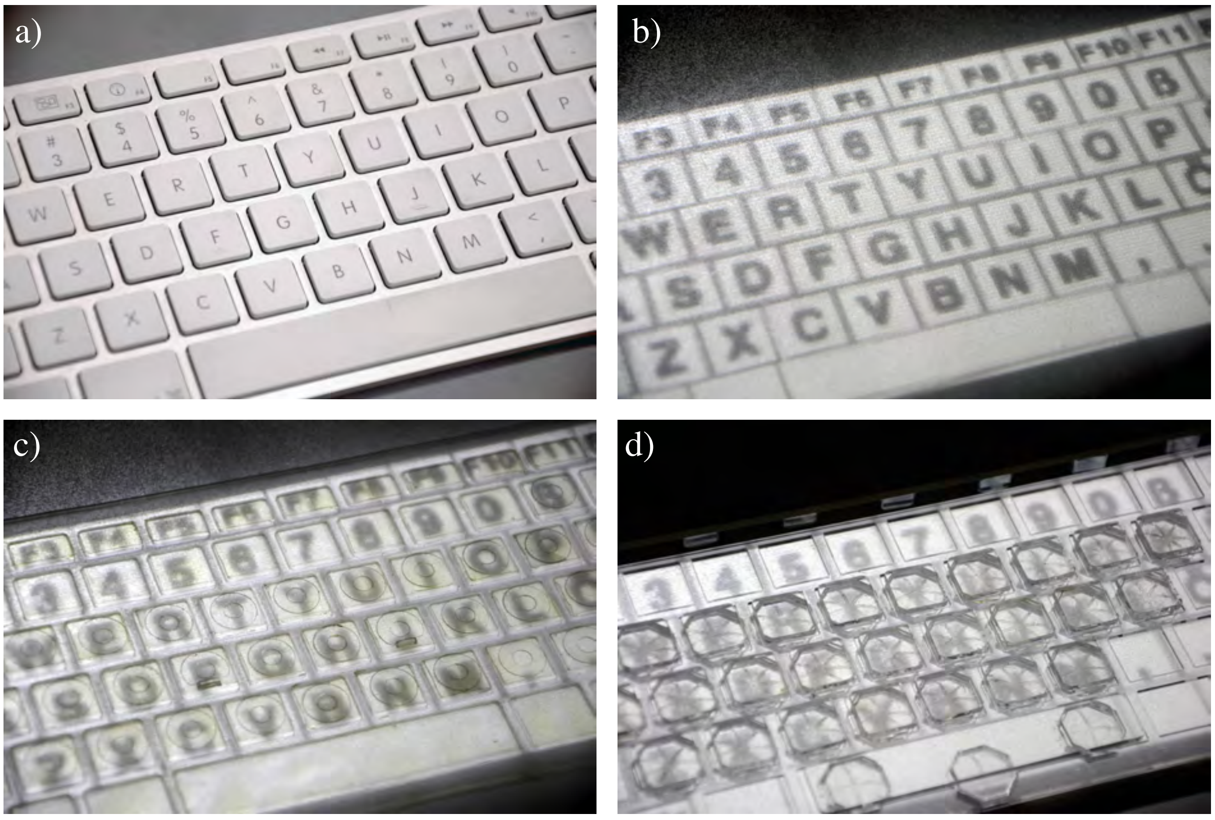


Figure 3.31: Different keyboards in typing performance study. a) Conventional Apple keyboard. Image brightened for better perception. b) On-screen keyboard. c) Flexible SLAP Keyboard. d) Rigid SLAP Keyboard.

foil raises a rigid key body and lets it hover above the surface. If a user presses the button, the spring is compressed and collapses beyond a certain pressure-point. The collapsing spring creates a noticeable “click” sound. It also presses the bottom foil against the surface and creates a short FTIR spot in the camera image (Fig. 3.29). An alternative design uses a small knob that is attached to the bottom of each key and penetrates the foil.

Foil springs provide an improved pressure point and auditive click feedback.

When the key is pressed, the knob touches the bottom foil and, thereby, the surface, creating a spot.

Stellar foil springs allow detection of keystrokes at boundaries.

The spring design is crucial to ensure a convenient haptic feedback. Our original design is shown in Fig. 3.30a. We used a leaf foil penetrated by a cylindric knob. The keys created a bright spot when being pushed down in the center. However, when pressed at the boundary, the foil did not collapse properly. Also, the keys varied in haptic feedback and tended to jam. The result of further iterations is a button with an embedded stellar foil button (Fig. 3.30b). It works well when being pushed down at the boundaries. Keys with rounded corners avoid jamming. Large keys like Space require multiple springs. To ensure that a similar force is required to push those, cross springs with lower resistance are employed. We also improved the manufacturing process to ensure nearly the same pressure point for all keys. Our prototype containing keys A to Z as well as Space, Enter, and Backspace is shown in Fig. 3.31d.

We believe that the rigid keyboard provides a better haptic sensation. Note though that this costs the flexibility of the silicone design of the original version.

3.8.3.2 Task

We asked participants to subsequently copy presented strings using each of the following four keyboards (Fig. 3.31):

1. Conventional keyboard: an off-the-shelf tethered Apple keyboard.
2. On-screen keyboard: a back projected virtual keyboard.
3. Flexible SLAP Keyboard: the original SLAP Keyboard based on a silicone protection layer (section 3.3.3.1).
4. Rigid SLAP Keyboard: the rigid keyboard with improved pressure point as described in the previous section.

We instructed all participants to type all strings *as fast and as accurate* as possible.

3.8.3.3 Test Setup

User study takes place at curved BendDesk table using vertical surface to display strings.

Our test setup is shown in Fig. 3.32. We used the BendDesk table as infrastructure that was described in section 2.3. It consists of three interactive areas that are merged together: a vertical board (100 cm × 43 cm), a horizontal tabletop (100 cm × 40 cm) and a connecting curve (100 cm × 16 cm, radius 10 cm). A 8.5 cm non-interactive raised rim in front of the horizontal surface allows to rest hands. The tabletop is mounted in an ergonomic

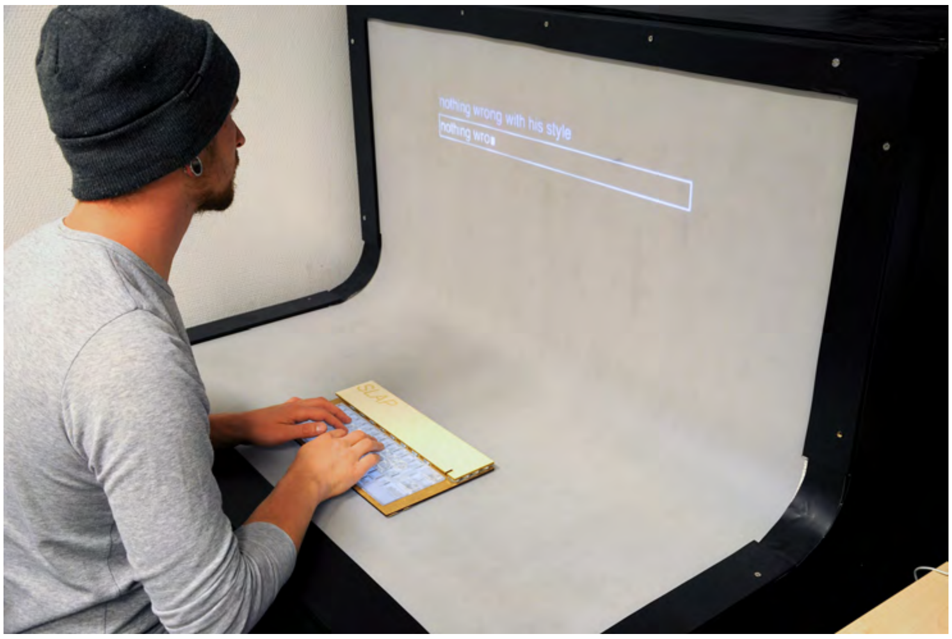


Figure 3.32: Test setup of keyboard performance test. Participants copy strings that are shown on the vertical surface. Room lighting is brightened up in the photo for better illustration.

height of 72 cm. We refer to [Weiss et al., 2010c] for a detailed description of the table. We chose this particular setup, because it mimics a common desk situation: Participants can sit at the table and can interact with a horizontal and a vertical surface.

Setup encourages user to avoid visual focus switches.

The test took place in a dim room with constant lighting conditions. Participants were sitting at the BendDesk table during the entire test. We chose a setup with two foci of attention: Following Bi et al. [2011], the keyboard of the current condition is centered in front of the participant, 17 cm away from the edge of the table. The string that participants had to enter was displayed on the vertical surface in the height of the eyes. An edit field beneath this string displayed the currently entered string. Due to the distance between input and output, we assumed that this setup would penalize focus switches between the keyboard and the output in terms of task completion time. We chose this design to encourage users to look at the screen and rely on haptic feedback.

QWERTY layout with Space, Enter, and Backspace

All keyboards showed a QWERTY layout. Users could press the keys “A” to “Z”, Space, Enter for confirming input, and Backspace for deleting the last entered character. To repeat letters, the respective keys had to be pushed multiple times, i.e., holding keys did not repeat letter input. The size and layout of keys was kept constant among all conditions.

English test strings are taken from standard set.

The strings in the test were chosen from the phrase set by MacKenzie and Soukoreff [2003]. It contains 500 English sentences with a length of 16 to 43 characters (average length is 28.6). Every sentence contained letters

"A" to "Z" and spaces. Users were instructed to enter sentences in lowercase although the test set contained very few capital letters (e.g., "I" or "Saturn").

SLAP Keyboards are fixed with weights.

To avoid accidental moving, the SLAP Keyboards were fixed using additional weights. These weights were attached in a way that they did not disturb typing or the perception of key labels.

FTIR tracking via single camera in table.

Detecting Keystrokes As opposed to the SLAP Table, BendDesk only uses FTIR to track touches. For highest tracking performance, we mounted a single camera in the table to capture a close-up of the bottom of the keyboard. The camera pointed to the surface in a steep angle to avoid that the participants' legs occluded the camera image (see bottom camera in Fig. 2.12 on page 27). We used a homography as described in section 3.5.1.1 on page 56 to compensate for the radial and perspective distortion.

Spots in key boundaries generate spot events. SLAP Keyboards discard areas outside key centers to avoid false detections.

Keystrokes of the on-screen and back projected keyboards were detected by checking for touch events within the rectangles beneath keys (section 3.3.3.1 on page 45). In the absence of DI, the tracking of the SLAP Keyboards had to be modified. A keystroke on these keyboards also pushes the key boundaries into the surface, producing spot events that potentially merge with spots in the keys' center and causing false positives. To avoid this, we masked the background frame of the tracker so that only a small pixel area beneath the center of each key was considered in the spot detection.

Delay is required between repetitive keystrokes.

To avoid false positives due to camera noise, we considered two spot events within a key's boundary as repetitive if they occurred at least with a delay of 150 ms. Otherwise, the second spot event was ignored.

3.8.3.4 Procedure

Procedure: Participants type test strings using all four keyboards.

After reception, participants sat down in front of BendDesk. After that, we informed them about the procedure of the user study in detail. We also instructed them that time is measured from the first keystroke in each trial. Accordingly, they had time to read each sentence before starting to type. We also acquired basic background information.

Participants subsequently conducted all four keyboard conditions (independent variable), while we randomized the order. Every condition contained 2 training trials that allowed subjects to become familiar with each keyboard and to learn how much pressure was required for a keystroke. After that, participants conducted 15 measured trials in which we measured the typing speed and accuracy. In each trial, a test string was displayed, and participants typed this string in the edit field. Once done, they confirmed their input by pressing Enter, which finished the trial. Note that a trial was over even if the entered string contained typing mistakes.

As dependent variables, we used standard metrices according to Wobbrock [2007]:

- *Words per minute (WPM)* defined as
$$\text{WPM} \;:=\; \frac{|T_{\text{entered}}| - 1}{S} \cdot 60 \cdot \frac{1}{5}$$
where T_{entered} is the entered string with $|\cdot|$ representing the number of characters. S is the duration in seconds from the first to the last letter in this trial.

- *Total error rate (TER)* is a ratio defined as
$$\text{TER} \;:=\; \frac{\text{IF+ INF}}{\text{C+ IF+ INF}}. \tag{3.1}$$
Let T_{entered} and T_{correct} be the entered and the expected correct string. Also, let $\text{MSD}(\cdot,\cdot)$ be the function computing the Levenshtein distance between both strings, i.e., the minimum number of insertion, deletion, and substitutions operations to transform one string into another. The variables in the total error rate are defined as
$$\begin{aligned} \text{C} &:= \max\{|T_{\text{entered}}|, |T_{\text{correct}}|\} - \text{MSD}(T_{\text{entered}}, T_{\text{correct}}), \\ \text{IF} &:= \text{number of backspaces}, \\ \text{INF} &:= \text{MSD}(T_{\text{entered}}, T_{\text{correct}}). \end{aligned}$$
Informally speaking, the three variables decompose the entered string into three disjunct sets: the correct characters (C), the corrected errors (IF), and the uncorrected errors (INF).

- *Adjusted words per minute (WPM_{adj})* penalizes typing mistakes in the WPM value
$$\text{WPM}_{\text{adj}} \;:=\; \text{WPM} \cdot (1 - \text{TER})^a$$
where we set a to 1.

Users were encouraged to give qualitative feedback between measurements. After the test, we also asked them to order all four keyboards according to the (subjective) typing efficieny that they enabled, starting with the fastest keyboard.

Our main hypotheses were:

Hypotheses for task completion time and typing correctness:

Conventional
$\gg$ Rigid SLAP
$>$ Flexible SLAP
$>$ On-screen

- *H1*: The conventional keyboard significantly outperforms all other keyboards in terms of speed and error rate.

- *H2*: The two SLAP Keyboards outperform the on-screen keyboard in terms of speed and error rate.

- *H3*: The rigid SLAP Keyboard enables a higher typing speed than the flexible version.

An average user test lasted about 21 minutes. In total, we measured and analyzed the input of 600 phrases typed with 21,171 keystrokes.

Dependent variable	Conventional		On-screen		Flexible SLAP		Rigid SLAP		
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	N
WPM	59.2	18.6	37.8	15.7	33.4	12.3	32.1	13.5	150
WPM_{adj}	58.8	18.5	36.7	14.9	32.6	11.8	31.3	12.8	150
TER[%]	7.1	7.5	10.7	9.4	12.6	9.0	12.8	8.3	150

Table 3.7: Results of keyboard performance study.

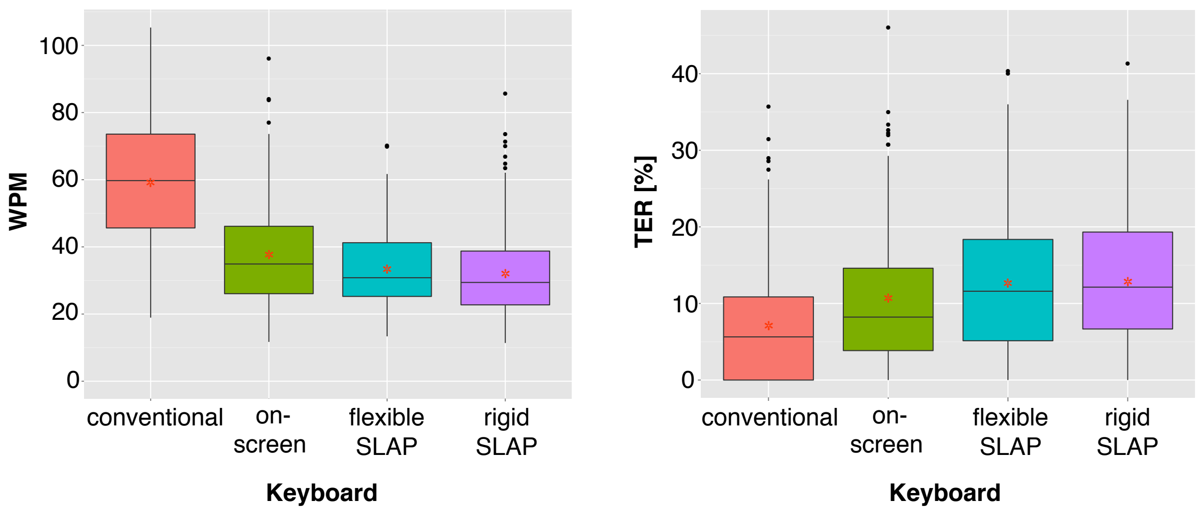


Figure 3.33: Words per minute (WPM) and total error rate (TER) depending on keyboard. In addition to the box plots, red asterisks denote mean values.

3.8.3.5 Participants

10 participants, who were experienced with typing

We tested 10 participants between 23 and 31 years old ($M = 26.0$, $SD = 2.4$), 1 was female. They were students recruited from our department, and all were experienced with typing, using a keyboard "multiple times a day". On a scale from 1 to 10, where 10 is the highest skill in typing, participants assessed their keyboard skills with 6.8 on average ($SD = 0.8$). We intentionally selected experts, because haptic feedback is more relevant if users interact in an eyes-free manner. Half of the participants (5) had formerly learned touch typing with ten fingers. All subjects were non-native English speakers. After the test, we raffled a 10 Euro gift coupon among the participants.

3.8.3.6 Quantitative Results

Results: Conventional $\gg$ On-screen $\not<$ Flexible SLAP $\approx$ Rigid SLAP

Our results are shown in Table 3.7 and Fig. 3.33. Fig. 3.34 shows the average words per minute and total error rate depending on time. The penalized adjusted words per minute values do not significantly differ from the WPM values and will not be discussed in the following.

Mixed-effects model analysis of variance shows main effect of condition in terms of WPM ($F_{3,531} = 150.65$, $p < .0001$) and TER ($F_{3,531} = 15.41$,

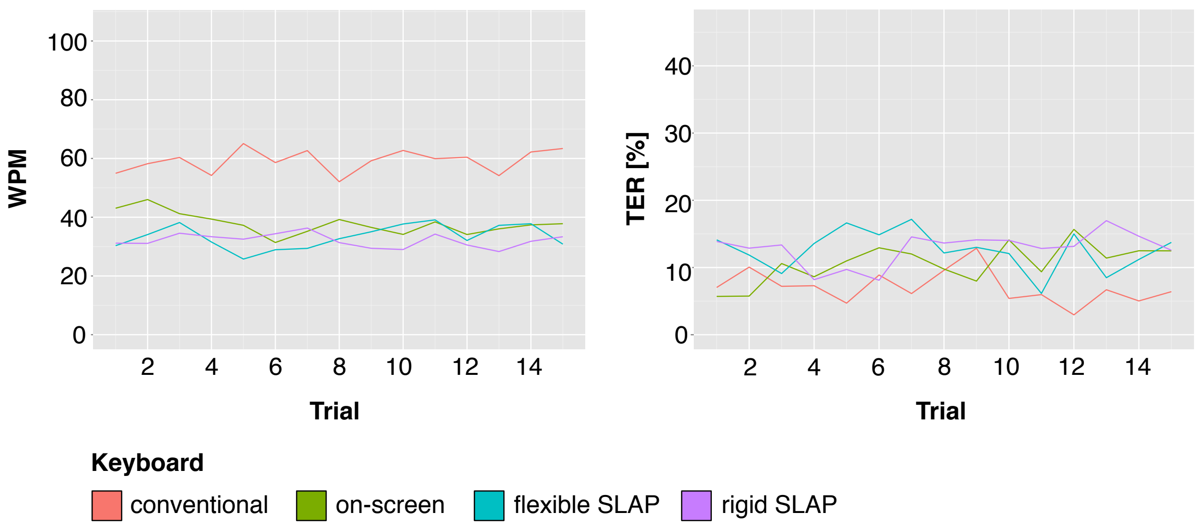


Figure 3.34: Average words per minute (WPM) and total error rate (TER) over trials depending on keyboard.

Condition pair	p**(WPM)**	p**(TER)**
conventional vs. rigid SLAP	<.0001*	<.0001*
conventional vs. flexible SLAP	<.0001*	<.0001*
conventional vs. on-screen	<.0001*	.0019*
on-screen vs. rigid SLAP	.0067*	.1237
on-screen vs. flexible SLAP	.0580	.2020
rigid vs. flexible SLAP	.8847	.9954

Table 3.8: Results of pairwise Tukey-Kramer HSD test. Asterisks denote significant results.

$p < .0001$). There was no significant main effect in trial and no significant interaction between trial and condition for WMP and TER. Participant was modeled as nominal random effect.

The results of a pairwise comparison using Tukey-Kramer HSD are shown in Table 3.8. The conventional keyboard outperforms all other keyboards in terms of words per minute and total error rate (supports H1). The differences among the virtual and the SLAP Keyboards are not significant, with the exception of the on-screen keyboard yielding a significantly higher typing speed than the rigid SLAP Keyboard (rejects $H2$ and $H3$).

3.8.3.7 Qualitative Feedback

Despite lack of haptic feedback, on-screen keyboard was ranked on second place, after conventional one.

Nearly all participants (9) considered the conventional keyboard as the most efficient one. Despite the limited haptic feedback, most subjects (6) placed the on-screen keyboard on the second rank. Two participants perceived the detection of keystrokes as more accurate as on the SLAP Keyboards. However, two different participants criticized that quickly repeating letters was difficult on the on-screen version. This was a result of our filter requiring 150 ms beneath repetitive keystrokes. Two participants emphasized the need to look at the keys while typing. This also matches our impression that some participants memorized strings before typing them with focus on the keyboard.

Mixed feedback on SLAP Keyboard. Tracking accuracy and missing orientation points were critized.

The SLAP Keyboards yielded mixed feedback. While the haptic feedback was appreciated, specific design issues impaired the user experience. Five participants stated that the labels of the SLAP Keyboards were difficult to read, which seemed to be even worse on the rigid version (2). Four of the five subjects who had learned touch typing before, criticized the missing orientation marks on the F and the J key on the rigid SLAP Keyboard. They also said that the lack of keys to the right of P and L made it hard to orientate the hands. For both keyboards, some participants (4-6) complained about false positives and false negatives in the keystroke detection. The preference of participants between the two SLAP Keyboards was mixed, and there is no clear favorite in terms of haptic feedback. One subject liked the definite pressure point of the rigid keyboard, while another one appreciated that the flexible version feels “smooth” and not so binary. Two participants found a “typewriter technique” with short, strong impulses to be most efficient for writing on the rigid SLAP Keyboard, while noting that these cause more exertion over time.

3.8.3.8 Discussion

Conventional keyboard performs best, because it is a well optimized device.

It is no surprise that the conventional keyboard outperformed all other versions (H1). Striking the key of a conventional keyboard means pressing on many decades of research and optimization. The haptic feedback of every key is iterated many times before a keyboard is released. Also, all users were well familiarized with this keyboard type.

SLAP Keyboard did not outperform on-screen keyboard but differences are small. Engineering issues should be solved.

Opposed to our hypothesis H2, the SLAP Keyboard did not outperform the on-screen keyboard. There is also a tendency that writing on on-screen keyboards is still faster than writing on our physical prototypes. However, the difference is small and only significant between the on-screen and the rigid keyboard. Given that a SLAP Keyboard can be easily handed over to other users, it remains a promising alternative to pure virtual keyboards. Most of the problems reported by participants are engineering issues: Tracking must be improved, and the keys should be produced in such a way that they are more translucent. The resolution of the BendDesk system was another limitation worsening the readability of the key labels.

The result that the SLAP Keyboards did not differ significantly in terms of WPM and TER was unexpected. Although the rigid SLAP Keyboard provides a clear pressure point, it does not outperform the flexible version. One reason could be the stiffness of the keys and the additional force required for typing. This should be improved in a future iteration. Also, the missing keys and orientation marks should be added. And, again, an improved tracking could reduce false detections, and maybe the difference due to haptics would be more significant.

3.8.3.9 Future Study Design

Four main factors influence the differences in input performance in this study:

- The haptic feedback of each keyboard
- The accuracy of keystroke detection
- The cognitive load due to focus switches between keyboard and text display
- The experience of the user to type on conventional and on-screen keyboards

Future study should isolate effects of haptic feedback, tracking accuracy, focus switches, and user experience.

In a future study, these variables should be isolated to understand their individual effects. A precise ground truth tracking method, e.g., a Vicon optical tracking system, should be used to isolate the impact of haptic feedback and to evaluate the accuracy of our spot-based tracking. Our study did not reveal how well participants type eyes-free with each keyboard. We observed a tendency of participants to look at the keys while using the on-screen keyboard. A gaze tracker should be employed to measure in which condition participants memorize strings and look mostly at the keyboard and in which they rely on haptic feedback. Alternatively, the string set could be replaced with random strings that make memorizing more difficult and require more focus on the input field. A detailed classification of users concerning their previous experiences with conventional and touch screens could reveal to which extent experience is a confounding variable.

Engineering issues should be solved.

The above mentioned engineering issues should be solved for a future study, including some modifications on the table infrastructure: The output resolution should be increased, and the non-interactive rim should be removed because two participants reported fatigue due to the slightly raised wrist.

Future study should test dynamic table setting.

Finally, it is worth testing typing performance in a dynamic interactive table setting involving multiple users, where ad hoc use, realigning, and handing over keyboards play a greater role.

3.9 Closing Remarks

SLAP Widgets are general-purpose controls that combine rich haptic feedback with a flexible visual appearance.

In this chapter, we presented SLAP Widgets, general-purpose tangibles for interactive tabletops. SLAP Widgets combine the rich haptic feedback of physical controls with the visual flexibility of GUI controls. After discussing related work, we explained the table and widget design, the underlying software framework as well as the mechanisms to detect widgets and to communicate their state to an application. We described a basic widget set, potential usage scenarios, and pairing techniques to associate widgets with on-screen objects. The chapter also showed how to extend the framework with further widgets and virtual objects. User studies demonstrated the potential of SLAP Widgets as alternatives to pure on-screen controls in terms of efficiency and learnability.

SLAP Widgets respect the nature of interactive tabletops.

Our design respects the specific dynamic nature of tabletops: SLAP Widgets can be placed on the surface ad hoc and removed again when not needed anymore. They coexist with direct manipulation input without interfering with it. The users' interaction with these controls happens *above* the surface without touching the surface directly. We designed the footprints and pairing gestures in such a way that false positives caused by finger touches are unlikely. That is, the input channel for interacting with SLAP Widgets is independent from normal direct touch input. This is a crucial benefit over dynamically arranging on-screen controls on a tabletop that would require a special modality to distinguish between input for interacting with controls and input for arranging UI elements. Furthermore, SLAP Widgets inherently support natural social protocols for exchanging controls. A physical SLAP Widget can be easily handed to another user. A user can also keep hold of a control to avoid other users grabbing it. Contrariwise, efficiently handing another user a pure virtual control requires special handoff techniques like those presented by Jun et al. [2008].

They coexist with other input methods without interfering with them. SLAP Widgets support social protocols.

SLAP Widgets are low-cost, lightweight, flexible, and easy to build. The underlying technology is hidden.

The strength of SLAP Widgets is their lightweight design. Consisting of materials such as silicone and acrylic, SLAP Widgets are low-cost, lightweight, and easy to build. Damaged parts can easily be replaced. Due to the absence of electronic parts, designers do not need deep engineering knowledge to craft widgets that are both useful and aesthetically pleasing. As passive devices, SLAP Widgets also do not contain cables that could possibly clutter the design. Following the spirit of Ubiquitous Computing [Weiser, 1991], the technology behind SLAP Widgets is hidden from the users.

SLAP Widgets are scalable interfaces.

SLAP Widgets fulfill the demand for scalable controls in several respects: First, SLAP Widgets can change their visual appearance on the fly and, therefore, they can be reconfigured for various purposes. Second, they can change abstract parameters. Therefore, an increase in the complexity of a table application does not necessarily imply an increase in the complexity of the tangibles. The ability to use SLAP Widgets ad hoc and remove them when not required anymore, also alleviates the issue of physical clutter that is often named as a limitation of tangibles (cf. [Shaer and Hornecker, 2010, p. 106]). The SLAP Framework follows this spirit and was implemented to support scalability: The creation and integration of new widgets is straightforward. Also, programming new virtual objects is easy. Thanks to the hierarchical structure, they can also be composed from existing ones. For example, a business card object could be designed as a `SGORect` containing one `SGOImage` and several `SGOText` objects.

Both input and output technology are integrated into the table's hardware *below the surface*. Touches and widgets footprints are seen by the table's camera, and the projector updates the graphics accordingly. Such an optical

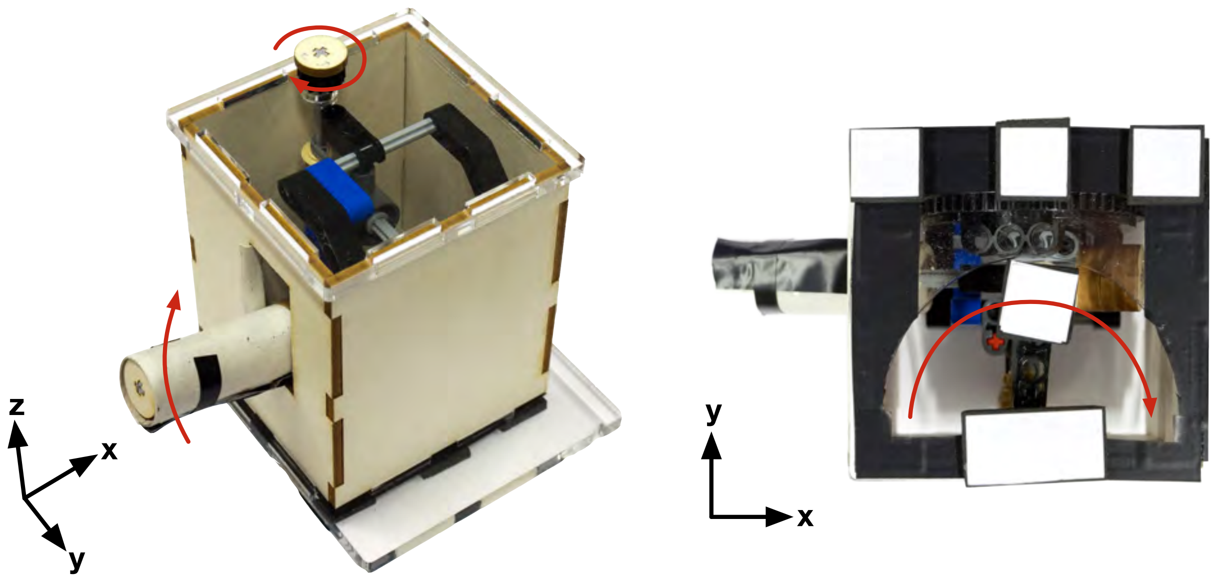


Figure 3.35: Tangible "Toxic Waste Cannon" designed by Norbert Dumont. Left: Physical prototype. Right: View on footprint from below. The tangible maps a rotation in the x/y plane to a rotation in the x/z plane, and communicates the turning angle to the footprint using a rotation in the x/y plane.

Design constraint: State must be communicated via footprint (flattened).

tracking setup is easy to construct and can also be employed to indirectly communicate physical properties, such as temperature or the filling level of liquid in a pot [Dietz and Eidelson, 2009]. On the downside, this method introduces design limitations. A SLAP Widget must be constructed in a way that its current state can be communicated through its footprint. Designers, therefore, must be able to *flatten* the three-dimensional state of a widget into a 2D unambiguous representation that is unique among all widgets. While this is easy for our basic widget set, it becomes more difficult if an object is high or involves mechanics that cannot be mapped directly to a surface position.

Large tangibles might require complex mechanics to communicate states to the footprint.

In order to gain a deeper insight, how designers create complex tabletop tangibles, we conducted a seminar, the "Media Computing Project", at our department in 2010. 11 students were asked to develop a tangible tabletop game of the type "Tower defense". Students were instructed to create a cooperative multiplayer game that combines direct touch interaction with tangibles. The goal of the players is to defend a base by placing defensive towers at the path of the attacking opponents. In the resulting game, the playing field as well as the invaders are projected on the screen. Users can grab tangible towers and place them on the field. Most towers can be modified or triggered by touching adjacent virtual controls. A remarkable tangible developed by the students is the so-called "Toxic Waste Cannon" that is illustrated in Fig. 3.35. A user can turn a physical knob on top of the tower to change the vertical angle of a muzzle. The tangible contains a complex mechanism that maps the horizontal rotation of the knob to a vertical rotation of the muzzle. Simultaneously, it horizontally turns a marker at the bottom of the tangible that communicates the angle to the camera in the table. However, due to the complex mechanics, this tangible

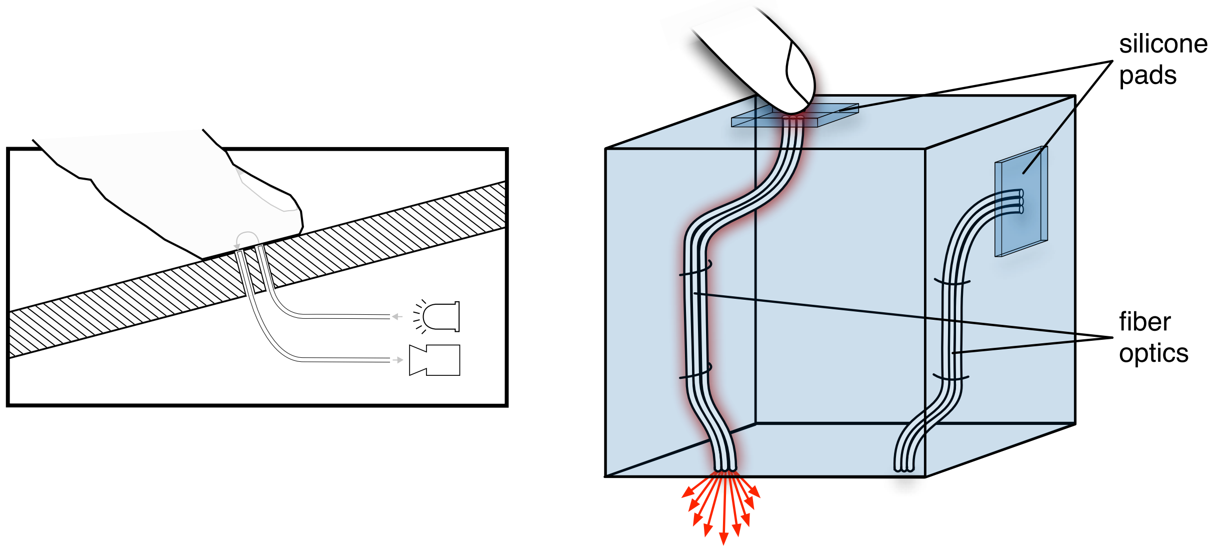


Figure 3.36: Tangibles with embedded fiber optics. Left: Principle of FlyEye. Fiber optics direct IR light to the finger and reflections back to a camera that captures touches. Image courtesy of Wimmer [2010]. Right: Application of this principle to a SLAP Widget. The control communicates the push state of two silicone buttons over a distance. Fiber optical cables connect two silicon pads with the bottom of the tangible. When using DI, IR light emitted from the table surface is injected into the fiber optics and leaves the cables beneath the pads. If the user pushes one of the pads, the IR light is reflected downwards through the fiber optics to the surface. This creates a visible IR spot in the camera. This tracking approach requires a good coupling between table surface and fiber optics.

is opaque. This limits the ability to change the widget's visual appearance via the table's back projection.

Alternative way: Fiber optics transfer marker states to the surface.

A promising alternative to transfer a widget's 3D state down to the tabletop without making it opaque are fiber optics. *FlyEye* by Wimmer [2010] embeds fiber optics into a tangible to communicate touches from the surface into a camera (Fig. 3.36, left). IR light is fed into a bundle of fiber optics that lead to the surface. When the user touches the surface, the IR light is reflected back into an additional set of fibers that direct to a camera. The camera captures the end of the fiber optics and derives touch states from them. Fig. 3.36 (right) shows an example that applies this principle to a SLAP Widget.

Fiber optics also allow to stack and compose tangibles.

Lumino by Baudisch et al. [2010] shows that fiber optics embedded into tangibles can also support the *modular* composition of physical objects. Lumino are stackable tangible blocks on tabletops whose z-order and orientation can be detected by the tabletop. Every block owns an individual marker at its bottom surface. Furthermore, it contains a bundle of fiber optics that transfers incoming light from its top to the bottom surface and composes it with an own marker. Thereby, a stack of blocks exposes a unique marker to the table's camera, encoding block IDs, orientations, and order. This principle could allow to construct SLAP Widgets that are composed of building blocks.

The concept of SLAP Widgets has also been transferred to non-optical tracking technologies that allow a thinner form factor. Yu et al. [2011] developed *Clip-on Gadgets*, passive physical controls that users can clip on mobile capacitive touch screens. They use a footprint concept similar to SLAP Widgets but communicate the widget's state by a capacitive link between the user's finger, conductive markers beneath the widget, and the electrodes in the touch screen. When the user touches a conductive part of the control, the capacity of the finger is transferred to the conductive markers, and spots appear in the sensor input image. With a sufficiently large surface of the conductive material, capacitive markers also create spots without being touched by the user. This technique works with most capacitive multi-touch screens, and can, therefore, be applied to mobile devices. Also, first commercial products have recently been released that use this technology, such as the Fling joystick[9]. More recently, Chan et al. [2012] introduced physical tangible widgets for capacitive screens, including sliders and knobs. Magnetic connectors between modular blocks also allow to stack and compose tangibles. The benefit of this technology is that transferring a state down to the footprint is easier, because conducting paths can be directed arbitrarily (as opposed to fiber optics). Using an improved manufacturing process, the cables can also be made so thin that they are barely visible. On the downside, constructing capacitive widgets is more difficult than building SLAP Widgets with optical markers.

Concept of SLAP Widgets has been transferred to non-optical technologies.

An alternative approach are widgets that use a pressure footprint on a resistive multi-touch screen to communicate their state. The Geckos project [Leitner and Haller, 2011] employs an Interpolating Force-Sensing Resistance (IFSR) sensor [Rosenberg and Perlin, 2009] on top of a magnetic grid. Circular permanent magnets at the bottom of the widgets provide a unique pressure footprint in the sensor that indicates type and state of the control. As opposed to previous approaches, Geckos can be attached to vertical (magnetic) surfaces as well. However, the system currently relies on top projection only. An integrated solution, involving a transparent IFSR sensor in combination with a magnetic plate behind a backlit LCD panel, could be feasible in future.

Vertical SLAP Widgets are imaginable using magnets and IFSR sensor.

While our SLAP Table prototype is relatively easy to build, its construction is also a limiting factor in terms of practicability. Both the projector and the camera demand a large volume beneath the surface and a fixed height. Due to its cubic frame, users tend to hit the table's sides with their knees. The NARROW SUBSTRUCTURE design pattern proposes to narrow the frame to make standing close to the table more convenient [Remy, 2010]. We implemented this pattern in the "Aachener Friedenstisch", an interactive exhibit in the city hall of Aachen. Visitors learn historic details about the Treaty of Aix-la-Chapelle (1748) by interacting with tangible blocks. The table surface is mounted on a structure that narrows down, allowing users to stand close to the table while interacting (Fig. 3.37).

Table construction limits practicability. Narrowed substructure would allow standing at table.

Typing on interactive tabletops remains an open issue. The SLAP Keyboard is still a prototype whose haptic feedback does not measure up with

[9]http://tenonedesign.com/fling.php

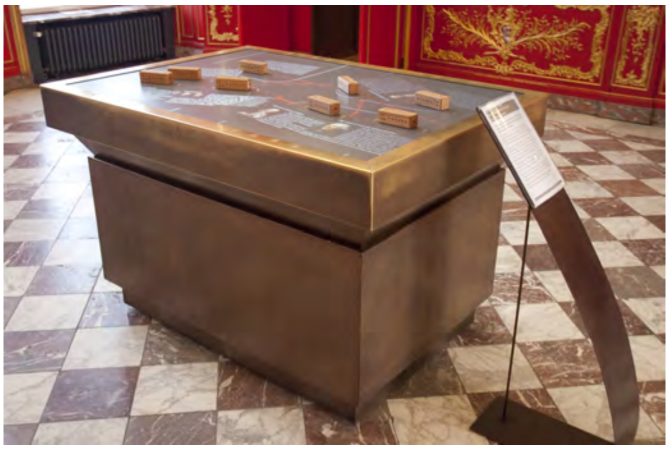


Figure 3.37: The "Aachener Frieden" table is an interactive table that enables interaction with tangible blocks. The narrowed substructure allows users to conveniently stand at the table.

Typing on interactive tabletops screen is still an open issue.

conventional physical keyboards. For intensive typing tasks on tabletops, augmentation of conventional keyboards could be a valid solution. The *Optimus Maximus*[10] is a physical keyboard with a 48 × 48 OLED display embedded in every key. Like the SLAP Keyboard, it can be dynamically relabeled according to the current application context. However, this keyboard is relatively expensive and tethered, which limits its applicability to touch screens. Block et al. [2010] use top projection on a conventional keyboard to relabel every key with a resolution of about 48 × 48 pixels. They also added touch sensors to detect whether users currently rest their fingers on one or multiple keys. The keyboard combines the well optimized haptic feedback of conventional keyboards with dynamic relabeling. Yet, it is still uncertain how well those keyboards integrate into a dynamic tabletop application where collapsibility plays an important role. Also, the need for a power supply either demands batteries, which have to be charged, or cables that consume space and potentially hinder quick hand-overs of controls.

Physical-visual consistency of SLAP Widgets might break down due to external events or remote input.

We mentioned *consistent feedback* as one of our requirements when designing SLAP Widgets. However, the physical-visual consistency easily breaks down when the software changes a widget's value. For example, imagine a SLAP Slider for controlling the volume of background music in a public place. At 6 pm, the system automatically reduces the volume to consider the neighborhood. While the back projection of the slider could be updated, the physical state would remain the same, breaking the tight coupling between the physical object and its dynamic visual appearance. An external event can also break this consistency, for example, in a remote collaboration application where two distant users work on a shared task. If both users have paired a slider with the same shared virtual object, the physical-visual state of one user becomes inconsistent as soon as the other one shifts the slider. In the next chapter, we will deal with this issue and provide a solution that maintains our design requirements.

[10]http://www.artlebedev.com/everything/optimus/

Chapter 4

Maintaining Consistency: Madgets

In the previous chapter, we introduced SLAP Widgets, general-purpose controls that combine haptic feedback with a dynamic visual appearance via back projection. The strength of SLAP Widgets is their lightness. As passive controls, they are low-cost, easy to build, and do not require any electronics. However, the benefits of haptic feedback come at the expense of interface consistency. The physical interaction between users and SLAP Widgets is one-directional: The table's software *reacts* on physical widget manipulation, but it cannot change the physical state itself; this back-direction is missing. This easily causes situations in which the consistency between a control's physical configuration and its back projected graphics breaks down. This not only destroys the illusion of a coherent tangible, it also causes ambiguous states in the user interface and, finally, limits the applicability of SLAP Widgets to productivity tasks.

SLAP Widgets are one-directional: System reacts on physical input but cannot change physical configuration.

In some situations, physical-visual consistency of SLAP Widgets easily breaks down.

This chapter addresses this issue and introduces a new class of tangible tabletop controls: Madgets. While maintaining the benefits of passive controls, Madgets preserve the physical-virtual consistency of tangibles and allow to transfer a variety of concepts known from conventional GUIs to tabletops. The key component of Madgets is electromagnetic actuation (section 4.5) in combination with fiber optical tracking (section 4.6). After motivating the issue and presenting related work, we will explain the hard-

Publications
Madgets was developed in collaboration with Florian Schwarz, who significantly contributed to the hardware design of the table and the actuation algorithm. Part of this work was first published as poster at the ITS '09 [Weiss et al., 2009a] and then as full paper at the UIST '11 conference [Weiss et al., 2010b]. It was also topic of the diploma thesis of Florian Schwarz [Schwarz, 2010], supervised by the author of this thesis. The preliminary study to explore the conflict between actuation and user control was designed together with Ronny Seidel. He conducted the study as part of his diploma thesis [Seidel, 2011] that was also supervised by the author of this thesis.

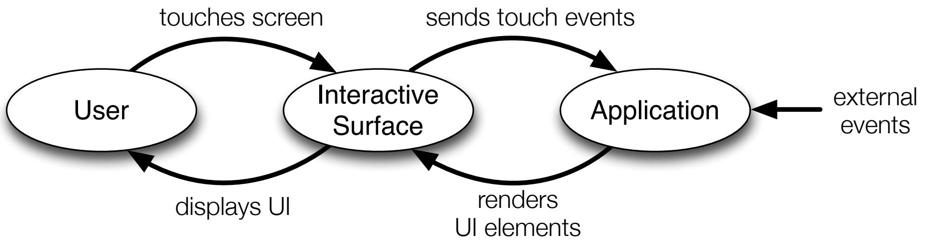


Figure 4.1: Bidirectional interaction between user, interface, and application in a touch-based GUI.

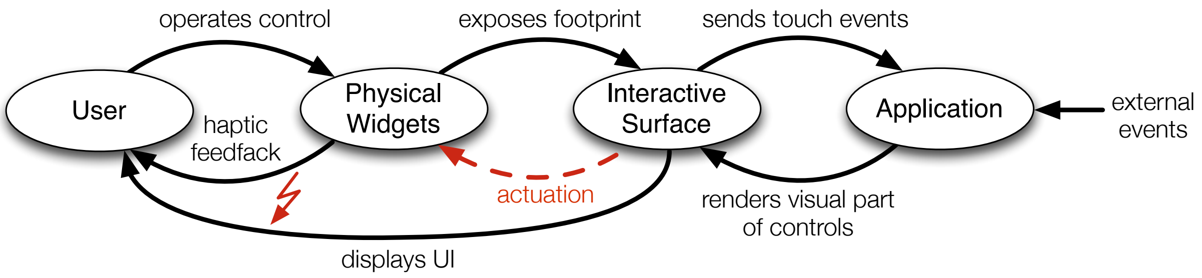


Figure 4.2: Typical interaction with SLAP Widgets on interactive tabletops. If an external event changes a widget's internal state, the table can update its visual back projection but not its physical state. This breaks the consistency between the visual and physical state of the control. An actuation mechanism, indicated by a red dashed arrow, is required for a complete bidirectional communication between user, interface, and application.

ware setup and the employed algorithms in detail. Finally, we will describe a variety of applications.

4.1 Unidirectional Interaction

GUIs inherently support bidirectional interaction.

A major reason for the success of touch-based GUIs in today's computing devices is their flexibility. The user interface can dynamically adapt to the current application context. There is no need to hardwire physical controls into a device anymore. Users can trigger actions and change values by tapping on-screen buttons or dragging virtual sliders or knobs with their fingers. Also, the underlying application can modify the UI, e.g., gray out a button to disable it, set the angle of a virtual knob, or adjust the range of a slider. The interaction between user, interface, and application is bidirectional (Fig. 4.1).

GUI applications can modify user interface.

Applications cannot change physical state of SLAP Widgets. This can lead to physical-visual inconsistencies.

Albeit SLAP Widgets provide richer haptic feedback than plain on-screen controls, the physicality of the tangibles also induces a gap in the communication process. When a user operates a widget on the tabletop, the tracking algorithm detects the changing footprint in the camera image, and the system modifies the widget's internal state accordingly. After that, the application updates the graphical representation of the control. However, the *visual* back projection is the only available back channel. The system cannot change the physical state of the control, it cannot influence the physical part of the user interface (Fig. 4.2). This can easily cause inconsistencies in the user interface, as the following scenarios illustrate:

SLAP Widget is associated with an object's parameter that changes autonomously. Physical position of slider is not updated.

Object Update Inconsistency *Bob intends to watch a video on the table. He pairs a SLAP Keypad with a video object for playback control, as well as a SLAP Slider for timeline navigation. He presses "Play" on the SLAP Keypad to start playback. After watching for a while, Bob decides to skip a part of the video. He drags the slider's handle to right but the video unexpectedly jumps backwards in the timeline.*

During playback the internal state of the video, the playback position, changed over time. However, as the slider's handle cannot be physically moved via software, it remained at its original position when playback was started. This yields an inconsistency between the object's state and the associated physical control. Such an inconsistency always occurs, if a SLAP Widget is associated with a parameter that changes autonomously or due to external events.

Multiple widgets associated with the same value. Operating one control does not physically update the others.

Inter-widget Inconsistency *Alice wants to navigate through a video that is displayed on the tabletop. She pairs a SLAP Slider with the video for timeline navigation and a SLAP Knob for frame-wise stepping (cf. video ethnography usage scenario in section 3.7.1). She shifts the slider's handle to the center in order to navigate to the middle of the video. She then continues working with the SLAP Knob for a while. Suddenly, she reaches the end of the video although the slider's handle is still centered, indicating the middle position of the video.*

Again, the physical state of the SLAP Slider ("middle position") was not updated and became *inconsistent* with the system state ("end of the video"). This issue could be mitigated by updating the *graphical* representation of the slider according to the position in the movie. However, in this case, the physical position would differ from the visual one. This not only breaks the illusion of a coherent tangible, it even impairs the user's mental model of the slider: If Alice now shifted the slider slightly to the right in order to move *forward* in the video, the video player would actually jump *back* to a position near the middle of the video. In general, if more than two continuous input widgets are paired with a single virtual object, inconsistencies are likely to occur.

A widget paired with an object immediately overwrites the existing value through its physical state.

Obtrusive Physical State *Bob intends to slightly increase the brightness of an image. He places a SLAP Slider on the tabletop. In the moment he pairs the slider with an image, the brightness jumps to a certain value although he has not moved the slider's handle at all.*

Since the physical state of a control cannot be changed software-wise, a widget obtrudes its physical state at the stage of pairing. Thus, when the user pairs a widget with an on-screen object, the physical state immediately sends a value to the object. Otherwise, the widget and the virtual object would be in an inconsistent state. A better solution would first adapt the physical state of the SLAP Widget just after it is paired with an object, and release it for interaction afterwards.

Remotely shared widgets are physically not kept in sync.

Remote Inconsistency *Alice and Bob conduct a remote simulation of a jet engine on two distant interactive tables. All virtual objects, including a visualization of a particle simulation, are mirrored on both tables via network connection. Both users pair a SLAP Knob to control the speed of the engine, and a set of SLAP Sliders to modify material parameters. Alice initially sets some experimental values for the material by using the sliders. Bob then gradually increases the speed. The simulation suddenly fails although—from Bob's perspective—all parameters seem to be within the engine's specification.*

What happened? Since Bob's SLAP Sliders have not physically moved after Alice's modifications, he was not aware of the modified material parameters. Thus, he accidentally used a speed that did not fit those parameters. Many controls, such as sliders, knobs, and toggle buttons, are both input and output devices. SLAP Widgets, however, can only provide consistent haptic output if they are exclusively applied by one user to a single virtual object, and if no external events change the object's state. This is a clear limitation for collaborative applications.

One-directional interaction hinders transfer of GUI features.

Finally, the table's inability to move and configure physical widgets avoids to transfer many GUI functions that are required for productivity tasks, including undo and redo, storing and loading a system state, or adapting the interface to the current application context.

Bidirectional interaction between user, interface, and application requires actuation.

SLAP Widgets signify a first step towards haptic feedback on tabletops, but the current design inherently hinders their use in productivity applications. For a bidirectional interaction between user, interface, and application, we need an *actuation* algorithm that ensures a malleable interface (Fig. 4.2): an interactive tabletop that can change its physical configuration on the fly, i.e., a system that can physically move and configure SLAP Widgets on the surface. In the following, we describe the body of related work that deals with this issue. Afterwards, we explain how we turned SLAP Widgets into actuated bidirectional controls.

4.2 Related Work

This section gives an overview on *actuated physical controls* that keep up a consistent mapping between their virtual and physical state. Furthermore, we estimate their usefulness for interactive tabletop applications.

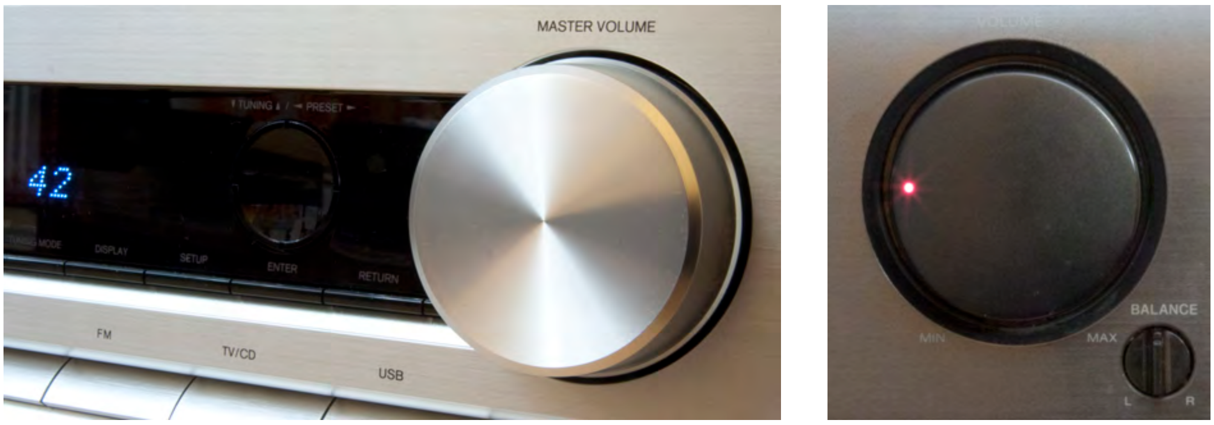


Figure 4.3: Volume knobs of HiFi audio receivers. Left: Infinite knob that changes volume relatively to the current value. It does not block at volume ranges and is easy to implement. However, an extra display is required to show the current volume. Right: Absolute knob with embedded analog volume indicator. Users can employ haptic memory to set a value. However, a motor is required to maintain digital-physical consistency when changing volume via remote control. Photos taken by the author.

4.2.1 Actuated Knobs and Sliders

Everyday example: Actuated volume knobs in HiFi audio systems

The issue of maintaing a virtual-physical consistency is already addressed in everyday devices. For example, most HiFi audio systems provide two ways to set the volume: a physical dial on the device and a remote control with pushbuttons that step-wise increase or decrease the loudness. Simpler systems avoid inconsistencies between the physical knob position and the actual volume value by using an infinite dial without physical range constraints. The current volume is only shown in the display (Fig. 4.3, left). The SLAP Knob is implemented in this way. More sophisticated systems use a knob that blocks at the minimum or maximum volume setting. A mark on the knob shows the current volume (Fig. 4.3, right). This design gives the audio volume a spatial meaning (angle) and, therefore, enables users to employ spatial memory when setting the volume in an eyes-free fashion. However, when a user changes the volume remotely, the system must turn the knob autonomously in order to match the new internal value. This requires at least a motor and an adequate driver inside the knob. While providing a richer haptic experience, such active controls always involve additional hardware, a higher implementation effort, and, eventually, higher production cost than passive ones.

Volume knobs with attached indicators embed motors to maintain consistency.

Actuated sliders in mixers allow to quickly save, load, and switch configurations.

Professional audio mixers often contain actuated sliders (Fig. 4.4). These physical sliders provide all benefits of physical controls and can be operated without looking. This is particularly important if a user has to concentrate on a different task. Furthermore, motors embedded in the sliders allow to load audio settings that have been created in advance. For example, imagine a sound engineer during a live music performance. He usually has to look at the stage while operating the controls. Just before the next song is performed, the engineer can change the volume of all instruments by loading a prepared configuration.

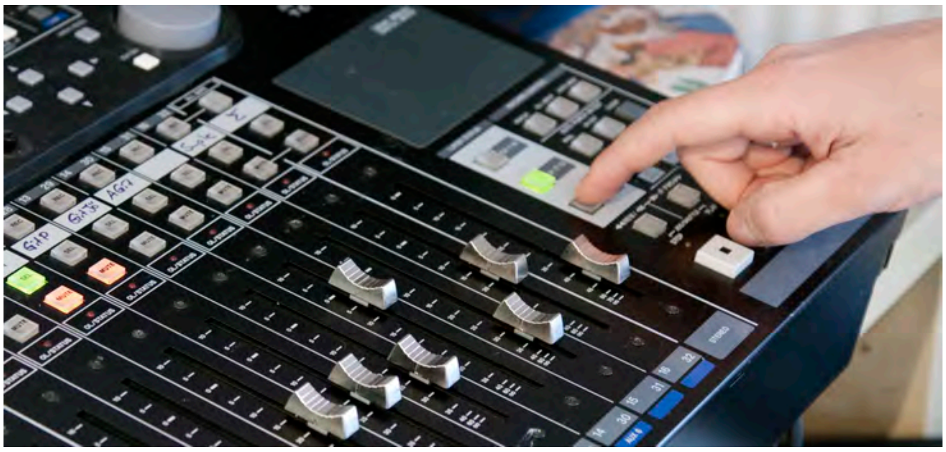


Figure 4.4: Hardware audio mixer. Motorized sliders allow to quickly save, load, and switch configurations. Photo taken by the author.

Motorized sliders have been explored as input and force feedback devices in HCI research.

In HCI research, motorized sliders have been explored for educational and musical purposes. The *Force Feedback Slider* is a motorized slider that takes force and position as input and as output channel [Shahrokni et al., 2006]. An application named *FeelTheBeat* uses a slider to physically output and input a time-based parameter in an audio sequencer. Through the actuated slider, users can see and feel the amplitude of a so-called sound envelope over time. They can also modify this parameter by moving the slider position during playback. Using a similar concept, *BounceSlider* provides multiple of these sliders to compose music from different samples. The purpose of this research is to "provide a tool for exploring perceived physical characteristics of *sound as an object*" [Gabriel et al., 2008, p. 128].

Embedding motorized sliders and knobs in interactive tabletops is difficult.

Motorized sliders and knobs are suitable for embedding in larger devices. However, their use on interactive tabletops is difficult. The need for a motor, controllers, tethering, or batteries induces a minimum size and weight factor, which makes it hard to integrate these controls into a dynamic tabletop application. Furthermore, they are complex and difficult to prototype.

4.2.2 Shape Displays

Superior goal of shape displays: Surfaces that can morph into arbitrary 3D shapes.

While most of today's interactive surfaces are planar, researchers envision interfaces that can change their shape arbitrarily; malleable physical devices that morph into any 3D physical interface, so-called *Shape displays*. The *Claytronics* project[1] plans to achieve this vision using "programmable matter", small scale nano robots that can build up arbitrary 3D structures in a modular way by creating and releasing interconnections [Goldstein et al., 2009]. The project is currently in the phase of simulation studies and large scaled robotic prototypes. Although first manufacturing processes for small robots have been published [Karagozler et al., 2009], it will probably require many years until a nano scale is achieved.

[1]http://www.cs.cmu.edu/~claytronics/

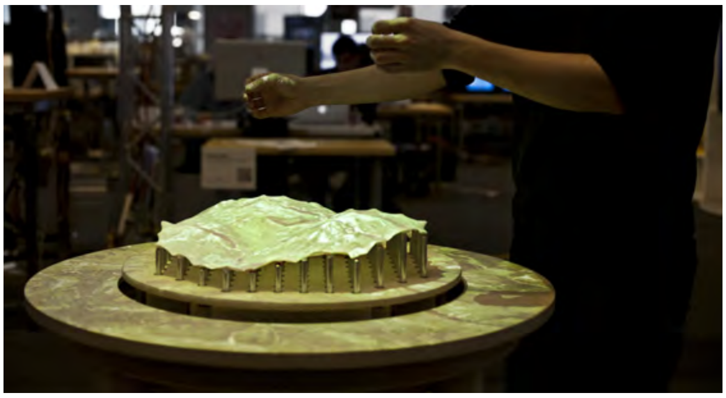


Figure 4.5: Relief contains 120 motorized pins that can generate height maps. Top projection on a Lycra surface provides a visual overlay. Users can perform input by pulling or pushing pins, by using external controllers, or via gestures above the surface by employing depth camera tracking. Image courtesy of Leithinger et al. [2011].

Yet, shape displays have been successfully implemented in form of dynamic 2D height map displays. Those displays consist of a 2D array of pixels whose height can be changed.

Shape displays have been implemented as malleable 2D height maps.

Already in 1966, Linvill and Bliss [1966] created a display that could output monochrome images in a tactile way. The device consisted of an array of small rods that could vibrate individually. Vibrating rods then formed a tactile binary 2D image which could be felt by blind users when putting their fingers on the surface. A practical application of such a technology are Braille displays. They allow visually impaired users to read and type at a computer via a dot-based haptic text representation (e.g., [Prescher et al., 2010]).

Practical example: Braille displays

FEELEX is a continuous terrain display that consists of a 2D array of rods with linear actuators. Each each rod can be set to an individual height [Iwata et al., 2001]. A deformable screen on top of the discrete rod array provides a continuous surface. To provide surface details, a projector renders a 2D texture onto this surface. Recently, Leithinger et al. [2011] published *Relief*, a more scalable version of this technique, and explored various interaction techniques (Fig. 4.5). *Lumen* employs *Shape Memory Alloys (SMA)* to move rods in a 2D array [Poupyrev et al., 2004]. They also added color lights in each rod yielding a coarse RGB display with an additional height channel. Furthermore, capacitive sensors allow users to interact with each rod.

Example implementations: 2D array of rods with linear actuators and Shape Memory Alloys

Harrison and Hudson [2009b] presented a pneumatic height display that can raise or emboss 3D buttons on a planar surface. The system is based on a flexible latex layer. The latex sheet is adhesively connected with an acrylic backing with cut-outs. These cut-outs determine areas that can be raised or lowered. A base under the backing closes the construction to a pressure chamber. An attached pneumatic pump allows to vary the pressure. With

Pneumatic actuation can be used to inflate or deflate buttons.

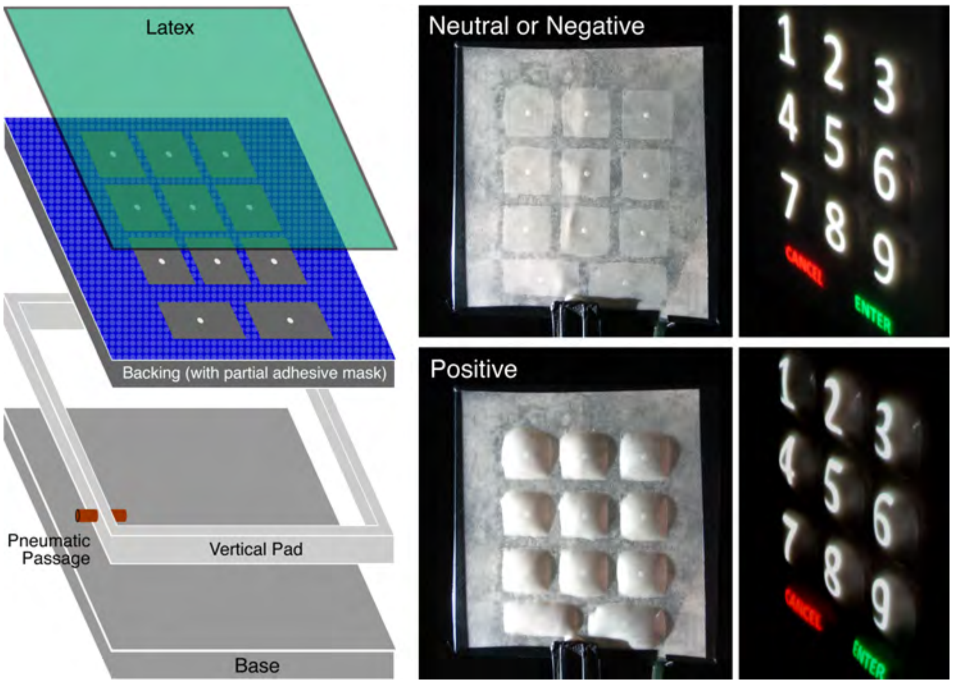


Figure 4.6: Pneumatic height display. Buttons are dynamically created by inflating the underlying chamber. The increased air pressure extends and raises the flexible surface in the button areas. Left: Technical concepts. Right: Example of an ATM keypad with back projection. Image courtesy of Harrison and Hudson [2009b].

default pressure, the latex sheet and, therefore, the interactive surface is flat. However, high pressure in the chamber raises the latex above the cut-outs (Fig. 4.6). Under lower pressure, the latex is embossed. The authors propose to use this technology to dynamically create or delete physical 3D buttons. Back projection and a camera behind the surface provide graphical output and multi-touch input, respectively. This system combines the flexibility of GUIs with dynamic 3D objects in a clever way. However, the haptic feedback created by a latex sheet is very limited. Furthermore, all possible height maps are predefined by the cut-outs in the acrylic backings.

Shape displays only create "2.5D" objects, not complex controls.

Shape displays provide dynamic malleable surfaces. However, all currently feasible implementations only create "2.5D" height maps. They cannot form complex controls with rich haptic feedback, such as sliders or knobs. Furthermore, shape displays are difficult to construct and limited in terms of resolution.

4.2.3 Actuated Tangibles on Tabletops

Several projects have introduced actuated tangibles for interactive tabletops to provide a bidirectional interaction between users and table objects, as well as to maintain physical-visual consistency. In the following, we describe these projects differentiating between active self-actuating objects and passive objects that are actuated indirectly.

4.2.3.1 Self-Actuating Tangibles

Self-actuating tangibles use embedded motors to freely move and align on a tabletop.

The *Planar Manipulator Display* by Rosenfeld et al. [2004] supports bidirectional interaction with active physical objects on a tabletop. Each object is based on a small motorized vehicle that can drive to any position on the table. The vehicles themselves do not perform complex computation but execute movement commands received from the underlying control system. A lateral-effect photodiode beneath the table senses the position and orientation of two pulsed photodiodes under each vehicle. This information is then sent to the control system. By mounting a physical cap on top of each vehicle, the device gains a semantic meaning. The authors present a sample application for interior architects. A room layout is projected from above. Every vehicle represents a piece of furniture that users can place everywhere on the table. When selecting a layout method, all other pieces rearrange themselves according to a particular configuration. Vehicles are moved simultaneously, and multiple users can interact at the same time. Also, physical arrangements can be saved and reloaded later.

The *Augmented Coliseum* is a similar project involving small physical robots on an interactive tabletop [Kojima et al., 2006]. The authors developed an augmented reality game that employs top projection to render a virtual game environment on the table surface. Furthermore, users can place actuated physical robots on the tabletop to interact with the virtual environment. Vice versa, a game logic and a physics simulation running in the background allow robots to interact with virtual objects. The vehicles in Augmented Coliseum track themselves and seemingly make autonomous decisions within the constraints of the game logic.

A more recent project called *Tangible Bots* combines a DSI multi-touch table setup with tangible robots [Pedersen and Hornbæk, 2011]. A Tangible Bot is a puck-like tangible mounted on two motorized wheels that allow for translation and rotation. A computer in the table tracks the tangibles and controls them by wirelessly transmitting actuation commands. Users can turn these tangibles like rotary knobs. Furthermore, a Tangible Bot can be rearranged by the system and, e.g., resist a user's turn by rotating into the opposite direction. This can be helpful to implement physical constraints.

Self-actuated tangibles allow quick movements but require large and heavy components that restrain use on tabletops.

Actuated vehicles and robots can move quickly and smoothly across a tabletop. However, they also involve electronics, such as motors and controller boards, which have to be powered. The Planar Manipulator Display and Tangible Bots require batteries in all tangibles that have to be recharged on a regular basis; the robots in the Augmented Coliseum are tethered. This restrains the flexibility of these devices when being used ad hoc with multiple users. Furthermore, actuated vehicles are relatively heavy, large,

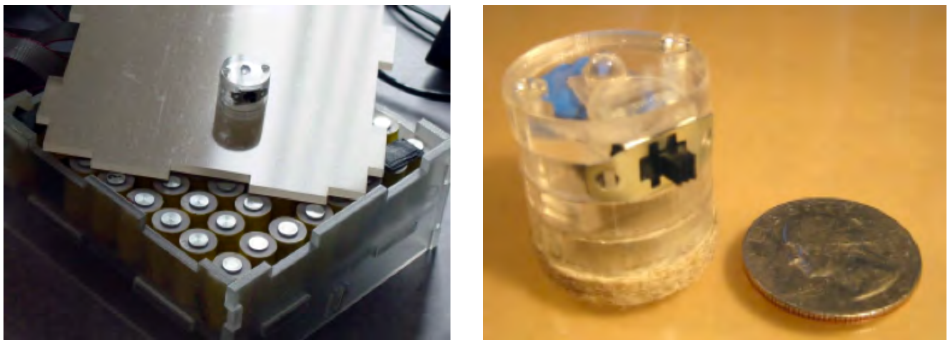


Figure 4.7: The Actuated Workbench. Left: The system bases on an array of electromagnets. The polarization and strength of each electromagnet can be changed individually to attract or repel magnetic pucks on the surface. Right: A magnetic puck contains a permanent magnet and an IR LED for tracking. A felt pad reduces friction during actuation. Image courtesy of Pangaro et al. [2002].

and difficult to prototype for designers without electrical engineering background.

4.2.3.2 Indirect Actuation

Indirect actuation embeds actuation hardware into the tabletop.

Some approaches incorporate the actuation mechanism into the tabletop hardware to move passive objects that do not contain electronic components for autonomous actuation.

Vibrating plates can move and align passive objects, but device is noisy and hard to combine with touch interaction.

The *Universal Planar Manipulator (UPM)* by Reznik and Canny [2001] is a horizontal vibrating plate that can move rigid objects on top of it. The friction of a rigid horizontal rotation of the plate creates a displacement field that moves all objects to certain directions. A set of distinct rotations creates a basis of displacement fields. These can be linearly combined to achieve an arbitrary horizontal translation for individual objects. That is, multiple objects can be moved at once in a round robin fashion, i.e., by successive distinct rotations of the single horizontal plate. Although most objects shown in the paper are circular, the authors state that the approach works for arbitrarily shaped rigid objects. The major benefit of this approach is that it does not require any electronics and, therefore, no batteries or power supplies in the tabletop objects. Designers can create tangibles without caring much about the actuation process. However, it remains arguable whether the UPM can be used for interactive tabletop applications: Due to the vibration patterns, the device is very noisy, and objects move rather slowly. Moreover, a direct touch interaction on the plate would be inconvenient during actuation.

The *Actuated Workbench* by Pangaro et al. [2002] employs a 2D array of electromagnets beneath an acrylic plate to move magnetic pucks across the surface (Fig. 4.7). A controller board can control the polarization and,

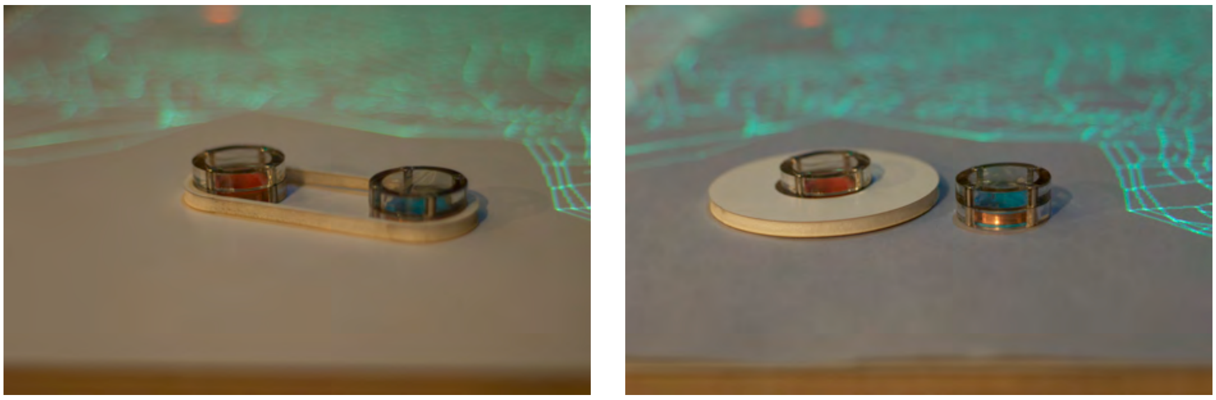


Figure 4.8: Mechanical constraints on actuated pucks that influence optimization. Left: A rubber band around two pucks enforces a maximum distance between them. Right: A disc around a puck ensures a minimum distance to other pucks. Image courtesy of Patten and Ishii [2007].

via pulse-width modulation, the strength of each electromagnet. A tabletop puck is an acrylic cylinder containing a strong neodymium permanent magnet as well as an IR LED mounted on top. The LED is used for tracking; it exposes its position as a spot seen by a camera mounted above the table. A felt pad beneath the puck reduces friction. The actuation principle is based on magnetism: An electromagnet near a puck can apply an attracting or repelling electromagnetic field to the embedded permanent magnet. The puck then moves towards or—in the opposite direction—away from the electromagnet, respectively. A combination of multiple electromagnetic fields allows to translate the puck continuously to any position on the table. Note that the strengths for all electromagnets have to be adapted in each step, according to the current position of the puck. The authors propose applications that transfer GUI mechanisms to physical pucks, such as load and save, undo, or search and retrieve of pucks. On a higher level, remote collaboration and interactive scientific visualizations of simulations are suggested.

Magnetic pucks can be actuated via an array of electromagnets. Attracting and repelling fields yield horizontal movement.

In a later project, Patten and Ishii [2007] add tangible constraints to the concept of passively actuated pucks. They introduce an application that intends to optimize the positions of cellular telephone towers on a map (Fig. 4.8, left). Each tower is represented by a physical puck. The application senses the positions of all pucks and computes a better arrangement according to certain criteria, e.g., a desired minimum distance between towers. It then tries to move the pucks to new positions via actuation, before repeating the tracking and computation step. The authors also introduce a set of physical constraints: A weight placed on a puck hinders it from being moved while all other pucks will still be translated in each step. A ring placed around a puck ensures a minimum distance between it and other objects. Contrariwise, a rubber band around two pucks determines a maximum distance (Fig. 4.8, right). These constraints allow users to physically intervene with the simulation and impose constraints in a tangible and intuitive way. The key benefit of this concept is that there is no need to actually track the tangible constraints. They are implicitly considered by

Mechanical constraints can augment actuated pucks in spatial optimization tasks.

the control loop: In every step, new puck positions are recomputed based on the current ones. If a puck does not move as predicted, e.g., because it is held in place by a weight, its deviant position is used for the next step.

Electromagnetic actuation is calm and smooth. Pucks are easy to build, but use for productivity tasks is limited.

The use of electromagnetic actuation allows a very calm and smooth actuation. Furthermore, tangible pucks are relatively simple to build, although they still require a battery to power the IR LED for tracking. However, pucks also provide rather limited haptic feedback. While being appropriate for spatial simulations and optimizations, their use for productivity tasks is limited. Complex tangible controls that contain moveable parts, such as a knob or a slider, cannot be actuated with the algorithms presented in this section.

4.3 Magnetic Widgets

Using our experience with SLAP Widgets and the findings from related work, we developed a new class of physical tabletop controls.

Design decision: no embedded batteries, motors, or cables as design constraints

The main goal in the concept phase was to create bidirectional controls that preserve the benefits of SLAP Widgets. Batteries, motors, and cables impose considerable constraints upon the designer; they introduce a minimum weight and distinct form factor, limit the mobility of the device, and, as mentioned before, they require a certain skill set of electrical engineering. Therefore, all kinds of motorized controls were not an option. The new controls should be lightweight, low-cost, and easy to build. We also wanted to keep the ability to dynamically change the visual appearance of the controls, because this property allows users to employ the same control for many purposes. Finally, inspired by the idea of Ubiquitous Computing, we aimed at a design that hides the underlying technology from the table.

Madgets are SLAP Widgets with attached permanent magnets for electromagnetic actuation.

Encouraged by the Actuated Workbench, we chose electromagnetism as base technology for actuating controls on the tabletop. It is a *calm* technology that does not require electronics parts in the tangibles. Furthermore, as it works over a distance, it can be hidden beneath the surface. To actuate controls across the surface, we extended SLAP Widgets by adding permanent magnets to its base, yielding a new class of actuated tabletop widgets: magnetic widgets, or *Madgets*.

A sample Madget is shown in Fig. 4.9. As SLAP Widgets, Madgets are made of transparent materials, such as acrylic, which allows for back projection and low-cost construction. Similarly to the Actuated Workbench, we actuate our controls by applying attracting and repelling electromagnetic forces. Permanent magnets are attached to the base of each Madget so that their poles are aligned orthogonally to the surface. By applying different force fields to these magnets, our system can move and configure physical controls on the tabletop (Fig. 4.10). Furthermore, we can actuate into the third dimension. For example, after attaching a permanent, we can raise a button's plate by creating a repelling magnetic field (section 4.8.2.1). Each

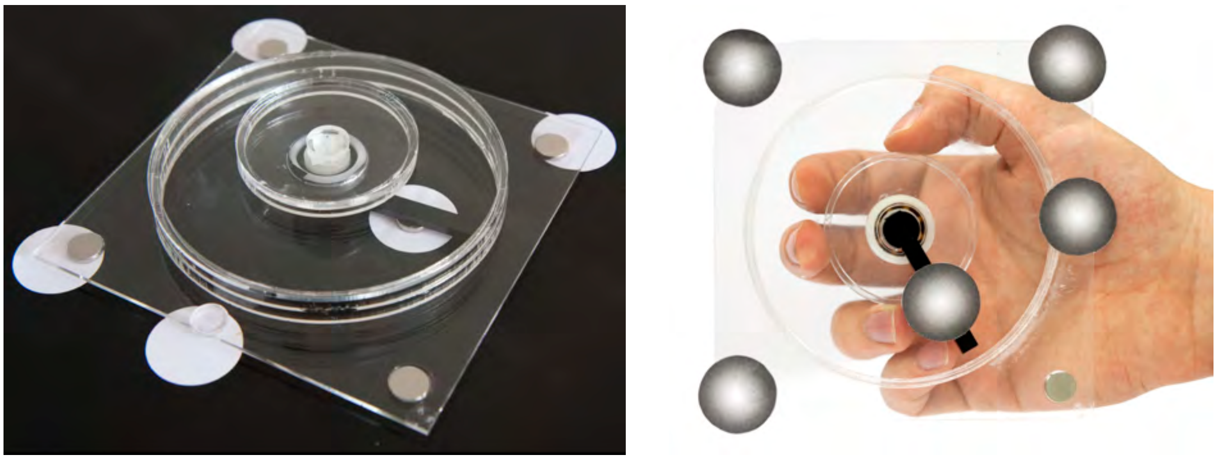


Figure 4.9: Madgets contain permanent magnets that enable passive actuation. Left: Prototype of the Knob Madget. Right: View from below. The footprint consists of gradient markers.

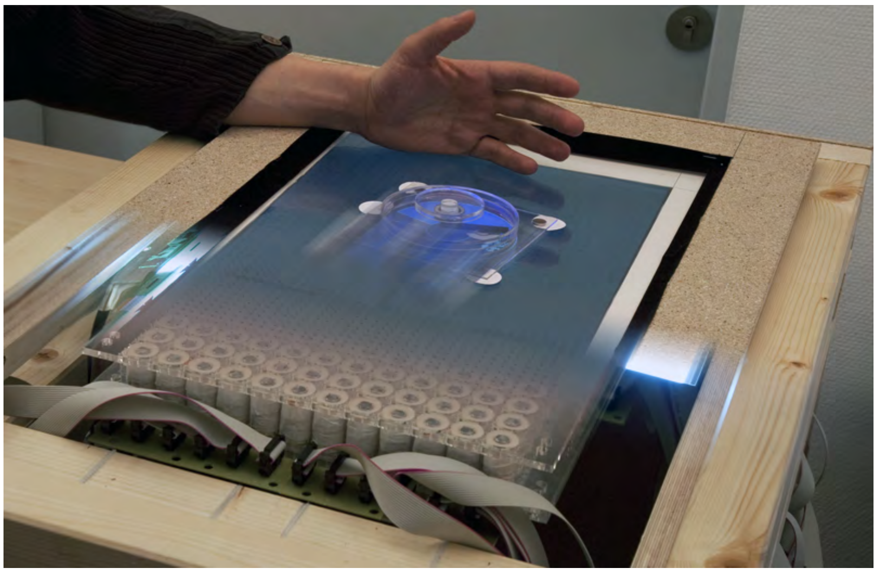


Figure 4.10: Principle behind Madgets. Madgets are actuated across the tabletop by applying electromagnetic forces to permanent magnets attached to the controls. The image shows a Knob Madget and the hardware beneath the table surface. Image adopted from [Weiss et al., 2010b].

Madget is tracked via circular markers in its base. Although the marker design is different and requires a novel tracking algorithm (section 4.6), the widget detection mechanism remains the same as was described in section 3.5.2.

4.4 System Overview

In order to track and actuate passive tangibles while changing their visual appearance on the fly, we developed a novel interactive table that merges

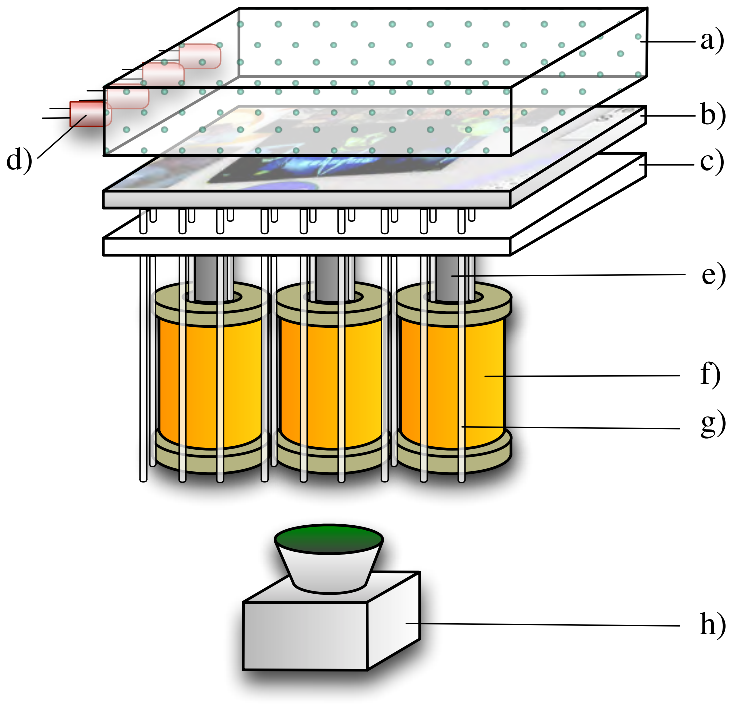


Figure 4.11: Schematic composition of Madgets Table surface. a) Endlighten acrylic. b) LCD panel. c) Electroluminescent foil. d) Infrared LED. e) Magnet core. f) Electromagnet. g) Fiber optical cable. h) Camera with attached IR pass filter. Image adopted from [Weiss et al., 2010b].

Madgets require a novel tabletop hardware.

these abilities into a compact design. In this section, we describe the conceptual design of our table. Sections 4.5 to 4.7 contain detailed explanations of the hardware and the underlying algorithms.

4.4.1 Surface

DSI is used for visual marker tracking.

As shown in Fig. 4.11, our table surface consists of multiple layers, adding up to a total thickness of 52 mm. The topmost layer is an *Endlighten* plate (Fig. 4.11a). It appears almost transparent to humans but contains micro particles that diffusively reflect light within the surface. LEDs (Fig. 4.11d) emit IR light into the plate that bounces within the Endlighten but also leaves the material to the top and the bottom. This enables DSI tracking: If a marker is placed on the surface, or if a user puts down a finger, it reflects outgoing infrared light downwards, where it can be detected by IR cameras in the table (Fig. 4.11h). The area of our table surface that is sensitive to input amounts to 40 cm × 25 cm.

LCD panel with EL foil creates visual output.

Directly beneath the Endlighten layer, an LCD panel (Fig. 4.11b) displays the graphical user interface, including the back projected visuals for each Madget that is placed on the table. An electroluminescent (EL) foil

(Fig. 4.11c) provides backlighting for the panel. We chose this material, because it can be cut and punched easily without loosing functionality.

Electromagnetic display applies forces to magnetic objects on the tabletop.

The core of our actuation method is an electromagnetic display, an array of electromagnets beneath the display layer (Fig. 4.11e-f). We can control the polarization and strength of every single magnet, and, therefore, apply attracting or repelling forces to the permanent magnets in each Madget. The magnetic fields reach beyond the other layers and can drag a Madget to a certain position, rotate it, or change its physical configuration.

Fiber optics pierce EL foil and allow to look past the electromagnets for tracking IR spots.

Since electromagnets are opaque, we cannot just place a camera beneath the surface for tracking IR spots. Instead, we employ a method similar to FiberBoard [Jackson et al., 2009]. A grid of polymer fiber optical cables (Fig. 4.11g) is placed between the magnets and inside their cores. All infrared light reflected from the surface is transmitted to cameras inside the table. Unlike the LCD panel, the EL foil is opaque in the IR spectrum. Therefore, we drilled holes into the foil. Each fiber optical cable begins beneath the LCD panel, penetrates the EL foil, passes the magnets, and ends just beneath the lowest surface layer. Using fiber optics, cameras in the table can "see" past the actuation hardware. Note though that due to the spacing of the fiber optics, the input resolution is significantly lower than in the SLAP Table. This is addressed by our tracking algorithm (section 4.6).

4.4.2 Architecture

Input concept and software architecture build upon the SLAP Framework.

Fig. 4.12 illustrates the architecture of Madgets. The input processing is similar to our SLAP Table. Cameras with IR pass filters monitor the surface, and deliver raw video footage to a Touch Detection Agent. This agent is a self-contained application that processes the camera input and searches for spots indicating finger touches or markers (section 4.6). It converts these spots into touch events and broadcasts them to all running table applications. The widget detection within the SLAP UITK finds footprints within the set of events (section 3.5.2). The SLAP UITK then handles the communication between widgets and virtual objects. It also renders the graphical user interface and outputs it to the LCD panel in the surface. The application layer sets up the virtual objects and reacts on system events. In addition, programmers can specify where and how the table shall actuate physical widgets.

Actuation Framework computes electromagnetic configuration for Madget actuation in closed loop control.

The *Actuation Framework* computes updates of the electromagnetic display depending on the current positions, orientations, and states of Madgets and on the desired target configuration. An actuation process always starts with a high-level command from the application layer to change the physical configuration of the table, e.g., *"move the knob to position (x,y) and turn it to 30°"*. The Actuation Framework then initiates a closed loop control. In each frame, the table retrieves the Madget's current position, rotation, and configuration from the widget detection, and computes the required

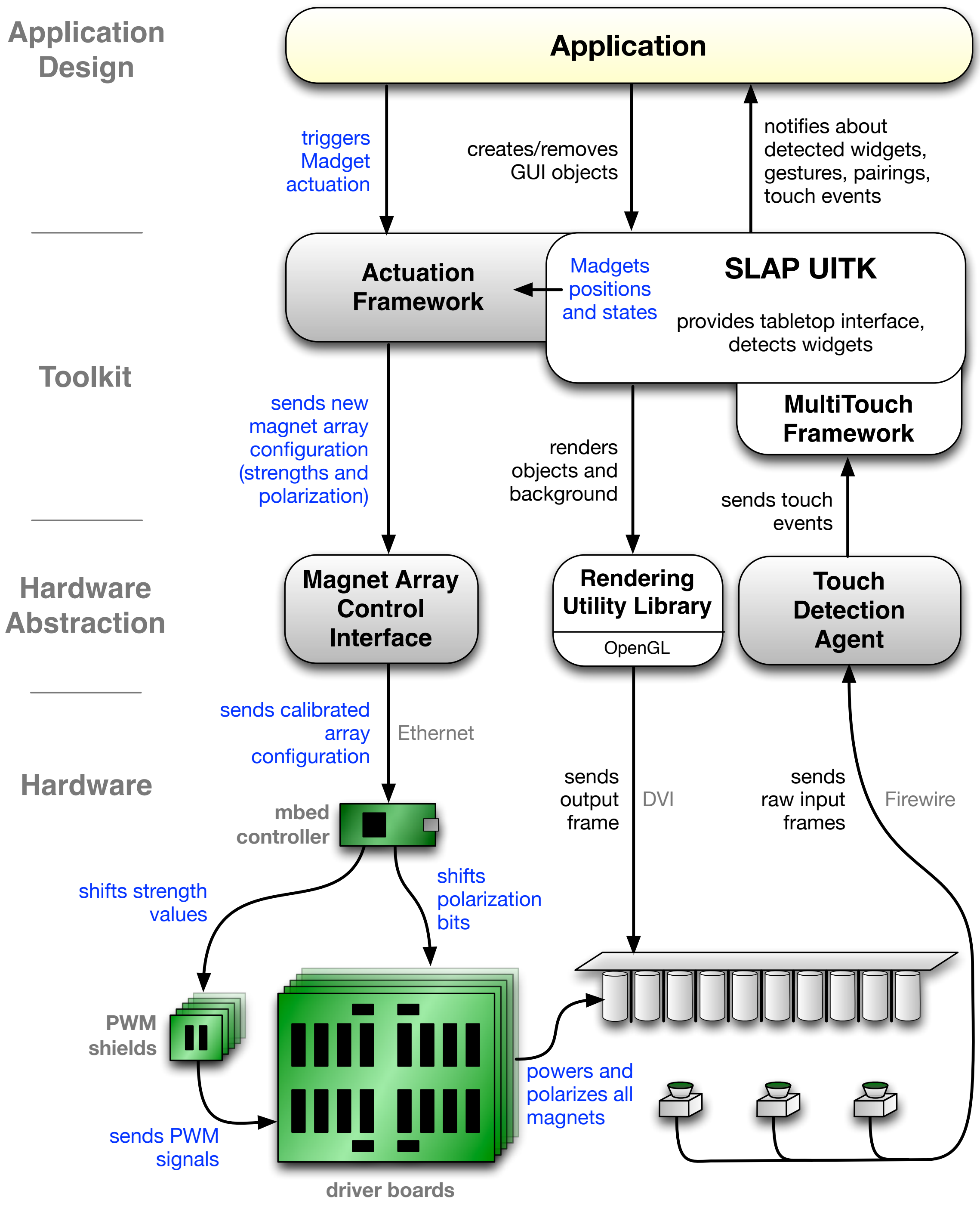

Application Design
Application
triggers Madget actuation
creates/removes GUI objects
notifies about detected widgets, gestures, pairings, touch events
Toolkit
Actuation Framework
Madgets positions and states
SLAP UITK
provides tabletop interface, detects widgets
MultiTouch Framework
sends new magnet array configuration (strengths and polarization)
renders objects and background
sends touch events
Hardware Abstraction
Magnet Array Control Interface
Rendering Utility Library
OpenGL
Touch Detection Agent
sends calibrated array configuration
Ethernet
Hardware
mbed controller
sends output frame
DVI
sends raw input frames
Firewire
shifts strength values
shifts polarization bits
PWM shields
powers and polarizes all magnets
sends PWM signals
driver boards

Figure 4.12: Architecture of Madgets. Blue texts indicate new communication channels in comparison to SLAP Framework (cf. Fig. 3.21 on page 63).

strengths and polarizations of every single electromagnet in the surface, in order to proceed the Madget closer to the target configuration. The close loop ends as soon as the Madget has reached its target.

In every frame, the Actuation Framework sends the desired power and polarization for all magnets to a *Magnet Array Control Interface*. This interface maps the 2D array configuration to calibrated indices (section 4.5.3),

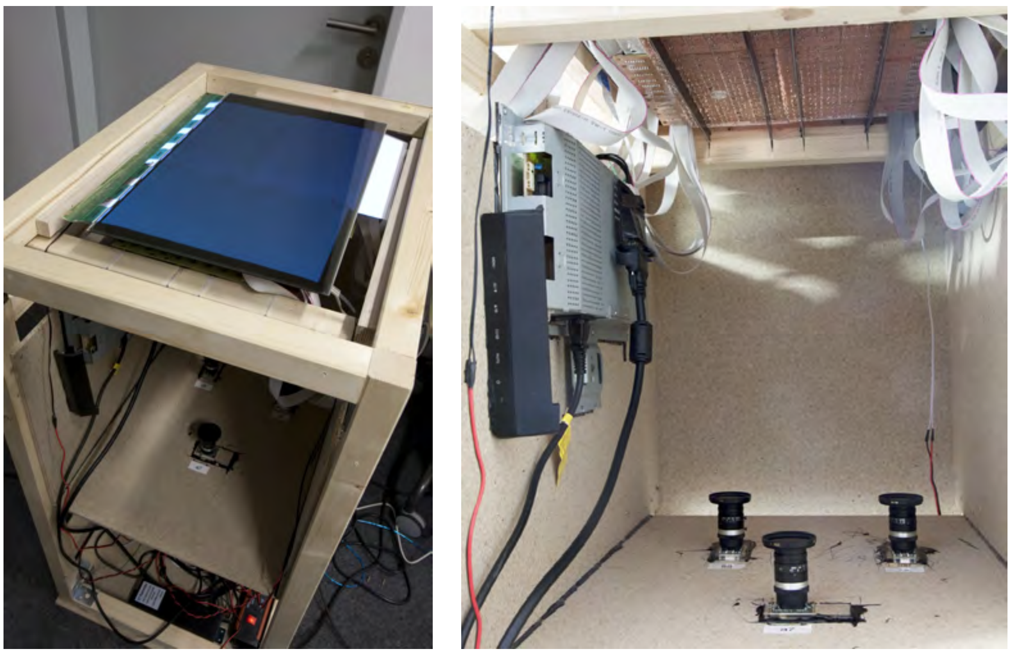


Figure 4.13: Implementation of Madgets Table. Left: The hardware is embedded into a wooden frame. Right: Cameras inside the table monitor the ends of fiber optical cables.

and sends this to an external hardware controller via Ethernet. This controller sends the strengths and polarization to *driver boards*, which power and polarize the electromagnets in the surface. The technical details of this process will be described in section 4.5.

The Magnet Array Control Interface maps 2D array configuration to hardware indices.

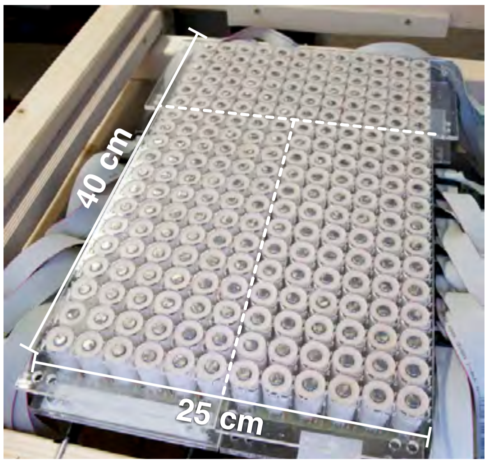
During development, our software, i.e., the top three layers of the architecture, ran on a Mac Pro with two 2.26 GHz Quad-Core Intel Xeon processors and 6 GB RAM.

4.4.3 Table Construction

The table is embedded in a wooden frame (Fig. 4.13, left). Metal rails spanned across the top provide support for the surface modules. All tracking cameras are mounted on a wooden board in the lower part of the frame (Fig. 4.13, right). Power supplies and controllers are placed in the bottom of the frame.

Modular design of surface ensures scalability.

We designed the surface in a modular way to ensure the scalability of the system. Our surface is composed of three independent modules. Each contains a subset of electromagnets and fiber optical cables, an EL foil, and a dedicated IR camera for tracking (Fig. 4.14, left). All electromagnets are supported by a 6 mm acrylic layer that is penetrated by the magnet's connecting cables and the fiber optics. Further acrylic layers hold the magnets in place horizontally. The connecting cables are glued to conductor boards that direct to pin headers at the sides. These headers are used to conveniently connect the electromagnets to the controller hardware. Fig. 4.14



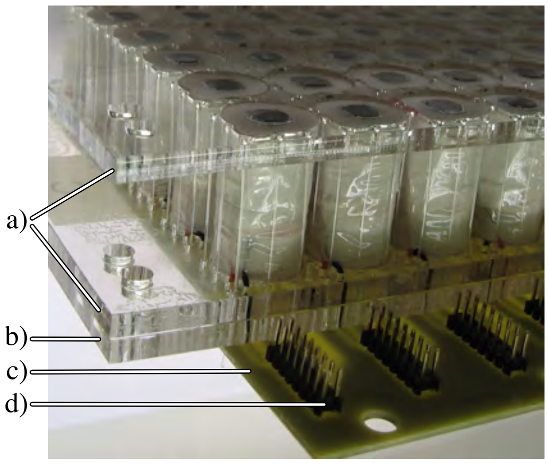


Figure 4.14: Implementation of Madgets Table surface. Left: Surface is composed of three modules. Modified image from [Weiss et al., 2010b]. Right: Electromagnets are mounted on an acrylic plate (b). Further acrylic layers (a) provide lateral support. Electromagnets are soldered to circuit boards (c) that provide pin headers for connection to controller hardware. Image courtesy by Schwarz [2010].

(right) shows the exact arrangement of layers. Note that the fiber optics also penetrate the conductor boards.

4.5 Electromagnetic Actuation

This section describes our electromagnetic actuation technique. After illuminating the hardware control and the preceding calibration process, we will explain the algorithm that computes the force fields to move, arrange, and configure our physical controls.

4.5.1 Hardware Control

Surface contains 19×12 electromagnets for actuation.

The overall surface contains 19×12 electromagnets. Each magnet amounts to a diameter of 19.5 mm and a length of 34.5 mm. It is wound around a plastic coil bobbin with 3500 turns of enameled 0.16 mm thick copper wire. The magnets are specified with a voltage up to 45 V DC, which yields a current of 0.32 A. However, we usually drive them at voltages between 30-40 V. Iron rods with a diameter of 8 mm and a height of 35 mm function as magnetic cores (Fig. 4.11e). We also worked with other core materials, such as manganese-zinc ferrite, as will be explained in section 4.9. We left a small 0.75 mm gap between the core and plastic cylinder to provide space for fiber optical cables. Every magnet is soldered to a thin conductor board beneath all surface layers. It contains pin headers to connect magnets from the sides of the surface.

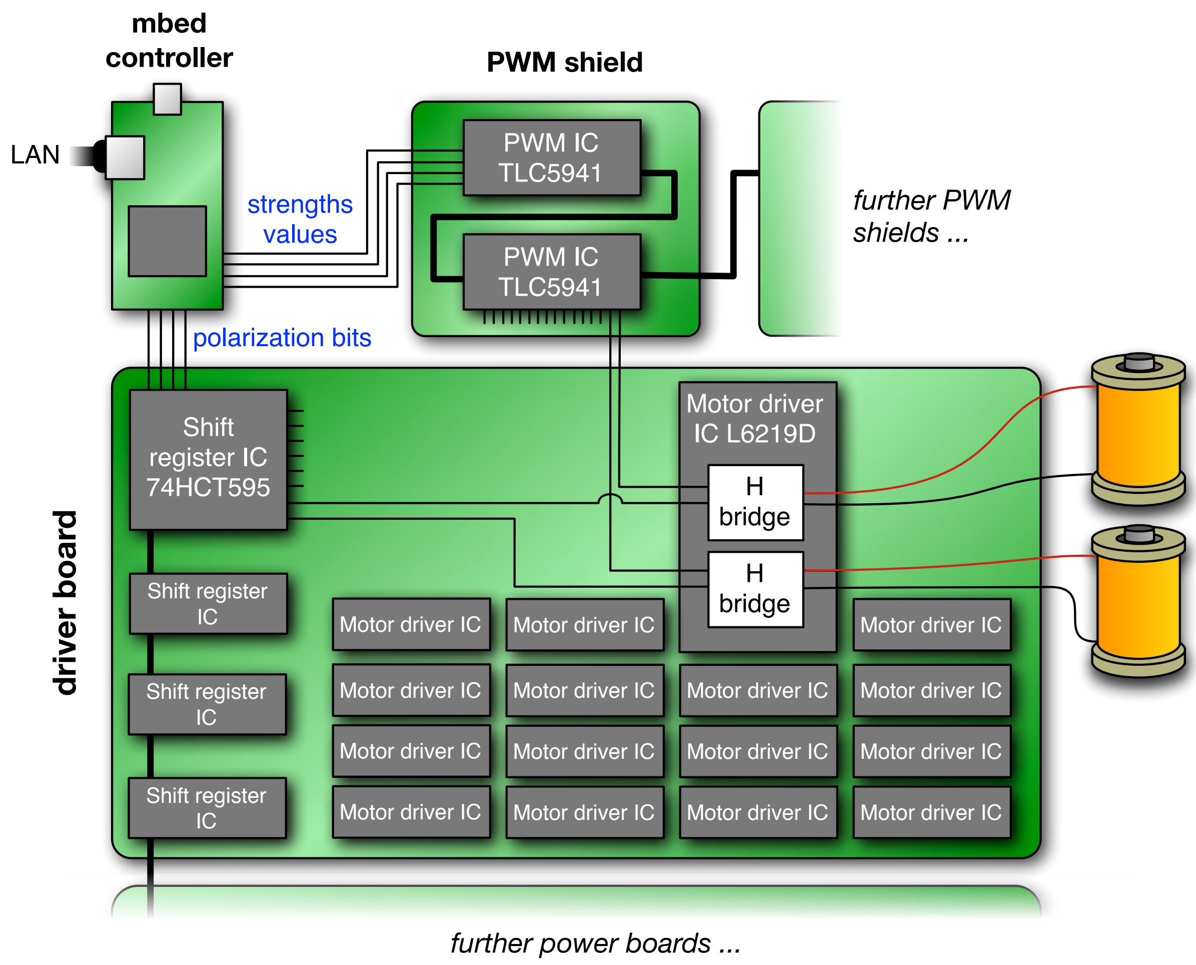


Figure 4.15: Circuit that controls strengths and polarizations of electromagnets.

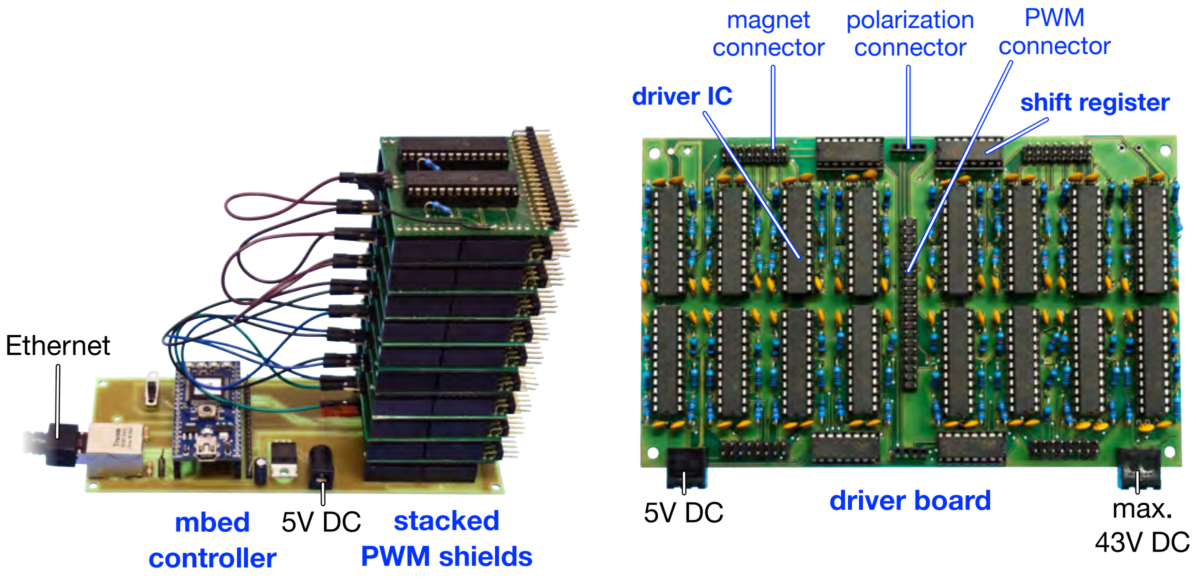


Figure 4.16: Main hardware components that control the electromagnetic array.

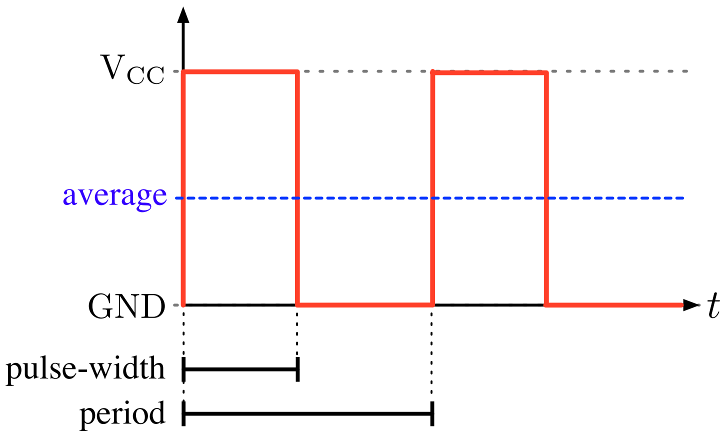


Figure 4.17: Pulse-width modulation (PWM). PWM generates an analog variable using a rectangular waveform containing "on" and "off" states. The effective voltage equals the average (blue dashed line) of the signal.

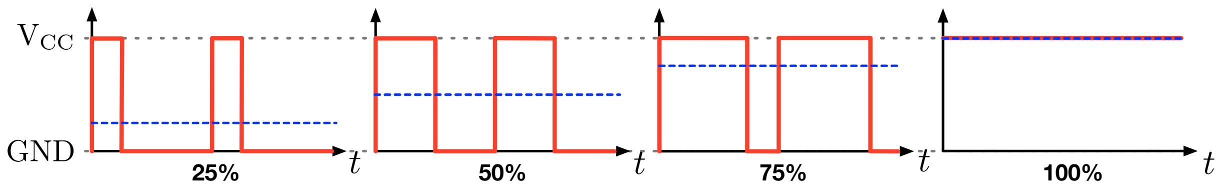


Figure 4.18: Examples for different PWM duty cycles.

Motor drivers control electromagnets.

The control circuit of the magnetic array is illustrated in Fig. 4.15, and the main hardware components are shown in Fig. 4.16. Two magnets are driven by a single L6219D motor driver integrated circuit (IC). Each driver runs at 5 V DC and requires a maximum of 43 V DC to power the attached magnets.

Strengths of electromagnets are varied via pulse-width modulation.

The strength of a magnet is varied via *pulse-width modulation (PWM)*. PWM is a method to generate an analog variable using a rectangular waveform (Fig. 4.17). Informally spoken, an analog value is created by switching a voltage "on" and "off". The period of the signal is usually constant. The "on time" within a period is called *pulse-width*. The percental *duty cycle* is the proportion of the "on time" in the period. If applied to electromagnets with a sufficiently high frequency, the effect of PWM is about the same as if the average voltage of the signal would be applied analogously. Sample duty cycles are shown in Fig. 4.18. The benefit of PWM is that it allows for a simple digital control of the electromagnets. An analog control of each individual electromagnet would considerably complicate the hardware design.

Motor drivers combine polarization bits and PWM signals via H-bridge to control electromagnets.

For each magnet, the driver needs two binary input signals. One to switch the magnet on or off according to the PWM signal, and one to specify the polarization of the output current. The motor driver switches the polarization of the output using two internal H-bridges. PWM generator ICs of type TLC5941 create PWM signals to vary the magnets' strengths. Each controller creates 12-bit (4096 steps) PWM signals for up to 16 independent channels. We daisy-chained 20 of these controllers in a stack to simultaneously control up to 320 magnets. The polarization bits for each motor driver are provided by 40 daisy-chained 74HC595 8-bit shift register ICs.

Electrical components are grouped in driver boards for better handling.

For better maintenance and handling, we grouped the above mentioned controllers into *driver boards*. Each board contains up to 16 motor drivers, 4 shift registers, and a connector for 32 parallel PWM signals. They also contain connectors for 5 V and up to 43 V DC for the electromagnets. We used 10 of these modules to control the 228 magnets. Note that with this configuration we can control up to 320 magnets.

Suppressor diodes and fuses protect motor drivers and electromagnets.

We protect the motor drivers with suppressor diodes connected in parallel with the attached magnets. It eliminates voltage peaks that occur when magnets are switched off or when their polarization is flipped. Furthermore, input current for each driver board is secured by 6.3 A fuses (T6.3L250V).

mbed controller is interface between software and actuation hardware.

An mbed NXP LPC1768 microcontroller[2] represents the interface between the software and the actuation hardware. It runs at 100 Mhz and contains an built-in Ethernet chip supporting fast communication between computer and controller. The software on the controller can be programmed using a web-based C++ compiler. At start-up, the framework connects to the controller via Ethernet and sends an authorization string. The controller then replies with a specific authorization confirmation including the software version. We employed this simple authorization procedure to identify version conflicts between framework and controller. Once the connection is established, the framework sends magnet array configurations via a 640 byte buffer for each frame. The buffer contains 16-bit integers for up to 320 magnets. The first bit in each integer encodes the polarization of a magnet, while the last 12 bits determine its PWM strength. Bits 2 to 4 are unused.

While polling the network, the run-loop of the mbed controller independently updates the magnets in every cycle and clocks the PWM generators. The mbed controller shifts PWM strengths into the PWM generators and polarizations into the shift registers via its digital outputs. A rising edge in the latch pin of both ICs assigns all PWM values and polarizations to the motor drivers in parallel.

4.5.2 Conventions

To ensure a correct actuation, the polarization of permanent magnets that are actuated on the tabletop must be consistent. Without loss of generality, we refer to the permanent magnet's pole that is attracted due to a positive electromagnetic polarization as the *negative pole*. Analogously, the other pole is called *positive pole*[3]. By convention, permanent magnets are

[2]http://mbed.org

[3]More common terms for poles of permanent magnets are "North" or "South". We use "positive" and "negative" for consistency with our mathematical actuation model.

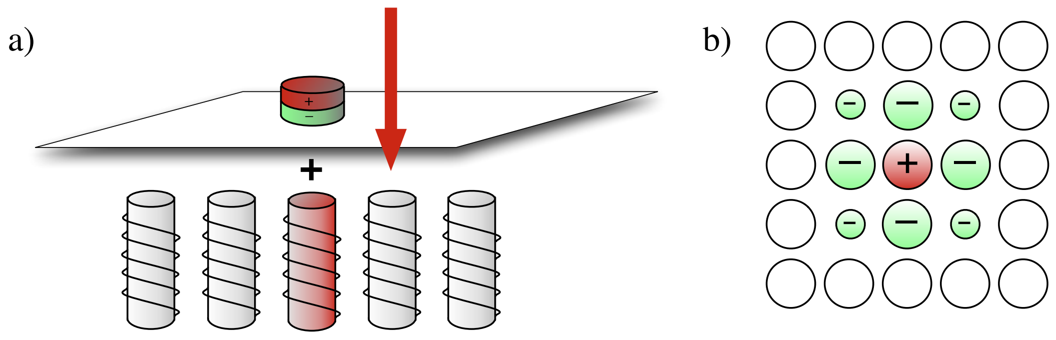


Figure 4.19: Conventions of actuation algorithm. a) The negative pole of a Madget's permanent magnet, which is attracted due to positive electromagnetic polarization, always faces downwards. Red arrow denotes force vector. b) Example array configuration diagram illustrating an attracting electromagnet in the center, surrounded by four repelling ones. Circle size represents strength of electromagnet.

always attached to Madgets so that their negative pole points downwards (Fig. 4.19a).

In the following, we will illustrate various configurations of electromagnetic subarrays. We show them as a matrix of circles, where each circle represents an electromagnet (Fig. 4.19b). Inactive magnets are shown as white circles. Positively polarized electromagnets are red circles containing a "+" sign. Negatively polarized electromagnets are denoted with a "−" sign in a green circle. The size of the cycle denotes illustrates the duty cycle, i.e., the strength of the electromagnet.

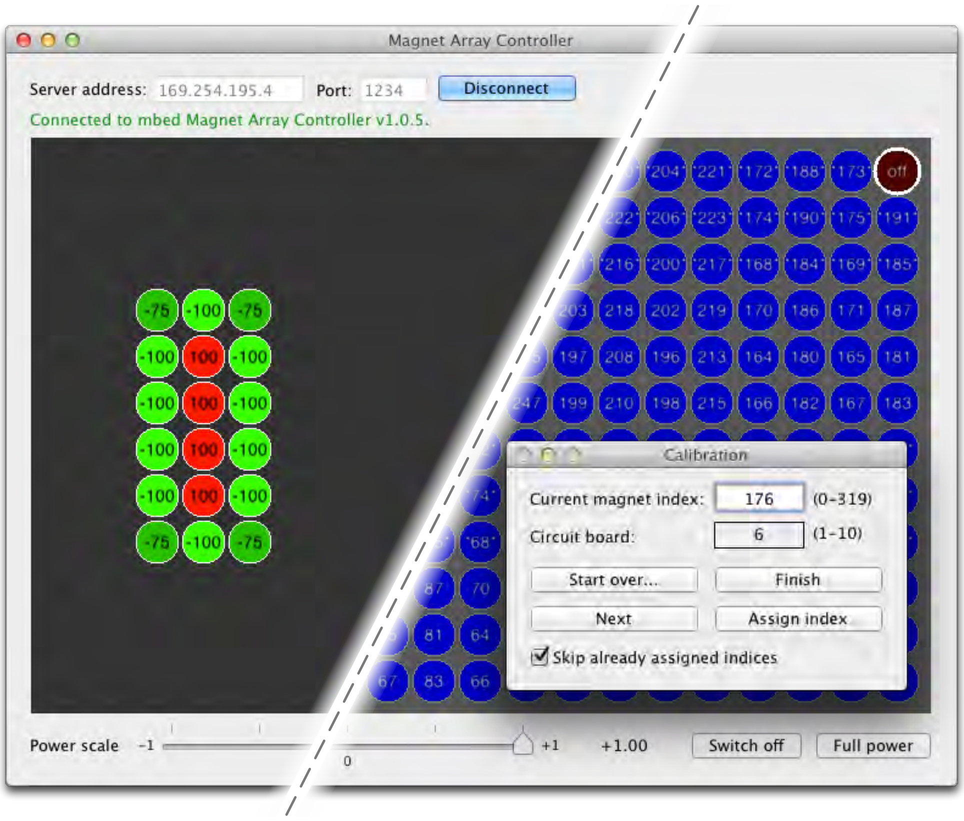
4.5.3 Calibration

Magnet Array Controller application provides interface for calibration process.

The mbed controller can access up to 320 indices, but only a subset of them points to electromagnets. Once the hardware is assembled, a one-time calibration step is necessary to map indices to distinct electromagnets in the array. The *Magnet Array Controller* application allows to specify this mapping (Fig. 4.20). The calibration requires a reference permanent magnet whose poles are designated.

Calibration: Indices are triggered, and user finds activated electromagnet and polarization by hovering reference permanent magnet.

After the user has started the calibration procedure, the system subsequently triggers all indices from 0 to 319 with a positive polarization. For every index, the user first checks whether an electromagnet is activated. An active magnet increases the power consumption, which is displayed on the power supply, and produces a subtle high-frequency sound. If an electromagnet is active, the user has to find its position by hovering the reference magnet above the array. After that he can click on the corresponding magnet represented in our software and assign the index to this magnet. Due to variations in manufacturing, the polarization of the electromagnet might be flipped. If the user detects that the negative pole of the reference magnet is not attracted, he can click on the magnet in our software and flip its polarization via a menu item. The result of the calibration is an array with


Magnet Array Controller

Server address: 169.254.195.4 Port: 1234

Connected to mbed Magnet Array Controller v1.0.5.

Calibration

Current magnet index: 176 (0-319)

Circuit board: 6 (1-10)

☑ Skip already assigned indices

Power scale -1 0 +1 +1.00

Figure 4.20: The Magnet Array Controller application provides a direct interface with the mbed controller. Left part: In default mode, the user can set the polarization and strength of each electromagnet using simple mouse or menu commands. Right part: In calibration mode, the user assigns PWM indices to electromagnets and flips incorrect polarizations.

19 × 12 entries corresponding to electromagnets in row-wise order. Each entry contains the index that accesses the magnet (or -1 if the magnet is not available) and a Boolean flag whether the polarization must be switched.

Calibration process extensive but needs to be performed only once.

This calibration process is time consuming and requires about 30–45 minutes for the entire table. It is easier if two persons perform it collaboratively: One person controls the calibration software, the other one searches for activated electromagnets. However, the table must only be calibrated a single time after assembly. Furthermore, we added a few features to simplify and speed up the process. First, an overlay can be displayed on the LCD panel of the table. It shows which magnets are already calibrated and helps to identify the row and column of the electromagnet that is currently active. As all driver boards are often wired and populated in the same way, the index schemes often repeat among boards in steps of 32 indices. Therefore, we added a "copy & paste scheme" function: A user can mark a rectangle of electromagnets, copy the scheme, and paste it with an offset at another position. We also provide a feature to quickly swap two indices. This can be helpful, if a scheme is pasted for a driver board that is slightly modified.

Copy & Paste functions speed-up process.

To test the calibration, the user places the reference magnet with the downfacing negative pole above an electromagnet in a predefined corner. Then

Calibration test: Single permanent magnet is actuated along entire surface.

he starts the “Successive rectangle test”. This test subsequently drags the magnet to every electromagnet on the table, triggering a single electromagnet in each step. The test starts in the corner of the table and proceeds every second to the next magnet in the row. Once a full row is tested, it continues with the next row in the opposite direction. If the reference magnet reaches the opposite corner, the table is correctly calibrated. If it stops in the meantime, the next electromagnet is probably not calibrated correctly. A reference magnet that jumps away from a position indicates that the polarization of an electromagnet is not adjusted correctly.

Magnet Array Controller provides interface to control electromagnets.

Once the table is calibrated, the *Magnet Array Controller* provides a simple GUI that allows users to activate single or multiple electromagnets and to change their polarization. They can also store and load array configurations. The program also provides functionality to conduct user tests that will be introduced in the next chapter.

4.5.4 Actuation Algorithm

Actuation bases on attracting and repelling electromagnetic forces.

Our algorithm actuates Madgets on the surface by applying magnetic fields to their attached permanent magnets. The actuation is based on two atomic forces: Attracting fields drag a Madget or parts of it into a certain direction, and repelling fields push it away. This section explains the algorithm that actuates a Madget on the tabletop. We begin with explaining existing approaches in literature and their limitations, followed by an overview and a detailed mathematical description of our technique.

4.5.4.1 From Single Pucks to Multi-Element Controls

The actuation of single small magnetic objects using electromagnetic arrays has been well investigated by related literature. The Actuated Workbench describes a variety of approaches to move magnetic pucks across the surface [Pangaro et al., 2002].

Manhattan Motion moves a magnetic puck to adjacent electromagnets in x- or y-direction, consecutively.

The simplest mechanism is the so-called *Manhattan Motion*. To move a magnetic puck to a discrete position on the surface, the puck is always attracted with full power to the closest electromagnet in x- or y-direction that decreases the distance to the target position (Fig. 4.21b). If the puck moves on top of an electromagnet, the next electromagnet is triggered. The name of the technique is derived from the behavior to attract the magnet either in horizontal or vertical direction until it reaches the destination. If only a single puck is placed on the table, a similar technique even works without tracking: By subsequently activating rows and columns of magnets from the boundaries towards the target position, the puck approaches that position (Fig. 4.21c). Manhattan Motion is a fast technique that is easy to implement. However, the achieved movement is jerky, and destinations between magnets cannot be reached.

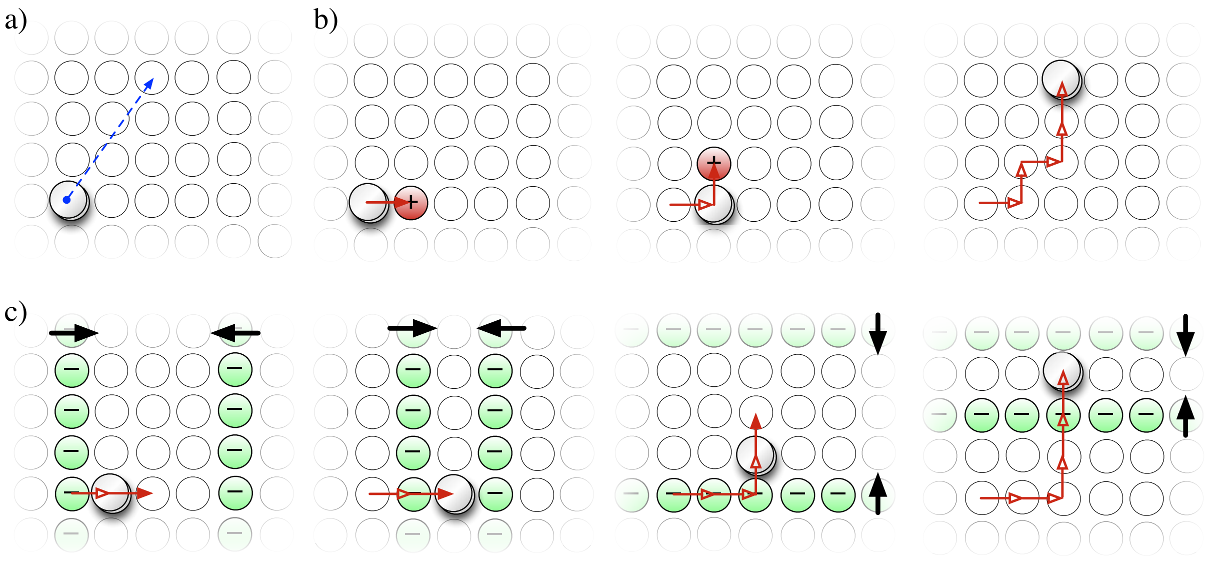


Figure 4.21: Actuating a single magnetic puck via Manhattan Motion. a) The puck shall be moved to a new position (blue arrow). b) If the initial position of the puck is known, it is actuated to the destination position by consecutively attracting it to the closest magnet in x- or y-direction towards the final position. c) A puck can also be moved by converging vertical and horizontal repelling lines of electromagnets. No tracking is required, but this approach only works for a single puck on the table.

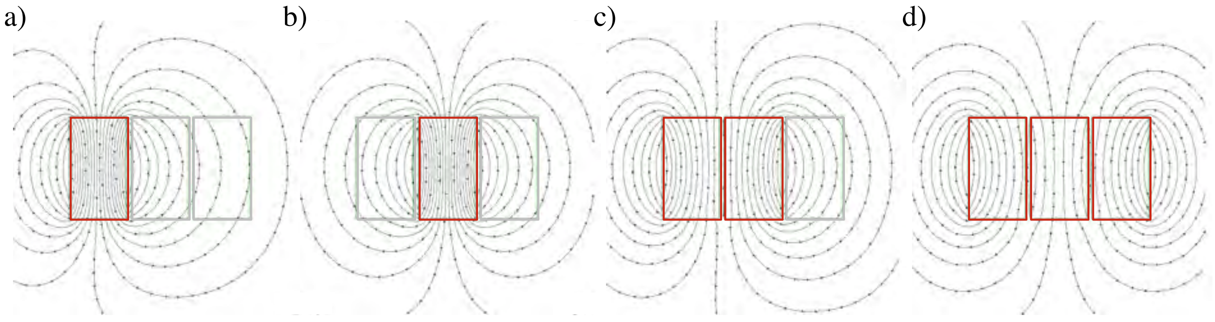


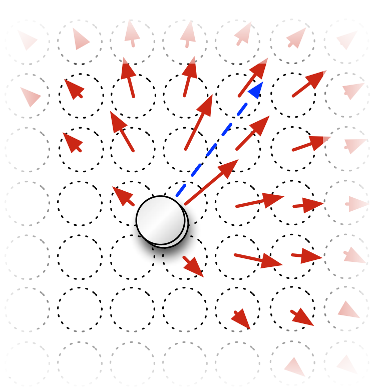
Figure 4.22: Illustration of flux lines depending on magnet activation. Red frames denote active, gray frames disabled magnets. a-b) If only a single magnet is activated, flux lines are centered at the magnet's core. c-d) If multiple magnets are activated, the flux lines move to the interpolated position. Image generated using ViziMag software [Beeteson, 2001]. Adapted version from [Pangaro et al., 2002].

For a continuous movement Pangaro et al. [2002] employ the principle of superposition. If multiple electromagnets are activated, the resulting force upon a permanent magnet equals the sum of forces that individually activated electromagnets would apply to that magnet. Although the interaction between electromagnets in the array is more complex, and this assumption is not absolutely correct, it is a valid approximation as shown in Fig. 4.22. By combining the forces of multiple electromagnets, the puck can be smoothly dragged into arbitrary directions and to "sub-magnet" positions on the surface. Inspired from antialiasing techniques in the field of Computer Graphics, the authors provide two smooth actuation mechanisms: *"Jet"-based antialiasing* combines attracting forces of many electromagnets

Principle of electromagnetic superposition allows continuous actuation.

Attraction into directions different than target direction smoothens the movement.

a)



b)

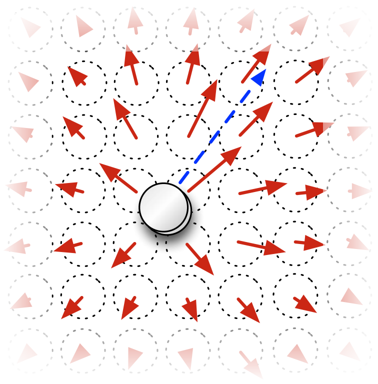


Figure 4.23: Antialiasing mechanisms for smoothly actuating a puck into a certain direction (blue arrow) according to Pangaro et al. [2002]. a) When using "Jet"-based antialiasing, all electromagnets in the direction of movement are activated but scaled according to the projection onto the movement vector. b) When using "Dot"-based antialiasing, electromagnets in the opposite direction are activated as well. All active electromagnets in the images produce attracting fields.

in the target direction of the puck (Fig. 4.23a). *"Dot"-based antialiasing* adds small attracting forces in the opposite direction (Fig. 4.23b). Albeit slower, the movement of the latter technique is smoother.

Linear Induction Motors move objects alongside a traveling magnetic field.

Another technique to move simple objects is presented by Yoshida et al. [2006]. The authors combine the principle of two Linear Induction Motors (LIM) [Noma et al., 2004] to create traveling magnetic fields on the surface. In contrast to the before mentioned approaches, a Linear Induction Motor powers the electromagnets with alternating current (instead of PWM) and a constant phase difference between coils in the desired direction. This induces current to a conductive non-magnetic object and creates force into the traveling direction. Figuratively speaking, the objects move on the bow wave of the magnetic fields.

Existing approaches work for pucks but cannot actuate complex controls.

All these approaches are suitable to smoothly translate rigid pucks on the tabletop. However, they do not allow to rotate objects, and arranging complex controls like Madgets, which are not circular and contain multiple elements, is not feasible with these techniques.

A Madget can consist of multiple rigid bodies that are connected by joints. For example, a slider consists of a base that is connected to a sliding knob via a linear joint. A knob combines a rigid base with a turnable arm via a rotational joint. Every rigid body is mounted on one or multiple permanent magnets. The base of a Madget contains at least two magnets to allow for rotational actuation; however, usually more magnets are attached to distribute the required actuation force to multiple magnets. Moving parts inside a Madget require as many magnets as they have degrees of freedom. Therefore, a sliding knob or a turning arm needs a single magnet for actuation.

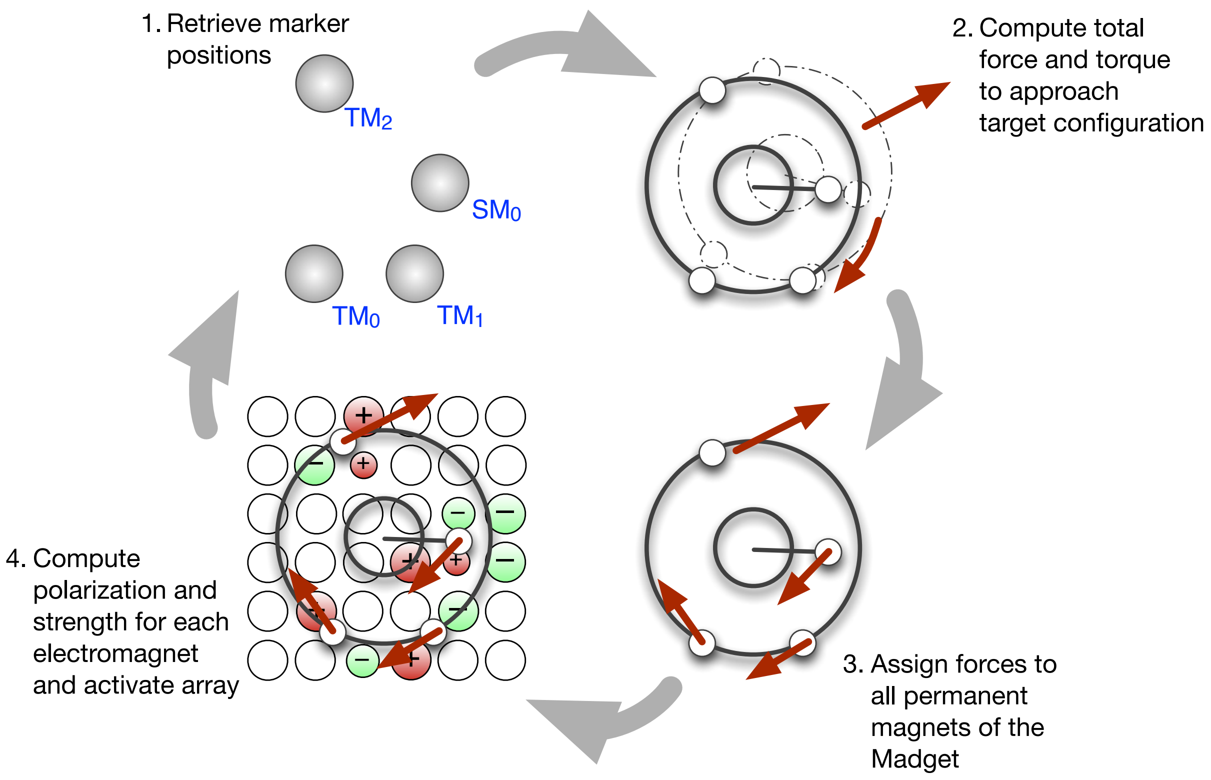

1. Retrieve marker positions
TM$_2$
SM$_0$
TM$_0$
TM$_1$
2. Compute total force and torque to approach target configuration
3. Assign forces to all permanent magnets of the Madget
4. Compute polarization and strength for each electromagnet and activate array
+
−

Figure 4.24: Control loop of actuation algorithm. The process is executed until the Madget reaches its target position and orientation.

All permanent magnets of a Madget inherently depend on each other, because they are connected to the same body. If we actuate a single permanent magnet, all other magnets on that body move as well (but not necessarily into the same direction). Also, every activated electromagnet influences every permanent magnet on the table. Our actuation algorithm must take the exact physical configuration of each control into account and consider the interdependencies of permanent magnets.

Permanent magnets of a Madget depend on each other.

Even though actuating complex controls is difficult, it enables a new actuation dimension: height. Parts of a Madget can be raised from the table using a repelling magnetic field. This is only stable if the part is embedded into a control that constraints the horizontal motion. Our algorithm must take this into account.

4.5.4.2 Overview

Our actuation algorithm begins with the request to actuate a Madget to a new position and/or to rotate it to a certain target angle. Depending on the Madget, the user can optionally specify a new inner state, e.g., a new rotation of the knob's arm.

Actuation starts with request to move and configure Madget.

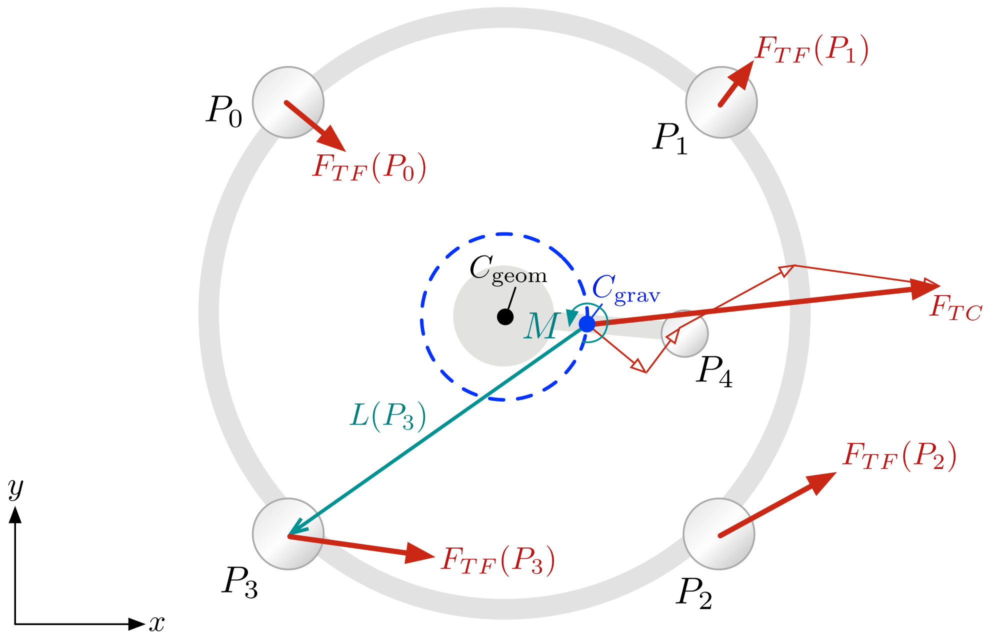


Figure 4.25: Forces and torque that are applied to a Knob Madget and its permanent magnets during an actuation step. Note that the center of gravity C_{grav} rotates around the geometric center C_{geom} depending on the angle of the knob's arm.

Our closed loop actuation algorithm now repeats the following steps, until the Madget reaches the target configuration (Fig. 4.24):

1. Retrieve the exact position of all permanent magnets from the tracking algorithm (section 4.6). Compute the current velocity and angular velocity of the Madget.
2. Compute the total force and torque that is required to actuate the Madget to the next position, orientation, and state (section 4.5.4.4).
3. Compute force vectors for all permanent magnets to achieve the total force and torque (section 4.5.4.5).
4. For each electromagnet in the array, compute the polarization and strength that is required to generate these forces (section 4.5.4.6).

4.5.4.3 Model

In this section, we derive the model of our actuation algorithm, after introducing the terminology and variables. We refer to Fig. 4.25 for an illustration of the force and torque values that will be discussed in the following.

Model is based on rigid body dynamics.

Our model is based on rigid body dynamics. Accordingly, multiple forces and torques that are applied to various points of the rigid body can be combined to a single force and torque applied to the center of gravity (COG).

The *position* of a Madget is defined by its projected 2D geometric center C_{geom}, i.e., the center of its 2D bounding box, because this is usually constant over time. The projected 2D center of gravity C_{grav} may vary according to a Madget's physical configuration. For example, the COG of a knob in Fig. 4.25 depends on the position of the rotating arm due to the changing weight distribution. The *absolute rotation angle* of a Madget in the horizontal plane is measured clock-wise against the y-axis vector $(0, 1)$.

Madget's position is defined with respect to geometric center, rotation with respect to y-axis vector.

We update the electromagnetic array with about 30 actuation frames per second (fps). In each frame, the Actuation Framework computes polarizations and strengths for all electromagnets. Then, the array triggers the electromagnets with this configuration for the entire frame length until the new configuration from the next frame is available. We denote the duration of a frame with $t_{\text{af}} \approx 0.033$ s.

Array is updated with 30 fps.

We denote electromagnets in the array with E_i with $i \in \{0, 1, ..., 227\}$. Without loss of generality, we assume that the indices are ordered line by line. For simplicity, we assume that a single Madget is actuated. Generalizing to multiple Madgets is straight forward, as will be explained later. We denote permanent magnets on that Madget with P_j with $j \in \{0, 1, ..., n_P - 1\}$ and n_P representing the total number of permanent magnets on the Madget. For further simplification, we assume that a Madget contains two types of permanent magnets: *Static magnets* belong to the rigid base of the Madget, *dynamic magnets* denominate all other magnets with at least one degree of freedom in respect to the base. $\text{Pos}(E_i)$ and $\text{Pos}(P_j)$ denote the 2D position of an electromagnet or permanent magnet on the surface, respectively. It is measured in cm and assumed as the projection of the magnet's center onto the local coordinate system of the table surface that is, without loss of generality, defined by the bottom left corner of the surface.

Static magnets belong to rigid base; dynamic magnets have at least one degree of freedom with respect to base.

For simplification, we only consider a Madget's static magnets for actuating it a new position in the following. Electromagnets E_i apply forces to these permanent magnets to actuate the Madget to a new position and absolute rotation angle.

We differentiate between *tangential* and *normal forces*. Tangential forces pull objects in horizontal direction across the surface plane. Analogously, normal forces push towards or away from the surface.

We differentiate between tangential and normal forces.

We first use a reference permanent magnet P_{Ref} with fixed properties to introduce all variables and later generalize to arbitrary magnets. Following our convention, the reference magnet's negative pole always faces downwards.

Forces are expressed with respect to reference permanent magnet.

With $F_{T\text{-max}}(P_{\text{Ref}})$ we denote the measured amount of tangential force that is required to drag the reference magnet P_{Ref} horizontally away from the position centered above a fully powered electromagnet (Fig. 4.26). Analogously, $F_{N\text{-max}}(P_{\text{Ref}})$ is the amount of normal required force required to

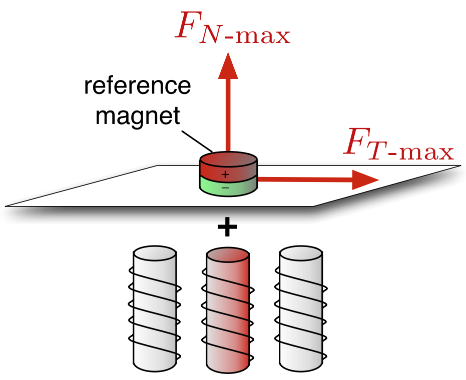


Figure 4.26: Measurement of bounds for normal and tangential forces using a reference magnet.

lift that magnet away from a fully activated electromagnet. Note that both values are scalars. All forces are declared in Newton.

$F_T(E_j)$ and $F_N(E_j)$ is the amount of tangential and normal force, respectively, that an electromagnet E_j with variable polarization and strength exerts on the reference permanent magnet P_{Ref} centered above E_j. The tangential force is bounded by

Bounds of tangential force

$$-F_{T\text{-max}}(P_{\text{Ref}}) \leq F_T(E_j) \leq F_{T\text{-max}}(P_{\text{Ref}}). \tag{4.1}$$

Note that $F_T(E_j)$ is a one-dimensional variable, whose sign indicates the polarization. A negative value denotes a repelling, a positive one an attracting force. The normal force $F_N(E_j)$ exerted by an electromagnet is approximately proportional to the tangential one. Thus, we write the normal force of an electromagnet as

Normal force is proportional to tangential one.

$$F_N(E_j) \quad = \quad \theta \cdot F_T(E_j)$$

where

$$\theta \quad := \quad \frac{F_{N\text{-max}}(P_{\text{Ref}})}{F_{T\text{-max}}(P_{\text{Ref}})}$$

describes the ratio between the maximum normal and tangential force applied by an electromagnet to the reference magnet. To consider permanent magnets with *different* properties than the reference magnet, we introduce two further ratios:

Scale factors map from reference to different permanent magnets.

$$\sigma_T(P_i) \quad := \quad \frac{F_{T\text{-max}}(P_i)}{F_{T\text{-max}}(P_{\text{Ref}})},$$
$$\sigma_N(P_i) \quad := \quad \frac{F_{N\text{-max}}(P_i)}{F_{N\text{-max}}(P_{\text{Ref}})}.$$

These factors describe how the required tangential and normal forces are scaled when a different permanent magnet P_i is used. Due to the convention that magnetic poles attracted by positive polarizations always point downwards (section 4.5.2), these factors are always positive. Note that we must distinguish between a tangential and normal factor, because the

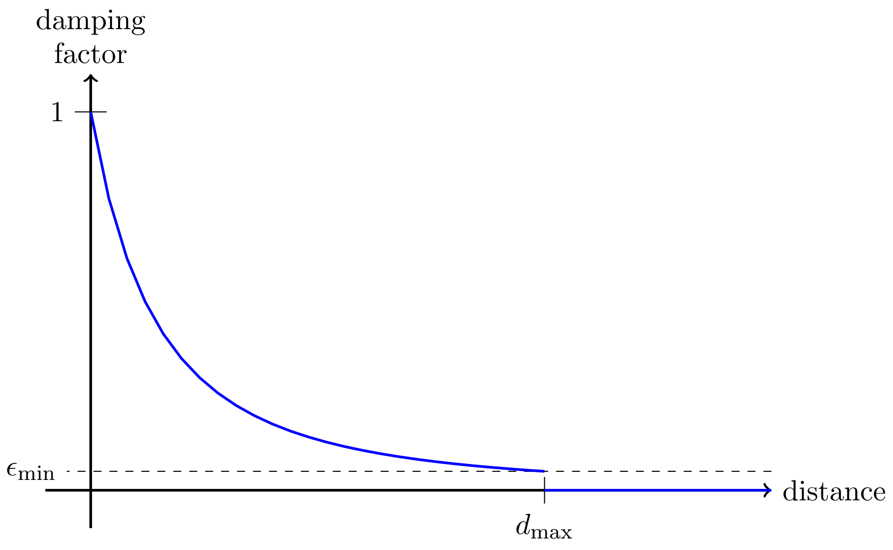


Figure 4.27: Damping of electromagnet's force depending on distance. Beyond the maximal relevant distance d_{max}, the damping factor is clamped to 0 (no influence).

size, form, and material properties of permanent magnets cause a different distribution of forces in both directions.

From a far distance, a magnet can be approximated as a magnetic dipole, whose field strength attenuates with the cube of the distance. Near the magnet, the fall off is more complicated: According to Biot-Sarvart law, it requires the evaluation of a line integral over the entire wire. Basically, this sums up the influence of every point on the wire to the distinct position. We discovered that a quadratic attenuation is appropriate to model attenuation near the magnet, an approximation that is also used by Pangaro et al. [2002]. Let

Distance between electromagnetic and permanent magnet

$$d(E_j, P_i) \quad := \quad \|\text{Pos}(E_j) - \text{Pos}(P_i)\|$$

be the Euclidian distance between an electromagnet E_j and a permanent magnet P_i. For performance reasons, we assume that the influence of an electromagnet beyond a maximum distance $d_{\text{max}} > 0$ is so small that it can be neglected. Assuming that the relative influence of an electromagnet at d_{max} equals ϵ_{min}, we define the damping function $\epsilon(x) : [0, \infty) \to [0, 1]$ with

Quadratic damping function

$$\epsilon(x) \quad := \quad \begin{cases} \dfrac{1}{\left(1 + \frac{x}{d_{\text{max}}} \cdot (\sqrt{\epsilon_{\text{min}}^{-1}} - 1)\right)^2} & x \leq d_{\text{max}}, \\ 0 & \text{otherwise.} \end{cases}$$

ϵ_{min} must be > 0, because the influence of an electromagnet to a distant magnet is theoretically never 0. Note that this damping function is just a clamped reparametrization of $f(x) = \frac{1}{x^2}$, so that $\epsilon(0) = 1$ and $\epsilon(d_{\text{max}}) = \epsilon_{\text{min}}$.

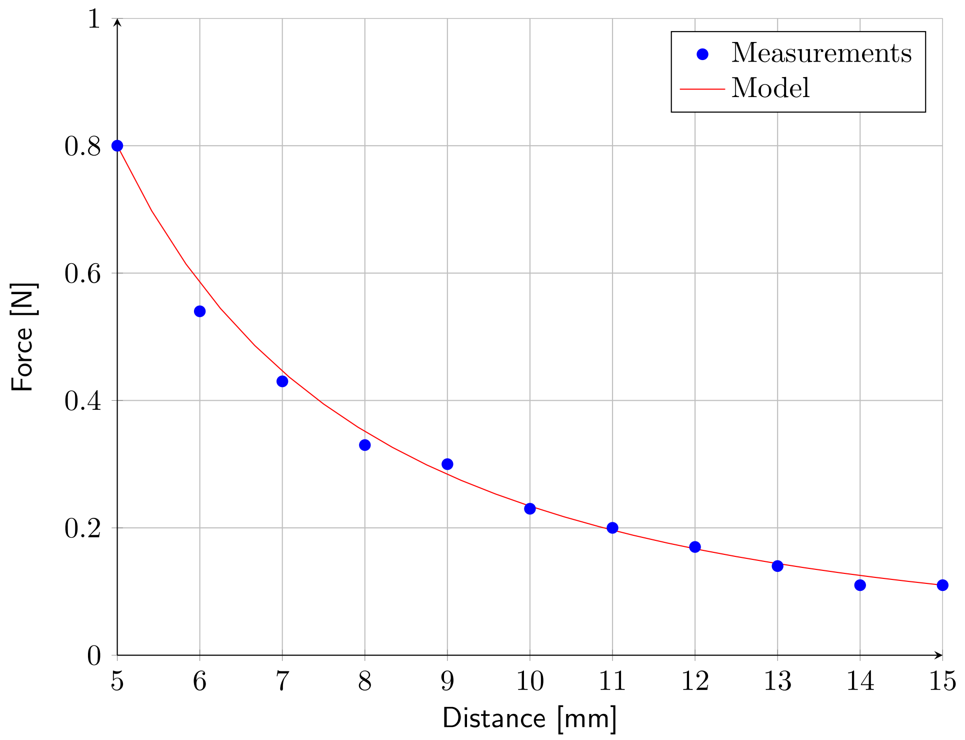


Figure 4.28: Adhesive force of an electromagnet from our array depending on distance to core. Single freely suspended electromagnet with iron core was driven at 40 V. Adhesive force equals the force that was required to detach a Neodymium permanent magnet (⌀ 10 mm, height 2 mm) from the top of the core. A 5 mm acrylic plate was placed between core and permanent magnet. Force was measured with a spring scale. The red curve fits our quadratic damping model with the measurements, i.e., $f(x) = 0.8 \cdot (1 + (x-5) \cdot 0.16968)^{-2}$ with $f(5) = 0.8$ and $f(15) = 0.11$.

Fig. 4.27 illustrates the damping function. Fig. 4.28 shows actual force measurement plots compared to our damping model.

Finally, we can compose the amount of normal force that a single electromagnet E_j with variable strength and polarization applies to a permanent magnet P_i in arbitrary distance:

Normal force of an electromagnet on a permanent magnet

$$F_N^*(E_j, P_i) \quad = \quad \underbrace{F_N(E_j)}_{\substack{\text{normal force of electromagnet}\\ \text{depending on strength}\\ \text{and polarization}}} \cdot \underbrace{\sigma_N(P_i)}_{\substack{\text{factor depending}\\ \text{on specific}\\ \text{permanent magnet}}} \cdot \underbrace{\epsilon(d(E_j, P_i))}_{\substack{\text{distance}\\ \text{damping}}}$$

Assuming the principle of superposition, the total *normal* force of multiple electromagnets affecting a single permanent magnet reads

Normal force of multiple electromagnets on permanent magnet

$$F_N(P_i) = \sum_j F_N^*(E_j, P_i) = \sum_j F_N(E_j) \cdot \sigma_N(P_i) \cdot \epsilon(d(E_j, P_i)). \qquad (4.2)$$

Without considering friction, the total *tangential* force applied by multiple electromagnets reads (cf. Fig. 4.30 on page 138)

$$F_T(P_i) = \sum_j \underbrace{F_T(E_j)}_{\text{tangential force of electromagnet}} \cdot \underbrace{\sigma_T(P_i)}_{\text{factor depending on specific permanent magnet}} \cdot \underbrace{\epsilon(d(E_j, P_i))}_{\text{distance damping}} \cdot \underbrace{\frac{\text{Pos}(E_j) - \text{Pos}(P_i)}{d(E_j, P_i)}}_{\text{normalized direction vector}} . \tag{4.3}$$

Tangential force of multiple electromagnets on permanent magnet

Note that $F_T(P_i)$ is a two-dimensional force vector in the plane of the tabletop.

The friction between a Madget and the acrylic surface decreases the tangential force that an electromagnet applies to each permanent magnet. In general, two kinds of friction forces are distinguished. The *static friction force*

$$F_{\text{st}} = \mu_{\text{st}} \cdot F_N \tag{4.4}$$

Static friction

is the force that must be exceeded to bring an object A on top of a surface B into motion. F_N denotes the normal force pointing downwards onto the surface. μ_{st} is a empirically determined *friction coefficient* between the two materials A and B. Once an object is in motion, the force

$$F_{\text{dyn}} = \mu_{\text{dyn}} \cdot F_N \tag{4.5}$$

Dynamic friction

is the minimum force required to keep the object moving.

In case of a permanent magnet P_i, the normal force depends on the weight and the magnetic normal force $F_N(P_i)$. Although the mass of a Madget is constant, the distribution of this mass for each permanent magnet is different and depends on the physical state of the Madget. Let $m(P_i)$ be the mass in gram distributed to the permanent magnet P_i. The static and dynamic friction force at this permanent magnet reads

$$F_{\text{st/dyn}}(P_i) := \max\Big\{ 0,\ \mu_{\text{st/dyn}} \cdot (\underbrace{m(P_i) \cdot g}_{\text{weight force}} + \underbrace{F_N(P_i)}_{\text{magnetic attraction force}}) \Big\},$$

Static/dynamic friction force at permanent magnet

with $g = 9.81 \frac{\text{m}}{\text{s}^2}$ denoting the standard gravity. Remember that $F_N(P_i)$ can also be negative. Assuming that the tangential force overcomes friction force, the final tangential force function including friction reads

$$F_{TF}(P_i) = \underbrace{\big(\|F_T(P_i)\| - F_{\text{st/dyn}}(P_i)\big)}_{\text{amount of magnetic force reduced by friction}} \cdot \underbrace{\frac{F_T(P_i)}{\|F_T(P_i)\|}}_{\text{normalized force vector}} . \tag{4.6}$$

Tangential force of multiple electromagnets on permanent magnet including friction

Our algorithm switches from $F_{\text{st}}(P_i)$ to $F_{\text{dyn}}(P_i)$ as soon as an object starts moving.

Finally, the total tangential force applied to the Madget's center of gravity C_{grav} reads (Fig. 4.25)

$$F_{TC} = \sum_i F_{TF}(P_i). \tag{4.7}$$

Total tangential force at COG

The total tangential torque M applied to the Madgets equals the sum of the z-components of the vector cross products between the tangential forces and levers of all permanent magnets. As we only consider motion in the plane, we retrieve the torque around the z-axis (cf. Fig. 4.25 again)

Total tangential torque

$$M = \sum_i L(P_i)_x \cdot F_{TF}(P_i)_y - L(P_i)_y \cdot F_{TF}(P_i)_x \tag{4.8}$$

where

Lever at permanent magnet

$$L(P_i) = \mathrm{Pos}(P_i) - C_{\mathrm{grav}}$$

is the 2D lever of permanent magnet P_i in the surface plane, and the subscripts $_x$ and $_y$ denote the x- and y-component of the vector, respectively.

4.5.4.4 Step 1: Compute Total Tangential Force and Torque

Programmer specifies target center position and angle.

Let us assume that a Madget is centered at C_{geom} with an absolute angle of α_{cur}. To actuate a Madget to a new position and orientation, the programmer specifies the Madget's target center position C'_{geom} and target angle α_{tar}. He also adjusts the actuation velocity of translation v_{tar} and rotation ω_{tar}. In a single frame of length t_{af}, the object is translated by

Offset of geometric center ...

$$\Delta C_{\mathrm{geom}} = \frac{C'_{\mathrm{geom}} - C_{\mathrm{geom}}}{\|C'_{\mathrm{geom}} - C_{\mathrm{geom}}\|} \cdot v_{\mathrm{tar}} \cdot t_{\mathrm{af}}$$

... and angle change depending on velocity and time.

and rotated by

$$\Delta\alpha = \mathrm{sgn}(\alpha_{\mathrm{tar}} - \alpha_{\mathrm{cur}}) \cdot \omega_{\mathrm{tar}} \cdot t_{\mathrm{af}}.$$

Geometric center must be converted to COG in every frame.

Note that our physical model bases on the center of gravity. Therefore, ΔC_{geom} is transformed into the offset center of gravity ΔC_{grav} in every frame.

In the first step of an actuation cycle, our algorithm assigns target force vectors to all permanent magnets that actuate the Madget to the next position and orientation in t_{af} seconds. Now assume that the magnet is *currently* moving with 2D velocity v_{cur} and angular velocity ω_{cur}. If we now apply the 2D tangential force F_{TC}, the Madget is accelerated with $\frac{F_{TC}}{m}$ where m is the measured mass of the widget. A torque of M yields a angular acceleration of $\frac{M}{J}$ with J as the Madget's mass moment of inertia. After an actuation frame of t_{af} seconds, the new velocity reads

Updated velocity and angular velocity depending on tangential force and measured mass

$$v_{\mathrm{new}} = v_{\mathrm{cur}} + \frac{F_{TC}}{m} \cdot t_{\mathrm{af}} \tag{4.9}$$

and the new angular velocity amounts to

$$\omega_{\mathrm{new}} = \omega_{\mathrm{cur}} + \frac{M}{J} \cdot t_{\mathrm{af}}. \tag{4.10}$$

The configuration of the electromagnetic array is determined in the beginning of the actuation frame and then stays fixed until the next frame.

Accordingly, both acceleration values are constant during the time interval t_{af}. Thus, we can assume that the velocity of a Madget changes linearly during the frame.

Velocity and angular velocity change linearly during frame.

Thus, the Madget's center of gravity has moved by

$$\Delta C_{\text{grav}} = \frac{v_{\text{cur}} + v_{\text{new}}}{2} \cdot t_{\text{af}} \tag{4.11}$$

and rotated by

$$\Delta\alpha = \frac{\omega_{\text{cur}} + \omega_{\text{new}}}{2} \cdot t_{\text{af}} \tag{4.12}$$

after the actuation frame.

To compute the required force and torque, we must regard the current velocity v_{cur} and angular velocity ω_{cur}. By combining equations 4.9 and 4.11, we compute the total tangential force for this frame:

Total tangential force and torque for single frame depending on velocity and time

$$F_{TC} = 2 \cdot m \cdot \left(\frac{\Delta C_{\text{grav}}}{t_{\text{af}}^2} - \frac{v_{\text{cur}}}{t_{\text{af}}} \right).$$

Analogously, we retrieve the total torque by combining equations 4.10 and 4.12:

$$M = 2 \cdot J \cdot \left(\frac{\Delta\alpha}{t_{\text{af}}^2} - \frac{\omega_{\text{cur}}}{t_{\text{af}}} \right).$$

4.5.4.5 Step 2: Assign Forces to Permanent Magnets

We now have to assign forces to permanent magnets so that the total tangential force and torque are achieved. More precisely, we are searching for $F_{TF}(P_i)$ for all permanent magnets P_i so that equations 4.7 and 4.8 are fulfilled:

Linear system of equations for forces applied to permanent magnets depending on total tangential force and torque

$$\begin{aligned} & F_{TC} = \sum_i F_{TF}(P_i) \\ \wedge \quad & M = \sum_i L(P_i)_x \cdot F_{TF}(P_i)_y - L(P_i)_y \cdot F_{TF}(P_i)_x. \end{aligned}$$

These equations yield a linear system of equations that is usually under-determined. Therefore, we apply further constraints to retrieve an optimal solution. To reduce power consumption, the ideal solution would minimize the sum of forces applied to permanent magnets:

System is under-determined. Optimal solution is non-linear.

$$\sum_i \|F_{TF}(P_i)\| \rightarrow \min.$$

However, this introduces a non-linear term which is difficult to optimize. Instead, we provide three constraints that allow for an efficient optimization:

Force vectors scaled versions of total tangential force

1. **Optimize for translation.** All force vectors at permanent magnets point into the direction of the total tangential force F_{TC}

$$F_{TF}(P_i) = \rho_i \cdot F_{TC}$$

with $\rho_i \in \mathbb{R}$.

Force vectors orthogonal to their lever

2. **Optimize for rotation.** All force vectors point into the direction orthogonal to their lever

$$F_{TF}(P_i) = \rho_i \cdot \frac{\perp L(P_i)}{\|\perp L(P_i)\|}$$

with $\perp L(P_i) = (-L(P_i)_y, L(P_i)_x)$ and $\rho_i \in \mathbb{R}$.

Both constraints can be solved with linear programming using

$$\sum_i |\rho_i| \quad \rightarrow \quad \min$$

as objective function and equations 4.7 and 4.8 as side conditions. The objective function minimizes the sum of absolute forces. However, due to the restriction of the solution space, only a local minimum is found that can involve peak forces at single electromagnets. As an alternative, we propose our third constraint:

Minimize squared lengths of tangential force vectors

3. **Optimize for least-squares forces.** Minimizes the sum of squared forces at all permanent magnets:

$$\sum_i \left(F_{TF}(P_i)_x^2 + F_{TF}(P_i)_y^2\right) \rightarrow \min. \tag{4.13}$$

Penalty of high forces on single electromagnets prevents overheat.

Due to the squaring of force, this constraint penalizes high forces at single permanent magnets. This is essential to prevent overheating of single electromagnets (see next section).

Lagrange multipliers yield linear system of equations.

We convert equation 4.13 into a linear system of equations by adding Lagrange multipliers, using equations 4.7 and 4.8 as side conditions:

$$\begin{aligned}\Lambda \quad = \quad & \sum_i \left(F_{TF}(P_i)_x^2 + F_{TF}(P_i)_y^2\right) \\ & + \psi_1 \cdot \Big(F_{TCx} - \sum_i F_{TF}(P_i)_x\Big) \\ & + \psi_2 \cdot \Big(F_{TCy} - \sum_i F_{TF}(P_i)_y\Big) \\ & + \psi_3 \cdot \Big(M - \sum_i \big(L(P_i)_x \cdot F_{TF}(P_i)_y - L(P_i)_y \cdot F_{TF}(P_i)_x\big)\Big).\end{aligned}$$

Partial derivatives yield the linear system of equations

$$\begin{aligned}
\frac{\delta\Lambda}{\delta F_{TF}(P_i)_x} &= & 2\cdot F_{TF}(P_i)_x - \psi_1 + \psi_3 \cdot L(P_i)_y = 0 \quad \forall i\\
\frac{\delta\Lambda}{\delta F_{TF}(P_i)_y} &= & 2\cdot F_{TF}(P_i)_y - \psi_2 - \psi_3 \cdot L(P_i)_x = 0 \quad \forall i\\
\frac{\delta\Lambda}{\delta\psi_1} &= & F_{TCx} - \sum_i F_{TF}(P_i)_x = 0\\
\frac{\delta\Lambda}{\delta\psi_2} &= & F_{TCy} - \sum_i F_{TF}(P_i)_y = 0\\
\frac{\delta\Lambda}{\delta\psi_3} &= & M - \sum_i \big(L(P_i)_x \cdot F_{TF}(P_i)_y - L(P_i)_y \cdot F_{TF}(P_i)_x\big) = 0.
\end{aligned}$$

We solve this underdetermined system of equations using linear programming, using the objective function

Solve system using linear programming.

$$\sum_i |F_{TF}(P_i)_x| + |F_{TF}(P_i)_y| \quad \to \quad \text{min.}$$

When moving a Madget across the table, we want to avoid friction that slows down the actuation. Therefore, we set all normal forces of static permanent magnets to zero, i.e., $F_N(P_i) = 0$. Furthermore, we scale up the tangential forces $F_T(P_i)$ to compensate for friction by weight force (equation 4.6).

Friction is compensated by zeroing normal forces and by scaling up tangential forces.

4.5.4.6 Step 3: Distributing Forces to Electromagnets

The last step of the algorithm determines the strength of all electromagnets to create the previously computed tangential and normal forces for every permanent magnet. Since every electromagnet influences all permanent magnets, we cannot just trigger electromagnets next to permanent magnets. As shown in Fig. 4.29, this can easily lead to conflicts: E_0 is triggered to actuate P_0 for turning the knob. However, E_0 also attracts P_1 unintentionally, which would move the control. This movement must be compensated by triggering further electromagnets. Therefore, we have to solve a linear system of equations that regards the dependencies between electromagnets and permanent magnets, and whose solution makes sure that all permanent magnets retrieve the correct force vectors.

Every electromagnet influences every permanent magnet. Systems of equations necessary.

Following equations 4.2 and 4.3, we define force factors:

Force factors for tangential and normal direction

$$\begin{aligned}
f_x(E_j, P_i) &:= \sigma_T(P_i)\cdot\epsilon(d(E_j,P_i))\cdot\frac{\mathrm{Pos}(E_j)_x - \mathrm{Pos}(P_i)_x}{d(E_j,P_i)},\\
f_y(E_j, P_i) &:= \sigma_T(P_i)\cdot\epsilon(d(E_j,P_i))\cdot\frac{\mathrm{Pos}(E_j)_y - \mathrm{Pos}(P_i)_y}{d(E_j,P_i)},\\
f_z(E_j, P_i) &:= \sigma_N(P_i)\cdot\epsilon(d(E_j,P_i))\cdot\theta.
\end{aligned}$$

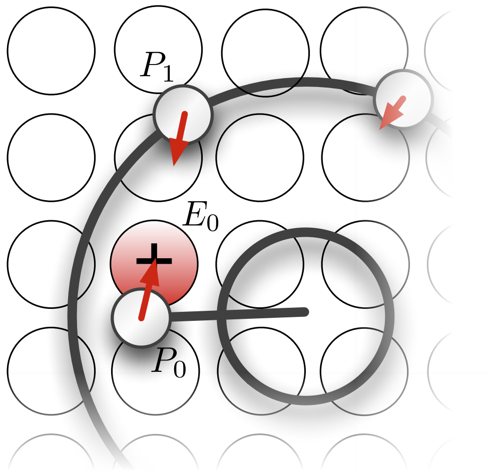


Figure 4.29: Force conflict while actuating a knob. E_0 is triggered to move P_0 for turning the arm, but P_1 is affected as well. Further electromagnets must now be activated to cancel out the Madget's motion while maintaining the desired force on P_1.

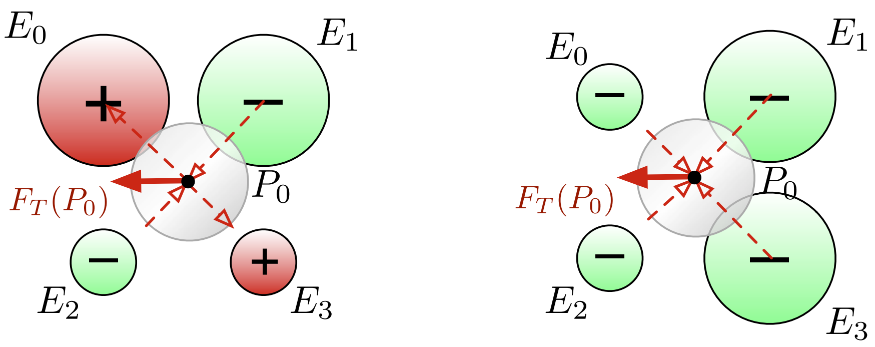


Figure 4.30: Computing the tangential force for a single permanent magnet using super position principle. Both cases result in the same vector but employ different electromagnetic configurations.

The first two force factors represent the amount of force with which an electromagnet E_j affects a permanent magnet P_i in tangential direction, separated for x and y component. $f_z(E_j, P_i)$ denotes the amount in normal direction. These factors only depend on the positions of the permanent magnets and are precomputed for every actuation frame.

The linear system of equations now reads

Linear system to find tangential forces for electromagnets

$$\begin{pmatrix} F_T(P_i)_x \\ F_T(P_i)_y \\ F_N(P_i) \end{pmatrix} = \sum_j \left(F_T(E_j) \cdot \begin{pmatrix} f_x(E_j, P_i) \\ f_y(E_j, P_i) \\ f_z(E_j, P_i) \end{pmatrix} \right) \quad \forall i \qquad (4.14)$$

System is usually under-determined.

where $F_T(E_j)$ are the unknown variables which are bounded by the maximum possible tangential force (equation 4.1). If solvable, this system of equations is usually under-determined, because forces exerted by distinct electromagnets can be substituted by other ones (cf. Fig. 4.30). Therefore, we add an objective function

Objective function minimizes absolute forces.

$$\sum_j \lambda_j \cdot |F_T(E_j)| \quad \rightarrow \quad \min$$

to minimize the absolute force applied by electromagnets. With Linear Programming, we can solve equation 4.14 under the maximum force constraint (equation 4.1) with this objective function.

Solved via Coin-CLP library

We use the *Coin-or linear programming library*[4] (Coin-CLP) to find the solution. It is designed to solve optimization problems in the form of

$$c^T \cdot x \quad \rightarrow \quad \min$$

with

$$\begin{aligned} r_{\text{lower}} &\leq Ax \leq r_{\text{upper}} \\ c_{\text{lower}} &\leq x \leq c_{\text{upper}}. \end{aligned}$$

Let m be the number of permanent magnets and n the number of electromagnets. As will be explained in the following, in our case A is a $3m \times 2n$ matrix, $c, x, c_{\text{lower}}, c_{\text{upper}} \in \mathbb{R}^{2n}$ and $r_{\text{lower}}, r_{\text{upper}} \in \mathbb{R}^{3m}$.

Absolute electromagnetic forces are separated into positive and negative ones.

Since the Coin-CLP solver does not allow absolute values in the objective function, we reformulate our problem. Electromagnets E_j are divided into electromagnets with positive polarizations, E_j^+, and those with negative polarizations, E_j^-. The respective tangential forces can now be expressed with positive numbers bounded by $F_{T\text{-max}}(P_{\text{Ref}})$:

$$\begin{aligned} & 0 \leq F_T(E_j^+) \leq F_{T\text{-max}}(P_{\text{Ref}}) \\ \wedge \qquad & 0 \leq F_T(E_j^-) \leq F_{T\text{-max}}(P_{\text{Ref}}). \end{aligned} \tag{4.15}$$

Accordingly, the final system of equations reads

Final linear system of equations for positively and negatively polarized electromagnets

$$\begin{aligned} \begin{pmatrix} F_T(P_i)_x \\ F_T(P_i)_y \\ F_N(P_i) \end{pmatrix} &= \sum_j \left(F_T(E_j^+) \cdot \begin{pmatrix} f_x(E_j^+, P_i) \\ f_y(E_j^+, P_i) \\ f_z(E_j^+, P_i) \end{pmatrix} \right) \\ &+ \sum_j \left(F_T(E_j^-) \cdot \begin{pmatrix} -f_x(E_j^-, P_i) \\ -f_y(E_j^-, P_i) \\ -f_z(E_j^-, P_i) \end{pmatrix} \right) \quad \forall i \end{aligned} \tag{4.16}$$

where $F_T(E_j^+)$ and $F_T(E_j^-)$ are the unknown variables. Note that the force factors for negative polarized electromagnets are inverted. Finally, the updated objective function reads

Updated objective function

$$\sum_j \lambda_j \cdot F_T(E_j^+) + \sum_j \lambda_j \cdot F_T(E_j^-) \quad \rightarrow \quad \min. \tag{4.17}$$

[4] https://projects.coin-or.org/Clp

Accordingly, we set

$$c := \begin{pmatrix} \lambda_1 \\ \vdots \\ \lambda_n \\ \lambda_1 \\ \vdots \\ \lambda_n \end{pmatrix}, \qquad x := \begin{pmatrix} F_T(E_1^+) \\ \vdots \\ F_T(E_n^+) \\ F_T(E_1^-) \\ \vdots \\ F_T(E_n^-) \end{pmatrix}$$

for the Coin-CLP solver. According to equation 4.15, x is bounded by

$$c_{\text{lower}} := \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \quad \text{and} \quad c_{\text{upper}} := \begin{pmatrix} F_{T\text{-max}}(P_{\text{Ref}}) \\ \vdots \\ F_{T\text{-max}}(P_{\text{Ref}}) \end{pmatrix}.$$

We relax the precision of the solution to improve solvability and run-time.

We further define

$$\begin{aligned} \Gamma_{\text{lower}}(x) &:= x - \gamma \cdot |x| \\ \Gamma_{\text{upper}}(x) &:= x + \gamma \cdot |x| \end{aligned}$$

with $\gamma \geq 0$ to relax the precision of the solution. A lower precision can improve solvability and run-time. The lower and upper bound of the solution now equals the target forces $F_T(P_i)$ and $F_N(P_i)$ within the precision bounds

$$r_{\text{lower}} := \begin{pmatrix} \Gamma_{\text{lower}}(F_T(P_1)_x) \\ \Gamma_{\text{lower}}(F_T(P_1)_y) \\ \Gamma_{\text{lower}}(F_N(P_1)) \\ \vdots \\ \Gamma_{\text{lower}}(F_T(P_m)_x) \\ \Gamma_{\text{lower}}(F_T(P_m)_y) \\ \Gamma_{\text{lower}}(F_N(P_m)) \end{pmatrix}, \; r_{\text{upper}} := \begin{pmatrix} \Gamma_{\text{upper}}(F_T(P_1)_x) \\ \Gamma_{\text{upper}}(F_T(P_1)_y) \\ \Gamma_{\text{upper}}(F_N(P_1)) \\ \vdots \\ \Gamma_{\text{upper}}(F_T(P_m)_x) \\ \Gamma_{\text{upper}}(F_T(P_m)_y) \\ \Gamma_{\text{upper}}(F_N(P_m)) \end{pmatrix}.$$

Finally, matrix A contains the force factors:

$$A := \begin{pmatrix} f_x(E_1^+, P_1) & \cdots & f_x(E_n^+, P_1) & -f_x(E_1^-, P_1) & \cdots & -f_x(E_n^-, P_1) \\ f_y(E_1^+, P_1) & \cdots & f_y(E_n^+, P_1) & -f_y(E_1^-, P_1) & \cdots & -f_y(E_n^-, P_1) \\ f_z(E_1^+, P_1) & \cdots & f_z(E_n^+, P_1) & -f_z(E_1^-, P_1) & \cdots & -f_z(E_n^-, P_1) \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ f_x(E_1^+, P_m) & \cdots & f_x(E_n^+, P_m) & -f_x(E_1^-, P_m) & \cdots & -f_x(E_n^-, P_m) \\ f_y(E_1^+, P_m) & \cdots & f_y(E_n^+, P_m) & -f_y(E_1^-, P_m) & \cdots & -f_y(E_n^-, P_m) \\ f_z(E_1^+, P_m) & \cdots & f_z(E_n^+, P_m) & -f_z(E_1^-, P_m) & \cdots & -f_z(E_n^-, P_m) \end{pmatrix}.$$

Note that every electromagnet introduces two columns ($2 \cdot n$ in total) due to the separation of positively and negatively polarized electromagnets. Every permanent magnet is represented by three rows ($3 \cdot m$ in total) as equation 4.16 introduces three equations per permanent magnet.

After solution, signed electromagnetic forces can be computed.

Once the solution is computed, the signed force for each electromagnet can be computed as follows:

$$F_T(E_j) = F_T(E_j^+) - F_T(E_j^-).$$

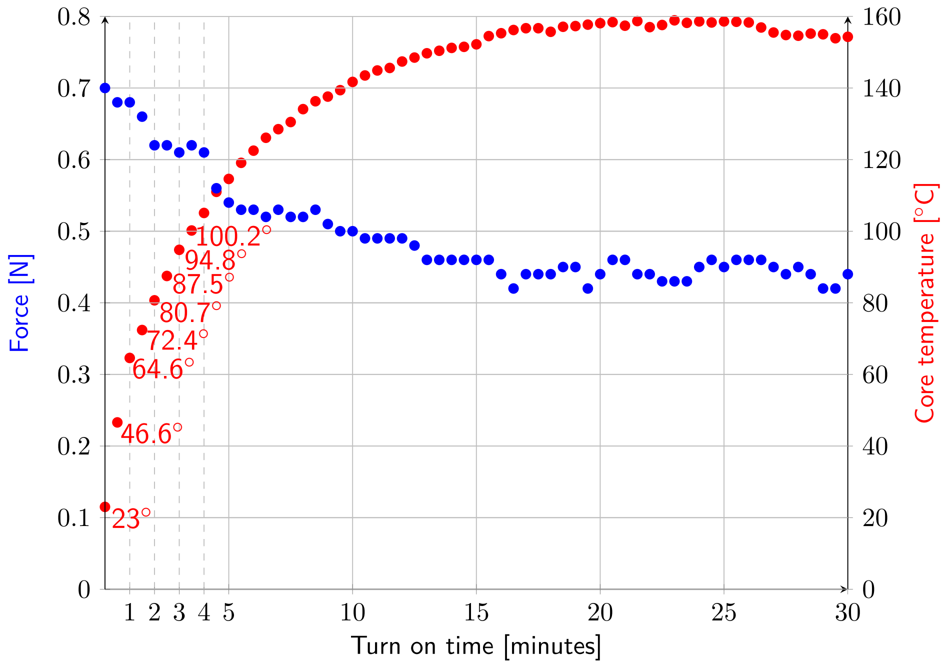


Figure 4.31: Core temperature and adhesive force of an electromagnet taken from our array depending on turn on time. Single freely suspended electromagnet with manganese-zinc ferrite core was driven at 40 V. Temperature was measured using optical IR thermometer pointed at the bottom of the core. Adhesive force equals the force that was required to detach a Neodymium permanent magnet (⌀ 10 mm, height 2 mm) from the top of the core. A 5 mm acrylic plate was placed between core and permanent magnet. Force was measured with a spring scale.

Either $F_T(E_j^+)$ or $F_T(E_j^-)$ is null. Otherwise, the solution would not be optimal in terms of the objective function.

Finally, force is mapped to polarization bit and PWM duty cycle.

Finally, the resulting force is mapped to the polarization bit and PWM duty cycle of the electromagnet. The polarization bit equals the sign of the tangential force, i.e., $\mathrm{sgn}(F_T(E_j))$. The PWM duty cycle is computed by linearly mapping $|F_T(E_j)|$ from $[0, F_{T\text{-max}}(P_{\mathrm{Ref}})]$ to $[0\%, 100\%]$.

Weights The weights λ_j in equation 4.17 influence the probability that an electromagnet is considered in the solution. In the simplest case, all weights are set to 1, which yields a solution that minimizes the sum of absolute forces exerted by electromagnets. However, more sophisticated weights can be helpful to protect the table's hardware or to smoothen the actuation.

Electromagnets heat up over time. This can reduce maximum force and damage hardware.

The longer an electromagnet is activated, the more heat it produces. This decreases the magnet's permeability and, thereby, reduces the maximum force it can apply (Fig. 4.31). We also noticed that, if magnets are steadily active for a long period, the fiber optical cables in the table begin to melt, damaging the tracking hardware in an irreversible way. Therefore, avoiding electromagnets to reach high temperatures is essential. Fortunately, the weights λ_j allow us to control the consideration of an electromagnet in the solution. By default, all weights could be set to 1. Once a magnet's sensor detects a temperature beyond 60° centigrade, its weight could be increased, making it less likely that this electromagnet is chosen for the solution. As our prototype table does not contain temperature sensors, we accumulate the activation time of an electromagnet instead. If a magnet's activation time exceeds an empirically determined threshold, λ_j is gradually increased. When the electromagnet is deactivated again, the activation time is decreased, and, therefore, the weight is lowered until it reaches 1 again.

Weights of hot electromagnets can be increased to reduce their probability of being selected for solution.

Usually, a default weight ($\lambda_j := 1$) prefers to actuate electromagnets that are close to permanents magnets. This can lead to jerky actuation. During a single actuation frame, the electromagnetic configuration is constant. However, since electromagnetic force strongly attenuates with increasing distance, a permanent magnet is not moving linearly but on a curve. This effect is the more considerable, the closer the activated electromagnets are to the permanent magnets, because the distance damping function $\epsilon(d(E_j, P_i))$ varies stronger during an actuation frame. By using electromagnets which are farer away, the actuation could be smoothed. To achieve this, the weight of an electromagnet could be set as follows:

Weighting close electromagnets stronger than distant ones smoothens actuation but also increases power consumption.

$$\lambda_j \quad := \quad \left(\frac{d_{\max}}{d_{\text{avg}}(E_j)}\right)^{\kappa}$$

where $d_{\text{avg}}(E_j)$ is the average distance of the electromagnet E_j to all permanent magnets on the table, and $\kappa \geq 1$ is a damping factor. Now, close electromagnets would retrieve a higher weight than those which are more distant. However, note that this technique also increases the power consumption and heat of each electromagnet.

4.5.4.7 Generalization

Multiple Madgets can be integrated into a single system of equations or actuated independently when sufficiently distant from each other.

Our algorithm can be easily extended to actuate multiple Madgets. First, the force distribution is computed for every single control (step 1 and 2). Then, the linear system of equations 4.16 is extended to consider the forces of *all* permanent magnets on the table (step 3). However, if too many Madgets are integrated into a single system, the performance of the Coin-CLP solver drops down. We can address this by subdividing the actuation into independent areas: If two Madgets are so far away that their electromagnets barely influence each other's permanent magnets, equation 4.16 can be solved for each Madget independently and in parallel. Currently, our algorithm actuates Madgets on a straight line from source to target positions. If multiple controls are actuated in parallel, they potentially collide on their paths. Theses events could by avoided by path finding algorithms, which have not been incorporated into our framework, yet.

Until now, we have only covered static magnets for moving and aligning Madgets. Dynamic magnets are used to change the physical *state* of control, such as the push state of a button, or the position of a slider.

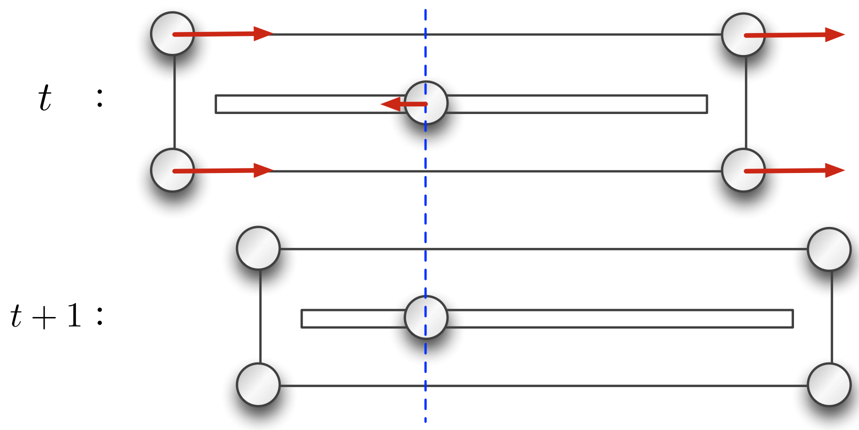


Figure 4.32: Relationship between static and dynamic markers during actuation. A Slider Madget is moved to the right while its handle is relatively moved to the left. Assuming zero friction, no force would be required to move the handle because its global position stays the same. However, the static friction force between sliding handle and base causes the handle to move as well. Therefore, a force in the opposite direction, canceling out friction, must be applied to the knob.

If a Madget contains dynamic magnets that have to be moved into tangential directions, like a knob's arm or a slider, the specific physical mechanism must be incorporated into the actuation model for every single Madget. Imagine an actuated slider with a handle that can be translated along one dimension (Fig. 4.32). If the Madget shall be translated to a new position while the dynamic magnet beneath the handle shall be moved to a new position simultaneously, the actuation algorithm must regard the physical dependency between the Madget's rigid base and the one-dimensional sliding handle.

Dynamic magnets must be integrated into the equations for every Madget individually.

In practice, it is easier to separate the actuation of the base magnets and the dynamic magnets into two subsequent steps. First, the Madget is moved and aligned to a new position and rotation. Then, all static magnets P_i are fixed ($F_N(P_i) = 0$ and $F_T(P_i) = 0$ for all i). Finally, the dynamic magnets P_d can be actuated without the dependency on the rigid base. In the example of the slider handle, the problem is reduced to finding an appropriate tangential force vector, i.e., $F_T(P_d) = v$ and $F_N(P_d) = 0$ where v points into the desired sliding direction.

Separating actuation of base and dynamic magnets is easier than model adaption.

Vertical actuation can be used to raise parts of a Madget, e.g., the plate of button (see section 4.8 for more examples). By setting $F_N(P_d) = f_r$ with $f_r < 0$ to a dynamic magnet P_d that is attached to a vertically supported element, this element can be raised from the table. Contrariwise, a positive force $f_r > 0$ can be employed to attract a spring loaded magnet to the table.

Vertical actuation can raise parts of a Madget.

4.5.5 Discussion

There is potential to improve actuation model with more accurate assumptions.

We have successfully applied the actuation algorithm described in this section to move and align multi-element controls on a tabletop. We chose this particular model, because it provides linear systems of equations that can be solved in real-time. However, some of the assumptions in the model are approximations that lower the accuracy of the solution. For example, we noticed that an electromagnet only creates a measurable magnetic field up from a minimum PWM. Also, electromagnets influence each other, which is not reflected in the model. In future work, this model should be enriched with the particular properties of the hardware design to generate a smooth, predictable actuation. Yet, additional non-linear parameters could impair the run-time of the algorithm.

4.6 Tracking

Precise tracking of Madgets is crucial for accurate actuation.

A precise tracking of Madgets on the surface is crucial to ensure an accurate close loop actuation. We employ DSI tracking to detect widget footprints, because visual tracking does not interfere with electromagnetic actuation. DSI is, furthermore, suitable to detect footprint markers as well as finger touches. Opposed to the SLAP Table, we cannot employ FTIR, because it would considerably blur the LCD panel due to the involved compliant layer.

4.6.1 Hardware Setup

DSI setup serves for tracking.

A ribbon of 108 LEDs radiates 850 nm IR light into the 6 mm Endlighten layer. An object touching the Endlighten layer reflects IR light downwards. An array of fiber optical cables transfers this light from the surface into the table. Each cable amounts to a diameter of 0.5 mm and a length of about 45 mm. It starts beneath the panel, penetrates the EL foil, passes the magnets, and pierces the conductor boards, which it is glued to. Finally, it ends about 2-3 mm beneath the conductor boards.

Fiber optics are placed around magnets and cores.

Fig. 4.33 shows the arrangement of cables: Four cables are placed between the magnetic cores and their coils, 12 around each magnet. The horizontal and vertical distance between cables varies between 6 mm in the core and 7.5 mm between inner and outer cables. In total, 58×37 fiber optical cables provide a low input resolution of 3.6 dpi. Similar to FlyEye by Wimmer [2010], we melted the ends of all cables using a laser cutter. This transforms them into small micro-lenses that improve the numerical aperture of the cable.

Three cameras capture surface.

Three Point Grey Firefly MV cameras, one for each module, with 3.5 mm lenses ($f/1.4$) inside the table capture the bottom caps of the fiber optical

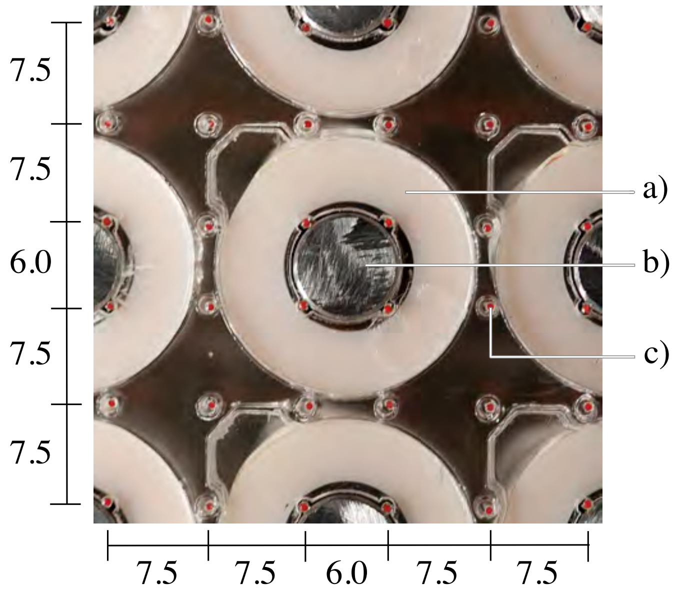


Figure 4.33: Alignment of fiber optical cables inside and between electromagnets as seen from above. Red light has been placed beneath the surface for the purpose of illustration. Numbers denote spaces in millimeter. a) Coil bobbin. b) Magnet core. c) Fiber optical cable. Modified image from [Weiss et al., 2010b].

cables. They produce gray scale frames at 30 fps in 640 × 480 pixels. Attached B+W 093 IR pass filters avoid interferences with visible light.

4.6.2 Tracking Algorithm

The tracking algorithm detects spots on the surface from the dot matrix of the fiber optical cables. This imposes three challenges to the tracking server:

Issues of tracking signal:

1. *The input signal is low resolution.* It is a strongly subsampled view of objects reflecting IR light into the table. This also affects marker design: A circular marker, always covering at least two dots, must have a diameter of about 15 mm.

   - Low resolution

2. *The input signal is noisy.* The LCD panel considerably attenuates the reflected IR light, which worsens the signal-to-noise ratio in the camera image.

   - Noise

3. *Every fiber optical cable is different.* All 2146 cables are hand-made, which implies that they slightly vary in light conductibility, curvature, and length. Furthermore, individual cables can be damaged yielding *dead dots.* Depending on the position, a dead dot causes the camera to be blind for a surface area of 1.82 to 2.25 cm^2. This must be compensated.

   - Varying quality of fiber optics

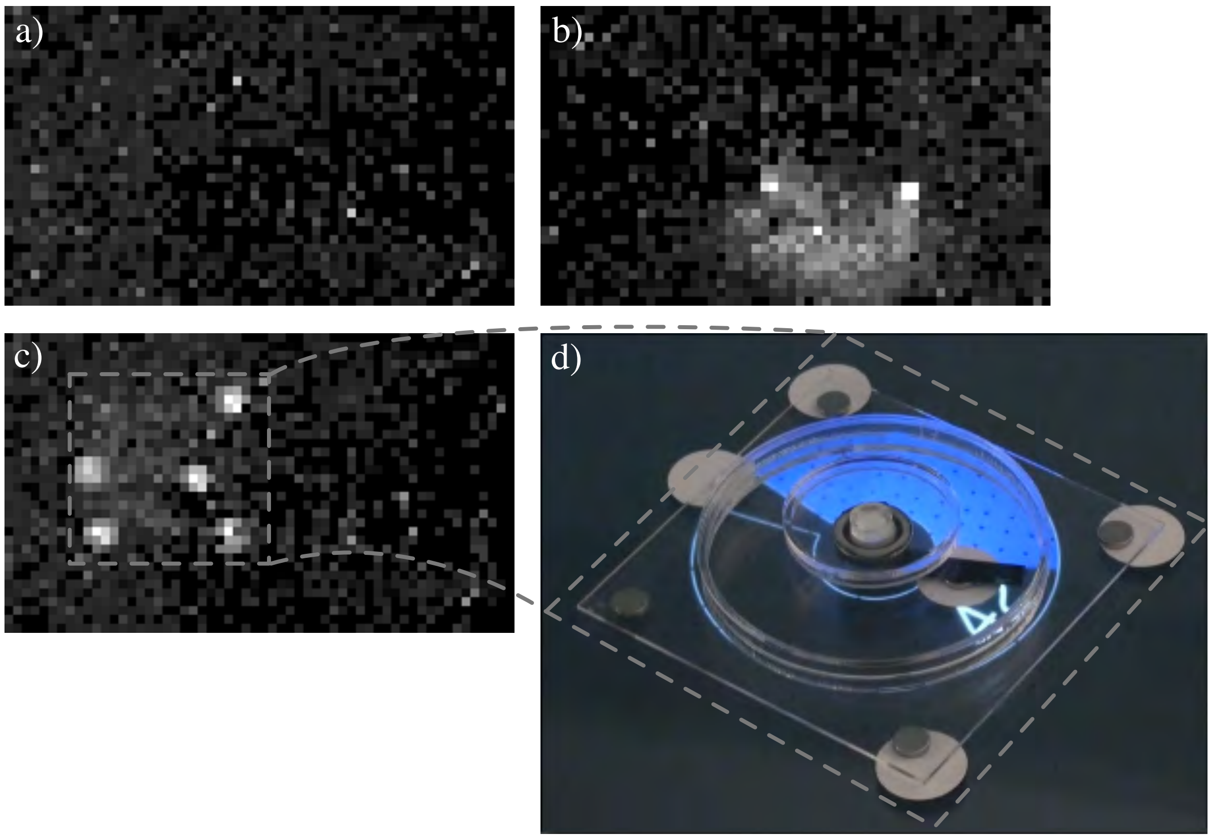


Figure 4.34: Input signal from the three cameras in the table. It is generated from the IR light that is emitted from the ends of the fiber optics. a) No objects are placed on the surface. The signal is noisy. b) Two fingers touching the surface. c) Footprint of a Knob Madget with gradient markers. d) Corresponding control with back projection as seen from above.

Sample input signals that illustrate these issues are shown in Fig. 4.34. Despite these challenges, the determination of marker positions must be accurate to maintain a correct control of the electromagnets during actuation. Note that the Figure does not show the raw camera input but the post-processed signal, as will be described in the following.

4.6.2.1 Gradient Fiducials

Resolution of fiber optics too low for unicolored markers.

The resolution of the dot grid is too low to just search for connected components in a binary image, as we described in section 3.5.1. A unicolored white footprint marker could not be mapped to an unambiguous position (Fig. 4.35a). We cannot increase the grid resolution, because the spacing between the fiber optics cannot be downsized. Instead, we increase the resolution of marker signal from a binary to a continuous scale. Inspired by the displayed-based measurement system to track tabletop robots by Kojima et al. [2006], we developed a novel marker design: circular *gradient fiducials*.

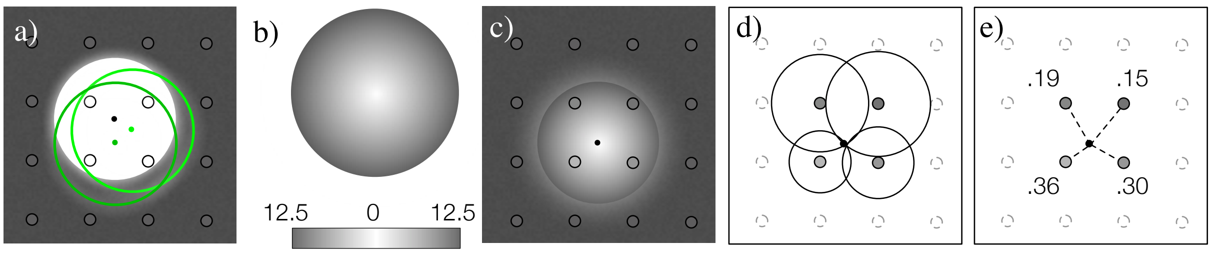


Figure 4.35: Principle of gradient fiducials. a) A unicolored marker placed on the fiber optical grid would yield an ambiguous localization. Alternative interpretations of the marker position are shown in green. b) A gradient fiducial is a circular marker with an imprinted gradient. Brightness maps to a radius between 0 mm and 12.5 mm. c) A gradient marker placed on the grid yields brightness values representing distances to the marker center. d) Each brightness value determines a circle on which the center of the marker can be. The actual position is on the intersection of all circles. e) We approximate the center position with an affine combination of the brightness values. Numbers denote normalized weights. Modified image from [Weiss et al., 2010b].

Solution: Imprinted brightness gradients on circular markers. Brightness encodes distance to center.

Our new markers are circles with a diameter of 25 mm. We imprinted a radial gradient, starting with white in the center, and linearly fading to dark-gray at the border (Fig. 4.35b). Thereby, the brightness of a pixel in the circle encodes the distance to the center. A gradient marker placed on the tabletop reflects the IR light according to the gradient. This yields a set of dots in the camera image, where the brightness value of each dot determines a circle on which the potential marker center lies (Fig. 4.35c-d). Accordingly, the actual center is located at the intersection of all circles. Thus, in theory, only three brightness dots are required to find the marker position.

In practice, the input signal is noisy (cf. Fig. 4.34a-c), and the circles do not meet in a single point. Instead, we have to find a position that minimizes the squared distance to all circles. Let p_i be the position of a dot and r_i its encoded distance to the marker's center. Then, the center c of the marker fulfills

Least-squares solution

$$\sum_i (\|c - p_i\| - r_i)^2 \quad \rightarrow \quad \min .$$

As this is a non-linear problem, we approximate the solution with the affine combination

Linear approximation

$$c \quad \approx \quad \frac{1}{\sum_i \omega_i} \cdot \sum_i \omega_i \cdot p_i \tag{4.18}$$

where the weight

$$\omega_i \quad := \quad r_{\text{marker}} - r_i \tag{4.19}$$

encodes the distance to the boundary of the marker, with r_{marker} denoting the marker's radius (Fig. 4.35e). Thus, dots in the center (brighter) are weighted stronger than dots at the periphery (darker).

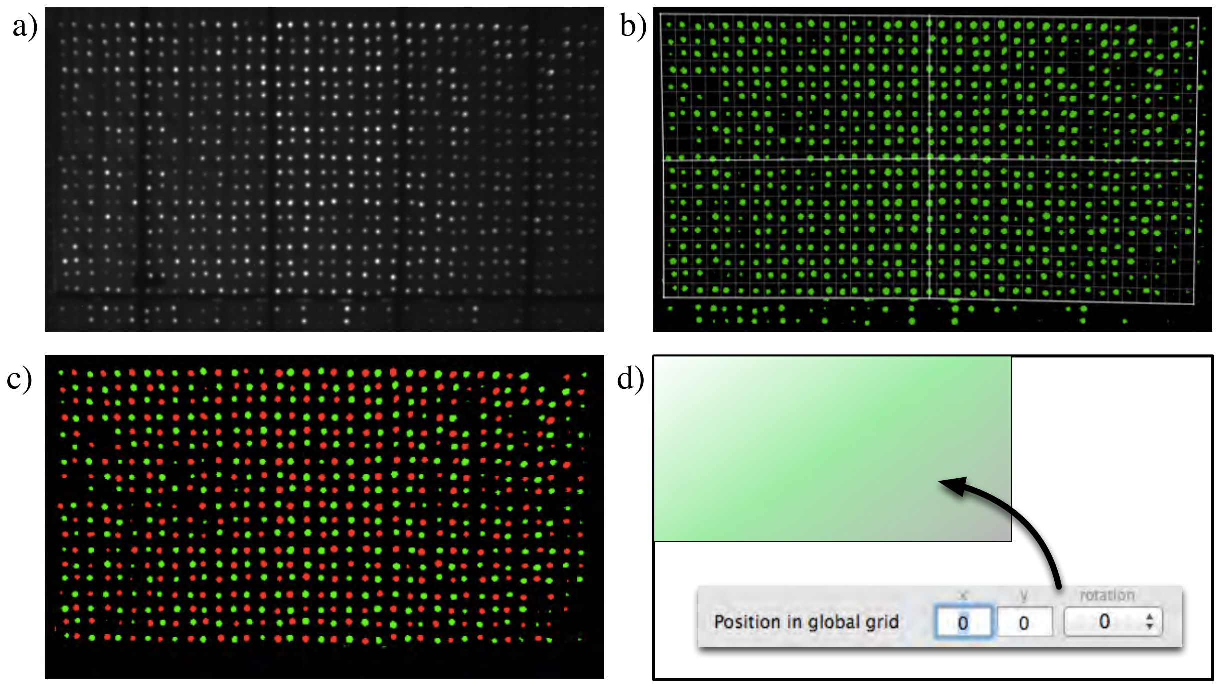


Figure 4.36: Main steps in the calibration process. a) User takes a foreground and a background image. The Figure shows the raw camera input when a white sheet is placed on the surface (foreground image). b) Active pixels (green) are determined by thresholding the difference between foreground and background pixels. The user then aligns a grid of cells onto the active pixels. c) Visualization of grid cells. The area of active pixels in every grid cell is alternatively colored with green and red color. If a dot is bicolored, or if subsequent dots are colored in the same way, the grid must be realigned. d) User specifies the position and rotation of the sub grid in the global grid.

We printed the gradients on bright paper using a laser printer, and employed a laser cutter to cut out circular markers. We chose the gradient so that the center reflects as much IR light as possible, while the periphery is as dark as possible but still bright enough to be distinguishable from the background.

4.6.3 Calibration

Before tracking objects on the surface, the three cameras in the table must be calibrated. Although this is an elaborate process, it has to be conducted only once. For every camera, the user has to conduct the following steps:

Configure camera so that all fiber optics are in focus and signal-to-noise ratio is maximized.

1. **Set camera parameters.** The aperture and zoom of the camera must work together so that the camera sensor receives as much light as possible with a depth of field that recovers all fiber optical caps. Remember that, due to variations in fabrication and assembly, the fiber optics have different lengths beneath the surfaces. Gain, brightness, shutter time, exposure time, and gamma must be specified in software to achieve the best signal-to-noise ratio.

2. **Active pixels.** Only specific pixels in the camera image belong to fiber optics, while the others are part of the background. These *active pixels* are found in this calibration step. First, the user removes all objects on the surface and lets the camera capture the *background image*. This image contains the minimum brightness for each pixel. Then, he places a white paper sheet on the table and captures a *foreground image* (Fig. 4.36a). As the paper sheet reflects most of the IR light downwards, this image contains the maximum possible brightness for each fiber optical cable. This equals the intensity that is reflected from the white center of a gradient fiducial. We retrieve the foreground and background image by averaging 50 camera frames to reduce noise.

   Determine which camera pixels belong to fiber optics.

   Both images define the possible brightness range for each pixel. We now consider a pixel as *active*, if the range exceeds a certain user-defined threshold $t_{\mathrm{ap}} > 0$ (Fig. 4.36b). Let **Bg** and **Fg** be the background and foreground image, respectively. We define the active pixel property of a pixel at position $\mathbf{x} = (x, y)$ as follows:

$$\mathrm{ap}(\mathbf{x}) \quad := \quad \begin{cases} 1, & \text{if } \mathbf{I}(\mathbf{Fg}, \mathbf{x}) - \mathbf{I}(\mathbf{Bg}, \mathbf{x}) \geq t_{\mathrm{ap}} \\ 0, & \text{otherwise.} \end{cases}$$

   Remember that $\mathbf{I}(\mathbf{Im}, \mathbf{x})$ denotes the intensity function that returns the pixel brightness at position $\mathbf{x}$ in an image **Im**.

3. **Grid alignment.** The input image is radially distorted, and only a part of it covers fiber optics. The low input resolution does not allow a touch-based calibration as the SLAP Table. Instead, the user manually drags a grid onto the image so that every cell contains exactly one cable (Fig. 4.36b). After specifying the grid resolution $gridX \times gridY$, i.e., the number of fiber optical cables in horizontal and vertical direction for this camera, he initially aligns the quadrangular grid by dragging its corners. To compensate for radial distortion, he can subdivide the grid by clicking a "+" button. In this case, the grid is transformed to a bicubic spline patch, whose interpolation points the user can drag. Every time the user subdivides the grid, he gains further degrees of freedom to align the patch. However, in practice, two subdivisions are sufficient to cover every fiber optical cable with a grid cell.

   Align grid so that every cell covers exactly one fiber optical cable.

   We index cells in a grid using a discrete number i with $i \in \{0, 1, ..., gridX \cdot gridY - 1\}$. Without loss of generality, we assume that the cells are ordered row by row. Once the grid is specified, we precompute a set $Cell_i$ for each cell. It contains all pixel positions covered by cell i. A color-coding of the fiber optics according to their grid index allows the user to validate the alignment of the grid (Fig. 4.36c).

   Now we can determine the brightness of each single fiber optical cable by computing a representative value of each cell in the grid. This value is computed by averaging the brightness of all active pixels. For a given camera frame **F**, we compute the representative of every

   For each cell, the brightness of the fiber optical cable is computed.

cell i as follows:

$$\text{Grid}^{\star}_{\text{cam}}(\mathbf{F}, i) \quad := \quad \frac{1}{\sum\limits_{\mathbf{x} \in Cell_i} \text{ap}(\mathbf{x})} \sum_{\mathbf{x} \in Cell_i} \text{ap}(\mathbf{x}) \cdot \mathbf{I}(\mathbf{F}, \mathbf{x}). \quad (4.20)$$

Dead dots are removed by averaging neighbors.

Few fiber optical cables might be damaged due to the manufacturing process. Only a small break in a cable suffices to eliminate its light conductivity, causing a *dead dot.* Replacing cables is difficult once the table is assembled (see discussion below). However, a dead dot causes a division by zero in equation 4.20, because no pixel in the cell is active. To avoid this case and to provide a reasonable approximation for the brightness in each dead dot, we define a new cell representative

$$\text{Grid}_{\text{cam}}(\mathbf{F}, i) := \begin{cases} \text{Grid}^{\star}_{\text{cam}}(\mathbf{F}, i) & \text{if } \{\mathbf{x} | \mathbf{x} \in Cell_i \ \wedge \text{ap}(\mathbf{x}) = 1\} \neq \emptyset, \\ \text{GridAvg}(\mathbf{F}, i) & \text{otherwise.} \end{cases}$$

where $\text{GridAvg}(\mathbf{F}, i)$ denotes the average of all non-dead dots in the 3×3 window around the cell. In the unlikely case that all dots in the neighborhood are invalid, we set $\text{GridAvg}(\mathbf{F}, i)$ to 0. However, a sub-matrix of 3×3 dead dots is unlikely and should be repaired. In practice, we first compute all cell representatives and detect dead dots. In a second pass, the dead dots are filled by averaging the neighbors.

Retrieve brightness range for cell representatives.

4. **Cell thresholds.** We precompute $\text{Grid}_{\text{cam}}(\mathbf{Bg}, i)$ and $\text{Grid}_{\text{cam}}(\mathbf{Fg}, i)$ from the background and the foreground image, respectively. Remember that the latter represents the brightness that is reflected from the center of a gradient fiducial marker (radius = 0).

   Furthermore, the user has to place a gray sheet of paper on the table that is colored with the darkest brightness of the gradient fiducial. The camera captures this *marker threshold image* $\mathbf{Gr}$. Then, we compute cell representatives $\text{Grid}_{\text{cam}}(\mathbf{Gr}, i)$ that encode the brightness level of IR light reflected from the periphery of a marker (radius = r_{marker}).

Specify global integration of grid.

5. **Global integration.** As each camera only covers the fiber optics of a single module, the grid must be integrated into the global grid that contains all 58×37 fiber optical cables. In this step, the user specifies the position and alignment of the sub-grid in global grid coordinates (Fig. 4.36d).

When all cameras are calibrated, the foreground, background, and marker threshold image of all cameras are combined into a global grid according to step 5 in the calibration. We denote these grids as $\text{Grid}_{\mathbf{Bg}}$, $\text{Grid}_{\mathbf{Fg}}$, and $\text{Grid}_{\mathbf{Gr}}$, respectively.

4.6.4 Image Processing Pipeline

Image processing pipeline:

With a calibrated system, we detect markers and finger touches in each frame as follows:

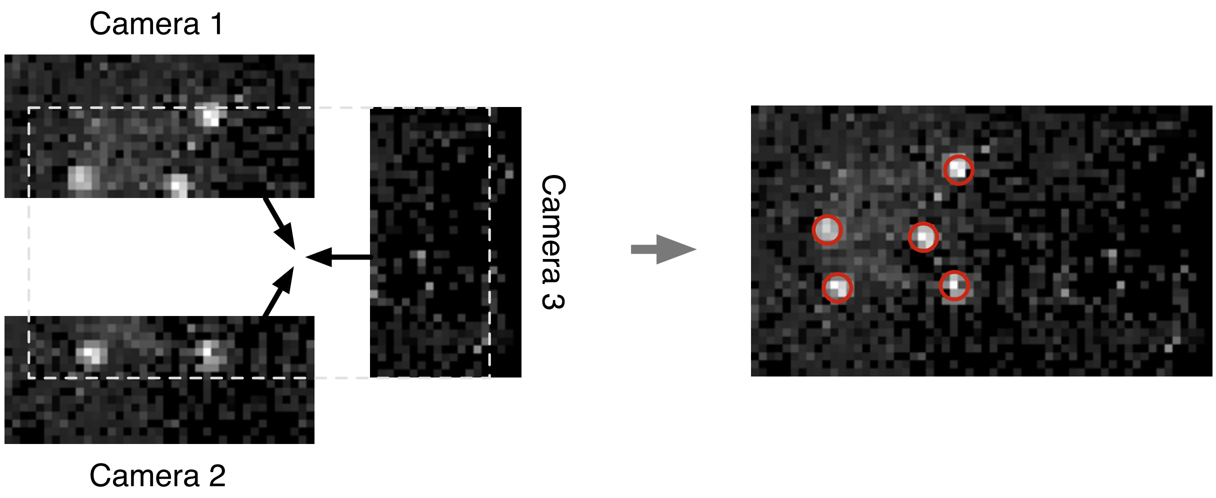


Figure 4.37: Image processing pipeline of fiber optical tracking. Left: Representatives are computed for every camera and composed into a global grid. Right: Marker detection.

1. **Global representatives.** For each frame **F** of the three cameras, we compute representatives $\mathrm{Grid}_{\mathrm{cam}}(\mathbf{F}, i)$ for all grid cells. Then, we combine all sub-grids into a single global grid $\mathrm{Grid}_{\mathbf{F}}$ according to calibration step 5 (Fig. 4.37, left).

   Retrieve global grid from three cameras.

2. **Detect markers.** We now search for all pixels that are potentially covered by a gradient fiducial (Fig. 4.37, right). We create a binary grid as follows:

   Detect markers and compute center by weighting normalized brightness values.

$$T(i) \quad := \quad \begin{cases} 1, & \text{if } \mathrm{Grid}_{\mathbf{F}}(i) > \mathrm{Grid}_{\mathbf{Gr}}(i) \\ 0, & \text{otherwise.} \end{cases}$$

   That is, all 1s in this grid correspond to dots in $\mathrm{Grid}_{\mathbf{F}}$ that are brighter than the IR reflection from the border of a Madget marker. We now search for connected components in T.

   Each component that contains at least 7 dots is considered as Madget marker. We compute its center using equation 4.18 with weights

$$\omega_i' \quad := \begin{cases} \dfrac{\mathrm{Grid}_{\mathbf{F}}(i) - \mathrm{Grid}_{\mathbf{Gr}}(i)}{\mathrm{Grid}_{\mathbf{Fg}}(i) - \mathrm{Grid}_{\mathbf{Gr}}(i)} & \text{if } \mathrm{Grid}_{\mathbf{Fg}}(i) - \mathrm{Grid}_{\mathbf{Gr}}(i) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

   These weights are equivalent to those in equation 4.19, because they are normalized in equation 4.18. In the center of a marker, the weights amount to 1 ($\mathrm{Grid}_{\mathbf{F}}(i) = \mathrm{Grid}_{\mathbf{Fg}}(i)$) and 0 at the boundary ($\mathrm{Grid}_{\mathbf{F}}(i) = \mathrm{Grid}_{\mathbf{Gr}}(i)$). From the boundary to the center, the weights increase linearly.

   Note that the arrangement of fiber optics is not a perfectly uniform grid (cf. Fig. 4.33). Thus, we have to compute the individual positions p_i in equation 4.18 depending on whether a fiber optical cable belongs to a core, or whether it is placed between magnets.

3. **Finger detection.** For finger touch detection, we first cancel out differences in light conductivity among individual fiber optical cables.

Detect finger touches from local maxima in remaining grid cells.

We compute a normalized version of the global grid:

$$\text{Grid}_{\text{norm}}(i) := \begin{cases} \dfrac{\text{Grid}_{\mathbf{F}}(i) - \text{Grid}_{\mathbf{Bg}}(i)}{\text{Grid}_{\mathbf{Fg}}(i) - \text{Grid}_{\mathbf{Bg}}(i)} & \text{if } \text{Grid}_{\mathbf{Fg}}(i) - \text{Grid}_{\mathbf{Bg}}(i) > 0, \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, $\text{Grid}_{\text{norm}}(i)$ maps grid dots to a scale from 0 to 1. We now search for local maxima that exceed a user-defined threshold $t_{\text{ft}} > 0$. That is, for grid cells i that fulfill

$$\begin{array}{rrcll} & \text{Grid}_{\text{norm}}(i) & \geq & t_{\text{ft}} & \\ \wedge & \text{Grid}_{\text{norm}}(i) & \geq & \text{Grid}_{\text{norm}}(j) & \forall j \in N_1(i) \end{array}$$

where $N_1(i)$ denotes the up to nine cells in the 1-neighborhood of cell i. All grid cells that fulfill this equation and are not covered by a Madget marker (step 2), are considered as finger touches. For sub-dot accuracy, we compute an affine combination of the grid positions in the 1-neighborhood using the normalized brightnesses as weights. The position of the finger touch for a local maximum at i finally reads

$$p_{\text{ft}}(i) = \frac{1}{\sum_{j \in N_1(i)} \text{Grid}_{\text{norm}}(j)} \cdot \sum_{j \in N_1(i)} \text{Grid}_{\text{norm}}(j) \cdot p_j.$$

4.6.5 Marker Placement

Markers must be placed in sufficient distance with respect to fiber optical grid configuration.

The marker design of Madgets is similar to that of SLAP Widgets. Markers must form a unique pattern that distinguishes the control from others, and communicates its ID and state. Due to the low resolution tracking, the distance between two markers must be large enough so that they are not merged by the tracking algorithm. That is, considering the fiber optical grid resolution and potential dead dots, designers should place adjacent markers with a space of at least 2 cm.

Permanent magnets can be directly mounted on circular markers to avoid further occlusion. Alternatively, their positions can be computed in relation to gradient markers, e.g., in the local coordinate system of the static markers (cf. section 3.5.2).

4.6.6 Discussion

Tracking is robust for marker detection. Simple touch interaction can be detected.

Despite the low resolution of the fiber optical grid, the tracking algorithm is able to reliably and robustly detect gradient markers for actuating Madgets on the surface. It can also detect finger touches and simple dragging gestures for transforming images. However, finger input accuracy beyond "photo sorting" would require a further refinement of the algorithm.

Due to the attenuation through the LCD panel, our first prototype of the table provided a very small signal-to-noise ratio. In the beginning, this

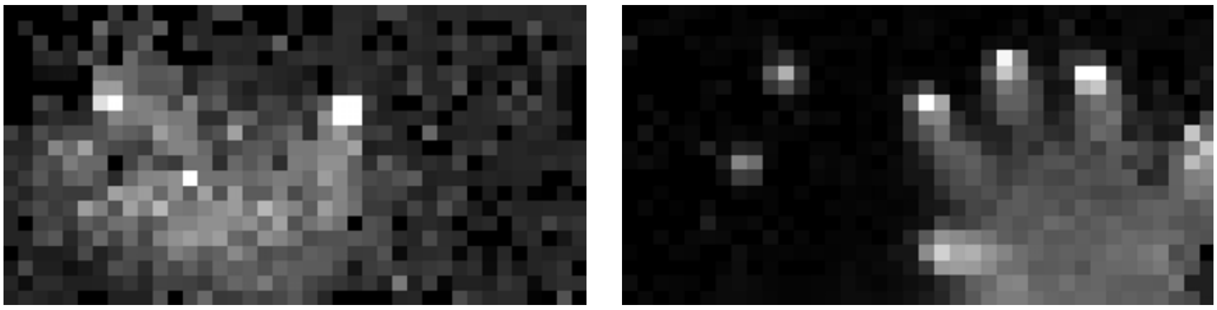


Figure 4.38: Effect of IR LED brightness on signal quality. Left: Normalized signal using LEDs in original prototype. Two fingers are touching the surface (extract from Fig. 4.34b). Right: Signal resulting from brighter LEDs. Five fingers (right) and two gradient markers (left) are placed on the surface.

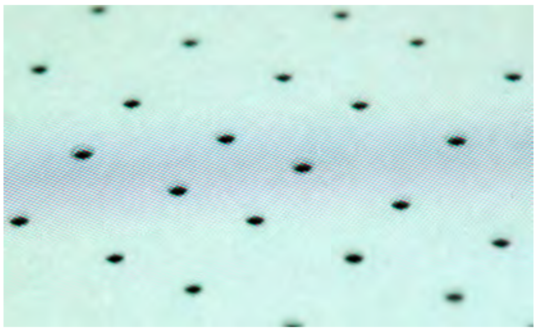


Figure 4.39: Close-up on active EL foil beneath LCD panel. No backlighting is provided where fiber optics penetrate the foil.

made it impossible to use the table in regular room lighting, because the (indirect) sunlight contains a lot of IR light. One way to address this issue is to use a thicker Endlighten surface. This would gather more IR light from the LEDs but considerably decrease the electromagnetic force that can be applied to Madgets. Instead, we improved the signal-to-noise ratio by embedding a new generation of brighter IR LEDs (Fig. 4.38).

Brighter LEDs improve signal-to-noise ratio.

We noticed that, up from a certain thickness of the Endlighten surface, marker tracking even works with unicolored circular markers. The reason is that fiber optics beneath the marker's center receive reflected IR light from all incident directions of the micro-lenses, while fiber optics in the periphery of the marker receive a smaller ratio. The result is a gradient-like IR reflection pattern. Using the same weighting as in the tracking algorithm described above, the center of the marker remains stable.

Unicolored markers also create gradient pattern.

4.7 Visual Output

We render the visuals of all Madgets and the GUI of the tabletop on a 24" TFT panel that we dismounted from a Samsung SyncMaster 2494LW monitor. We display the interface in a resolution of 1920 × 1080 pixels. In

LCD panel with EL foil backlighting displays interface.

contrast to the input channel (3.6 dpi), our panel produces high resolution output at 93 dpi (Fig. 4.34d on page 146). An EL foil, powered by an inverter (Epicenter E220U-972-A2), provides the backlighting of the panel. We used a mill to drill 1.3 mm holes into that foil to let all fiber optical cables pass this layer.

EL foil is easy to process but darker than other approaches.

The EL foil is rather thin (0.5 mm), and it is easy to process. Conventional backlighting, like fluorescent tubes or LEDs, needs more space and a diffusor layer. This would increase the space between magnets and surface and considerably limit the magnetic forces. However, the EL foil is darker than other approaches and its luminance decreases during its lifetime. Also, the holes cause small dark circles in the graphical output, because no backlighting is provided there (Fig. 4.39). This could be corrected with a calibrated white light source inside the table. The fiber optical cables would transmit this light to the surface and backlight the drilled dots. An IR block filter in front of the light source would avoid interferences with the tracking algorithm.

Holes could be filled with visible light from light source inside table.

4.8 Applications

The ability to actuate multi-element controls and to hold Madgets in place while moving parts of it, enables a variety of applications and allows to transfer traditional GUI techniques to tangible controls. These applications are described in this section.

4.8.1 General-Purpose Widgets

Madgets enable bidirectional communication between user, interface, and system for general-purpose controls.

Madgets provide a bidirectional communication between user, interface, and system. Users can interact with Madgets as with SLAP Widgets. When placed on the tabletop, our system detects a control, updates its graphical representation, and reacts on user input through the marker footprint. Beyond that, the *system* can change the *physical* state of each Madget. For example, a user can place a slider on the tabletop and pair it with an on-screen video player. Once the video is started, the slider's physical handle follows the position in the video, maintaining a consistent mapping between the virtual playback state and the physical slider. If desired, the user can navigate to a different time by dragging the handle to a new position. Inter-widget inconsistencies, as described in section 4.1, can also be avoided.

4.8.1.1 Persistence

Loading and saving configurations is a standard feature of conventional applications that is usually not feasible for tangible interfaces. Our actu-

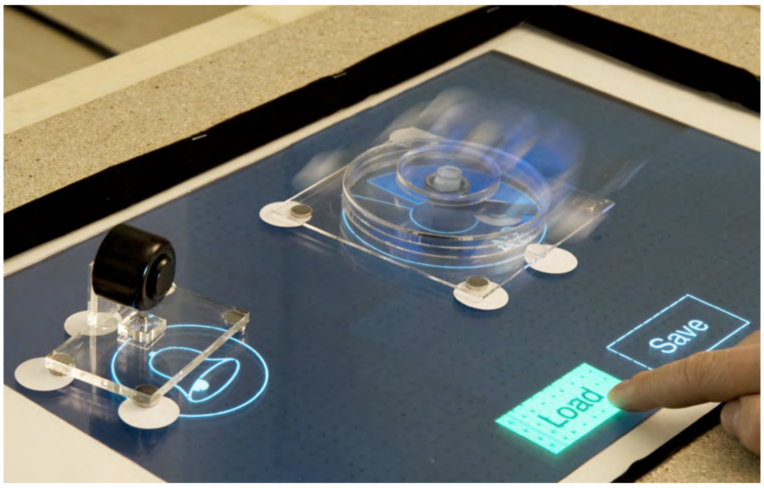


Figure 4.40: Actuation allows to save and restore the physical states of Madgets.

ation algorithm brings this GUI concept to tangible tabletop applications (Fig. 4.40). This is especially helpful if multiple persons work at the same table subsequently.

Actuation allows to load and save physical configurations.

Imagine a user, Alice, who employs several Madgets for a certain task. After a while, she decides to postpone the remaining work and presses an on-screen "Save" button before leaving the table. On the next day, she wants to continue the task but notices that someone else used the table in her absence and rearranged the controls. Thus, she presses "Load" and selects her previous configuration from a list. Accordingly, the application requests Alice to place some missing controls on the surface. Once she has done that, the application restores the visual interface and actuates all Madgets so that they match the physical configuration from the moment Alice saved the setup.

GUI concepts like undo and redo can be brought to the table.

Also well-known concepts like undo and redo can now be transferred to physical tabletop controls. For example, a user could draw a counterclockwise circle gesture next to a Madget to "reverse the time" and undo the last user's physical action. This would then move a Madget or a part of it to its previous position, respectively. Contrariwise, a clockwise circle gesture could trigger redo.

4.8.1.2 Remote Collaboration

Madgets are synchronized and enable telepresence when collaborating remotely.

As proposed for actuated pucks [Pangaro et al., 2002], remote collaboration can now be implemented without facing remote inconsistencies. Let us have another look at our initial example in section 4.1.

Alice and Bob share a knob to control the speed of a simulated engine and several sliders to set specific parameters. This time, both users employ

Madgets instead of SLAP Widgets. When Alice changes parameters, these are immediately send via network to Bob. The actuation algorithm now physically updates *Bob's set of sliders. Thus, he can see the changes when the sliders' handles move to new positions. Bob can adjust the engine speed accordingly, and avoid a failure of the simulation.*

The actuation of Madgets triggered from distant users enables *tangible presence* as described by Brave et al. [1998]. It makes users more aware of each other's actions and enables a more immersive collaboration.

4.8.1.3 Actuation by Gesture

Tangibles can be moved via gestures.

Reaching distant virtual objects on interactive tabletops and across multiple devices is a topic that has received much attention (e.g., [Nacenta et al., 2005]). Using actuation, some of these concepts can be transferred to reach distant *physical* controls. Imagine a knob is not within a user's grasp. By drawing a circle, using a tabletop gesture like the I-Grabber [Abednego et al., 2009], or by executing a 3D pointing gesture using a depth camera like the Kinect, the system could move the control to the user. Contrariwise, a wipe gesture could move it out of the way to provide interaction space. Techniques like this could make physical controls as flexible as their virtual counterparts.

4.8.2 Going 3D: Height

Madgets enable height as novel actuation dimension.

In related work, electromagnetic actuation was limited to tangential movement of pucks. A key feature of our technique is the actuation into the vertical direction. By holding a Madget in place while applying a normal force on a subpart of it, objects can be raised from the table.

4.8.2.1 Buttons

Buttons can be realized by vertically actuating plates with embedded magnets.

A simple but powerful example is our Radio Button Madget (Fig. 4.41). Like all Madgets, it consists of a rigid base with attached permanent magnets for positioning and alignment on the surface. The essential parts of the control are three magnetic buttons. Each button contains an acrylic plate with a permanent magnet glued to its center. It is embedded in a vertical box that avoids tangential motion. By default, gravity drags the plate to the bottom. When applying repelling electromagnetic fields to its permanent magnet, the plate is raised. Each box contains a hole at the top that lets users feel the plate. It is slightly narrowed to retain the plate in the box. Users can easily push down an actuated plate against the magnetic force; a repelled plate provides a spring-like resistance. Similar to a SLAP Keypad (section 3.3.3.2), a push down event is communicated via a spot to the table's cameras.

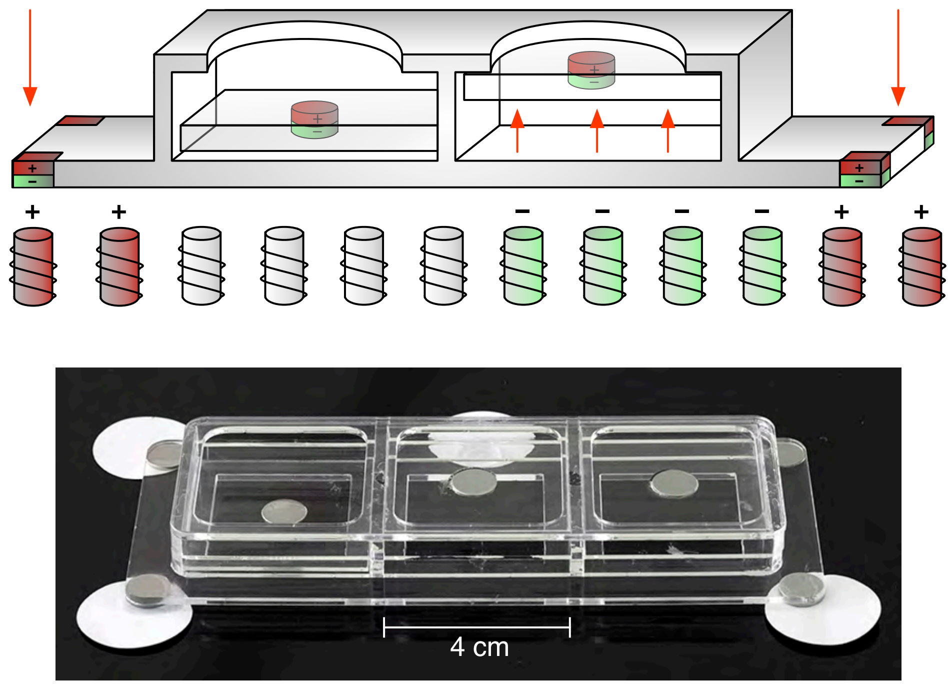


Figure 4.41: Radio Button Madget. Top: Actuation concept. The Madget is held in place with attracting electromagnetic fields. The active button is not actuated and pushed down by gravity. Inactive buttons are raised with repelling fields. Bottom: Working prototype with three buttons. Modified image from [Weiss et al., 2010b].

This Madget provides a simple way to implement a three-state radio button: The actuation is inactive for the selected button (down), while all other button plates are repelled (up). When the user presses a different button, the tracking algorithm detects this change, updates the internal state and the back projection. Then, the system disables the actuation for this button while raising all other "unchecked" buttons. It can also be used as a keypad with simple push or toggle buttons. Note that the Madget provides a richer haptic feedback than the SLAP Keypads. Users cannot just feel the boundaries of each key but also their push state (up or down). Furthermore, the magnetic resistance in combination with an acrylic plate provides a natural "click" sound and a better pressure point than a silicone layer.

4.8.2.2 Clutch

Subsequent actuations allow more complex mechanisms, like physically disabling a button.

Vertical and tangential actuations can be executed subsequently to implement dynamic physical constraints. In analogy to GUI buttons that are unavailable or "grayed out", our Blockabel Button Madget represents a pushbutton that can be locked (Fig. 4.42). Primarily, it consists of a pushbutton that is raised with repelling electromagnetic fields. By default, user can easily push it to trigger actions. Beyond that, it provides a *clutch mechanism*. The button can be locked by first raising the plate with nor-

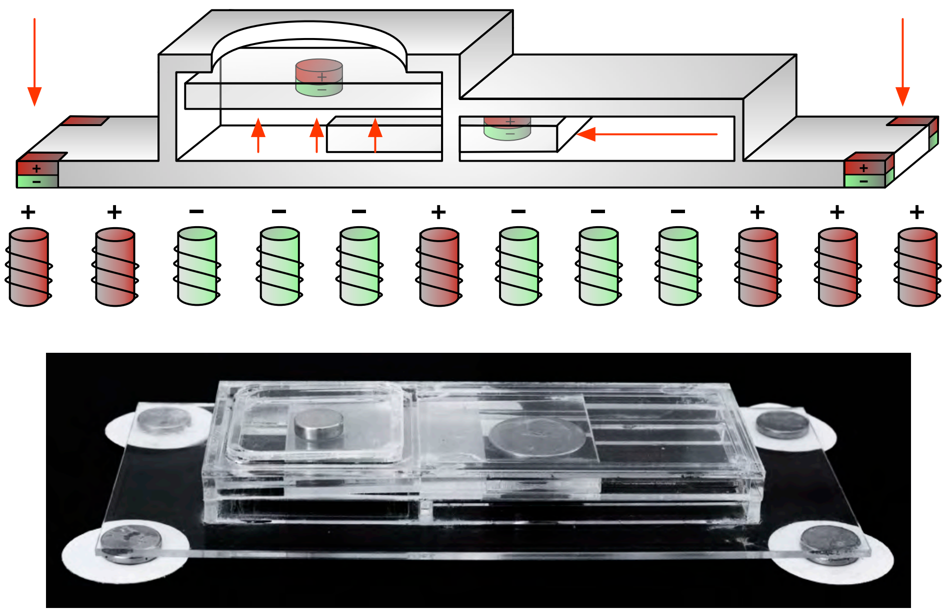


Figure 4.42: Blockable Button Madget with clutch mechanism. Top: Actuation concept. In default mode, the button's plate is raised using magnetic forces and can be pushed down. By raising the button's plate, and then translating the blocking bar beneath the plate, the button is locked. It cannot be pushed anymore. Actuation in reverse order unlocks the button again. Bottom: Prototype. Modified image from [Weiss et al., 2010b].

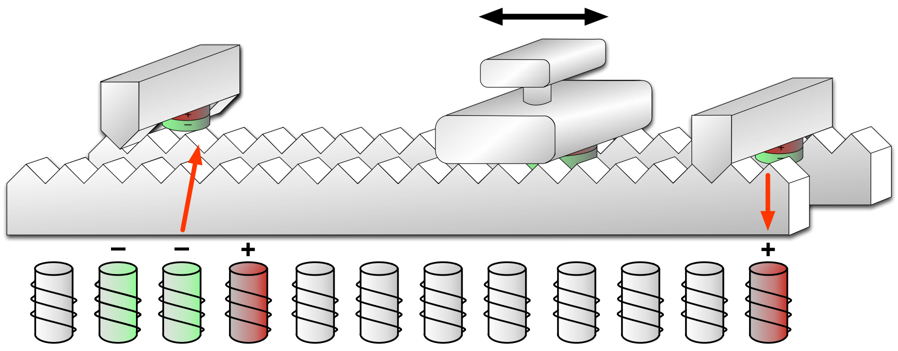


Figure 4.43: Concept of Slider Madget with dynamic range limits. Range limits stick in notches of the sliding bar. They can be raised and placed to new positions using magnetic actuation.

mal forces and then horizontally moving an acrylic bar beneath the plate. Now the button cannot be pushed down; it is locked. The inverse actuation sequence releases the button again.

This concept can be applied to many other controls. For example, a slider with dynamic physical range limits is imaginable. Imagine a slider that implements these limits with acrylic blocks fixed in a notch bar. Each block contains a permanent magnet. The position of a limiting block can be

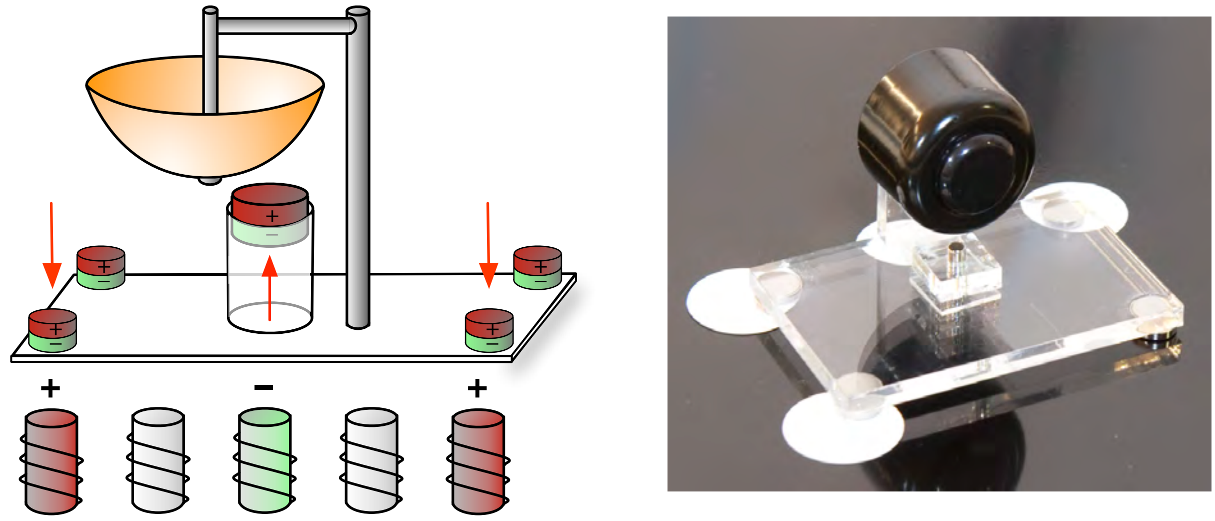


Figure 4.44: Bell Madget with vertically actuated beater. Left: Actuation concept. While the Madget is held in place, the beater is pushed against the bell with a strong repelling electromagnetic impulse. Right: Working prototype. Modified image from [Weiss et al., 2010b].

changed by first raising it via a normal force, then moving it to a new position, and, finally, releasing it again by deactivating actuation (Fig. 4.43).

4.8.2.3 Mechanical Audio Feedback

Vertical actuation can create localized audio feedback.

Vertical actuation can also be employed to create mechanical audio output. Fig. 4.44 shows our Bell Madget. It consists of a rigid base with a small acrylic mount that centers an off-the-shelf bicycle bell above the base. A cylindric magnet (⌀ 4 mm, height 12.5 mm), the beater, is embedded into the base so that it leaves a gap of 5.8 mm to the bell's sounding box when actuation is disabled. A strong repelling electromagnetic impulse lets the beater bump the bell and produce a ping sound. The volume can be varied via the duty cycle of each impulse.

The Bell Madget can be used to create a *localized* audio feedback on the surface. Placed next to a virtual object, it can notify the user about events. For example, a single ping could inform about a new message in a chat window, while a high-frequent repetitive ringing could communicate critical errors, such as connection loss during a remote collaboration. It is also imaginable to use Bell Madgets with different tones for different applications. Note that this local feedback can barely be accomplished with speakers arranged around or inside the table.

4.8.3 Force Feedback

Actuation can enrich the pure physicality of multi-element controls with *active* force feedback.

4.8.3.1 Vibration

Programmers can simply let a Madget vibrate by repeatedly assigning small tangential or normal forces. They just have to make sure that the control remains close to its current position.

Small repeated force impulses let a Madget vibrate for user feedback.

This can be useful to call the user's attention to a Madget. For example, a shared control could vibrate if a remote user starts operating it. Or a continuous control like a knob or slider could vibrate while a user sets its associated value within a critical range. Vibration *patterns* could also communicate more complex information.

Vibration feedback can also indicate discrete steps in continuous controls, for example, semantic marks in a video timeline. A designer could implement a simple beat feedback, similar to the Bell Madget, by attaching a closed vertical acrylic pipe to a control. Every time a mark in the continuous scale is reached, a permanent magnet in the pipe is repelled, hits the ceiling, and lets the control vibrate shortly. The control via software allows to dynamically adapt the "haptic marks".

4.8.3.2 Resistance

Resistance of controls can be varied using different electromagnetic forces.

The resistance of a control can indicate its importance and act as a further output channel. A button causing an irreversible action, such as "Close without save", could be harder to press. A volume slider shifted beyond a certain threshold could be harder to move. Those effects can be implemented by changing the strengths of electromagnetic fields. Varying the duty cycle of the electromagnets raising a button's plate changes the perceived resistance of a button. Attracting a knob's arm to the surface makes it harder to move. We will present concrete techniques to change the resistance of controls in the next chapter.

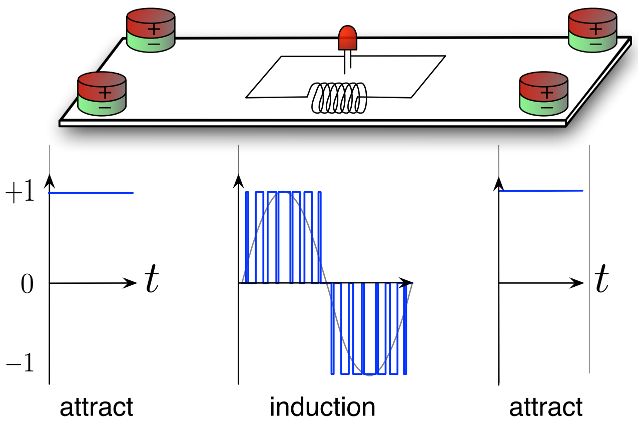
4.8.4 Water Wheel Madgets

Electromagnetism allows to transfer electrical and mechanical power.

Madgets are passive objects that do not employ any batteries or tethered power supplies. Yet, electromagnetic fields can be used to transfer energy to them. In analogy to the ancient mechanical drives, we call these controls Water Wheel Madgets.

4.8.4.1 Inductive Energy Transfer

In the same way as electrical tooth brushes are charged, we can create electric current inside a Madget via electromagnetic induction. An example is shown in Fig. 4.45. The so-called Induction Madget consists of a rigid base with an embedded coil. This coil is connected to a low power LED. If



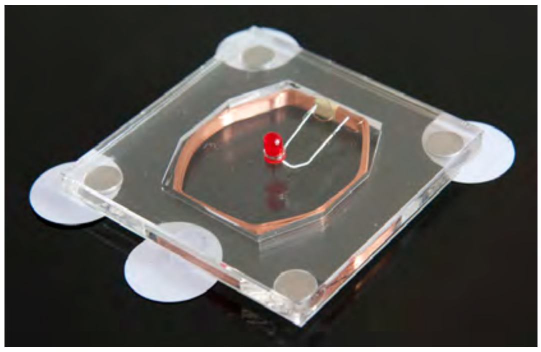


Figure 4.45: Madget with LED powered by induction. Left: Actuation concept with corresponding PWM signal. Permanent magnets at the corners hold the Madget in place. A sinusoid PWM signal applied to the electromagnets beneath the middle induces current to the circuit. Right: Working prototype. Modified image from [Weiss et al., 2010b].

we apply a sinusoidal PWM signal at the electromagnets beneath the coil, current is induced to the coil, which lets the LED blink. This technique allows us to add low power electronic parts to a Madget, such as LEDs or simple sensors, *without* imposing a large form factor or weight to the design. A more complex example is a Madget containing a coil, a proximity sensor, and an IR LED that is directed to the table surface. As soon as a user's hand approaches the sensor, the LED could create an IR spot that would be detected by the tracking algorithm. Therefore, proximity could be used as further input modality. Also, memory chips or radio transmitters could be embedded.

A sinusoidal PWM signal can induce current in a coil that is embedded in a Madget.

4.8.4.2 Motors

Electromagnetic force also allows for mechanical power transfer. The Gear Wheel Madget in Fig. 4.46 consists of a pivot-mounted gear wheel with permanent magnets attached to opposite sites of the wheel. Our actuation algorithm can spin this wheel by dynamically applying tangential forces. Assuming an accurate tracking, we can even vary the speed of the rotation. This Madget could be a base building block for more complex controls, such as small robots.

Actuation can turn a wheel in a Madget, which then acts as a mechanical building block.

4.8.5 Prototyping Physical Properties

Electromagnetic actuation allows designers to simulate physical properties of controls and to change them on the fly. This can be especially helpful for prototyping. We will describe and evaluate this concept in the next chapter.

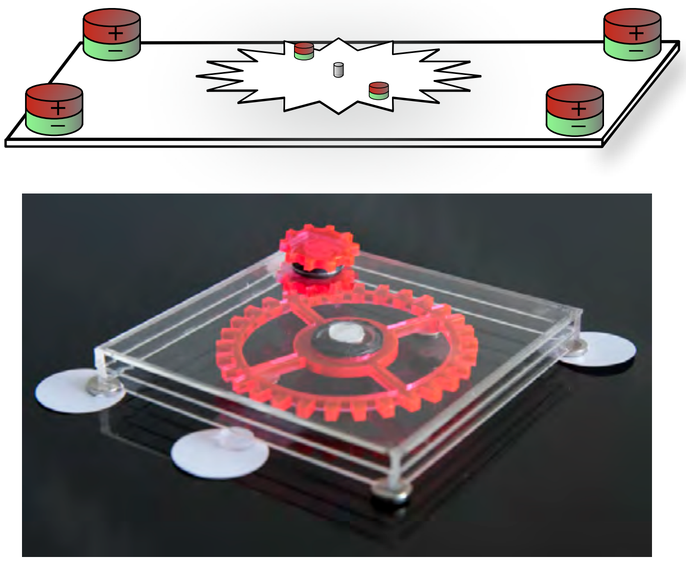


Figure 4.46: Gear Wheel Madget. Top: Prototype concept. The pivot-mounted gear is rotated by applying tangential forces to its attached permanent magnets. Forces are directed orthogonally to the pivot point. Bottom: Building block prototype. The bottom wheel is rotated by electromagnetism. The top wheel could be used to drive attached mechanics. Modified image from [Weiss et al., 2010b].

4.9 Implementation Challenges

When implementing the Madgets Table, we faced engineering challenges that had to be solved to achieve a reliable actuation. The most important ones are described in this section.

Actuation algorithm bases on a set of approximations to achieve high frame rate.

A Madget should be actuated as smooth as possible. However, as mentioned in section 4.5.5, our actuation algorithm includes a set of approximations that reduce the non-linear problem of finding a correct electromagnetic configuration to linear systems of equations. One assumption is that during an actuation frame, the strength and polarization of every electromagnet remains constant. This is only true if the duration of an actuation frame t_{af} converges to 0 ms; that is, the shorter an actuation frame, the smoother the actuation. While the tracking and actuation algorithms require a minimum timespan to compute the next frame, the communication between the software and the hardware controller turned out to be the major bottleneck. In our original prototype, we used an Arduino Mega with an ATMega1280 microcontroller to trigger the electromagnets. It was connected to the computer via USB. However, the bandwidth between the Mac computer and the controller limited our frame rate to 30 fps. Furthermore, driver incompatibilities required us to add a further computer with Microsoft Windows as a mediator between the Mac and the Arduino. The later introduced mbed controller solved these issues and removed the additional computer from the system design. The mbed controller contains an on-board Ethernet chip that can be accessed by simply soldering a RJ45 plug to distinct

mbed controller improved actuation frame rate compared to original Arduino controller.

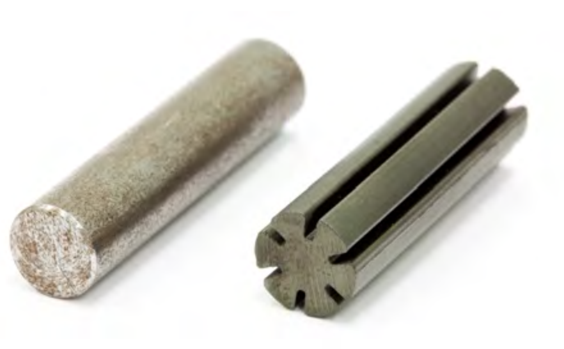


Figure 4.47: Different core materials. Left: Iron core. Right: Manganese-zinc ferrite core with recesses to avoid eddy currents.

pins on the controller. Via Ethernet, we can update the electromagnetic array with 60 fps, yielding a frame length of $t_{\mathrm{af}} \approx 0.0166$ s. Furthermore, based on a web compiler, the mbed controller can be programmed in a platform independent way.

Manganese-zinc ferrite cores are less prone to eddy currents than iron cores.

The choice of the electromagnets' core material is crucial for the performance of the actuation. The original Madgets Table uses cylindric iron rods, because they yield a high permeability. However, iron cores are highly prone to eddy currents, which causes them to heat up and to considerably lose power over time. They also tend to saturate such that the core's magnetization becomes irreversible. In a later iteration of the table, we use manganese-zinc ferrite cores (Fig. 4.47). Usually applied for welding, these cores provide an electromagnetic field strength similar to iron rods but contain recesses that minimize eddy currents. We refer to Scherz [2007] (p. 134–138) for a detailed explanation of various core materials.

Active cooling should be embedded into actuation hardware.

The Madgets Table does not contain a heat sink or an active cooling mechanism like a fan. Although single electromagnets can be switched off if they become too hot, the surface had to be passively cooled down after we actuated controls for about 30 minutes. This phase can be accelerated by placing cooling pads onto the surface. However, in a future prototype, a cooling mechanism should be integrated into the hardware. Note though that our compact integration of actuation and tracking makes it hard to add further components into the surface. A smaller actively cooled prototype of the table, which relies on tracking from above, will be presented in section 6.5.

Assembly process of table is very extensive. Future design should support more efficient maintenance.

The entire table is assembled by hand. All fiber optics have to be placed carefully into the grid. To speed up this process, we developed and iterated a set of tools, e.g., to cut fiber optics in pieces of constant length, to arrange fiber optics around a core, or to hold a module in place. We also defined processes to test the system and to detect errors. However, the maintenance of the hardware can be tricky and time demanding. While controllers can be replaced easily, modifying the surface after assembly is difficult. It usually requires to detach all layers and remove the single modules from the table's frame. For example, replacing a fiber optical cable necessitates to remove the old one with a cordless screwdriver and then glue a new one to the bottom board of the module. If a cable is placed inside an electromagnet, all four inner fiber optics have to be replaced as they are taped to the core. Replacing an electromagnet is even more difficult, because it is soldered to the bottom board. A future iteration of the system should be constructed in such a way that it supports a more efficient maintenance.

4.10 Actuation versus User Control

Some users perceive autonomously actuated objects as living beings.

When actuating controls two psychologic aspects have to be considered. First, there is a tendency of users to perceive actuated controls as living beings. When watching a demonstration video of an actuated button that was translated across the table in a jerky way, several users independently made statements like: “Oh, look it walks like a beetle.” The so-called Media Equation [Reeves and Nass, 1996], which claims that users treat computers as real people, certainly also applies to actuated controls. They appear as autonomous objects, which gives users a living impression. A similar observation has been stated by Rosenfeld et al. [2004] in the context of the Planar Manipulator Display:

> “We also noted a strong tendency for users to anthropomorphize the behavior of moving physical objects, even when those objects were seemingly personality-free furniture models. For example, a bug in the control systems code sometimes caused objects to perform small random motions near their target location before coming to stop. We were surprised how many observers gave descriptions of this behavior such as “he was nervous until he found his spot”. While this tendency may not be particularly relevant for this application, it suggests that active physical presentation might be especially suited to simulations involving human behaviors and social relationships.”
>
> [Rosenfeld et al., 2004, p. 6]

Designers must balance importance of actuation with users’ desire for control.

The second aspect that must be considered by application designers is the user’s potential loss of control. While actuation is certainly reasonable to maintain the physical-visual consistency, it also implies that the user is *not in control* when a Madget moves. This can yield a feeling of uncertainty when operating physical controls.

4.10.1 Preliminary Study

Preliminary study: Collaborative ordering task with a pretended remote user

In order to retrieve a better understanding of the potential conflict between actuation and user control, we conducted a pilot user study. Participants performed a simple remote collaboration task. Four numbered actuated pucks were placed on the surface that had to be brought into a certain order (Fig. 4.48). As a second table was not available, the remote user was simulated using a Wizard-of-Oz approach: The test examiner intervened

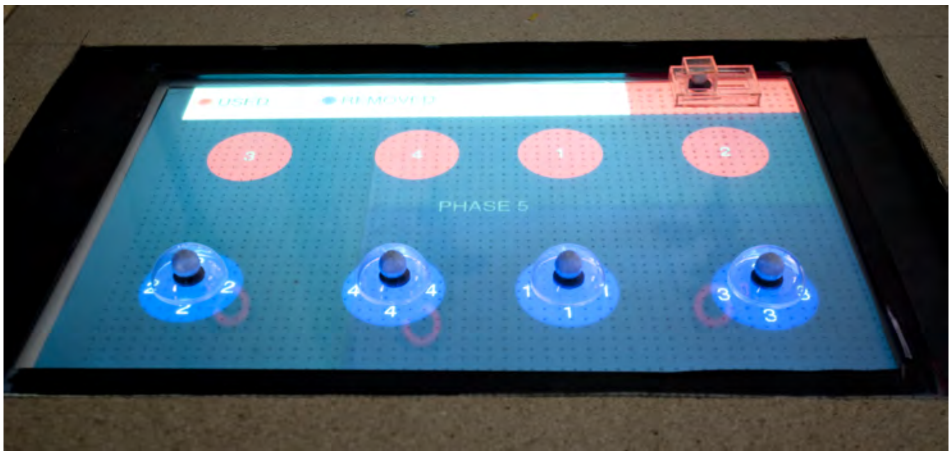


Figure 4.48: Setup of preliminary study to explore conflicts between actuation and user's desire for control. The user's task was to order numbered pucks in collaboration with a pretended remote user. Image courtesy of [Seidel, 2011].

with the interaction by dragging pucks with a mouse on a separate conventional desktop GUI. While users could directly move the physical pucks on the surface, the examiner employed electromagnetic actuation to do the same. He pretended to be a remote user, who supported the ordering task but made mistakes occasionally. The intention of this study was to find out which feedback methods were suitable if actuation was triggered by external events.

The main findings of this preliminary study are:

- *Starting actuations distract users.* Every time an actuation of a puck started that users did not anticipate, they tended to interrupt their task, and watch what happened. In some cases, this distraction even caused users to forget their current operation.

- *Visual feedback improves user experience and collaboration.* We conducted the study with and without a visualization of the actuation path. Users appreciated to be able to see where an actuation puck was moving. In these situations, some users even supported the actuation: For instance, if they saw that an actuation trajectory would likely pass through another physical object, users tended to move this one out of the way. Predictability of actuation turned out to be an important ingredient to a feeling of being in control. The sole announcement of an upcoming actuation via a flashing halo around a puck or the display of the already passed trajectory was not considered important by most users.

- *In the absence of direct contact to the remote person, users tended to communicate via physical pucks.* Participants requested a direct contact to a remote user, e.g., via audio or video, to coordinate collaborative actions. In the absence of this connection, they used the tangibles for communication. If a puck was actuated, e.g., into the wrong direction, some participants moved it repeatedly into the opposite direction, in order to "convince" the remote person.

- *Having control is essential.* In one condition, we provided participants a switch to disable the influence of the remote user. This switch basically turned off electromagnetic actuation. Many participants appreciated this control, especially if the simulated remote user made mistakes. Yet, they also agreed that such a "kill switch" can strongly impair remote collaboration and lead to deadlocks in the interaction if both users mutually exclude each other's influence.

Future work: Study with two tables and two collaborating users.

In future work, the study should be iterated and conducted using two actuation tables with two real users. The Wizard-of-Oz prototype only allows a limited generalization. However, this pilot study already shows the importance of predictability and the need for designers to regard the user's desire for control when using actuation.

4.11 Closing Remarks

In this chapter, we introduced Madgets, tangible general-purpose controls that extend the concept of SLAP Widgets with actuation, ensuring a bilateral interaction between user, interface, and tabletop. We discussed related work and explained the table hardware and control flow of the Madgets Table. We showed how to track Madgets using low-resolution fiber optical tracking, and how to actuate complex controls using electromagnetic actuation. We also explored the design space of potential applications that make use of horizontal and vertical actuation mechanisms. Finally, we illuminated crucial design challenges that we had encountered.

Madgets embody the same benefits as SLAP Widgets and maintain physical-visual consistency.

The Madgets Table maintains the consistency between the physical and the virtual state of tangible controls. Yet, Madgets embody the same benefits as SLAP Widgets. They are lightweight and use the table's back projection for dynamically changing the controls' appearances. In addition, they are passive; they neither require heavy batteries nor cables that could clutter the interface. They are easy to construct and do not necessitate an electrical engineering background. Finally, Madgets are low cost and robust. Even if a Madget breaks, it is easy and cheap to replace. Using the actuation algorithm described in section 4.5, many concepts known from GUIs can now be integrated into tangible tabletop applications. The actuation itself is calm and unobtrusive.

Lightness of Madgets comes to the cost of complex table design.

The lightness of passive actuated controls comes to the cost of a rather complex table design. Our table prototype was difficult to construct, hard to maintain, and, compared to the SLAP Table, rather costly. Furthermore, the power consumption of the electromagnetic array is quite high. It is likely, though, that after further iterations of the prototype, production costs and power consumption could be significantly reduced. The main factor that complicates the table construction is the fiber optical tracking. Besides the maintenance and assembly effort, the algorithm needs larger footprint markers (⌀ 25 mm) than SLAP Widgets, and the use of cameras increases the depth of the table construction, making it less flexible. A resistive tracking technology like the already mentioned Interpolating Force-Sensing Resistance (IFSR) sensor [Rosenberg and Perlin, 2009] could improve tracking and, thereby, increase actuation accuracy. Also, it would considerably reduce the size of the base frame. An alternative is the use of sensor coils, as described by Hook et al. [2009], or Hall effect sensors. This could detect the position of permanent magnets and, thereby, supersede the relatively large paper-based markers. However, special consideration must be taken to avoid interferences between the tracking and the electromagnetic actuation technology.

In future, tracking could be replaced with thin form factor techniques.

The forces that the electromagnets can exert are limited. In practice, users can normally overcome the force fields of electromagnets, e.g., if a Madget is attracted to the surface with maximum force. The restricted force also imposes constraints to the designer: Madgets must not be too heavy, and moving parts should be supported on bearings to avoid friction. Stronger electromagnets could alleviate this issue. Note though that these would also increase the power consumption.

Applicable electromagnetic force is limited, which imposes design constraints.

As aforementioned, the synthesized force strongly attenuates depending on the distance to the electromagnet. In our prototype setup, this implies that most actuation mechanisms must be realized within a short range above the surface. Permanent magnets placed beyond that distance only receive minor forces. Similar to the flattened 2D representation of a Madget's state (cf. section 3.9), complex 3D actuations must be implemented so that the permanent magnets are close to the surface; maybe using concepts as Gear Wheel Madgets (section 4.8.4.2). However, for free 3D actuation of controls, like hovering objects, a different setup would be inevitable. The recently published *ZeroN* device by Lee et al. [2011] represents a promising starting point for this kind of actuation: It can translate magnetic spheres through a 3D space using a magnetic levitator.

Range for effective actuation above the surface is small.

In this chapter, we showed how to move and configure tangible controls and how to maintain their physical-visual consistency. In the next chapter, we go a step further and apply electromagnetic actuation to simulate physical effects like friction and resistance in general-purpose controls.

Chapter 5

Rendering Physical Effects

The previous chapter explained how electromagnetic actuation enables bidirectional general-purpose tangible controls that can be arranged on the surface by the software. This actuation technique also maintains the consistency between the physical control and its graphical representation. In this chapter, we apply the concept of electromagnetism to the design process of physical controls.

Developing physical controls requires many design-implementation-evaluation cycles.

Iterative design is a useful and successful method for designing user interfaces [Nielsen, 1993; Buxton and Sniderman, 1980]. The development of physical controls is a matter of many development cycles, comprising design, implementation, and evaluation phases [Baecker et al., 1995]. Early low fidelity prototypes include storyboards, sketches, or 3D models. They are useful to gain high level user feedback that reveals rough and conceptual design issues. Medium fidelity prototypes introduce first physical objects made of cardboard, wood, foam, or 3D printed ABS plastic. They allow users to experience the form factor of the prospective device and to provide feedback about ergonomics. They can already contain simple mechanical elements, like attached buttons. High fidelity prototypes look similar to the final product. Also, the physical *feel* of the device is mostly implemented, e.g., the resistance of a button, or the way it feels to drag a slider handle. Such a prototype can also include basic electronics to communicate the control's state. In each development cycle, a prototype is evaluated, and the feedback is incorporated into the next cycle.

Iterative design is time and cost efficient.

The benefits of iterative design are time and cost efficiency. Early prototypes are cheap and simple to build. In this phase, revising the concept is unproblematic, while undetected conceptual mistakes produce high cost in later phases when the product design has already evolved. During the process, prototypes become more and more refined while the feedback from

Publication
Part of the work in this chapter was published as short paper at the CHI '11 conference [Weiss et al., 2011a].

users is more and more detailed after each iteration. The more iterations are conducted, the more elaborate and expensive prototypes become. Iterative design can be considered as a hierarchical design technique, with few high level decisions in the beginning and many precise refinements in the end.

Building tabletop controls is easy. However, while look is dynamic, feel is fixed after assembly.

Building physical controls for interactive tabletops like SLAP Widgets or Madgets is relatively easy. Using tools like a laser cutter, they can be quickly crafted from low-cost materials, such as wood, acrylic, or silicone. A control's position, orientation, and status can be communicated via optical markers without the need for electrical sensors. Furthermore, the table's back projection allows for quick visual relabeling and feedback. However, while the *look* of such tangibles can be changed dynamically, its *feel* is usually fixed and cannot be changed once a tangible is assembled. If a user study reveals that a knob is too smooth-running, or a button is too lightweight to press, the mechanics must be reconstructed. This can be an extensive and costly process, because new user studies have to be arranged to evaluate further design iterations. However, some of a control's physical properties can be changed dynamically using electromagnetic actuation. Thereby, a user could test multiple haptic configurations within a single test session. An adaptive feel is also a powerful non-visual feedback channel.

Iterating the feel is usually costly but electromagnetic actuation can change feel dynamically.

In this chapter, we explain how electromagnetic actuation can be used to render certain physical properties in tangible controls. These are perceived weight, spring resistance, friction, and notches. We can change these properties on the fly via adaptive electromagnetic force fields. Our techniques not only save effort and costs when iterating general-purpose controls. They also enable dynamic physical properties as a further feedback channel.

5.1 Haptic Rendering in General-Purpose Devices

Haptic Rendering deals with the synthesis of haptic effects.

Synthesizing haptic effects is the main focus of the field of *Haptic Rendering*. We refer to Salisbury et al. [2004] for an overview. The field has originated many haptic devices that were used in Virtual Reality applications, as well as commercial force feedback mice and joysticks, that found their way into the gaming industry.

3D force feedback devices have been used to enrich 2D GUI widgets with force feedback.

A common device used for haptic rendering is the *PHANTOM* [Massie and Salisbury, 1994]. It is a conventional 6DOF force feedback pen for haptic rendering of 3D objects in virtual reality environments. The exact position and orientation is tracked. Internal brakes and actuators provide force feedback and can, e.g., hinder the user to penetrate a virtual object. The PHANTOM provides very accurate 3D force feedback within a certain force range. Besides haptic rendering of 3D objects and surfaces (e.g., [Walker and Tan, 2004]), it has been used to enrich virtual 2D GUI elements with force feedback [Miller and Zeleznik, 1998, 1999]. However, the pen only

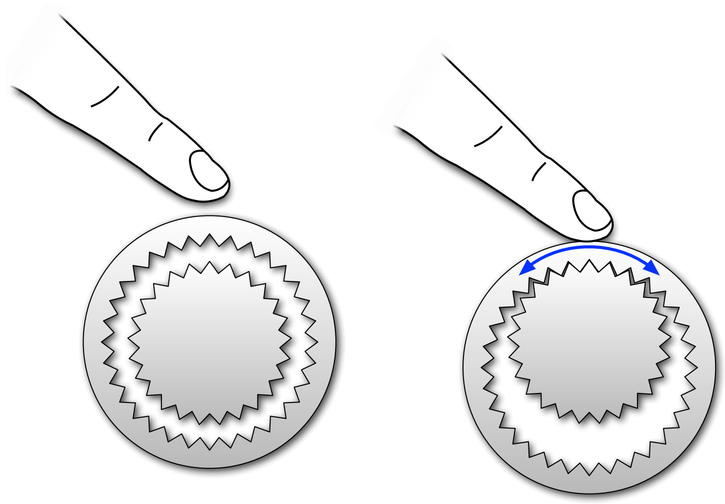


Figure 5.1: Concept of haptic clutch for media navigation as described by Snibbe et al. [2001]. Navigation is modeled as shoving a virtual inner wheel by firmly grasping and turning an outer one. Upon release of the control, the inner wheel keeps turning according to an inertia model.

provides indirect feedback. Beyond that, the PHANTOM device is a heavy stationary device, which makes it unsuitable for interactive tabletops.

Adjustable haptic properties of general-purpose controls can be used as feedback for navigation through media.

Snibbe et al. [2001] suggest to use dynamically adjustable haptic properties of general-purpose controls as an output channel when navigating through media. They present a set of single axis actuated displays equipped with motors, pressure sensors, and brakes. Using their prototype devices, they can simulate different frictions, inertia, and resistances in knobs and sliders. The authors demonstrate various applications in the field of media browsing. An example is a physical knob with an underlying *Haptic Clutch* model. It consists of a virtual outer wheel on top of an inner one (Fig. 5.1). The speed and direction of the inner wheel controls the frame rate and playback direction of a video. If the user grasps the physical knob firmly and rotates it, he virtually shoves the inner wheel. The video plays in the given pace. If the user releases the knob, the virtual inner wheel keeps moving, and the video continues. A user can stop the video by firmly grasping the knob without turning it, or further shove the inner wheel. When turning a knob in *Haptic Fisheye* mode for navigation to a data set, the strength of the user's grasp influences the resolution of browsing. Another example is *Foreshadowing* that uses an increasing haptic texture feedback applied to the knob to indicate the user that he is near a tick marks in a data collection.

Physical effects in general-purpose controls can influence affective responses.

Swindells et al. [2007] investigated the effect of various physical parameters, like the friction, inertia, or detents of a knob, on task completion time and emotional responses of users. In example application contexts, they propose to vary the "feel" of a control as a communication channel. For example, a knob setting critical values in a power plant could feel "unpleasant" if an operator sets risky values, while feeling smooth within normal ranges. The authors conducted a user study measuring task completion time and affective response when operating several active or passive haptic controls, and show that "physical control renderings of position-, velocity-, and acceleration-based effects can significantly influence affective responses" [Swindells et al., 2007, p. 941].

Example: compressible tangible that can simulate various internal springs

Changing physical properties has also been applied to the field of actuated tangibles. For example, *SqueezeBlock* by Gupta et al. [2010] is a compressible tangible that simulates different internal springs. It uses an internal motor attached to a mechanic to render dynamic spring constants and to change them on the fly. This allows, e.g., to provide a haptic "click" response if the user squeezes the tangible beyond a certain level. Hemmert et al. [2010] designed a mobile phone prototype that is able to change its center of gravity by using two orthogonally aligned actuated sliders with an attached weight.

Haptic rendering usually requires hardware that limits applicability on interactive tabletops.

General-purpose controls with varying physical properties have barely been implemented for interactive tabletops. A main reason for this is that haptic rendering often requires extensive hardware, such as motors and brakes. In the following, we show how electromagnetic actuation allows changing physical properties of passive tabletop controls while maintaining their character as lightweight and low-cost tangibles.

5.2 Using Magnetism to Induce Physical Effects

We render physical effects by applying dynamic electromagnetic normal forces to permanent magnets in tangible tabletop controls.

We can render physical effects by applying dynamic electromagnetic normal forces to permanent magnets in tangible tabletop controls or parts of it. For our prototypes and proof of concept studies, we used a single module of the Madgets Table and fastened our controls on the surface. The module contains 12×6 electromagnets with iron cores that span a total interactive area of 25.2 cm × 12.6 cm. Since no tracking was required, and in order to maximize the electromagnetic forces, we detached the LCD panel and EL foil, and replaced the Endlighten layer with a 5 mm acrylic plate. We ran each electromagnet at 35 V. Using the Madgets Table, the strength of each electromagnet is varied over the PWM duty cycle. Values linearly scale between 0% (disabled magnet) and 100% (fully powered electromagnet).

5.2.1 Perceived Weight

Weight of tangible is determined by its mass.

The weight of a tangible is determined by its mass, that is, by the kind and amount of materials involved in its construction. This is a fixed quantity that cannot be changed once the object is built. Yet, using electromagnetic actuation, we can influence the *perceived weight*.

When dragging an object across the tabletop, the user perceives the friction force of the object, that is, the product of friction coefficient and normal

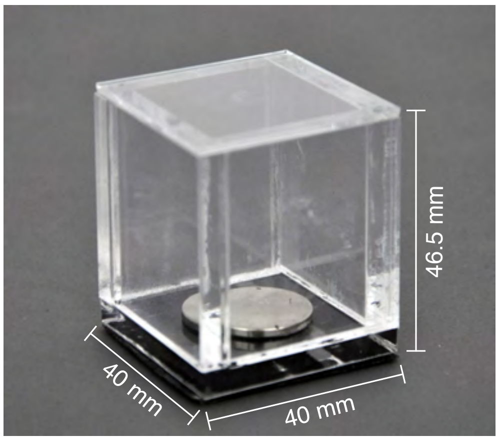


Figure 5.2: Weight Madget. The perceived weight can be varied by applying normal forces to the embedded permanent magnet.

force (cf. equations 4.4 and 4.5). To raise it from the table, she has to overcome the normal force. Naturally, the normal force of an object depends on its mass and the earth gravity. Yet, according to section 4.5.4, we can apply further normal forces to permanent magnets that are attached to a control. By attracting these magnets to the surface, we increase their normal force and, thereby, the normal force of the entire tangible. Consequently, the control is harder to move or detach from the surface; the perceived weight is increased. Contrariwise, repelling fields can decrease the perceived weight.

Perceived weight can be increased by dynamically attracting permanent magnet in tangible.

The ability to dynamically change the perceived weight with varying electromagnetic fields opens a new feedback channel for tabletop tangibles. Weight can inform the user about the state of an object. For example, assume a file management application involving tangibles to represent folders. In analogy to Auditory Icons [Gaver, 1986], the weight of a tangible could encode the size of a file or the number of files in a folder; full folders are harder to move than empty ones. By this, the *virtual* content of folders would gain a *physical* manifestation.

Weight can inform about state of tangible, e.g., amount of virtual content.

Our *Weight Madget* prototype is shown in Fig. 5.2. It consists of a simple acrylic box (base area 40 mm $\times$ 40 mm, height 46.5 mm) with an embedded circular permanent magnet ($\varnothing$ 20 mm, height 2 mm) that is glued on top of the 5 mm base plate. To reduce tangential friction, we mounted a felt pad (thickness 0.25 mm) beneath the box. We measured the varying normal force required to detach the tangible in dependence of the duty cycle using a spring scale. As shown in Fig. 5.3, we can linearly increase the normal force of the tangible. If the electromagnets are disabled, the normal force only consists of the weight force. Note that below 40% no increase in normal force was measurable. Since the permanent magnet polarizes the iron cores in the electromagnets, a minimum threshold must be overcome to create pulling forces.

Weight Madget = acrylic box + embedded permanent magnet

Normal force can be increased linearly.

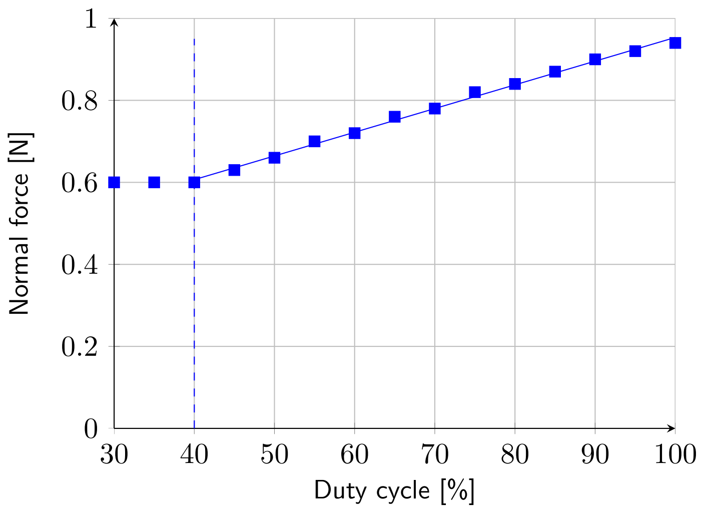


Figure 5.3: Normal force measured on Weight Madget depending on duty cycle. A minimum threshold of about 40% must be overcome to create a force towards the surface.

Effect only works reliably if tangible is touching the surface.

Note the illusion of varying weight only works if the tangible touches the surface. Due to the strong attenuation of force with increasing distance to the electromagnets, the applied normal forces derate as soon as the object is detached from the surface. Considerably stronger electromagnets and accurate tracking of height would be necessary to compensate for this.

5.2.2 Friction

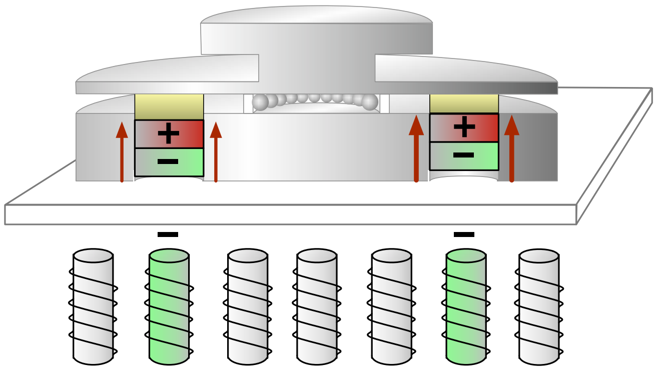
Friction of controls communicates properties and triggers emotions.

The required force to operate a continuous control can communicate properties about the value that is manipulated. For example, a knob that affects crucial parameters of a large machine is harder to turn (which also avoids accidental input), while a knob controlling a less important value is smooth-running. However, also emotional aspects are important in the so-called *affective design*. For example, despite the trend of moving a majority of functions to a remote control and on-screen displays, expensive hi-fi systems always provide a heavy knob that feels “valuable”, as opposed to low-end compact systems. Another already mentioned example are washing machines that provide heavy knobs giving the user a feeling of mechanical control, although the knob merely triggers a micro controller.

Example: increase knob's friction in critical value ranges

Dynamically changing a control's friction could be used as a state indicator of target objects. A slider Madget could change its stiffness according to the virtual object it is paired with. Or a knob controlling a machine could produce a higher resistance if a user tries to set risky values.

In continuous controls, the friction force denotes the resistance of the moveable part against linear or rotational movement. We developed a prototype knob, the Friction Knob Madget, whose resistance can be modified in real-



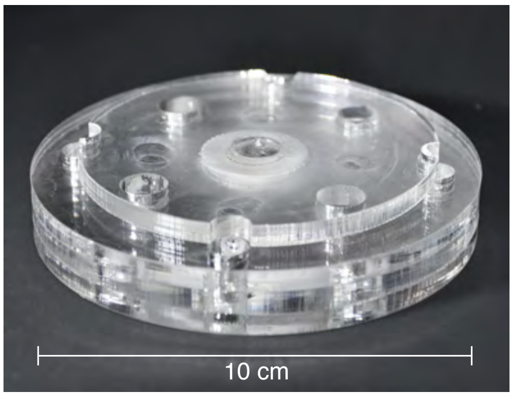
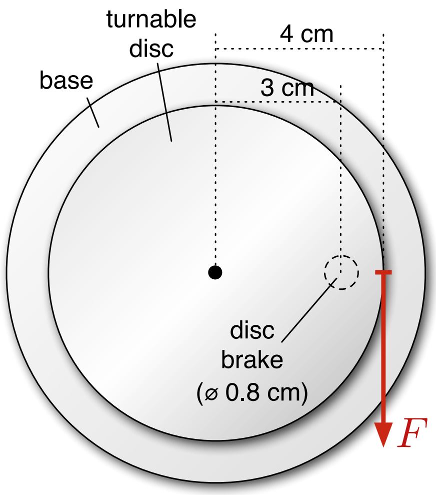


Figure 5.4: Friction Knob Madget. Left: Actuation concept. Brake pads (yellow) are pushed against the turnable disc using electromagnetic forces. Right: Physical prototype used in our measurement. The control can contain up to four disc brakes beneath the disc. In the picture, one brake is embedded (top right).



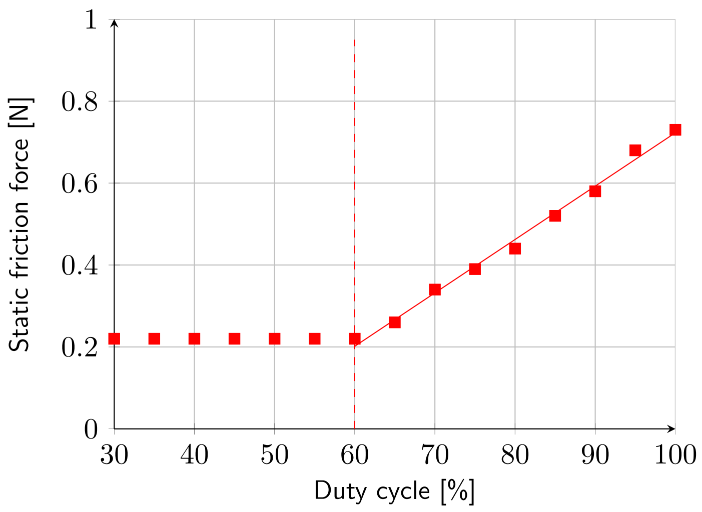


Figure 5.5: Static friction force measured at knob with electromagnetic brake mechanism. Left: Measurement method. Force was measured 4 cm away from the center, orthogonally to the lever. Right: Resulting force depending on duty cycle. The minimum required duty cycle to lift the brake block is about 60%.

time via electromagnetic fields. Our control is shown in Fig. 5.4. It consists of a solid circular acrylic base with a pivot-mounted handle on top. The handle consists of a small disc at the top for convenient turning, and a large disc with the same size as the base. Four disc brakes are embedded in cylindric holes in the base. Each cylindric brake consists of a permanent magnet, an acrylic layer, and a thin brake pad at the top.

Friction Knob Madget = knob + vertically actuated disc brakes

If all electromagnets are disabled, the knob can be rotated easily due to a ball bearing between base and handle. By applying a repelling field to a permanent magnet, the respective brake rises and the pad touches the handle disc. Now friction forces resist the rotation. The friction coefficients depend on the material of the pad and the acrylic as well as on the normal forces. By varying the duty cycle, the normal force can be regulated. Fur-

Friction force is varied by adapting normal force applied to disk brakes.

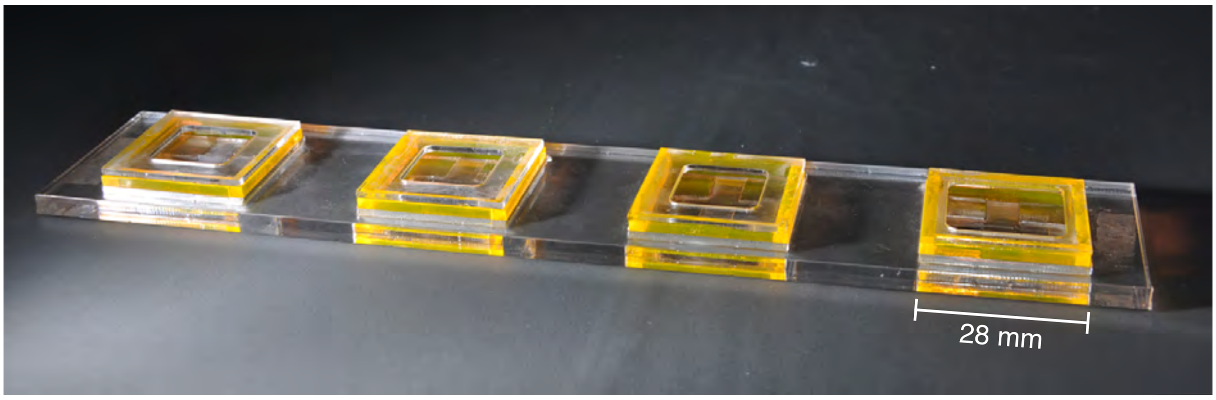


Figure 5.6: Prototype of pushbuttons that provide varying spring resistances through electromagnetic actuation.

thermore, brake pads can be easily replaced and equipped with different materials to control the friction coefficients. Our prototype contains four brakes. This provides a wider friction range and allows to combine different friction coefficients.

Static friction force can be changed linearly via electromagnetic actuation.

We measured the static friction force against duty cycle at our prototype, containing one circular brake pad (⌀ 0.8 cm) with sandpaper coating. We measured the force with a spring scale that was attached to a lever 4 cm away from the center into the direction of rotation (Fig. 5.5, left). As illustrated in Fig. 5.5 (right), we can nearly linearly influence the knob's static friction force via the duty cycle. Note that a minimum of 60% was required to raise the disc brake so that it touches the disc.

5.2.3 Spring Resistance

Buttons with varying resistance are difficult to implement but can, e.g., reduce typing errors.

The resistance of pushbuttons is usually realized with an embedded spring. Buttons with *altering* resistances are rare, because they are difficult to implement. Yet, a recent study proves the usefulness of this feature. Hoffmann et al. [2009] introduced *TypeRight*, a tactile feedback that can vary the stiffness of every key. Their study reveals that typing is less error-prone if unlikely keys are harder to press than likely ones. The likelihood of a key is derived from dictionary look-ups of the currently entered substring. While adapting a button's resistance usually requires elaborate mechanics, electromagnetic actuation allows to simulate changing resistance via varying normal forces.

Button prototype contains four plates with embedded permanent magnets.

Fig. 5.6 shows prototype buttons supporting dynamic resistances. The construction resembles the Radio Button Madget (section 4.8.2.1). Each button amounts to a size of 28 mm × 28 mm. To increase magnetic forces, we placed four flat permanent magnets in the corners of the button's plate. Again the plate is raised by applying repelling electromagnetic fields to permanent magnets. Users can push down the plate 5 mm against the magnetic force. As the force rises with increasing proximity to the electro-

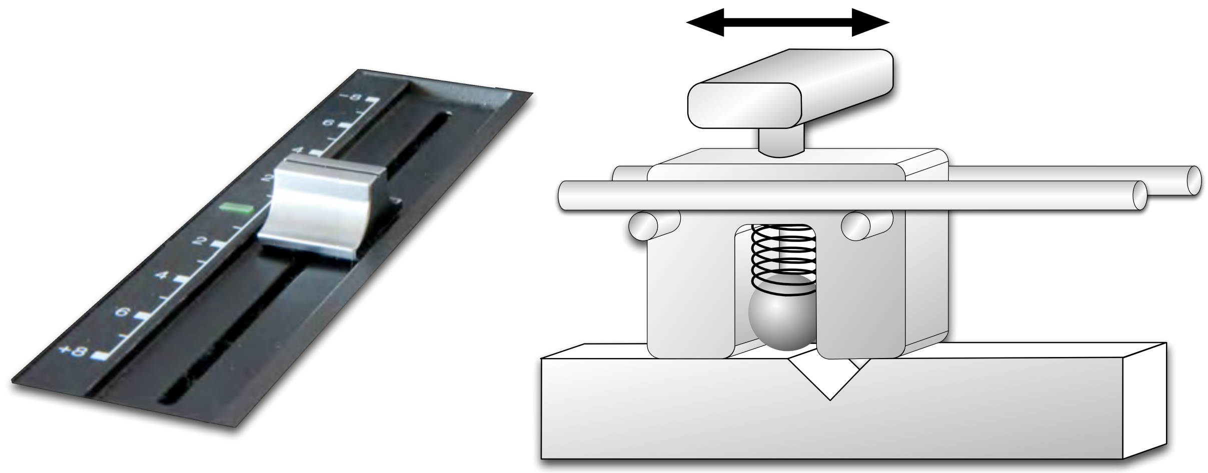


Figure 5.7: Physical notch mechanism. Left: Pitch fader in a turntable with a snapping notch at zero level. Photo taken by the author. Right: Possible implementation of a notch mechanism in a physical slider. If the sliding knob reaches a notch position, an embedded spike snaps into an indentation. To move the knob further, the user must overcome the spring force of the spike.

magnets, this simulates an embedded progressive spring. We can *vary* the resistance of this spring by applying duty cycles between $R_{\min}$ and 100%, where $R_{\min}$ is the minimum duty cycle that is required to raise the plate to the top.

5.2.4 Dynamic Notches

Notches mark discrete positions in continuous controls. If a user moves, e.g., a slider or knob to such a mark, the handle snaps, and a certain force must be overcome to move the handle to a different position. This is a common feature for many everyday devices and usually fulfills one of these purposes:

Notches mark discrete positions in continuous controls.

1. Notches can physically limit the number of allowed states to a discrete set, e.g., when choosing the program at a washing machine's knob (Fig. 1.1 on page 2). Intermediate states cannot be chosen, because the knob automatically snaps to the nearest notch.
2. Notches can produce haptic feedback every time the user performs a discrete step. For example, most jog wheels use notches to indicate frame-by-frame navigation (Fig. 1.3 on page 5).
3. Important points in a scale can be marked by notches. For example, the pitch fader of the turntable in Fig. 5.7 (left) lets the handle snap at the zero level, because this is the default position.

The positions of notches are usually statically implemented by a mechanism like that in Fig. 5.7 (right). However, *dynamic* notches could provide a variety of applications. For example, a time-line slider used for video

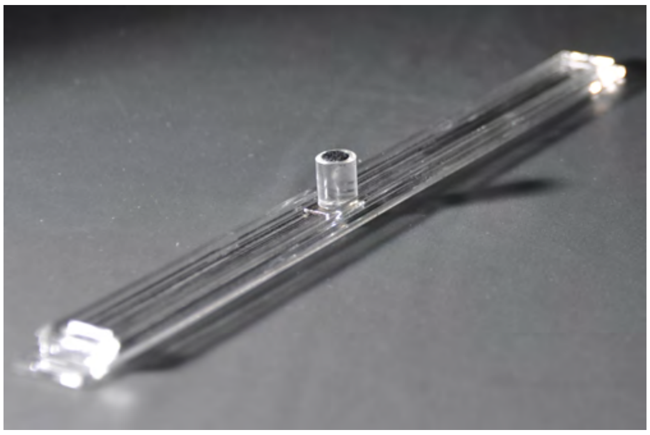


Figure 5.8: Prototype of slider with dynamic notches. The acrylic handle is mounted on top of a permanent magnet.

navigation could activate notches at scene marks on demand. Besides the usual browsing by dragging the slider handle, a quasi-mode could activate these notches for quick scene-to-scene navigation. Another example are physical knobs for navigation through hierarchical circular menus (cf. section 3.3.3.4). The knob would update the number of notches depending on the number of entries in the current submenu.

Notches can be implemented via electromagnetic fields.

While embedding dynamic notches in conventional controls is difficult and requires powered components such as brakes or motors, they can easily be implemented via electromagnetic fields. Fig. 5.8 shows our slider prototype that supports dynamic notches.

Slider prototype contains handle mounted on permanent magnet.

Similarly to the SLAP Slider (section 3.3.3.3), it is made of acrylic and contains a handle that can be shifted to set a distinct value. The handle consists of a permanent magnet at the bottom and an acrylic cylinder on top, connected via a small acrylic block. It can be horizontally slid across the thin base layer over a distance of about 17 cm. To minimize friction, we roughened this layer and coated it with graphite powder from a pencil. The slider is aligned on top of nine electromagnets.

Dynamic notch is an attracting magnetic field next to two repelling ones.

We simulate dynamic notches by creating attracting electromagnetic fields beneath the notch positions. These fields attract the permanent magnet that is attached to the handle. Accordingly, if a handle is shifted near a notch position, the user feels a force towards the notch; and when released, the handle snaps to that position. This effect can be increased with repelling magnetic fields between the notches. Fig. 5.9 illustrates the actuation scheme for different numbers of notches. Note that notches can be removed easily by switching off all magnets. In that case, the slider transforms back to a purely continuous control.

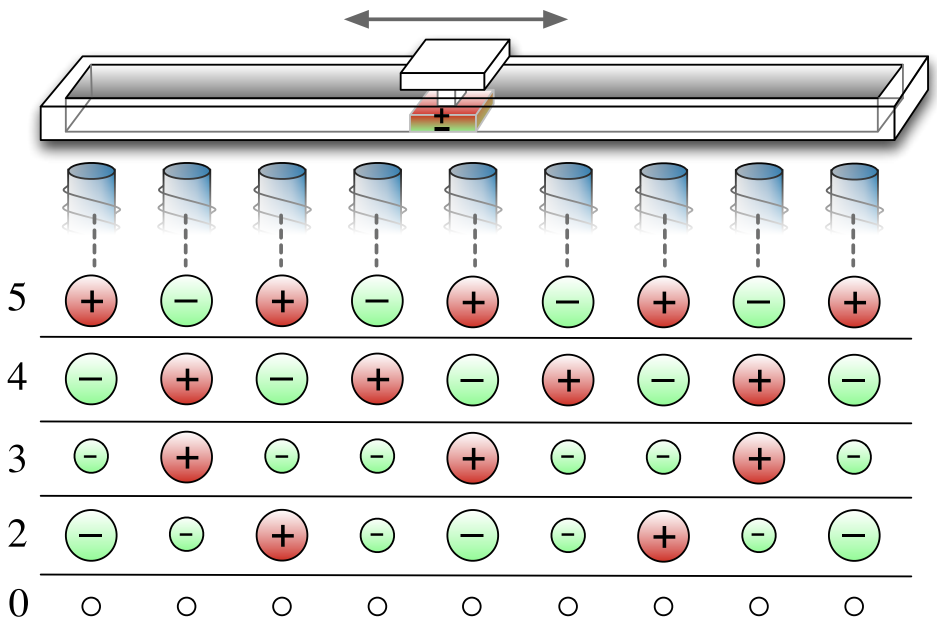


Figure 5.9: Actuation scheme for simulation of two, three, four, and five notches in slider. Circle size represents duty cycles (large = 100%, small = 60%). If all electromagnets are turned off, the slider is continuous (lowest row). Modified image from [Weiss et al., 2011a].

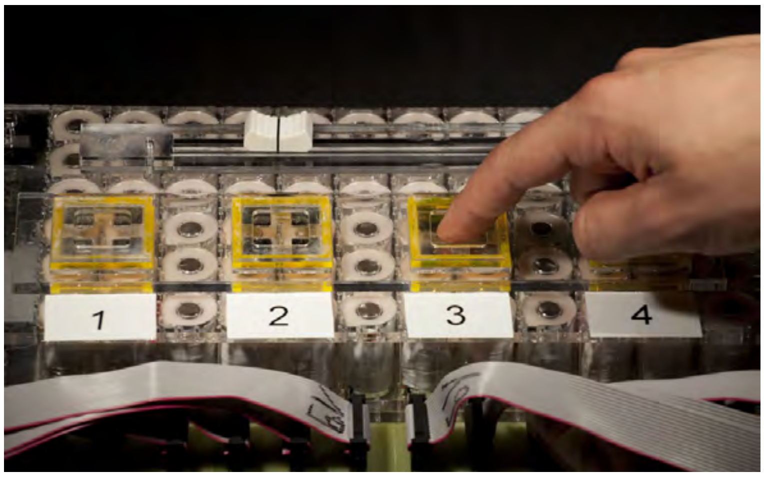


Figure 5.10: Experimental setup for testing buttons with varying resistances (bottom) and slider with dynamic notches (top). Image adopted from [Weiss et al., 2011a].

5.3 User Studies

We evaluated our actuation concepts in two user studies. The goal of these studies was to prove the concept that electromagnetic actuation can simulate different physical effects so that they can be reliably perceived by users.

User tests were conducted for proof-of-concept.

11 participants (3 female) between 23 and 30 years old ($M = 26.3$, $SD = 2.5$) performed two tests in a single session. The user tests involved recognition tasks that were conducted in a double-blind way. Neither the participant nor the instructor could *see* the current condition. Participants had to rely on their haptic sense, while the instructor only initiated the random trials and recorded results. The test setup that the participants faced is shown in Fig. 5.10.

We used the *Magnet Array Controller* application, which was introduced in section 4.5.3, to control the user test. It provides functionality to choose the user test, store the participant's gender and age, and to enter results into a database for later evaluation. The configuration of the electromagnetic array is invisible to the instructor during the entire user test.

5.3.1 Spring Resistance

We first tested whether participants can reliably distinguish different button resistances synthesized by our electromagnetic array. We also intended to gain insight into the output resolution for spring resistances of our specific prototype.

5.3.1.1 Methodology

Task: Push four actuated buttons and order them according to resistances.

Four equal buttons were placed on the electromagnetic array in front of the participant. All buttons were made of the same materials and constructed in the same way using a laser cutter. In each trial, all buttons were raised with different normal forces exerted from the 2×2 electromagnets beneath each plate. Participants could push all buttons as many times as they wanted. They were then asked to sort the buttons by strength in descending order and report the sequence according to the IDs printed in front of the buttons (Fig. 5.10). The instructor then entered the result into our software. Since stronger buttons rise faster than weaker ones, participants had to focus a sign (a smiley printed on paper) at the wall behind the setup *before* the buttons were raised. This avoids that visual clues aid the participant's decision process. A break of 10 seconds between two subsequent trials was added to cool down the surface.

Three conditions involved two, three, and four buttons.

Relative increase of resistance was constant.

We tested three conditions: The strengths of two, three, or four buttons had to be distinguished, respectively. In every condition, we used the smallest duty cycle required to raise a plate, 60%, for the weakest button and 100% for the strongest button. The weakest button requires about 0.040 N to be pushed down, the strongest one about 0.160 N. Following the non-linear relation between strength of stimulus and perception (Weber-Fechner law), we made sure that the *relative* increase between buttons was constant. That is, for the three button condition, we used the duty cycles 60%, 77.5%, and 100%, four buttons were actuated with 60%, 71.1%, 84.3%, and 100%. Note that the plates of all active buttons were pushed against the top; they were always at the same level in the beginning of each trial.

Ten trials per condition with randomized strengths

After reception and instructions about the test procedure, each participant performed ten trials per condition. The test started with two buttons, followed by three and then four buttons. The order of strengths was randomized in each trial. We measured the number of incorrectly reported sequences per condition and participant. We hypothesized that participants could differentiate up to four different levels of resistance.

5.3.1.2 Results

No participant had difficulties to order the strengths of two buttons in all ten trials. In the three button condition, participants made 0.82 mistakes on average ($SD = 0.98$), while five participants correctly determined all sequences. When four different button strengths were given, participants incorrectly reported 1.64 orders on average ($SD = 1.57$), while four participants made no error at all.

5.3.1.3 Discussion

Results provide proof of concept that button strengths can be created in perceivable way.

Our results provide a proof of concept that we can generate different button strengths in a perceivable way. Despite the relatively low forces required to push down buttons, participants could mostly distinguish up to four different button resistances with our prototype, where the four button condition caused higher error rates. Mistakes were equally distributed among all buttons. Thus, minor variations in assembly among the four buttons did not cause the incorrect reports.

Resolution of button strengths depends on various factors that must be balanced.

The resolution of dynamic button resistances is specific to our system and depends on several parameters: A lighter button plate, a shorter actuation distance, or a higher voltage per electromagnet increases the maximum force that can be applied, and, therefore, provides a wider range of distinguishable resistances. Also, more powerful electromagnets could improve the resolution. Permanent magnets should be chosen carefully: Larger permanent magnets provide stronger fields but also introduce additional weight that antagonizes the actuated normal forces. In general, a precise construction process using appropriate tools like a laser cutter is crucial to generate replicable results.

Other types of simulated springs require tracking of vertical button position.

The presented actuation method only simulates progressive springs. To synthesize other types of springs, such as linear ones, a precise tracking of the plate's height in combination with rapidly updating electromagnetic fields would be required. Then, the force of the virtual spring at the current height could be mapped to a duty cycle value that regards the attenuation of the magnetic field.

5.3.2 Dynamic Notches

In the second user test, we tested how well different electromagnetic configurations can simulate dynamic notches in a slider control.

5.3.2.1 Methodology

Conditions involved two, three, four, or five dynamic notches.

Participants operated the slider control that is shown in Fig. 5.10. We replaced the handle of our original prototype (Fig. 5.8) with a professional one from an audio equalizer, because it provided better grip when being operated with a single finger. In each trial, we discretized the slider with two, three, four, or five notches using dynamic electromagnetic fields. Fig. 5.9 shows the different electromagnetic configurations. A notch was implemented as an attracting field and amplified using repelling fields around it.

Task: Operate slider control without looking and report number of notches.

Before the first trial, participants placed the index finger on the slider. To ensure that they only made use of their haptic sense, they were not allowed to look down at the control during the test. Instead, they looked at our sign at the wall. In each trial, participants were asked to report the number of notches they felt. They could shift the slider to the left and right as many times as needed. Afterwards, they named the number they perceived to the instructor, who entered it into the software.

Number of notches was randomized.

After explaining the test procedure, every participant performed ten trials. The number of notches was randomized between two and five for each trial. Again, a 10 second break provided time for a cool down of the hardware.

5.3.2.2 Results

On average, participants made 1.46 mistakes ($SD = 1.51$) in ten trials. The condition involving four notches was correctly recognized in all cases but one (3.7% false rejects). Participants misinterpreted 11.5% and 11.1% of three and five notch condition, respectively. The two notch condition was missed in nearly one third (30%) of the cases.

5.3.2.3 Discussion

Dynamic notches can be simulated via alternatively polarized electromagnets.

The results show that dynamic notches can be simulated via alternatively polarized electromagnets. The combination of an attracting field with two adjacent repelling ones creates a strong attraction towards the notch position. The recognition rates were quite high for three to five notches. However, the two notch condition was not recognized rather frequently. The probable reason for this was the distance of 8.4 cm between the two notches. If the handle was shifted near the middle of the two notches, the attracting force was not high enough to overcome friction and to drag the handle to one of the two positions. To solve this issue, a more active actuation is required: The handle could be first dragged by "intermediate notches" until it would be close enough to the actual notch to snap to its position. This requires a precise tracking of the handle.

Beyond a certain notch distance, active actuation is required.

Our test setup optimally aligns the slider so that each notch position is exactly placed above an electromagnet. Providing notches at intermediate positions for sliders that are placed in any direction on the surface is more difficult. The resolution of the array could be increased, but this approach is clearly limited by physical and economical constraints. Alternatively, the notches could be dynamically synthesized by interpolating fields of adjacent electromagnets. An accurate tracking of the handle's position and a high update rate of the array are crucial to achieve this.

For arbitrary alignment of slider control, exact tracking and interpolated magnet actuations are needed.

5.4 Closing Remarks

In this chapter, we explained how to use the electromagnetic actuation of our Madgets system to dynamically change physical properties of tangible tabletop controls. After discussing related work, we showed how to simulate effects, such as perceived weight, friction, spring resistance, and dynamic notches. Our measurements and user studies provide a proof of concept that these properties can be varied via electromagnetic forces, in levels that are differentiable by users. The concept allows designers to quickly iterate those properties without rebuilding the control and without incorporating electronic parts. Also, properties can be dynamically changed in real-time, which leads to a further valuable feedback channel.

As mentioned in previous chapters, only the area close to the surface is suitable for electromagnetic actuation. This inherently limits the design space for simulating physical properties to shallow controls or mechanisms that can be controlled indirectly from the surface. Although the controls are easy to construct, designers have to learn the specific system parameters to make use of the full haptic resolution. For example, choosing a permanent magnet for a button that is too heavy can cause the button's plate not to rise at all. A future prototype could provide stronger electromagnets to offer a larger range of electromagnetic forces and to enlarge the actuation area above the surface.

Rendering is only possible near to the surface.
Controls are easy to construct, but designers have to learn specific system parameters.

Our prototypes were aligned with the electromagnets to maximize the applied force. For an arbitrary alignment of controls on the surface, forces must be interpolated from multiple electromagnets as described in section 4.5.4.

Until know, we have examined tangible controls on interactive tabletops to improve haptic feedback. SLAP Widgets and Madgets provide precise and eyes-free input. However, in some situations, attaching tangibles to an interactive surface is difficult, e.g., when operating vertical or small touch screens. In the next chapter, we will explain a feedback technique based on electromagnetism that produces a haptic sensation at the user's finger when hovering above the surface. The solution does not involve tangibles and only requires minimal equipment worn by the user.

Chapter 6

Beyond Tangibles: FingerFlux

In the previous chapters, we introduced general-purpose tangibles that enrich interaction on interactive tabletops with haptic feedback. Electromagnetic actuation maintains the coherence between physical objects and their virtual counterparts. Tangibles are beneficial on large interactive tabletops where gravity naturally holds them in position and only a relatively small ratio of the screen is covered by physical objects. However, when it comes to smaller touch screens and those that are not aligned horizontally, tangibles are less practical than pure on-screen controls.

The use of tangible controls on vertical or small touch screens is difficult.

The research community has originated many techniques that provide haptic feedback on small touch screens. A common drawback of most existing methods is their limitation to "on touch" feedback: Haptic feedback is only provided in the moment the user touches the surface. In this chapter, we demonstrate how electromagnetic fields can be employed to produce active force feedback on the surface and in the space above. This concept, called *FingerFlux*, enables a series of applications, including *a priori* feedback, input correction, user guidance, or moderate physical constraints when interacting with on-screen controls.

Problem of most haptic feedback methods on touch screens: Feedback occurs first when surface is touched.

6.1 Haptic Feedback on and Above Surfaces

There is a large body of research that addresses the improvement of haptic feedback on interactive surfaces and devices. Most of them can be assigned

Publications
Part of the work in this chapter was published as short paper at the UIST '11 conference [Weiss et al., 2011b]. It was also presented as demonstration at this conference.

to one of three categories. *Contact-based feedback* generates a haptic sensation directly on the surface when the user touches it. *Mediator-based feedback* transfers surface information via a physical object. *Non-equipped 3D feedback* generates haptic feedback in the 3D space above the surface without any equipment worn by the user.

6.1.1 Contact-Based Feedback

Contact-based methods create feedback in the moment of finger contact. Common example: Vibration feedback when touching mobile screen.

Contact-based feedback is common in mobile devices. An integrated actuator lets the entire device vibrate if certain events occur, such as button clicks. Examples for those actuators are unbalanced motors rotating eccentrically weighted shafts or piezo-based bending motors [Poupyrev et al., 2002]. Different tactile sensations can be produced by varying frequency and amplitude of the actuator signals. For example, Fukumoto and Sugimura [2001] propose to create a "click" feedback after a user has tapped the screen by a single vibration pulse or short burst. Watanabe and Fukui [1995] use ultrasonic vibrations in the order of micrometers to convert a rough surface into one that feels smooth. As the entire device vibrates, those approaches are usually limited to small devices and single-touch interaction.

Pin arrays with vibration and thermal feedback

Yang et al. [2006] introduced a tactile display containing a 6 × 5 pin-array of piezoelectric bimorphs. In addition to multi-point vibration and micro shape feedback, a thermoelectric heat pump in combination with a water cooling mechanism also provides dynamic *thermal* feedback.

Electrocutaenous displays directly stimulate nerves in skin.

Electrocutaenous displays, such as *SmartTouch* [Kajimoto et al., 2003], *directly* stimulate nerves in the skin by applying small current pulses (1-3 mA for 0.2 ms). A matrix of 4 × 4 electrodes embedded into a small tactile display exerts currents though the finger as soon as it is placed over the surface.

Electrostatic feedback applies forces that create vibration sensation at finger.

Electrovibration feedback allows to simulate different kinds of surface friction but works only if finger slides.

A less intrusive method is electrostatic feedback. Tang and Beebe [1998] developed an array of 7×7 electrodes on a wafer. By applying a high voltage between wafer and electrodes, an electric field between a finger's skin and the electrode forms. This field creates an electrostatic force that alternately attracts and repels the finger and, thereby, creates a sensation of texture when the user slides on the surface. *TeslaTouch* employs a transparent electrode layer beneath an insulating layer [Bau et al., 2010]. It creates electrovibration feedback by applying a periodic electrical signal to the electrode. This gives rise to a periodic attractive force to a sliding finger, which is perceived as vibration or friction. TeslaTouch provides subtle haptic feedback and allows users to differentiate between various virtual textures using their haptic sense. Unlike most other approaches, it can also be used on large surfaces. However, a drawback of this method is that it only provides feedback for fingers that slide on the surface. Also, different feedback at multiple points, e.g., for different fingers, cannot be created.

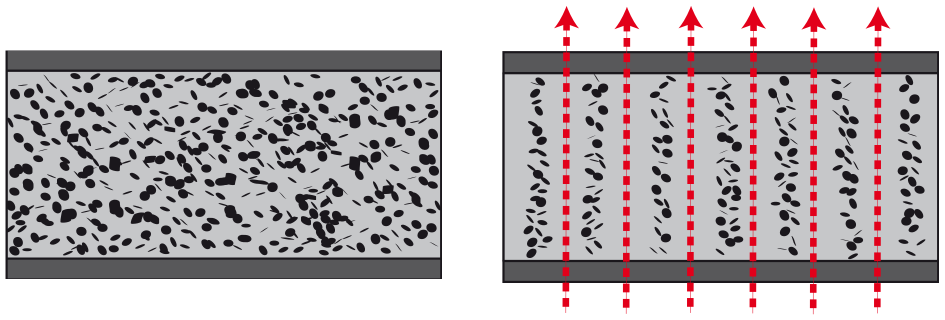


Figure 6.1: Principle of MudPad. Left: By default, particles float freely within the magnetorheologic fluid (low viscosity). The surface feels soft. Right: If a homogenous magnetic field is applied, the particles arrange along the flux lines (high viscosity). The surface becomes stiff. Image courtesy of Jansen et al. [2010].

MudPad changes viscosity of surface by aligning iron particles in surface pouch using electromagnets.

MudPad by Jansen et al. [2010] produces localized haptic feedback by modifying the *viscosity* of the surface material. The surface consists of a pouch filled with magnetorheologic fluid, a suspension of carrier fluid and carbonyl iron particles. Like Madgets, an electromagnet array is placed beneath the surface, whose electromagnets can be controlled individually. By default, the surface is soft and can be pushed easily. If an electromagnet is activated, the particles in the fluid align along the magnetic flux lines; the fluid becomes stiff (Fig. 6.1). According to Jansen et al., the stiffness of the surface can be continuously varied from a viscosity like water to the stiffness of "peanut butter". Within the resolution of the array, MudPad allows to create individual haptic feedback for multiple points on the surface.

Drawback of contact-based feedback: Differentiation between surface palpation and action trigger is difficult.

Contact-based feedback is suitable for many applications. Some of the techniques can be easily integrated into existing devices without requiring any equipment worn by the user. A common drawback of this method is that it only produces *a posteriori* feedback; a haptic sensation is given *after* the user has touched the surface. This can be problematic if touches trigger actions, e.g., when users unintentionally pushbuttons. To distinguish between a user "palpating the interface" and "triggering an input", techniques like modes or time-based interface are required that increase cognitive load and potentially give rise to input errors.

6.1.2 Mediator-Based Feedback

Mediator-based feedback communicates haptic feedback via a physical object operated by the user.

Mediator-based feedback is produced via a physical object that the user holds or that is mounted on the user's hand. The object is tracked and generates haptic feedback depending on its current position.

The *Haptic Tabletop Puck* by Marquardt et al. [2009] is a tangible block for interactive tabletops (Fig. 6.2). The user can drag this device across the table and feel the haptic feedback it produces. A camera in the table tracks

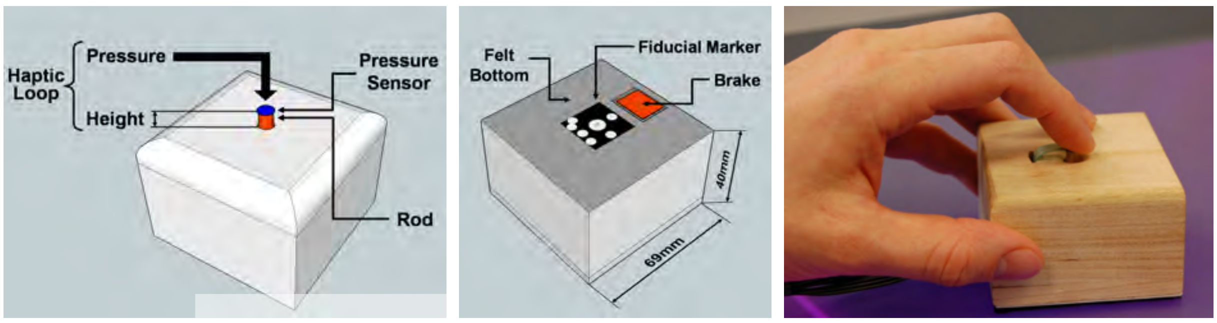


Figure 6.2: The Haptic Tabletop Puck is a tangible that mediates virtual information from the surface via haptic feedback to the user's hand. Image courtesy of Marquardt et al. [2009].

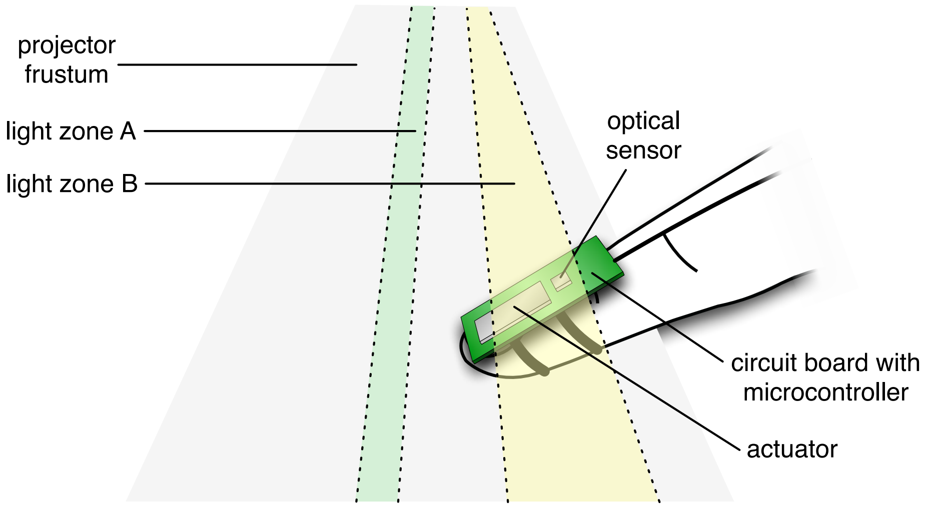


Figure 6.3: Principle of Senseable Rays. A projector emits time-modulated spatial patterns in various light zones. The user wears a tethered circuit board containing an optical sensor and an actuator. If the optical sensor at the finger is held into one of the light zones, the circuit board decodes the light pattern and triggers the actuator accordingly.

Tangibles with embedded actuators can transfer spatial information to the user's finger or hand.

the position of the puck via its fiducial marker. Two components provide haptic feedback. A brake pad on the bottom can slow down dragging across the tabletop. A rod on top of the tangible can dynamically adjust its height, and acts as pressure sensitive pushbutton at the same time. Marquardt et al. present a variety of applications. One example is the haptic exploration of terrain displayed on the table: When moving the puck across the virtual terrain, the rod's height reflects the altitude, vibration patterns applied to the rod represent temperature, while the type of terrain maps to the strengths of the brake; a puck being moved across virtual mountains is more "breaky" than on vegetation.

Mediators can also provide haptic feedback in the 3D space above the surface. *Senseable Rays* by Rekimoto [2009] is a combined tracking and actuation device worn at the index finger (Fig. 6.3). A projector above a surface renders time-modulated light patterns to encode different volumes above the surface. If the user places his index finger into one of these volumes, a photo-detector senses the specific light pattern and triggers the actuator at the finger accordingly. This setup is rather simple and does not require an external tracking detector, albeit the external projector is part of the tracking.

3D force feedback devices, as used in Virtual Reality, are very accurate but indirect and stationary.

The PHANTOM device, which was mentioned in the previous chapter, can be considered as a mediator-based feedback device that produces accurate 3D force feedback. An alternative is the *Magnetic Levitation Haptic Device*[1]. It consists of a handle that can be levitated in six degrees of freedom using a set of permanent magnets and electromagnets. This device allows for 6DOF control in virtual reality scenes and has also been used to synthesize textures [Unger et al., 2008; Grieve et al., 2009]. However, the stationary design and the indirect manipulation limits the use of both devices on interactive tabletops.

Exosceletons are helpful to guide the user's hand or arm, but they are intrusive and require difficult setup.

Exosceletons attached to the user's hand or the entire arm [Bergamasco et al., 1994; Wusheng and Tianmiao, 2003] can provide elaborate force feedback while interacting in 3D space above the surface. Users can feel virtual objects and obstacles in the scene. While providing rich feedback, these methods require much setup time, which limits their use in ad hoc scenarios. Also, these technologies are tethered and heavy. A recent technique employs electrodes to directly stimulate and control muscles in the user's hand [Tamaki et al., 2011]. A goal of this project is to teach manual tasks, such as learning a musical instrument. However, besides the difficult setup, this technique is very intrusive, which might impair the user experience.

Mediator-based feedback is localized and scales well, but ad hoc use on tabletops is often limited.

The major benefit of mediator-based feedback is that it provides localized haptic feedback through a single device. As long as the actuating device is tracked, it is independent of the size of the surface or space it is used for. Thus, those techniques scale well and, unlike contact-based feedback, they allow 3D and a priori feedback in the context of tabletops. The downside is the requirement to wear equipment, which limits "walk-up-and-use" scenarios. Also, the actuating devices require motors, cables, and further hardware that make them heavy, less robust, and intrusive.

6.1.3 Non-Equipped 3D Feedback

3D feedback without equipment worn by the user is still an unsolved issue.

The research on systems that produce 3D feedback *without* any equipment worn by the user is still in its infancy. One potential approach uses an array of air jets nozzles that can be activated individually to create a resisting force at a 3D position above the surface [Suzuki and Kobayashi, 2005]. However, the system currently requires the user to hold a scoop-like "air receiver" to gather an air stream to perceive its force. One reason might be

[1]http://butterflyhaptics.com/

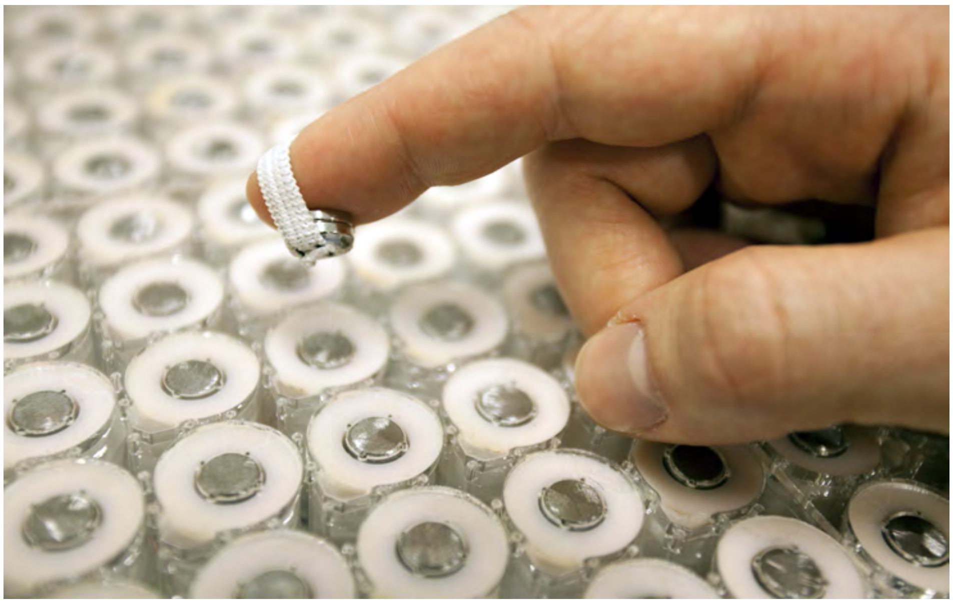


Figure 6.4: Concept behind FingerFlux. Electromagnetic fields reach well beyond the surface and apply forces to a permanent magnet that is attached to the user's finger. The permanent magnet passes these forces to the skin where they induce a haptic sensation. The table's backlighting and LCD display have been removed in this photo for illustration purposes.

that the jets considerably diffuse with increasing distance from the surface. Also, the authors state that the system is rather noisy.

Arrays of air-jets or ultrasound emitters can create localized feedback in 3D, but use on touch-based surfaces is difficult.

Hoshi et al. [2010] use an array of ultrasound emitters to create unrestricted tactile feedback in the air. The method bases on a principle called *acoustic radiation pressure*. The phase and intensity of every emitter can be controlled individually. By overlaying the ultrasound waves, a focal point in 3D space is generated. This point contains a higher air pressure that can be felt by the user.

Both systems provide promising alternatives to mediator-based feedback methods. However, integrating air-jet nozzles and ultrasound emitters into interactive surfaces is difficult.

6.2 Near-Surface Haptic Feedback Using Electromagnetic Fields

FingerFlux creates haptic sensation by applying magnetic fields to permanent magnet at user's finger.

FingerFlux is a novel haptic output method that provides haptic feedback above the surface. It combines electromagnetic actuation with a permanent magnet attached to the user's fingertip (Fig. 6.4). We leverage the fact that synthesized electromagnetic fields reach well beyond the surface. Using these fields, we can influence the permanent magnet that is connected to the finger's skin. Electromagnetic forces applied to the permanent magnet are directly passed to the finger and create a haptic sensation. An attracting field drags the finger towards the electromagnet beneath the surface, a repelling one pushes it away. This is a unique feature, because most other haptic feedback methods only provide either vibration feedback or repelling forces. The ability to *attract* a finger opens a design space for new applications, as will be explained later in the chapter. Although FingerFlux uses a permanent magnet as force mediator, it is relatively unobtrusive, because it does not require the user to wear heavy parts or active electronics.

Haptic feedback bases on composition of attractive and repelling forces.

These two forces, attraction and repulsion, represent building blocks for more complex feedback in the area near the surface. For example, quickly flipping the polarization of an active electromagnet causes a proximate finger to be alternately repelled and attracted with high frequency, creating a vibration feedback. As mentioned in previous chapters, electromagnetic forces strongly attenuate depending on distance, and they are the strongest when users place the finger directly on the surface. Yet, our user studies will reveal that haptic feedback is perceivable up to 35 mm above the surface. Therefore, FingerFlux can be considered as *2.5D mediator-based haptic feedback*.

FingerFlux creates 2.5D mediator-based haptic feedback.

FingerFlux uses Madgets Table as infrastructure.

For our initial prototype and user studies, we employed the Madgets Table that was described in chapter 4. We later developed a smaller, more compact version of the setup (section 6.5). The electromagnets contain iron cores and are powered at 40 V DC and 255 mA. We taped two cylindric neodymium permanent magnets (⌀ 10 mm, height 2 mm) to the user's finger. We used elastic insulation tape, because it provided a higher haptic sensation than rigid tapes. We followed our convention by making sure that the magnets' negative pole always faced downwards to the surface. Note that the two magnets could also be replaced with a single stronger one.

6.3 Applications

FingerFlux provides haptic feedback and supports eyes-free interactions while interacting with touch-screens. Beyond the benefits of contact-based haptic feedback, our technique enables two novel concepts: feeling the interface *before* touching the surface and directional feedback. We will describe these in the following.

6.3.1 Feeling the Interface

Electromagnetic fields reach beyond the surface and can be used to render a haptic representation of the interface in the area above the surface.

6.3.1.1 A Priori Feedback

FingerFlux lets users feel the surface before touching it.

FingerFlux can let users feel elements of an interface before they (involuntarily) initiate an action. This does not just allow for eyes-free interaction but also supports visually impaired people that could otherwise not use the interface.

A single electromagnet exerting a moderate repelling force could create a "haptic bump" on a virtual button. Stronger repelling fields could augment controls that are inactive and "grayed out". Those controls would push approximating index fingers away, indicating that they are not available. Vibration feedback, synthesized by electromagnets with quickly flipping polarization, could accentuate buttons that trigger a critical consequence. For example, if a user accidentally approaches a button labeled with "Discard changes" without looking, the vibration feedback could avoid an unintentional loss of data. Also, vibration *patterns* are imaginable that communicate button labels to blind users, maybe using a simplified version of morse code. Finally, FingerFlux could be used as a private output channel, making certain interface elements only sensible for users carrying permanent magnets.

6.3.1.2 Rendering Objects

Simple lines and shapes can be rendered above the surface.

FingerFlux also allows designers to render lines and simple shapes by arranging repelling electromagnetic fields in a certain form. This could make objects on touch screens perceivable for visually impaired users. Assuming a high density of electromagnets, rendering geographic information like street maps is also imaginable. In this case, streets would be modeled using attracting fields, while other objects would exert repelling forces. Using various strengths for different electromagnets could finally be used to generate 2.5D objects. Note that, due to the nature of magnetic fields, FingerFlux only provides "elastic" feedback and cannot create height maps or sharp features.

6.3.2 Directional Feedback

Attracting electromagnets provide directional feedback.

Most haptic feedback techniques on touch screens provide binary feedback: If a user touches a screen at a certain position, a haptic sensation occurs, otherwise not. Contrariwise, the lessons learned from actuating tangibles (chapter 4 and 5) allow us to generate *directional* feedback: We can attract the finger to a certain position, drag it into a direction, or constrain a user's movement.

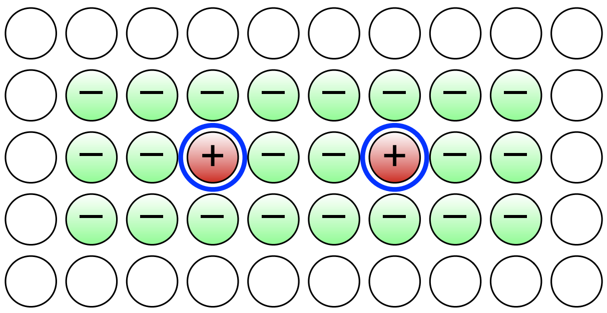


Figure 6.5: Actuation concept to reduce drifting at two on-screen buttons. Users are attracted to the buttons' centers and repelled from the immediate surrounding areas.

6.3.2.1 Reduce Drifting

Attracting finger to positions of on-screen buttons can reduce drifting during eyes-free operation.

A common issue when interacting with touch screens in an eyes-free manner is drifting [Brown et al., 2003]. When pushing on-screen buttons on a touch screen without looking, fingers gradually drift over time. Without visual contact, input becomes error-prone. Although there exist techniques to deal with uncertain input if users press in between two buttons (e.g., [Schwarz et al., 2010]), these techniques only work if drifting does not accumulate too much. Thus, touch screens require visual attention. However, in some situations, eye contact with the interface is not always available or represents a security issues, e.g., when driving a car [Bjelland et al., 2007]. This in one reason why operating touch-screens while driving is prohibited in many countries.

FingerFlux can reduce drifting when operating on-screen buttons without looking. Instead of creating a haptic bump, an attracting electromagnetic field can drag the user's finger towards a button's center every time he approaches it. Repelling fields around the button increase this effect. This actuation concept is illustrated in Fig. 6.5. It compensates input errors at *every* button press so that they do not accumulate over time. Thereby, the need for visual attention is effectively reduced. The method generates *feedforward* by correcting input errors *before* they happen. We will investigate this feature in detail in our user study in section 6.4.2.

Feedforward: Input is corrected before it happens.

The same technique can be applied to create a snap-to-grid function on tabletops. By creating attracting magnetic fields beneath certain grid points, a user's finger snaps to these positions when approaching them.

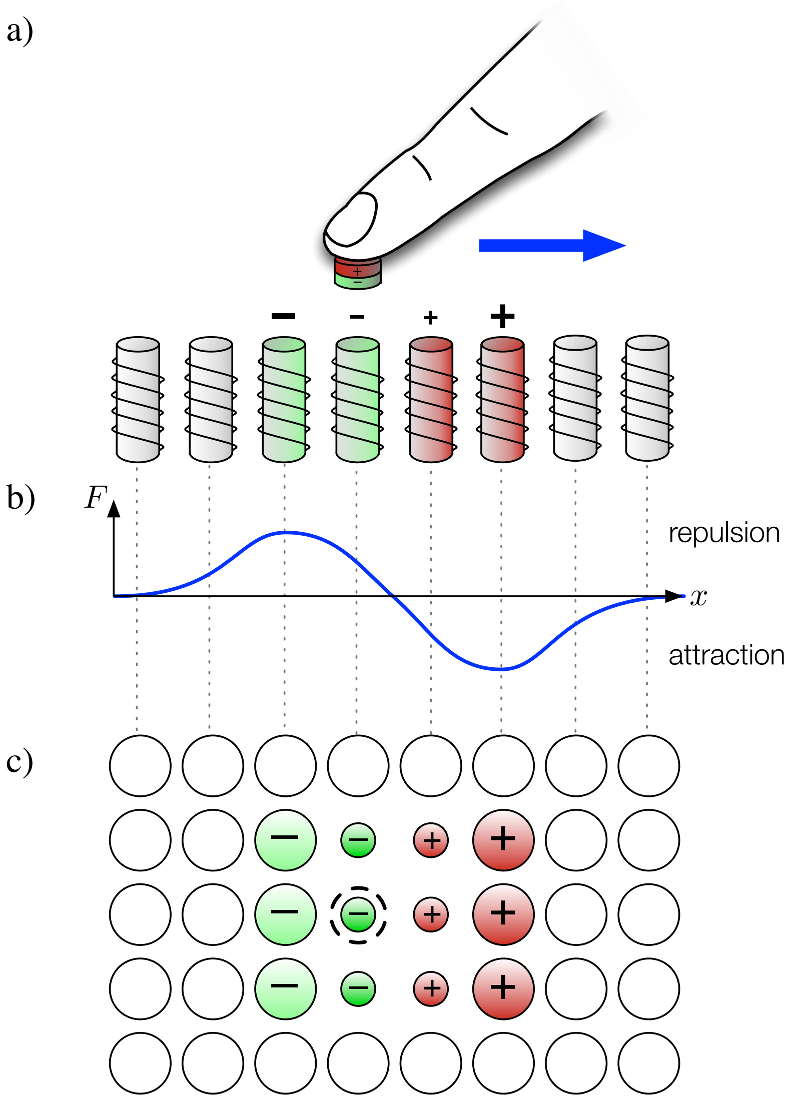


Figure 6.6: Actuation concept for guiding user to the right. a) Side view. b) Illustration of force wave created by actuation gradient. c) Actuation scheme as seen from top. Dashed circle denotes position of finger.

6.3.2.2 Guiding the User

Madgets actuation concepts allow to guide user's finger across surface.

The same actuation concept for moving a tangible across the surface can be applied to drag a user's finger into a certain direction when it hovers above the surface. This can be helpful to teach gestures or to guide blind users to certain points on a touch screen.

Fig. 6.6 shows a gradient actuation pattern to guide a finger to the right. The magnet beneath the finger creates a moderate repelling force to lift it from the surface. Strong attracting fields to the right drag the finger into the desired direction, while strong repelling fields from the left pushes it away from the opposite one. Rotating this pattern by 90 degrees provides guiding schemes for up, left, and down direction, respectively.

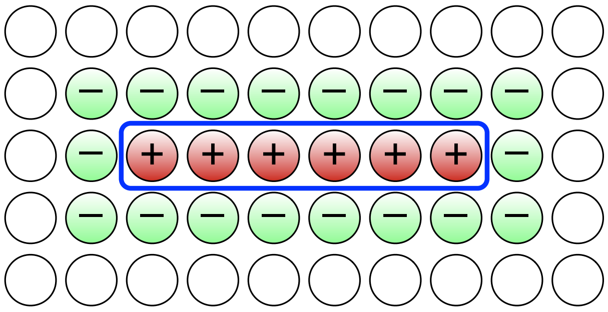


Figure 6.7: Actuation concept of slider with moderate constraint. Dragging within slider area is easy, while leaving it lets users push against repelling forces.

6.3.2.3 Moderate Physical Constraints

Aligned attracting electromagnets surrounded by repelling ones create moderate physical constraints.

We can generalize the aforementioned actuation scheme for buttons to synthesize moderate physical constraints on the surface. Fig. 6.7 illustrates an actuation scheme beneath an on-screen slider. Attracting fields in the slider area and repelling fields around it make it easy to drag the virtual handle in horizontal direction, but dragging vertically or leaving the slider range requires more force. As electromagnets can be reconfigured on the fly, the slider size can be adapted dynamically. Dynamic notches as described in section 5.2.4 can be generated with stronger attracting fields at notch positions.

6.4 User Studies

We conducted two user studies to learn to which extent FingerFlux can provide haptic feedback in the area above the surface, and to explore whether our technique is suitable to correct input a priori, i.e., before the user touches the surface.

6.4.1 Height

Test: Up to what height can users perceive haptic feedback?

In our first study, we intended to determine the maximum distance above the surface at which users can reliably perceive haptic feedback. We chose vibration as feedback technique, because pilot studies revealed that it produces a stronger sensation than constant forces. Alternating repelling and attracting forces created the largest possible force amplitude that the system can create.

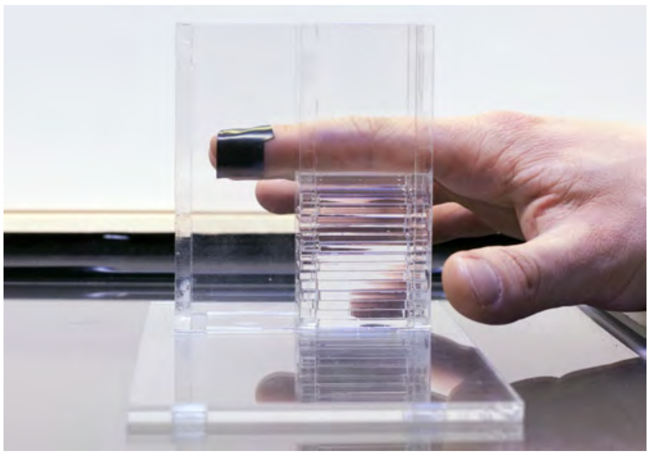


Figure 6.8: Experimental setup of height user study. Image adopted from [Weiss et al., 2011b].

6.4.1.1 Test Setup

Our test setup is shown in Fig. 6.8. A participant placed his finger in an acrylic box that was open to the top and towards the user. The box contained a stack of acrylic plates. During the test, the stack supported the finger in a way that the fingertip was exactly held above an active electromagnet. We controlled the distance of the fingertip to the surface with the number of plates in the stack. Each acrylic plate amounted to a thickness of 3 mm while the bottom plate of the box was 5 mm thick.

6.4.1.2 Procedure

Task: Report whether or not haptic stimulus was given within a time interval.

Participants used our above mentioned prototype consisting of two cylindric neodymium magnets taped to the index finger. In each trial, a test subject placed his finger above an electromagnet. Then, he heard two beep sounds. Between these two sounds, the system randomly chose among two conditions:

Independent variable is the presence or absence of vibration stimulus.

- Condition "No Haptic Feedback": No stimulus was created. The electromagnet beneath the fingertip remained inactive.
- Condition "Vibration Feedback": Stimulus was presented. The electromagnet beneath the finger created vibration feedback by quickly alternating polarization with 10 Hz at full power.

When the participant heard the second beep, he reported whether or not he had felt the stimulus. The instructor entered the answer into the software and started the next trial. In the beginning of the test, the distance between fingertip and surface amounted to 20 mm. After every ten trials, the distance was increased by placing another 3 mm plate on the stack.

The test ended after ten trials at a height of 65 mm. Thus, in total, a participant performed 160 trials (16 height steps × 10 trials).

Double-blind test design

The test was double-blind. Neither the participant nor the instructor could *see* or *hear* whether a stimulus was presented. The latter is crucial because active electromagnets produce subtle sounds. To avoid the presence of sound as a confounding variable, we activated a distant electromagnet in the "No Haptic Feedback" condition. It was too far away from the fingertip to create a noticeable haptic sensation but generated the same sound as the magnet in the "Vibration Feedback" condition. Note that the volume of the sound was so quiet that its position could not be determined. Also, the power consumption, which is displayed to the instructor on the power supply, was equal in both conditions. The software logged the participants' reports and automatically checked the correctness of the answers without presenting them to the instructor during the test.

We define the *maximum height* that can be reliably sensed as the maximum height at which *all* participants give 10 correct answers. Note that the probability for *guessing* the right result of all trials for a given height is only about 0.1%[2].

6.4.1.3 Participants

We tested eight participants between 23 and 28 years old ($M = 25.6$, $SD = 2.10$). One participant was female. All subjects were right-handed but one.

6.4.1.4 Results

Vibration feedback was reliably sensed up to 35 mm above surface.

Fig. 6.9 shows our results. All participants were able to reliably detect the presence or absence of vibration feedback up to 35 mm above the surface. Participant 6 even differentiated signals up to 65 mm.

6.4.1.5 Discussion

Perceivable volume above surface large enough to design haptic feedback for hovering fingers.

Our results show that FingerFlux can produce haptic feedback in a useful volume above the surface. The volume is large enough to use electromagnetic actuation as an haptic output channel for users hovering their fingers above the interface. However, due to the strong attenuation of electromagnetic forces depending on distance, smaller spaces between finger and surface should be preferred if strong feedback is desired.

Result is system-specific.

We tested a specific prototype for a proof of concept. Other systems would probably yield a different volume size. The intensity of the feedback and

[2] $\frac{1}{2}^{10} = 0.0009765625 \approx 0.1\%$.

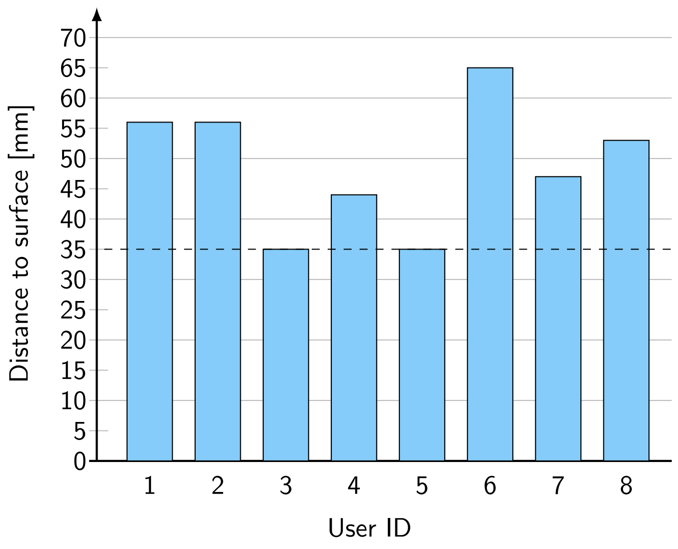


Figure 6.9: Results of height user study. The bars indicate the surface distances up to which each participant did not make any recognition mistakes. Modified image from [Weiss et al., 2011b].

the maximum perceivable height depend on many hardware-specific parameters, such as the configuration of the electromagnets, the applied voltage, the quality and size of the permanent magnets as well as the current temperature of the active electromagnets. Beyond that, our results suggest that there exist inter-individual differences between users regarding the ability to feel haptic sensations.

Inter-individual differences apply.

6.4.2 Reduce Drifting

Test: Can FingerFlux reduce drifting when operating on-screen buttons?

In the second user study, we investigated our claim that FingerFlux can correct input errors a priori by applying directional forces. We tested if electromagnetic attraction can reduce drifting when operating on-screen buttons eyes-free.

6.4.2.1 Test Setup

Again, participants wore our prototype consisting of two combined permanent magnets on the index finger of the dominant hand. During the test, the participant was standing in front of the table. Two circular buttons with a diameter of 25 mm were displayed on the surface. They were horizontally arranged with a distance of 40 mm, which equals a gap of two electromagnets. The buttons were drawn as outline by default and filled when the participant held them down by touching the surface within the circle area.

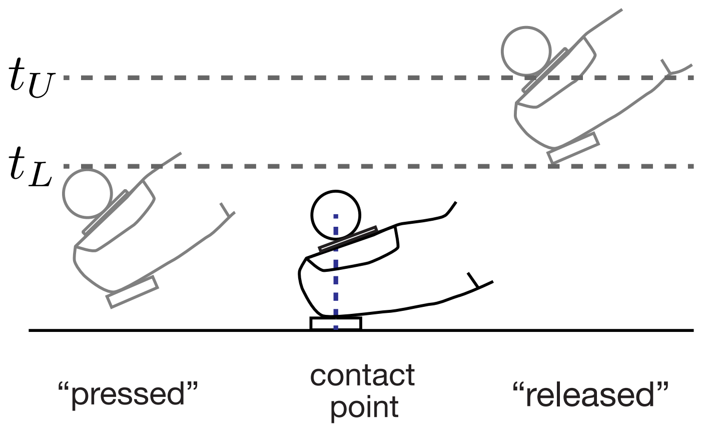


Figure 6.10: Finger touch detection via Vicon tracking system. The 2D contact point equals the projection of the lowest 3D position between the hysteresis thresholds in the plane. t_U and t_L denote the upper and lower threshold, respectively. Modified image from [Weiss et al., 2011b].

We tracked fingers using Vicon system.

We used a Vicon optical tracking system for precise tracking of the finger position. IR cameras around the table captured the 3D position of a retroreflective marker attached to the top of the user's fingertip. The 3D position was measured in millimeter and in local coordinates of the table: The front left corner of the surface determined the origin, the long and short edge the x- and y-axis, respectively. The z-axis equaled the normal vector of the surface. A one time calibration process was required to define this local coordinate system.

We derived button push events from the 3D trajectories as shown in Fig. 6.10: When the Vicon marker on the finger was closer than $t_L = 15$ mm to the surface, a virtual button was considered as “pressed”. When moving it beyond a height of $t_U = 17$ mm, the button was “released”. We derived theses thresholds from a pilot study. This hysteresis thresholding avoided repetitive button presses due to sensor noise. The *contact point* was the projection of the closest position to the surface (lowest z-coordinate) into the x-y plane during the period in which the button was “pressed” (Fig. 6.10).

6.4.2.2 Procedure

Task: Alternately press left and right button as precise as possible without looking.

The participants' task was to alternately press the left and the right on-screen button as precisely as possible without looking at the surface (Fig. 6.11). Participants were not allowed to rest the hands on the surface during our measurements. We tested two different conditions (independent variable):

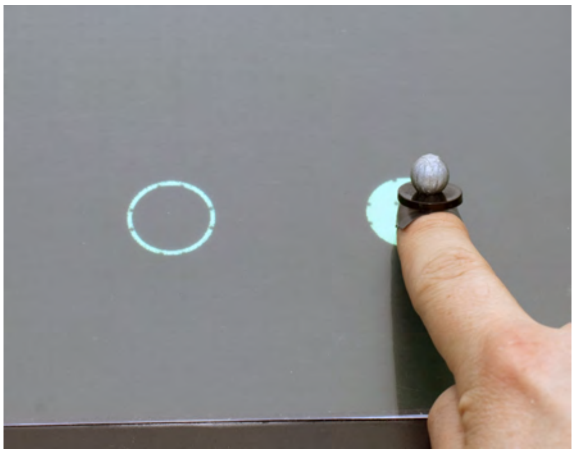


Figure 6.11: Experimental setup of drifting user study.

Independent variable: Presence or absence of forces attracting to buttons.

- Condition "Non-Haptic": No haptic feedback is created. The electromagnet array is deactivated.
- Condition "Active Feedback": The electromagnets beneath the buttons create attracting force, while the surrounding ones create repelling ones. Thereby, a finger nearing the button is attracted to its center.

Within-subject test design, counter-balanced conditions

Participants executed both conditions in a single test while we counterbalanced the order to prevent learning effects. In the beginning of each condition, all test subjects conducted a training phase where they could push the buttons while looking at them. After the initial instructions, participants were aware that the measured trials were conducted eyes-free. Thus, they could memorize the buttons' position during the training phase.

In each condition, participants conducted 15 visual training presses, followed by 45 measured presses with closed eyes. Participants started and ended with the left button. During the test, we logged the contact points of all button pushes. A trial equaled a trigger of the left button, excluding the first one where the participants hovered the index finger above the button. Thus, a test subject performed 7 training and 22 measured trials per condition.

Dependent variable is the cumulative drift over time.

As dependent variable we measured the *cumulative drift* according to Brown et al. [2003]: This value denotes the Euclidean distance between the first contact point on the left button to the contact point of each successive trial. According to Brown et al., the cumulative drift increases over trial number if no visual feedback is given. We hypothesized that the active feedback by FingerFlux could prevent cumulative drifting.

Note that the position of the Vicon marker might have slightly differed among participants. However, this did not impair the results, because we measured the drifting *relative* to the initial left button press.

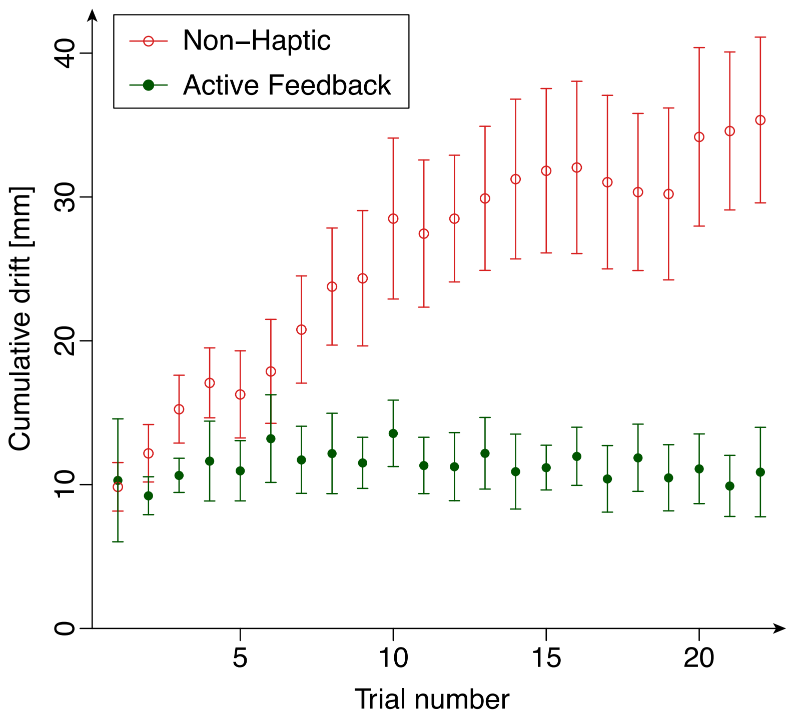


Figure 6.12: Comparison of drifting between "Non-Haptic" and "Active Feedback" condition. Plot shows inter-subject means of cumulative drift in each measured trial. Error bars represent inter-subject standard error. Image adopted from [Weiss et al., 2011b].

6.4.2.3 Participants

We tested 10 participants between 23 and 29 years old ($M = 25.6$, $SD = 2.17$). Two participants were female. All users were right-handed.

6.4.2.4 Results

Absence of haptic feedback yields significant drifting. No significant drifting occurs when feedback is provided.

Fig. 6.12 shows the average cumulative drift of "Non-Haptic" condition (plotted in bright red) versus the active feedback condition (plotted in green). Our study confirms the results by Brown et al. [2003]: If no haptic feedback is given, the participants' cumulative drift significantly increases over trials ($F(1, 219) = 194.11, p < .001, r = 0.69$). Yet, with active haptic feedback, there is no significant drifting ($F(1, 219) = 3.51, p = .062$).

The result is confirmed when looking at the actual contact points of a sample user in Fig. 6.13. The arrows illustrate the order of trials. Apparently, the contact points in the haptic condition (green) are clustered and move into no particular direction. In the non-haptic condition, however, contact points (red) gradually drift downwards.

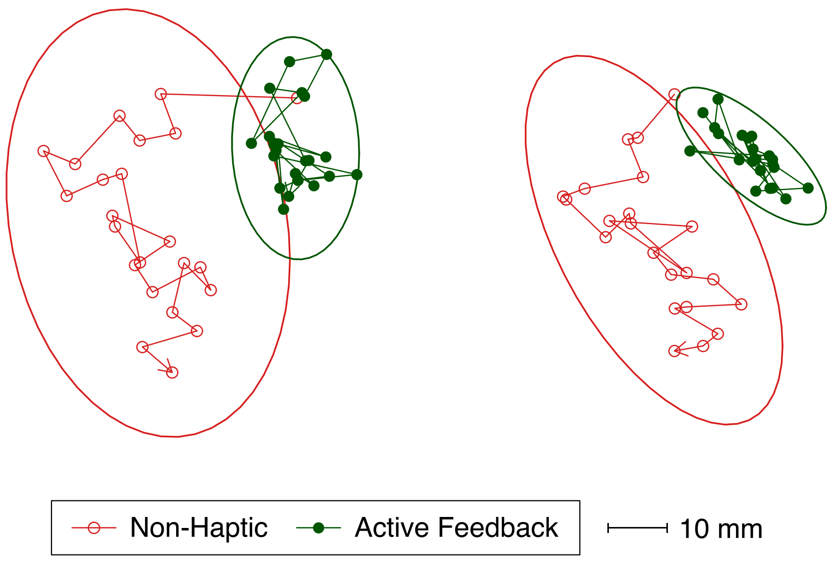


Figure 6.13: Contact points from a sample user in drifting study. Successive contacts are connected by arrows. Ellipses denote the 95% confidence level contours of the normal-probability distributions. Modified image from [Weiss et al., 2011b].

6.4.2.5 Discussion

FingerFlux can reduce drifting when operating on-screen buttons eyes-free.

The results show that FingerFlux can significantly reduce drifting when operating on-screen buttons eyes-free. As in the previous test, the strength of the attraction is specific to our prototype and depends on the particular hardware configuration and factors like the applied voltage and the choice of permanent magnets.

Permanent magnet on fingertip might influence touch performance.

Equipping the user with a permanent magnet can potentially impair the touch precision, because it differs from the direct finger contact that is common for interactive surfaces. This was not an issue in our study, because participants wore the magnet in both conditions. The actual position of the permanent magnet might also have an effect on the performance. According to Holz and Baudisch [2011], users employ visual clues to derive a mental model of where the contact point of a finger touch is. In our study, the electromagnets attracted the finger to a distinct position which does not necessarily match a participant's mental model. This is not an issue in our case, because the buttons were pressed eyes-free. However, in situations where visual and haptic feedback is provided, the position of the permanent magnet should be carefully chosen to avoid interferences between the user's visual and haptic model.

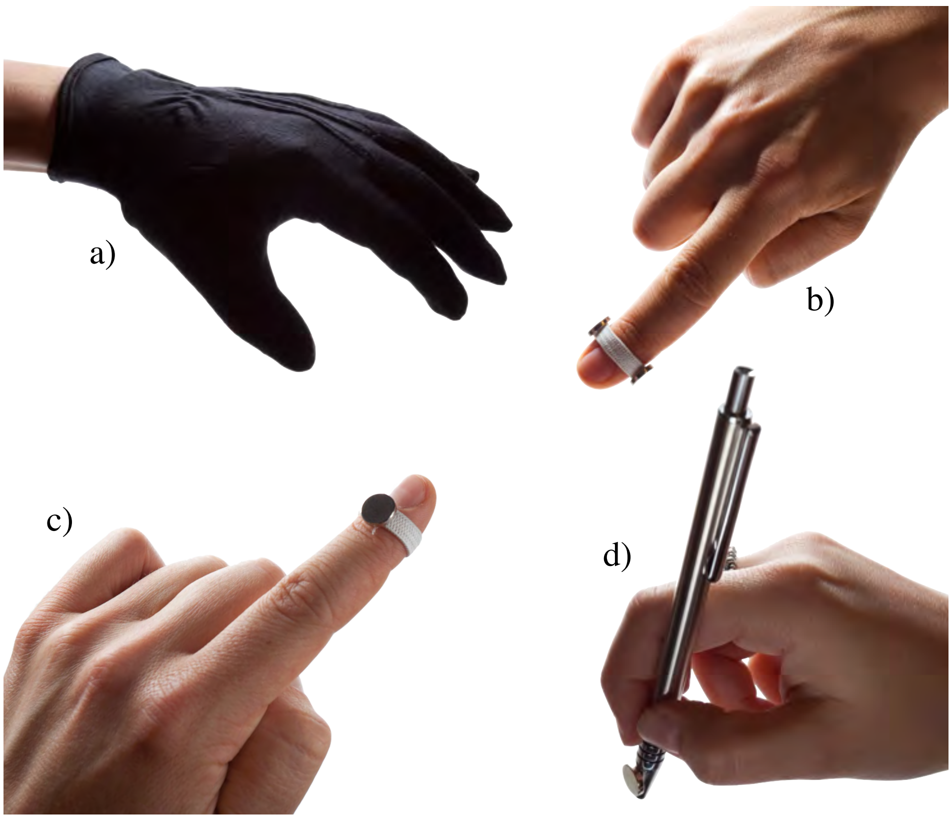


Figure 6.14: Future FingerFlux design concepts. a) Glove with embedded magnets. b) Magnet placed next to a finger tip, c) on top of a finger nail, or d) on a stylus.

6.5 Future Design Concepts

For our FingerFlux prototype, we placed a permanent magnet beneath the fingertip. While sufficient to prove our concept, a magnet between the finger and the surface might reduce the tactile sensation at the finger tip, especially when sensing touches or shearing forces while dragging. Yet, as shown in Fig. 6.14, other designs are straightforward.

Permanent magnet on top of fingernail preserves natural touch feedback but lowers intensity of electromagnetic feedback.

Following Ando et al. [2007], the magnet can be placed on the fingernail to superimpose the virtual haptic feedback to the real world feedback of the surface (Fig. 6.14c). On the one hand, this would preserve the haptic sensation at the finger when touching the surface. On the other hand, the additional distance to the electromagnets would lower the maximum force that can be applied. Beyond that, the skin beneath the finger is considerably more sensitive than the fingernail. Magnetic nail polish could be an alternative of the permanent magnets. It is less noticeable but would presumably yield a smaller magnetic response.

Magnets can be placed next to the finger.

Placing two magnets *next* to the finger could be a compromise that provides a strong response to magnetic forces without placing a magnet beneath the finger tip (Fig. 6.14b). The magnets could be attached to a ring as shown in the *Nenya* project that uses magnets for input [Ashbrook et al., 2011]. In this case, the actuation algorithm must regard that two magnets at different positions must be controlled independently.

Magnets embedded into a glove provide multi-point feedback but also attract each other.

When designing FingerFlux, we first envisioned a glove with embedded permanent magnets (Fig. 6.14a). Multiple magnets could provide haptic feedback at different fingers and other locations on the palm. In general, a glove is convenient, because it is quick and easy to don. However, an inherent challenge when designing such a glove is that permanent magnets attract each other. They have to be placed in a way that they do not influence each other considerably. A glove with embedded magnets also tends to collapse if it is doffed. Nevertheless, a glove represents a promising design.

In body modification scene, permanent magnets have been implanted into fingers.

The idea of using permanent magnets to feel electromagnetic fields has already been realized in the body modification scene. Some users implanted magnets into their fingers to gain a "sixth sense". In several blog posts, users appreciate the ability to raise ferromagnetic objects with a finger only or to feel electromagnetic fields that are, e.g., emitted from a transformer. However, it is questionable if this will become a mainstream trend, especially due to the potential health hazard (e.g., [Kruavit and Numhom, 2008]).

Magnets can also be attached to stylus. This requires tracking of orientation.

FingerFlux could also be used to enhance indirect input methods. Attached to a stylus (Fig. 6.14d), a permanent magnet could, e.g., render a virtual texture, which a user draws on, in a tactile way. Note though that a stylus can be turned in three degrees of freedom. As the orientation of the permanent magnet's pole can vary significantly, this concept requires an accurate tracking of the stylus' orientation.

Implementation into commercial system is feasible.

The Madgets Table is too large to embed it into a conventional system like a car's touch screen. Yet, the implementation of FingerFlux in a commercial system is technically just around the corner. In order to demonstrate FingerFlux at the UIST 2011 conference, we developed a smaller compact version of the electromagnetic array, as shown in Fig. 6.15. It contains 6×5 electromagnets and requires only a single driver board. Unlike the Madgets Table, it uses manganese-zinc ferrite cores. Cooling is provided by placing all electromagnets on a heat sink. Two attached fans dissipate the heat away from the array. The prototype does not provide graphical output, but a backlit LCD panel or a projector could be easily added. To ease shipment, we used a single visual OptiTrack FLEX:V100 camera running at 100 fps in 640×480 for tracking. It was mounted on a tripod above the surface, facing downwards. The 3D position of the user's permanent magnet was derived from the x and y coordinate and the radius of a reflected marker on top of the fingertip.

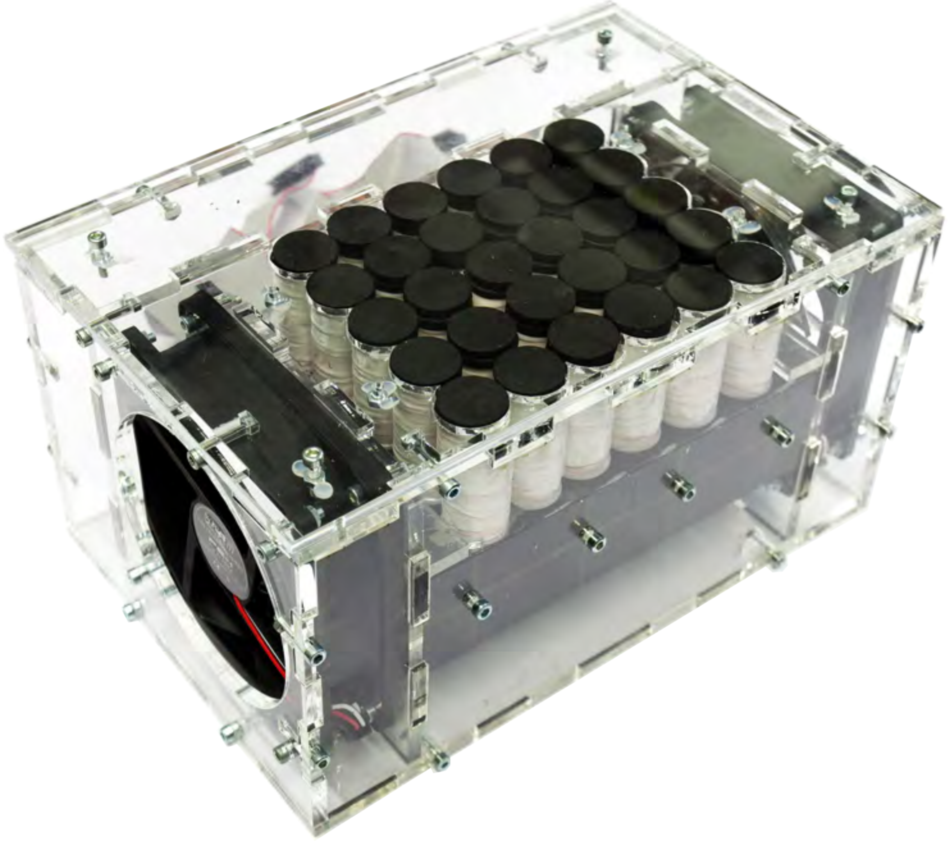


Figure 6.15: New FingerFlux table prototype containing 6 × 5 electromagnets placed on a heat sink and cooled actively. Fingers are tracked using an optical tracking system.

6.6 Closing Remarks

In this chapter, we presented FingerFlux, a haptic feedback method that provides a physical sensation in the near space above an interactive surface. It bases on electromagnetic actuation in combination with a minimal equipment worn by the user, a permanent magnet attached to the user's finger. After discussing related work, we explained our design concepts and potential applications. We presented two user studies that reveal that FingerFlux works in a usable height above the surface, and that it can effectively reduce drifting when operating on-screen buttons without looking. Finally, we illuminated and discussed future design concepts.

FingerFlux provides active, a priori feedback.

FingerFlux is one of the few haptic output methods that provide feedback above the surface without mounting a complex device to the hand. Unlike most contact-based feedback methods, which just inform the user about a surface feature, FingerFlux provides *active* force feedback by employing attracting and repelling directional forces. Users perceive haptic feedback *before* they perform input.

FingerFlux can be helpful for tasks in which visual attention is restricted.

FingerFlux can be helpful in situations where visual attention to a touch screen is unavailable or risky. For example, Pitts et al. [2010] conducted a study on users driving in a car simulator while operating a touch screen. The results suggest that haptic feedback reduces the total time that users spent looking at the screen during the test, while the effect is significant if visual feedback is delayed or absent. We believe that FingerFlux could further reduce this timespan and, thereby, increase the security for such tasks. This should be investigated in a future study.

Equipment with a permanent magnet might reduce user experience but is worth the trouble.

The requirement to wear a permanent magnet might still be a limiting factor in terms of user experience. For haptic interaction with the surface, the magnet must be donned. Yet, this might be worth the trouble if it means that visually impaired users can interact with touch screens who would otherwise have to rely on audio output only. Apart from that, sighted users can employ FingerFlux on demand only, e.g., if they have to focus on a different task while operating a touch screen.

Interpolating magnetic fields is practicable alternative to maximizing array resolution.

Our prototype only provides a limited resolution, albeit it is probably sufficient for many button-based interfaces. The resolution can be increased with smaller and longer electromagnets in the array while optimizing the core and coil material. However, the resolution cannot be scaled down arbitrarily, because there is a physical limitation for minimizing electromagnets. As aforementioned, interpolating forces from adjacent electromagnets might be a more practical solution.

Robustness against tilting requires tracking of finger's orientation.

FingerFlux works best if the magnet is held in parallel to the table, i.e., if the negative pole faces downwards and is orthogonal to the surface. Tilting the finger more than 180 degrees inverses the applied force. However, we did not notice such behavior at our users. To make the system absolutely robust against tilting, the fingers' orientation must be detected, and the electromagnetic forces must be updated according to the magnet's pole.

Depth cameras could replace Vicon tracking.

We used a Vicon system to track the finger in our user studies, which is impractical for ad hoc use. Depth cameras like the Kinect could replace the finger detection. While requiring a higher computer vision effort, no markers would be required anymore. An alternative could be to track the magnets directly, using the aforementioned sensor coils. Also, magnetometers have been successfully applied to sense the position of magnets attached to a finger [Harrison and Hudson, 2009a] or a ring [Ashbrook et al., 2011]. Again, the influence of the synthetic electromagnetic fields generated by the array must be deducted for such a tracking method.

Differences in haptic sensation among users are crucial and must be taken into account.

No "one size fits all" solution

We observed considerable individual differences among users in terms of haptic perception. We demonstrated FingerFlux to over a hundred users at the UIST 2011 conference. The feedback was mixed and ranged from users who only felt vibration but no directional forces, to those who could be easily guided in x and y direction. If multiple magnets are placed on the hand, the varying receptor density in the skin also gives rise to a different sensitivity at every single position. Therefore, designers face inter- and inner-individual differences, which makes a "one size fits all" feedback solution inadequate. A preceding calibration method could solve this; as we set our audio volume when listening to music, we could also adjust the intensity of feedback when operating a FingerFlux device.

Chapter 7

Conclusion

Interactive tabletops represent a promising alternative to conventional desktop computers, especially for collocated collaborative work: Multiple users share the same workspace and perform a common task. Direct manipulation and awareness of each other's actions allow for natural interaction with the system and between users working together. However, productivity applications are still a rare good on interactive tabletops. One reason is the lack of controls that enable an efficient input of abstract parameters and text while respecting the dynamic nature of the tabletop user interface. In this thesis, we developed such controls and brought the haptic qualities of conventional buttons, sliders, knobs, and keyboards to interactive tabletops. In the following, we summarize our contributions and give an outlook on future research.

Goal of the thesis: Haptic general-purpose controls that support efficient input of abstract parameters and text on interactive tabletops.

7.1 Contributions

We introduced SLAP Widgets, physical controls that can be paired with virtual objects on interactive tabletops for precise, eyes-free input. They allow to enter abstract parameters and to type texts, which enables their use in productivity applications. The philosophy behind these controls is to respect the nature of interactive tabletops. Two main aspects distinguish our concept from conventional physical controls. First, SLAP Widgets are passive, which implies that no cables or heavy components are embedded. This makes them lightweight and untethered, a property that is crucial in the context of dynamic tabletop applications, for which the composition of the user interface and the number of users can change quickly. SLAP Widgets can be placed on the table ad hoc, physically handed over to other users, and removed as soon as they are not required anymore. Second, the visual appearance of SLAP Widgets can be adjusted via the table's back projection, which makes them versatile and ensures scalability.

SLAP Widgets enable precise, eyes-free, ad hoc input for general-purpose tasks.

We presented a software framework for developing tabletop applications that support direct manipulation of digital content and ad hoc use of SLAP

Widgets. It is designed in such a way that the underlying widget detection and rendering processes are hidden from the application designer. Nevertheless, the framework enables simple integration of new virtual objects and physical controls.

Our user studies showed that SLAP Widgets can outperform on-screen controls in terms of task completion time and precision, especially in tasks where the control and the data that is manipulated are placed at different locations. Our qualitative study revealed that the concept and handling of our haptic general-purpose controls is easy to learn and intuitive. We presented a study on typing that indicated that the SLAP Keyboard is a potential alternative to an on-screen version. However, additional iterations will be required to reach the haptic feedback and reliability of conventional keyboards.

Madgets maintain physical-virtual consistency.

We developed Madgets to solve inconsistencies that can occur between the physical and virtual state of a control, potentially breaking the illusion of a coherent tangible. To the favor of designers of tabletop controls, we designed Madgets in such a way that they keep all the advantages of SLAP Widgets: passiveness, lightness, and a dynamic visual appearance. We introduced a novel actuation algorithm that allows to move and transform physical controls via dynamic electromagnetic fields. Furthermore, we presented a tracking algorithm based on fiber optics that senses objects and fingers on the surface under low resolution camera input. We demonstrated several novel applications of electromagnetic actuation that reach beyond the maintenance of visual-physical consistency. Especially vertical actuation is a powerful concept that has barely received attention in the field of interactive tabletops.

Electromagnetic actuation allows to vary physical properties in controls.

We also showed how to simulate physical properties in haptic controls by varying electromagnetic forces. Our measurements and proof of concept studies indicated that we can reliably control perceived weight, friction, spring resistance, and number of notches. Our techniques not only support rapid prototyping of controls, they also enable dynamic physical properties, which can be adjusted at run-time, as information and feedback channel.

FingerFlux provides active, near-surface haptic feedback.

Vertical actuation takes advantage of the effect that electromagnetic fields reach well beyond the table's surface. We employed this effect to extend the haptic feedback channel to the space above the tabletop. We developed FingerFlux, an output method that creates haptic feedback in the area near the surface. Haptic sensations are created by exerting electromagnetic fields on a permanent magnet that is attached the user's finger. Our first study showed that users can sense haptic feedback in a useful area near the surface. In contrast to other haptic output methods, FingerFlux provides *active* feedback that can guide the user. This was confirmed by our second study. It revealed that FingerFlux can effectively reduce drifting when users operate virtual buttons without looking. FingerFlux is especially useful for small or non-horizontal touch screens, where the applicability of physical controls is difficult.

7.2 Future Work

This thesis provided the foundation for haptic general-purpose controls on interactive tabletops. At the same time, it highlighted new opportunities for further research.

General-purpose controls could be implemented on tabletops with different tracking technologies.

Optical tracking was the predominant method for detecting fingers and objects on interactive surfaces. A basic setup can be implemented without the need for industrial manufacturing. The publication of FTIR to a wide audience gave rise to a large community of tabletop researchers and an flourishing do-it-yourself movement, which iterated and improved visual tracking methods. During the development of our projects, new tracking technologies evolved, and today's non-optical methods are becoming affordable. It is likely that capacitive and resistive screens will play a greater role in future tabletop systems, because they are thinner and less sensible to ambient lighting. Bringing the concept of SLAP Widgets to these devices is clearly an interesting direction for future work. First projects have already been published that track controls on touch screens with non-optical sensing. Capacitive or resistive tracking could also replace the rather complex fiber optical tracking of Madgets.

SLAP Widgets and Madgets could be brought to non-horizontal surfaces and mobile devices.

SLAP Widgets and Madgets base on horizontal surfaces, relying on gravity to keep them in place when not being used. Yet, other angles are imaginable, e.g., placing general-purpose controls on vertical surfaces like whiteboards. This is a challenging issue, because holding a control in place becomes more difficult on non-horizontal surfaces. Actuating these controls would require a novel mechanism. Another platform worth looking at are touch-based smart phones and tablets that have received much interest in recent years. General-purpose controls that are attached to these devices can considerably enhance the user experience.

Pure addition of haptic feedback does not necessarily increase usability. Further iterations are required, involving an interdisciplinary team.

An important lesson we learned from developing SLAP Widget prototypes is that the pure addition of haptic feedback does not imply an increase in effectiveness. This especially mattered for the SLAP Keyboard. Although all keys of the rigid version provided a clear pressure point and mimicked the behavior of keys of a conventional keyboard, their haptic feedback did not yield better results than a pure on-screen version. As all other widgets, the keyboards ran through many iterations until their haptic experience was satisfying. However, further prototypes will be required to reach the quality of a conventional keyboard. This will necessitate an industrial manufacturing process and an interdisciplinary team of researchers, designers, and engineers.

Further potential for electromagnetism in interface design

The unique property of electromagnetic actuation is the ability to influence objects over a distance in a calm, unobtrusive way. In this thesis, we demonstrated various applications of this technology. Yet, we believe that there is still unexplored potential to exploit electromagnetism in interaction design. For example, a recent project uses an array of electromagnets to arrange rheologic fluid on a surface to create "programmable blobs" [Wakita et al., 2011]. It is imaginable that, some day, electromagnetic actuation could be used to assemble complex 3D interfaces.

Area of conflict between actuation and user control is an interesting research direction.

Madgets maintain consistency and enable new actuation dimensions. However, our preliminary studies indicate that the need for actuation and the users' desire to have control over the UI result in a conflict. For example, the predictability of actuation plays an important role for the user experience. In collaborative scenarios where physical widgets are shared, conflicts are likely occur. Finding rules to overcome these and deriving guidelines to let users feel in control while providing a consistent UI is an exciting and important future research direction.

Investigation of inter-individual differences in haptic perception could improve sensations created by FingerFlux.

FingerFlux is a suitable method to provide haptic feedback above the surface with a minimal equipment worn by the user. However, the demonstration of FingerFlux to a wide audience showed us how strongly inter-individual differences influence the haptic perception. In future work, these differences should be investigated in more detail. The result could be a one-time per-user calibration whose results could be incorporated into the actuation algorithm. A further promising direction is the guidance of the user's finger across the surface. Refining this feature could significantly help visually impaired users to interact with touch screens.

Development of productivity applications is required for further investigation of general-purpose controls.

On a higher level, productivity applications should be implemented and tested on interactive tabletops. As conventional GUI applications cannot just be migrated to tabletops, a design from scratch might be advisable that involves tabletop experts, UI designers, and target users. A result of this research could be concrete guidelines or design patterns for developing productivity applications on tabletops. Our general-purpose controls should be evaluated in the light of these applications, in long term studies and under real-world conditions. Also, pairing techniques should be compared in a controlled experiment.

Rapid prototyping tool could improve the creation of general-purpose controls.

Finally, it is worthwhile to take a closer look at the process of designing general-purpose controls. The visualization and event logic of our widgets are specified in the software. Alternatively, a rapid prototyping tool could be developed that supports the creation and incorporation of SLAP Widgets and Madgets without the need for writing programs. We also believe that the concept of using a Madget as a building block for composing controls with more complex mechanics is a promising direction for future work.

7.3 Closing Remarks

We presented a set of physical general-purpose controls, haptic feedback methods, and interaction concepts that enable precise and eyes-free input on interactive tabletops. Our haptic controls are especially useful for entering abstract parameters and text, which is crucial in productivity applications. Nevertheless, our techniques can be combined with other input methods. There might be situations when tapping an on-screen button might be faster than grabbing a SLAP Keypad, pairing it and pushing the key. SLAP Widgets and Madgets are designed in a way that they can coexist with other interaction concepts. They are especially suitable for tasks that require precise, eyes-free input in an ad hoc workspace.

SLAP Widgets and Madgets can coexist with other techniques.

During the development of all systems, we faced many engineering challenges and there is a great potential for further iterations and improvements. In the recent decades, interactive tabletops were part of a very agile and dynamic research movement that was and still is producing many novel technologies. It is likely that concepts like SLAP Widgets, Madgets, or FingerFlux can soon be realized with smaller form factors and improved technology.

Improved implementations with smaller form factors are just around the corner.

We hope that this thesis will be an inspiration for other researchers who intend to create interactive surfaces that bridge the gap between the digital and the physical world, between dynamic graphical user interfaces that encourage a natural, collaborative interaction and devices that allow users to employ their full spectrum of haptic abilities.

Bibliography

M. Abednego, J. Lee, W. Moon, and J. Park. I-Grabber: expanding physical reach in a large-display tabletop environment through the use of a virtual grabber. In *ITS '09: Proceedings of ACM International Conference on Interactive Tabletops and Surfaces*, pages 61–64, 2009.

C. Alexander, S. Ishikawa, and M. Silverstein. *A pattern language: towns, buildings, construction.* Oxford University Press, 1977.

H. Ando, J. Watanabe, M. Inami, M. Sugimito, and T. Maeda. A fingernail-mounted tactile display for augmented reality systems. *Electronics and Communications in Japan (Part II: Electronics)*, 90(4):56–65, 2007.

A. N. Antle, A. Bevans, J. Tanenbaum, K. Seaborn, and S. Wang. Futura: design for collaborative learning and game play on a multi-touch digital tabletop. In *TEI '11: Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*, pages 93–100. ACM, 2011.

D. Ashbrook, P. Baudisch, and S. White. Nenya: subtle and eyes-free mobile input with a magnetically-tracked finger ring. In *CHI '11: Proceedings of the 2011 annual conference on Human factors in computing systems*, pages 2043–2046. ACM, 2011.

R. M. Baecker, J. Grudin, W. A. S. Buxton, and S. Greenberg. *Readings in Human-Computer Interaction: toward the year 2000, second edition.* Morgan Kaufmann Publishers Inc., 2nd edition, 1995.

J. Barrett and H. Krueger. Performance effects of reduced proprioceptive feedback on touch typists and casual users in a typing task. *Behaviour & Information Technology*, 13(6):373–381, 1994.

O. Bau, I. Poupyrev, A. Israr, and C. Harrison. TeslaTouch: electrovibration for touch surfaces. In *UIST '10: Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 283–292. ACM, 2010.

P. Baudisch, T. Becker, and F. Rudeck. Lumino: tangible blocks for tabletop computers based on glass fiber bundles. In *CHI '10: Proceedings of the 28th international conference on Human factors in computing systems*, pages 1165–1174. ACM, 2010.

J. S. Beeteson. *Visualising magnetic fields: numerical equation solvers in action.* Academic Press, Inc., 2001.

H. Benko and D. Wigdor. Imprecision, inaccuracy, and frustration: the tale of touch input. In *Tabletops - Horizontal Interactive Displays*, Human-Computer Interaction Series, pages 249–275. Springer London, 2010.

H. Benko, A. D. Wilson, and R. Balakrishnan. Sphere: multi-touch interactions on a spherical display. In *UIST '08: Proceedings of the 21th annual ACM symposium on User interface software and technology*, pages 77–86. ACM, 2008.

M. Bergamasco, B. Allotta, L. Bosio, L. Ferretti, G. Parrini, G. Prisco, F. Salsedo, and G. Sartini. An arm exoskeleton system for teleoperation and virtual environments applications. In *Proceedings of the IEEE International Conference on Robotics and Automation, 1994*, pages 1449–1454. IEEE, 1994.

X. Bi, T. Grossman, J. Matejka, and G. Fitzmaurice. Magic desk: bringing multi-touch surfaces into desktop work. In *CHI '11: Proceedings of the 2011 annual conference on Human factors in computing systems*, pages 2511–2520. ACM, 2011.

C. M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer-Verlag New York, Inc., 2006.

H. V. Bjelland, T. Hoff, C. A. Bjørkli, and K. I. Overgard. A case study of a touch based interface for in-car audio systems. *Design Journal*, 10(1): 24–34, 2007.

F. Block, M. Haller, H. Gellersen, C. Gutwin, and M. Billinghurst. VoodooSketch: extending interactive surfaces with adaptable interface palettes. In *TEI '08: Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 55–58. ACM, 2008.

F. Block, H. Gellersen, and N. Villar. Touch-display keyboards: transforming keyboards into interactive surfaces. In *CHI '10: Proceedings of the 28th international conference on Human factors in computing systems*, pages 1145–1154. ACM, 2010.

J. Borchers. *A pattern approach to interaction design.* John Wiley & Sons, Inc., 2001.

S. Brave, H. Ishii, and A. Dahley. Tangible interfaces for remote collaboration and communication. In *CSCW '98: Proceedings of the 1998 ACM conference on Computer supported cooperative work*, pages 169–178. ACM, 1998.

L. E. Brown, D. A. Rosenbaum, and R. L. Sainburg. Limb position drift: implications for control of posture and movement. *Journal of Neurophysiology*, 90(5):3105–3118, 2003.

W. Buxton and R. Sniderman. Iteration in the design of the human-computer interface. In *Proceedings of the 13th Annual Meeting, Human Factors Association of Canada*, pages 72–81, 1980.

X. Cao, A. D. Wilson, R. Balakrishnan, K. Hinckley, and S. E. Hudson. ShapeTouch: leveraging contact shape on interactive surfaces. In *TABLETOP '08: 3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems*, pages 129–136. IEEE, 2008.

L. Chan, S. Müller, A. Roudaut, and P. Baudisch. CapStones and ZebraWidgets: sensing stacks of building blocks, dials and sliders on capacitive touch screens. In *CHI '12: Proceedings of the 30th international conference on Human factors in computing systems*, pages 2189—2192. ACM, 2012.

P. H. Dietz and B. D. Eidelson. SurfaceWare: dynamic tagging for microsoft surface. In *TEI '09: Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, pages 249–254. ACM, 2009.

P. Dragicevic and Y. Shi. Visualizing and manipulating automatic document orientation methods using vector fields. In *ITS '09: Proceedings of ACM International Conference on Interactive Tabletops and Surfaces*, pages 65–68. ACM, 2009.

F. Echtler, A. Dippon, M. Tönnis, and G. Klinker. Inverted FTIR: easy multitouch sensing for flatscreens. In *ITS '09: Proceedings of ACM International Conference on Interactive Tabletops and Surfaces*, pages 29–32. ACM, 2009.

R. Fiebrink, D. Morris, and M. R. Morris. Dynamic mapping of physical controls for tabletop groupware. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 471–480. ACM, 2009.

L. Findlater, J. O. Wobbrock, and D. Wigdor. Typing on flat glass: examining ten-finger expert typing patterns on touch surfaces. In *CHI '11: Proceedings of the 2011 annual conference on Human factors in computing systems*, pages 2453–2462. ACM, 2011.

G. W. Fitzmaurice, H. Ishii, and W. Buxton. Bricks: laying the foundations for graspable user interfaces. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 442–449. ACM Press/Addison-Wesley Publishing Co., 1995.

M. Fukumoto and T. Sugimura. Active click: tactile feedback for touch panels. In *CHI EA '01: Extended Abstracts on Human factors in computing systems*, pages 121–122, 2001.

R. Gabriel, J. Sandsjö, A. Shahrokni, and M. Fjeld. BounceSlider: actuated sliders for music performance and composition. In *TEI '08: Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages 127–130. ACM, 2008.

W. W. Gaver. Auditory icons: using sound in computer interfaces. *Human-Computer Interaction*, 2(2):167–177, 1986.

L. Giusti, M. Zancanaro, E. Gal, and P. L. T. Weiss. Dimensions of collaboration on a tabletop interface for children with autism spectrum disorder.

In *CHI '11: Proceedings of the 2011 annual conference on Human factors in computing systems*, pages 3295–3304. ACM, 2011.

S. C. Goldstein, T. C. Mowry, J. D. Campbell, M. P. Ashley-Rollman, M. De Rosa, S. Funiak, J. F. Hoburg, M. E. Karagozler, B. Kirby, P. Lee, P. Pillai, J. R. Reid, D. D. Stancil, and M. P. Weller. Beyond audio and video: using claytronics to enable pario. *AI Magazine*, 30(2), 2009.

T. Grieve, Y. Sun, J. M. Hollerbach, and S. A. Mascaro. 3-D force control on the human fingerpad using a magnetic levitation device for fingernail imaging calibration. In *WHC '09: Proceedings of the World Haptics 2009 - Third Joint EuroHaptics conference and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 411–416, 2009.

T. Grossman and D. Wigdor. Going deeper: a taxonomy of 3D on the tabletop. In *TABLETOP '07: Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 137–144. IEEE, 2007.

S. Gupta, T. Campbell, J. R. Hightower, and S. N. Patel. SqueezeBlock: using virtual springs in mobile devices for eyes-free interaction. In *UIST '10: Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 101–104. ACM, 2010.

J. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pages 115–118. ACM, 2005.

C. Harrison and S. E. Hudson. Abracadabra: wireless, high-precision, and unpowered finger input for very small mobile devices. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 121–124. ACM, 2009a.

C. Harrison and S. E. Hudson. Providing dynamically changeable physical buttons on a visual display. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 299–308. ACM, 2009b.

B. Hartmann, M. R. Morris, H. Benko, and A. D. Wilson. Augmenting interactive tables with mice & keyboards. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 149–152. ACM, 2009.

F. Hemmert, S. Hamann, M. Löwe, J. Zeipelt, and G. Joost. Weight-shifting mobiles: automatic balancing in mobile phones. In *CHI EA '10: Extended Abstracts on Human Factors in Computing Systems*, pages 3081–3085, 2010.

O. Hilliges, D. Baur, and A. Butz. Photohelix: browsing, sorting and sharing digital photo collections. In *TABLETOP '07: Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 87–94. IEEE, 2007.

O. Hilliges, D. Kim, and S. Izadi. Creating malleable interactive surfaces using liquid displacement sensing. In *TABLETOP '08: 3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems*, pages 157–160. IEEE, 2008.

O. Hilliges, S. Izadi, A. D. Wilson, S. Hodges, A. Garcia-Mendoza, and A. Butz. Interactions in the air: adding further depth to interactive tabletops. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 139–148. ACM, 2009.

U. Hinrichs, M. Hancock, C. Collins, and S. Carpendale. Examination of text-entry methods for tabletop displays. In *TABLETOP '07: Second Annual IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 105–112. IEEE, 2007.

S. Hodges, S. Izadi, A. Butler, A. Rrustemi, and B. Buxton. ThinSight: versatile multi-touch sensing for thin form-factor displays. In *UIST '07: Proceedings of the 20th annual ACM symposium on User interface software and technology*, pages 259–268. ACM, 2007.

R. Hofer, D. Naeff, and A. Kunz. FLATIR: FTIR multi-touch detection on a discrete distributed sensor array. In *TEI '09: Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, pages 317–322. ACM, 2009.

A. Hoffmann, D. Spelmezan, and J. Borchers. TypeRight: a keyboard with tactile error prevention. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 2265–2268. ACM, 2009.

C. Holz and P. Baudisch. The generalized perceived input point model and how to double touch accuracy by extracting fingerprints. In *CHI '10: Proceedings of the 28th international conference on Human factors in computing systems*, pages 581–590. ACM, 2010.

C. Holz and P. Baudisch. Understanding touch. In *CHI '11: Proceedings of the 2011 annual conference on Human factors in computing systems*, pages 2501–2510. ACM, 2011.

J. Hook, S. Taylor, A. Butler, N. Villar, and S. Izadi. A reconfigurable ferromagnetic input device. In *UIST '09: Proceedings of the 22nd annual ACM symposium on User interface software and technology*, pages 51–54. ACM, 2009.

M. S. Horn, E. T. Solovey, R. J. Crouser, and R. J. K. Jacob. Comparing the use of tangible and graphical programming languages for informal science education. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 975–984. ACM, 2009.

T. Hoshi, M. Takahashi, T. Iwamoto, and H. Shinoda. Noncontact tactile display based on radiation pressure of airborne ultrasound. *IEEE Transactions on Haptics*, 3(3):155–165, 2010.

E. L. Hutchins, J. D. Hollan, and D. A. Norman. Direct manipulation interfaces. *Human-Computer Interaction*, 1(4):311–338, 1985.

H. Ishii. Tangible bits: beyond pixels. In *TEI '08: Proceedings of the 2nd international conference on Tangible and embedded interaction*, pages xv–xxv. ACM, 2008.

H. Ishii and B. Ullmer. Tangible bits: towards seamless interfaces between people, bits and atoms. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 234–241. ACM, 1997.

H. Iwata, H. Yano, F. Nakaizumi, and R. Kawamura. Project FEELEX: adding haptic surface to graphics. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 469–476. ACM, 2001.

S. Izadi, S. Hodges, S. Taylor, D. Rosenfeld, N. Villar, A. Butler, and J. Westhues. Going beyond the display: a surface technology with an electronically switchable diffuser. In *UIST '08: Proceedings of the 21st annual ACM symposium on User interface software and technology*, pages 269–278. ACM, 2008.

D. Jackson, T. Bartindale, and P. Oliver. FiberBoard - compact multi-touch display using channeled light. In *ITS '09: Proceedings of ACM International Conference on Interactive Tabletops and Surfaces*, pages 25–28. ACM, 2009.

Y. Jansen, T. Karrer, and J. Borchers. MudPad: tactile feedback and haptic texture overlay for touch surfaces. In *ITS '10: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 11–14. ACM, 2010.

S. Jorda, G. Geiger, M. Alonso, and M. Kaltenbrunner. The reacTable: exploring the synergy between live music performance and tabletop tangible interfaces. In *TEI '07: Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 139–146. ACM, 2007.

L. Jun, D. Pinelle, C. Gutwin, and S. Subramanian. Improving digital handoff in shared tabletop workspaces. In *TABLETOP '08: 3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems*, pages 9–16. IEEE, 2008.

H. Kajimoto, M. Inami, N. Kawakami, and S. Tachi. SmartTouch - augmentation of skin sensation with electrocutaneous display. In *HAPTICS '03: Proceedings of the 11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 40–46. IEEE, 2003.

S. K. Kane, M. R. Morris, A. Z. Perkins, D. Wigdor, R. E. Ladner, and J. O. Wobbrock. Access overlays: improving non-visual access to large touch screens for blind users. In *UIST '11: Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 273–282. ACM, 2011.

M. E. Karagozler, S. C. Goldstein, and J. R. Reid. Stress-driven MEMS assembly + electrostatic forces = 1mm diameter robot. In *IROS '09:*

Proceedings of the IEEE International Conference on Intelligent Robots and Systems. IEEE, 2009.

K. Kin, M. Agrawala, and T. DeRose. Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. In *GI '09: Proceedings of Graphics Interface 2009*, pages 119–124. Canadian Information Processing Society, 2009.

M. Kojima, M. Sugimoto, A. Nakamura, M. Tomita, M. Inami, and H. Nii. Augmented coliseum: an augmented game environment with small vehicles. In *TABLETOP '06: First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pages 3–8. IEEE, 2006.

K. Koukouletsos, B. Khazaei, A. Dearden, and M. Ozcan. Teaching usability principles with patterns and guidelines. In *Creativity and HCI: From Experience to Design in Education*, pages 159–174. Springer Verlag, 2009.

A. Kruavit and S. Numhom. Magnet implantation into a dice game dealer's digital tips with late thumb tip infection: an iatrogenic criminal operation. *The Thai Journal of Surgery*, 29:97–100, 2008.

J. Lee, R. Post, and H. Ishii. ZeroN: mid-air tangible interaction enabled by computer controlled magnetic levitation. In *UIST '11: Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 327–336. ACM, 2011.

S. Lee and S. Zhai. The performance of touch screen soft buttons. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 309–318. ACM, 2009.

D. Leithinger, D. Lakatos, A. DeVincenzi, M. Blackshaw, and H. Ishii. Direct and gestural interaction with relief: a 2.5D shape display. In *UIST '11: Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 541–548. ACM, 2011.

J. Leitner and M. Haller. Geckos: combining magnets and pressure images to enable new tangible-object design and interaction. In *CHI '11: Proceedings of the 2011 annual conference on Human factors in computing systems*, pages 2985–2994. ACM, 2011.

J. Leitner, J. Powell, P. Brandl, T. Seifried, M. Haller, B. Dorray, and P. To. FLUX: a tilting multi-touch and pen based surface. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 3211–3216. ACM, 2009.

J. G. Linvill and J. C. Bliss. A direct translation reading aid for the blind. *Proceedings of the IEEE*, 54(1):40–51, 1966.

I. S. MacKenzie and R. W. Soukoreff. Phrase sets for evaluating text entry techniques. In *CHI EA '03: Extended Abstracts on Human factors in computing systems*, pages 754–755, 2003.

J. Mankoff, S. E. Hudson, and G. D. Abowd. Interaction techniques for ambiguity resolution in recognition-based interfaces. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology*, pages 11–20. ACM, 2000.

N. Marquardt, M. A. Nacenta, J. E. Young, S. Carpendale, S. Greenberg, and E. Sharlin. The haptic tabletop puck: tactile feedback for interactive tabletops. In *ITS '09: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 85–92. ACM, 2009.

N. Marquardt, R. Jota, S. Greenberg, and J. A. Jorge. The continuous interaction space: interaction techniques unifying touch and gesture on and above a digital surface. In *INTERACT '11: Proceedings of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part III*, pages 461–476. Springer-Verlag, 2011.

T. H. Massie and J. K. Salisbury. The PHANToM haptic interface: a device for probing virtual objects. In *Proceedings of the 1994 ASME International Mechanical Engineering Congress and Exhibition*, volume DSC 55-1, pages 295–302, 1994.

N. Matsushita and J. Rekimoto. HoloWall: designing a finger, hand, body, and object sensitive wall. In *UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology*, pages 209–210. ACM, 1997.

T. Miller and R. Zeleznik. An insidious haptic invasion: adding force feedback to the x desktop. In *UIST '98: Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 59–64. ACM, 1998.

T. Miller and R. Zeleznik. The design of 3D haptic widgets. In *I3D '99: Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 97–102. ACM, 1999.

C. Müller-Tomfelde and M. Fjeld. Introduction: a short history of tabletop research, technologies, and products. In *Tabletops - Horizontal Interactive Displays*, pages 1–24. Springer London, 2010.

M. A. Nacenta, D. Aliakseyeu, S. Subramanian, and C. Gutwin. A comparison of techniques for multi-display reaching. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 371–380. ACM, 2005.

J. Nielsen. *Usability Engineering.* Morgan Kaufmann Publishers Inc., 1993.

H. Noma, S. Yoshida, Y. Yanagida, and N. Tetsutani. The proactive desk: a new haptic display system for a digital desk using a 2-DOF linear induction motor. *Presence: Teleoperators and Virtual Environments*, 13 (2):146–163, 2004.

T. Paek, K. Chang, I. Almog, E. Badger, and T. Sengupta. A practical examination of multimodal feedback and guidance signals for mobile touchscreen keyboards. In *MobileHCI '10: Proceedings of the 12th international conference on Human computer interaction with mobile devices and services*, pages 365–368. ACM, 2010.

G. Pangaro, D. Maynes-Aminzade, and H. Ishii. The actuated workbench: computer-controlled actuation in tabletop tangible interfaces. In *UIST*

'02: Proceedings of the 15th annual ACM symposium on User interface software and technology, pages 181–190. ACM, 2002.

J. Patten and H. Ishii. Mechanical constraints as computational constraints in tabletop tangible interfaces. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 809–818. ACM, 2007.

E. W. Pedersen and K. Hornbæk. Tangible bots: interaction with active tangibles in tabletop interfaces. In *CHI '11: Proceedings of the 2011 annual conference on Human factors in computing systems*, pages 2975–2984. ACM, 2011.

M. J. Pitts, G. E. Burnett, M. A. Williams, and T. Wellings. Does haptic feedback change the way we view touchscreens in cars? In *ICMI-MLMI '10: International Conference on Multimodal Interfaces and the Workshop on Machine Learning for Multimodal Interaction*, page 38. ACM, 2010.

I. Poupyrev, S. Maruyama, and J. Rekimoto. Ambient touch: designing tactile interfaces for handheld devices. In *UIST '02: Proceedings of the 15th annual ACM symposium on User interface software and technology*, pages 51–60. ACM, 2002.

I. Poupyrev, T. Nashida, S. Maruyama, J. Rekimoto, and Y. Yamaji. Lumen: interactive visual and shape display for calm computing. In *SIGGRAPH '04 Emerging Technologies: Conference Abstracts of the 31st annual conference on Computer graphics and interactive techniques*, page 17. ACM, 2004.

D. Prescher, O. Nadig, and G. Weber. Reading braille and tactile ink-print on a planar tactile display. In *ICCHP '10: Proceedings of the 12th international conference on computers helping people with special needs*, pages 482–489. Springer-Verlag, 2010.

B. Reeves and C. Nass. *The media equation: how people treat computers, television, and new media like real people and places.* Cambridge University Press, 1996.

J. Rekimoto. SenseableRays: opto-haptic substitution for touch-enhanced interactive spaces. In *CHI EA '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, pages 2519–2528, 2009.

J. Rekimoto, B. Ullmer, and H. Oba. DataTiles: a modular platform for mixed physical and graphical interactions. In *CHI '01: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 269–276. ACM, 2001.

C. Remy. A pattern language for interactive tabletops in collaborative workspaces. Master's thesis, RWTH Aachen University, 2010.

C. Remy, M. Weiss, M. Ziefle, and J. Borchers. A pattern language for interactive tabletops in collaborative workspaces. In *EuroPLoP '10: Proceedings of the 15th European Conference on Pattern Languages of Programs*, pages 9:1–9:48. ACM, 2010.

D. S. Reznik and J. F. Canny. C'mon part, do the local motion! In *ICRA '01: Proceedings of the IEEE International Conference on Robotics and Automation*, pages 2235–2242. IEEE, 2001.

I. Rosenberg and K. Perlin. The UnMousePad: an interpolating multi-touch force-sensing input pad. In *SIGGRAPH '09: Proceedings of the 36th annual conference on Computer graphics and interactive techniques*, pages 1–9. ACM, 2009.

D. Rosenfeld, M. Zawadzki, J. Sudol, and K. Perlin. Physical objects as bidirectional user interface elements. *IEEE Computer Graphics and Applications*, 24(1):44–49, 2004.

K. Salisbury, F. Conti, and F. Barbagli. Haptic rendering: introductory concepts. *Computer Graphics and Applications, IEEE*, 24(2):24–32, 2004.

P. Scherz. *Practical Electronics for Inventors*. McGraw-Hill, Inc., 2nd edition, 2007.

J. Schöning, J. Hook, T. Bartindale, D. Schmidt, P. Olivier, F. Echtler, N. Motamedi, P. Brandl, and U. v. Zadow. Building interactive multi-touch surfaces. In *Tabletops - Horizontal Interactive Displays*, Human-Computer Interaction Series, pages 27–49. Springer London, 2010.

m. c. Schraefel, G. Smith, and P. Baudisch. Curve dial: eyes-free parameter entry for GUIs. In *CHI EA '05: Extended abstracts on Human factors in computing systems*, pages 1146–1147, 2005.

F. Schwarz. Madgets: actuated translucent controls for dynamic tangible applications on interactive tabletops. Master's thesis, RWTH Aachen University, 2010.

J. Schwarz, S. Hudson, J. Mankoff, and A. D. Wilson. A framework for robust and flexible handling of inputs with uncertainty. In *UIST '10: Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 47–56. ACM, 2010.

S. D. Scott, K. D. Grant, and R. L. Mandryk. System guidelines for co-located, collaborative work on a tabletop display. In *ECSCW '03: Proceedings of the eighth conference on European Conference on Computer Supported Cooperative Work*, pages 159–178. Kluwer Academic Publishers, 2003.

R. Seidel. An analysis of the conflict between the user control and the need for physical-visual consistency in tangible tabletop interaction. Master's thesis, RWTH Aachen University, 2011.

O. Shaer and E. Hornecker. Tangible user interfaces: past, present, and future directions. *Foundations and Trends in Human-Computer Interaction*, 3(1-2):1–137, 2010.

A. Shahrokni, J. Jenaro, T. Gustafsson, A. Vinnberg, J. Sandsjö, and M. Fjeld. One-dimensional force feedback slider: going from an analogue to a digital platform. In *NordiCHI '06: Proceedings of the 4th Nordic conference on Human-computer interaction*, pages 453–456. ACM, 2006.

K. A. Siek, Y. Rogers, and K. H. Connelly. Fat finger worries: how older and younger users physically interact with PDAs. In *INTERACT '05: Proceedings of the 2005 IFIP TC13 international conference on Human-Computer Interaction*, pages 267–280. Springer-Verlag, 2005.

S. S. Snibbe, K. E. MacLean, R. Shaw, J. Roderick, W. L. Verplank, and M. Scheeff. Haptic techniques for media control. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology*, pages 199–208. ACM, 2001.

J. Steimle. *Pen-and-paper user interfaces - integrating printed and digital documents.* Human-Computer Interaction Series. Springer, 2012.

N. Sultanum, S. Somanath, E. Sharlin, and M. C. Sousa. "Point it, split it, peel it, view it": techniques for interactive reservoir visualization on tabletops. In *ITS '11: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 192–201. ACM, 2011.

Y. Suzuki and M. Kobayashi. Air jet driven force feedback in virtual reality. *IEEE Computer Graphics and Applications*, 25:44–47, 2005.

C. Swindells, K. E. MacLean, K. S. Booth, and M. J. Meitner. Exploring affective design for physical controls. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 933–942. ACM, 2007.

A. Tabard, J. Hincapié-Ramos, M. Esbensen, and J. E. Bardram. The eLabBench: an interactive tabletop system for the biology laboratory. In *ITS '11: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 202–211. ACM, 2011.

E. Tamaki, T. Miyaki, and J. Rekimoto. PossessedHand: techniques for controlling human hands using electrical muscles stimuli. In *CHI '11: Proceedings of the 2011 annual conference on Human factors in computing systems*, pages 543–552. ACM, 2011.

H. Tang and D. Beebe. A microfabricated electrostatic haptic display for persons with visual impairments. *IEEE Transactions on Rehabilitation Engineering*, 6(3):241–248, 1998.

J. Underkoffler and H. Ishii. Illuminating light: an optical design tool with a luminous-tangible interface. In *CHI '98: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 542–549. ACM Press/Addison-Wesley Publishing Co., 1998.

J. Underkoffler and H. Ishii. Urp: a luminous-tangible workbench for urban planning and design. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 386–393. ACM, 1999.

B. Unger, R. Hollis, and R. Klatzky. The geometric model for perceived roughness applies to virtual textures. In *HAPTICS '08: Proceedings of the 2008 Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 3–10. IEEE, 2008.

N. Villar and H. Gellersen. A malleable control structure for softwired user interfaces. In *TEI '07: Proceedings of the 1st international conference on Tangible and embedded interaction*, pages 49–56. ACM, 2007.

J. Wagner. SLAP, silicone illuminated active peripherals. Master's thesis, RWTH Aachen University, 2009.

A. Wakita, A. Nakano, and N. Kobayashi. Programmable blobs: a rheologic interface for organic shape design. In *TEI '11: Proceedings of the fifth international conference on Tangible, embedded, and embodied interaction*, pages 273–276. ACM, 2011.

M. Waldner, J. Hauber, J. Zauner, M. Haller, and M. Billinghurst. Tangible tiles: design and evaluation of a tangible user interface in a collaborative tabletop setup. In *OZCHI '06: Proceedings of the 18th Australia conference on Computer-Human Interaction*, pages 151–158. ACM, 2006.

L. Walker and H. Z. Tan. A perceptual study on haptic rendering of surface topography when both surface height and stiffness vary. In *HAPTICS '04: Proceedings of the 12th international conference on Haptic interfaces for virtual environment and teleoperator systems*, pages 138–145. IEEE, 2004.

J. R. Wallace and S. D. Scott. Contextual design considerations for co-located, collaborative tables. In *TABLETOP '08: 3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems*, pages 57–64. IEEE, 2008.

T. Watanabe and S. Fukui. A method for controlling tactile sensation of surface roughness using ultrasonic vibration. In *Proceedings of IEEE International Conference on Robotics and Automation, 1995*, volume 1, pages 1134 –1139 vol.1. IEEE, 1995.

M. Weiser. The computer for the 21st century. *Scientific American*, 265 (3):94–104, 1991.

M. Weiss, R. Jennings, J. Wagner, J. D. Hollan, R. Khoshabeh, and J. Borchers. SLAP: silicone illuminated active peripherals. In *TABLETOP EA '08: Extended abstracts of 3rd IEEE International Workshop on Horizontal Interactive Human Computer Systems*, pages 37–38, 2008.

M. Weiss, F. Schwarz, and J. Borchers. Actuated translucent controls for dynamic tangible applications on interactive tabletops. In *ITS EA '09: Extended Abstracts of ACM International Conference on Interactive Tabletops and Surfaces*, 2009a.

M. Weiss, J. Wagner, Y. Jansen, R. Jennings, R. Khoshabeh, J. D. Hollan, and J. Borchers. SLAP widgets: bridging the gap between virtual and physical controls on tabletops. In *CHI '09: Proceedings of the 27th* *international conference on Human factors in computing systems*, pages 481–490. ACM, 2009b.

M. Weiss, J. Wagner, R. Jennings, Y. Jansen, R. Khoshabeh, J. D. Hollan, and J. Borchers. SLAP widgets: bridging the gap between virtual and physical controls on tabletops. In *CHI EA '09: Extended Abstracts on Human Factors in Computing Systems*, pages 3229–3234, 2009c.

M. Weiss, J. Wagner, R. Jennings, Y. Jansen, R. Khoshabeh, J. D. Hollan, and J. Borchers. SLAPbook: tangible widgets on multi-touch tables in groupware environments. In *TEI '09: Proceedings of the 3rd International Conference on Tangible and Embedded Interaction*, pages 297–300. ACM, 2009d.

M. Weiss, J. D. Hollan, and J. Borchers. Augmenting interactive tabletops with translucent tangible controls. In *Tabletops - Horizontal Interactive Displays*, Human-Computer Interaction Series, pages 157–180. Springer Verlag, 2010a.

M. Weiss, F. Schwarz, S. Jakubowski, and J. Borchers. Madgets: actuating widgets on interactive tabletops. In *UIST '10: Proceedings of the 23nd annual ACM symposium on User interface software and technology*, pages 293–302. ACM, 2010b.

M. Weiss, S. Voelker, C. Sutter, and J. Borchers. BendDesk: dragging across the curve. In *ITS '10: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 1–10. ACM, 2010c.

M. Weiss, C. Remy, and J. Borchers. Rendering physical effects in tabletop controls. In *CHI '11: Proceedings of the twenty-ninth annual SIGCHI conference on Human factors in computing systems*, pages 3009–3012. ACM, 2011a.

M. Weiss, C. Wacharamanotham, S. Voelker, and J. Borchers. FingerFlux: near-surface haptic feedback on tabletops. In *UIST '11: Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 615–620. ACM, 2011b.

M. Weiss, G. Herkenrath, L. Braun, and J. Borchers. Text entry on interactive tabletops using transparent physical keyboards. In *CHI '12 Workshop on Designing and Evaluating Text Entry Methods*, 2012.

G. Welch and G. Bishop. An introduction to the kalman filter. Technical report, University of North Carolina at Chapel Hill, 1995.

P. Wellner. Interacting with paper on the DigitalDesk. *Communications of the ACM*, 36(7):87–96, July 1993.

W. White. Method for optical comparison of skin friction-ridge patterns, 1965. U.S. Patent 3,200,701.

A. D. Wilson. Using a depth camera as a touch sensor. In *ITS '10: Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, pages 69–72. ACM, 2010.

R. Wimmer. FlyEye: grasp-sensitive surfaces using optical fiber. In *TEI '10: Proceedings of the fourth international conference on Tangible, embedded, and embodied interaction*, pages 245–248. ACM, 2010.

R. Wimmer, F. Hennecke, F. Schulz, S. Boring, A. Butz, and H. Hußmann. Curve: revisiting the digital desk. In *NordiCHI '10: Proceedings of the 6th Nordic Conference on Human-Computer Interaction Extending Boundaries*, page 561, 2010.

J. O. Wobbrock. Measures of text entry performance. In *Text entry systems: Mobility, accessibility, universality*, pages 47–74. Morgan Kaufmann Publishers Inc., 2007.

J. O. Wobbrock, M. R. Morris, and A. D. Wilson. User-defined gestures for surface computing. In *CHI '09: Proceedings of the 27th international conference on Human factors in computing systems*, pages 1083–1092. ACM, 2009.

C. Wusheng and W. Tianmiao. Design of data glove and arm type haptic interface. In *HAPTICS '03: Proceedings of the 11th Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems*, pages 422–427. IEEE, 2003.

G. Yang, K. Kyung, M. A. Srinivasan, and D. Kwon. Quantitative tactile display device with pin-array type tactile feedback and thermal feedback. In *ICRA '06: Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3917–3922. IEEE, 2006.

S. Yoshida, H. Noma, and K. Hosaka. Proactive desk II: development of a new multi-object haptic display using a linear induction motor. In *VR '06: Proceedings of the IEEE conference on Virtual Reality*, pages 269–272. IEEE, 2006.

N. Yu, S. Tsai, I. Hsiao, D. Tsai, M. Lee, M. Y. Chen, and Y. Hung. Clip-on gadgets: expanding multi-touch interaction area with unpowered tactile controls. In *UIST '11: Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 367–372. ACM, 2011.

Index

Typeset September 6, 2012