FAKULTÄT FÜR !NFORMATIK

Exploring Multi-Touch Interaction

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medieninformatik

eingereicht von

Georg Christoph Kaindl

Matrikelnummer 0125448

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung:
Betreuer: Ao. Univ.-Prof. Dipl.-Ing. Dr.techn. Peter Purgathofer

Wien, am 12. Oktober 2009 ....

Unterschrift Verfasser Unterschrift Betreuer

FAKULTÄT FÜR !NFORMATIK

Exploring Multi-Touch Interaction

Diploma Thesis

as fullfillment of the requirements for the degree

Master of Science

within the master studies in

Media Informatics

submitted by

Georg Christoph Kaindl

Student-Number 0125448

at the
Department of Computer Science at the Vienna University of Technology

Supervision:
Supervisor: Ao. Univ.-Prof. Dipl.-Ing. Dr.techn. Peter Purgathofer

Vienna, October 12th, 2009 ....

Signature Author Signature Supervisor

Abstract (English)

The thesis aims to establish a departure point for designers to explore the rich opportunities offered by multi-touch interaction. The viability of multi-touch interaction as a technique in HCI is outlined by discussing the role of touch surfaces from the perspective of current trends in HCI such as experiential design and user-centric approaches, as well as by drawing a link to established theoretical frameworks such as activity theory. A summary of the state of the art of multi-touch research is presented with a focus both on aspects of hardware sensor and software technology as well as proposed interaction paradigms and metaphors. In order to enable the exploration of multi-touch interaction on large surfaces and screens, simple camera-based hardware setups suitable for assembly without a technological background are presented with the goal of providing designers with a prototyping platform for their endeavors. Finally, Touché is introduced, a vision-based tracking application to be used in conjunction with the aforementioned self-assembled hardware, providing designers with both hardware and software technologies to explore multi-touch interaction on affordable setups.

Abstract (Deutsch)

Die Diplomarbeit setzt sich zum Ziel, einen Ausgangspunkt für den explorativen Zugang zu den zahlreichen Möglichkeiten von Multi-Touch Interaktion zu schaffen. Die Sinnhaftigkeit dieser Interaktionstechnik wird umrissen, indem sowohl auf ihre Rolle im Bezug zu aktuellen Trends innerhalb der HCI Forschung eingegangen, etwa Experience Design und User-Centric Design, aber auch indem für eine Verbindung zu etablierten theoretischen Werkzeugen wie Activity Theory argumentiert wird. Eine Zusammenfassung des aktuellen Stands der Forschung im Multi-Touch Bereich wird sowohl mit Augenmerk auf Sensor- und Software-Technologie, aber auch auf vorgeschlagene Interaktionsparadigmen und -metaphern gegeben. Um die Auseinandersetzung mit groflächiger Multi-Touch Interaktion zugänglich zu machen, werden kamera-basierte Hardwaresysteme vorgestellt, die für den Eigenbau ohne technische Vorkenntnisse geeignet sind. Ebenfalls wird Touché präsentiert, eine kamera-basierte Tracking-Lösung für den Gebrauch zusammen mit jenen Hardware Systemen, sodass Designer sowohl über die nötige Hardware, als auch Software verfügen können, die für eine Auseinandersetzung mit Multi-Touch Interaktion notwendig sind.

Inhaltsverzeichnis

1 Introduction

The technology community being passionate about a novel input method is a rare occasion. Multi-touch interaction breaking out of the laboratories and pushing into the mainstream marks such an occurrence: Spear-headed by highly acclaimed products such as Apple's iPhone [5] and Microsoft's Surface [19], multi-touch is no longer just an ingredient for extravagant visions of the future, but not yet a bland bullet point on a feature list either. It is something that technologists, designers and consumers alike are excited about, a major selling point even for some.

Companies such as Natural UI Europe [84] have sprouted to develop hard- and software solutions for multi-touch installations. Technology providers such as NextWindow [85] and N-Trig [83] market exciting hardware that enables multi-touch interaction on a wide variety of different surfaces. On the packaging of Microsoft's upcoming Zune HD gadget [20], multi-touch is advertised prominently among other key features such as HD video playback and games. On a more anecdotal note, the large flat panel screens featured in the various booths at technology conferences are nowadays in dire need of "do not touch" signs, something that used to go without saying just a couple of years ago.

At this stage in the lifecycle of a novel input method however, it is advisable not to be blinded by the sheer hype, but to scrutinize the technology and its promises in an objective manner. It is not yet clear whether multi-touch interaction is destined to become a short-lived chapter in the history of human-computer interaction, much like the flamboyant data gloves of the 1980s, or a lasting addition to the arsenal of the interaction designer of the 21st century. Thus, while multi-touch input is undeniably here at the moment, we can not yet be sure if it is indeed here to stay.

As much as it is the technologists' responsibility to create feasible ways of detecting multiple touch-points on a vast array of different surfaces and track their movement over time in a reliable manner, it is up to the designers to harness this data stream in order conceive real-world applications that make it worthwhile to establish multi-touch interaction as a permanent asset in their toolbox. At the current stage of development, the multi-touch community is decidedly stifled in both respects: Neither is affordable off-the-shelf hardware capable of multi-touch sensing available with a good choice of different surface sizes, nor are powerful and flexible software solutions for rapid multi-touch experimentation and prototyping widely adopted. To a certain degree, this manifests itself in a dire shortage of innovative ideas, but rather in a flood of impressive technology demonstrations of little practical use. Within the community, it is a frequent jab at these developments to figuratively call out yet another demonstration of zoomable and rotatable photographs on a bland canvas – a sujet so frequently employed that writing another implementation has become the unofficial entrance exam into the multi-touch community.

This is not to say that there are no sizable efforts happening in the scientific community. Quite on the contrary, publications about the integration of multi-touch with table-top systems, technologies to enable sensing of multi-touch data and other results pertaining to this area are rather abundant. The difficulty, however, lies in building upon these results within a larger community of designers that is not restricted to the self-sufficient realm of universities and research agencies, but does also invite graphic and interaction designers, artists, DIY enthusiasts and similar fractions, who have traditionally been important agents and stakeholders in shaping the future of technology.

Thus, the aim of this thesis is fourfold: First, the motivation for further research derived from the abundant promises heralded by multi-touch technology is detailed. Second, the state-of-the-art of multi-touch research in respect to both sensory technology and integration into viable software solutions will be discussed. Third, an overview will be given about how multi-touch surfaces of arbitrary size can be built trivially, using off-the-shelf components and materials that are readily available. Fourth, a software solution termed "Touché" [55] will be introduced, which serves both as a tracker application to stream multi-touch data to arbitrary clients, but also as a framework to rapidly develop rich multi-touch software, and has been developed as an accompaniment to the thesis.

2 Thesis Structure

The presentation of previous work in the field of multi-touch interaction will be opened by shedding light on the motivation behind employing this paradigm as an input method: A detailed overview of the aims, constraints and limitations of multi-touch interaction will be given, as well as its importance in the realm of experience design.

Different implementations of multi-touch surfaces, either as a stand-alone device or as part of a larger installation, will be introduced and their unique properties be discussed in detail. This includes forms of interaction that are possible only because of special properties in the sensors, such as the ability to not only sense touch, but also hovering, or the ability to distinguish between multiple stakeholders in a multi-user environment.

Next, interaction techniques and metaphors from the literature revolving around multi-touch interaction will be summarized, establishing an understanding of the common ground onto which current research is focused.

The following chapter describes multiple possibilities of building a multi-touch capable surface out of off-the-shelf materials, focusing on optical devices driven by cameras and computer vision software.

The final chapter details a software architecture suitable for a tracker application that is to be used in conjunction with a camera-based multi-touch surface. Several algorithms from the field of computer vision are described in order to build a multi-staged tracking pipeline that is capable of inferring multiple touch positions from a camera image, tracking their motion over time and distributing the extracted touch data to client software. Finally, Touché will be introduced, an open-source tracker application and multi-touch framework that has been developed in conjunction with the thesis and is currently in production use at multiple sites throughout the world.

3 Previous Work

Despite having garnered mainstream attention only a handful of years ago, multi-touch has been studied as a concept for more than 25 years [58], [67], [16], [66]. Reduced to its core, it refers to a surface's ability to track multiple touch points (preferably generated by fingers, but possibly also by pens or other aids) consistently during their lifetime. This means that multi-touch input is closely related to gestural and chordic interaction [120], the former referring to the deduction of user intentions by analyzing the temporal behavior of one or several touch points over their lifetime on the surface, the latter describing the use of multiple distinct fingers on one or two hands as a core feature of an interaction, much like a pianist uses different finger configurations to trigger a seemingly infinite amount of chords.

Contrary to the impression one may get from following current technology news outlets, multi-touch interaction is not limited to screens, where the touch-sensitive surface is layered over the display itself and thusly leading to a direct coupling between input and objects, but common surfaces such as walls, tables or drawing tablets are frequently imbued with such sensing capabilities as well [67]. Thus, multi-touch input exists as both direct and indirect input. Another consequence is that multi-touch plays an important role in the field of ubiquitous computing as well, which is concerned with creating smart objects that can respond to, assist and be employed in an ad-hoc manner by multiple users in their vicinity [115]. Similarly, it is used in tangible computing environments, where an object can either be itself touch-sensitive, or a surface on which the objects are arranged is [113].

Multi-touch interaction can exist on a vast array of different surfaces, varying in size, shape and texture: For instance, it is nowadays a common feature of PDAs and smartphones such as Apple's iPhone [5], for which small screens and areas of interaction are characteristic, often employed on table-top systems such as Microsoft's Surface [19], but also on even larger surfaces like interactive walls, that can accomodate for many concurrent users and provide a sizable space for display and interaction, and where physically reaching remote areas of the screen becomes a serious concern [60]. Surfaces can be rectilinear, round [53] or arbitrarily shaped. In fact, some sensors can even accomodate for surfaces that are not planar, but can exhibit an arbitrary topography with hills and valleys, be rounded or even spherical [106], [94].

While the potential of multi-touch is tremendous, different surfaces and applications have requirements that might even stand in conflict with the goals that designers want to achieve by embedding multi-touch interaction into the product. Thus, a firm understanding of the motivation behind multi-touch input, its various forms and implementations as well as the manyfold interaction paradigms and metaphors that it can provide is essential in order to create successful products that benefit from multi-touch interaction.

3.1 Motivation

Multi-touch is frequently heralded as a step towards more natural forms of interaction than can be afforded by using a mouse and a keyboard, which gains particular importance due to the fact that electronic appliances are pushing strongly into every aspect of our daily lives, dissolving the mental model of computers being there to simply "get the work done". [97].

The main motivation of this chapter is to provide a detailed look at the promises made by multi-touch interaction, as well as how these promises fit into the goals of today's trends in HCI. Additionally, it is deemed important to point out potential theoretical frameworks that can be used for the evaluation of multi-touch surface interaction. Only in the light of such theoretical considerations it becomes constructive to provide hard- and software-technologies for designers to build their own multi-touch hardware in order to experiment with and evaluate their ideas.

3.1.1 Naturalness

Proponents of ubiquitous computing paradigms evangelize a world in which electronics are invisibly embedded into our surroundings and are accessed through intelligent interfaces [115]. A core facet of such intelligent agents are natural interfaces, which are characterized by blending into the environment and provide modes of input that are seamless with respect to the barrier between the world of humans and the world of computers. Examples are smart surfaces that can sense when they are being touched and by whom, such as coffee tables or office walls that can seamlessly turn into a newsreader or a collaborative whiteboard.

An early discussion of multi-touch input as a natural way of interfacing with a computer has been led by the creators of the VIDEOPLACE installation in 1985 [61]: They describe two different types of human intelligence, one dealing with the complex symbolic, rational deduction process ruled by logic in which computer scientists traditionally pride themselves, and the other refers to the ability to understand and manipulate the world, as well as create mental models of locality, which is mostly defined by our basic human heritage. While especially scientists excel (and aim to excel) in the former, the general population is firmer in the latter, often not even aiming to achieve excellence in the logical thought process. Thus, a consequence is to move away from interaction models inspired by analytical problem solving and programming, such as command lines and stringing complex symbolic operation together with mouse movements, towards a paradigm that is more akin to how humans tackle problems in the real world: Through understanding of space and time as well as natural assumptions about how we can manipulate the world. In fact, this is an assumption that designers of multi-touch enabled interactive technology frequently use as a starting point in today's context as well.

While the meaning of naturalness becomes intuitively clear through such descriptions, it is hard to define rigid guidelines about which properties cause an interaction to be perceived as natural. The creators of the *DigitalDesk* [119], an early result in the field of augmented reality that has become a classic publication and one of the first in which multi-touch concepts have been discussed in detail, note that multi-touch interaction enables the user to harness their kinesthetic memory – the delicate muscle memory that we develop during our lifetime when manipulating our surroundings. It summarizes the natural skills and mental models of how the environment responds to our actions and how we can plan and infer suitable actions based on this ability to predict the results. For instance, it is evidently clear how a piece of paper on a desk can be rotated with a hand. One of the goals of DigitalDesk is to project such forms of interaction into the digital realm. In the case of rotation, DigitalDesk allows the user to do just that: Placing fingers or the whole hand on the object to rotate it. It is reported that users find this interaction to be intuitively clear, not requiring any sort of metaphorical thinking: The metaphor becomes the interaction itself.

The perceived naturalness of an interaction is directly related to the cognitive load that its execution places on the user [40]. For a natural interaction, the implementation and the metaphor it is based on steps into the background and makes room for the task itself. For instance, rather than wrestling with the meaning and placement of control points and handles in order to rotate and translate an object on the screen, the use of a two-finger gesture is not only reported to feel more natural, but also to be easier to remember, up to the point where it happens intuitively in response to the desire to perform such an operation.

Pfeiffer et al. formulate this phenomenon as “user interface friction” [89], which is an umbrella term for the degree of how fluid, reactive and unencumbered an interaction is perceived by users. For instance, stringing together a complex affine transform consisting of a translation, a rotation and a scale factor in a traditional mouse-based image editor is a sequential task that requires multiple mode switches. Thus, it poses a significant amount of friction as the user is required to translate her intention into the computer’s domain of distinct transformations that have to be performed in sequence, using different user interfaces (e.g. translation via direct mouse mapping, rotation and scaling via handles) in each mode. Conversely, this transform can be performed on a suitably designed multi-touch system with a single gesture, mapping the single task the user has in mind into a single interaction with the system. With the drastic reduction of complexity comes a considerably lower amount of perceived interface friction.

The immediacy between the mapping of an interaction between metaphor, implementation and execution describes only one side of the meaning of naturalness. Another aspects pertains less to the interaction itself, but to its situational and social context. To remain with the example of rotating an object, consider the communicative meaning of rotating an object on a shared workspace such as a multi-touch enabled table-top system [42]: Orienting an object towards oneself signals a sense of ownership and privacy. Rotating it towards another participant, even only as a subtle hint, can signal an invitation for comments or at least to have a short peek. Similarly, objects are often given the same orientation and put in a pile in order to signal their belonging together within a given context. Thus, an interaction paradigm should afford for such subtle meanings in order to be perceived as a natural extension to the situational context. Rotating a digital document with a mouse and rotation handles, taking into account reaching for the mouse, the mode switch, moving the pointer and finally quitting the mode again, is too cumbersome to simply communicate that a partner should have a look. In fact, the slowness and lack of elegance would most likely eclipse the intended meaning and lead to a lack of understanding - what did you just do this for? Conversely, a multi-touch based gesture can map this meaning into the digital realm seamlessly, being very closely modeled after the real-world example.

On a broader scale, humans tend to very rapidly and naturally switch between different modes of working during a single task, both alone and in collaborative contexts [95]. As an example, a group of users might work together at a table, while individual workers rapidly alternate between phases of sharing information with others and working alone, whereas another user might play a mediating role between them. Similarly, a single user might be working on multiple documents at once, rapidly rearranging them on her desk into suitable configurations for different sub-tasks. Consequently, a perceivably natural interaction has to accomodate for such transitions in an elegant and seamless manner as well. Multi-touch shows potential in this aspect of naturalness: On one hand, it forgoes the mouse and any other form of tethered input device, obviating the need to reach for anything other than the actual interaction surface when the user returns to her work with the computer [105], providing for a fluid and unencumbered transition between periods of working with the computer and working on something else. Similarly, the intuitive metaphors inspired by work with real-world documents and objects that multi-touch input can accomodate for enables multiple users in a collaborative environment to draw upon their knowledge of social norms of shared workspaces without having to translate these norms into a different realm that corresponds to the computer's implementation of loosely coupled interaction metaphors: Since users are, for instance, well-versed at working together on documents on a table, multi-touch interaction allows them to transfer these skills to a digital table-top system directly, rather than prompting them to reinterpret these skills through the lens of a mouse- or keyboard-based interface.

Nevertheless, the virtue of modeling interaction after examples from the analog world is inherently different from creating simulations of real processes. Weiser [116] takes this idea to the point by asking how intelligent agents should behave in an ubiquitous computing environment. Should they act like humans, or should they appear human-like, but more obedient? In the same vein, should a touch-enabled table-top application simulate an office desktop, or should it only be partly inspired by it? At its core, the question that a designer has to ask herself is whether the interaction from the real world that her interface is inspired by is in itself already so free of trouble and potential misunderstandings that it is worth to model it precisely in a digital realm, despite her being mostly freed from constraints imposed by the real world. Thus, naturalness is not to be seen as a recreation, but as an augmentation of the interaction paradigms a design is based upon. As an example, consider a table-top system in which users can slide documents towards each other across the surface. While a very natural interaction, a real-world implementation would suffer from difficulties such as friction causing a document to come to rest in the middle of the table, out of the recipient's reach, or a document might slip off the table altogether [2]. Therefore, the interaction itself requires a considerable amount of attention to perform: A user must carefully slide the document with just the right amount of force and in the correct direction, a task demanding both planning and practice in its execution. Conversely, the digital implementation can sacrifice realism in order to make the task possible without requiring any of these skills, while still being perceived as easy to understand and natural.

3.1.2 Experiential Qualities

Experience design is a comparatively new trend within HCI research. Rather than engaging the challenge of designing compelling user interfaces from the perspective of the tool itself (e.g. by aiming to make to the tool more efficient, easier to learn and wield), the focus rests on the experiential qualities that a tool can invoke in a user during the usage scenario. Consequently, it is in an inherently anthropocentric approach. These qualities are generally stoked by the many impressions and feelings that a system recreates over time and in interplay with the user, how it responds and behaves in a dialogue between human and machine [71].

McCarthy et al. [77] note that experience is not something that is restricted to the time of actual use: Rather than ceasing together with the experienced quality, it evolves over time and space as the user reflects upon her sensual and emotional memory of the interaction in terms of herself and her cultural and social surroundings. Naturally it is difficult to formalize rigid guidelines to evoke ephemeral qualities such as certain emotions in users, due to the fact that experience is something that is actively created through repeated reflection. However, McCarthy et al. distill four threads along which experience is formed, in order to showcase potential for a designer to actively plan the way in which users will live through and with the interaction at hand. These qualitative threads are *compositional*, referring to the way in which the multiple parts forming the experience fit together into a coherent whole, such as how a narrative structure is created, how potential actions are made visible, how consequences and their causes become apparent, and how plausible the experience stays in itself, *sensual*, describing the overall atmosphere created by the artifact as well as the sum of cognitive impressions such as colors and textures, *emotions* and their role in coloring the experience, and *spatio-temporal*, relating to the role of space and time, e.g. the pacing of an interaction and the binding to specific places and environments that have special meanings to the user.

Following up on these four threads, six dimensions of the sense-making proces are introduced, which can be related to each of the threads in order to describe the complex interplay of the mind, the senses and the outside world in the forming of rich experiences. These dimensions are *anticipating*, the fact that an experience is never unprejudiced, and expectations when approaching a large touch-screen are considerably different than when approaching a keyboard/mouse combination, *connecting*, referring to the immediate judgement about a situation that is made without much thought, but remains crucial even in latter stages when active reflection re-colors the experience, *interpreting*, the process of forming a mental model of the situation and figuring out what we can do, should do and how we feel about it, *reflecting*, describing the thorough evaluation that most often succeeds an interaction, *appropriating*, the complex thought process in which we inter-relate an experience with our previous experiences and to our sense of self, and *recounting*, referring to the way we tell others and ourselves about an experience and how it is colored in the context of a narrative, rather than an a private thought process, including our exaggerations and omissions.

Applying this model to the concept of multi-touch, it becomes evident that such interfaces hold a vast potential to cater to the different threads and dimensions of experience formation. As an example, on a direct interaction multi-touch screen, narratives can be formed solely through the qualities of the interaction itself [2].: When applying a simple drag-style interaction, different objects could behave in subtly different manners in order to create a narrative about their qualities. A heavy stone might be "hard" to drag, lagging behind the finger and requiring considerable amount of work to budge it across the screen. Similarly, a feather might respond to a virtual gust of wind when a finger is being dragged nearby. Following up on a different thread, some multi-touch surfaces allow for the employment of arbitrary textures [124], [94], [24], enabling designers to use wood panels, cloth and many other materials as part of the experience. Thus, due to the removal of inherent constraints imposed by traditional input devices that require additional hardware to couple a user's hand to the interaction, the freedom of the designer to shape the sensual and emotional response of an interaction is vastly increased.

While the the concept of "naturalness" detailed previously is an adequate example, the number of different experiential properties that can be imagined is astronomical. On a related note, such qualities can be rather complex and broad in scope and are not to be trivialized into unspecific notions such as an interaction feeling "nice", since such impressions make it difficult to design new interactions around them. Jonas Löwgren [70], [69] introduces two complex experiential qualities that are salient examples of the scope that such qualities are concerned with: *Pliability* and *fluency.*

Pliability refers to the degree in which an interaction feels involving and tightly coupled to the user's manipulation, encouraging and facilitating the exploration of the artifact at hand without the fear of being overwhelmed by its requirements or power, up to a point of general serendipity in use. In detail, it refers to the way in which a user can indulge herself into the interaction while experiencing its complexity as the gradual unfolding of different options in a manageable manner, following a dramaturgy of exploration that can be carefully crafted by the designer of the artifact. At the same time, the insight that a user gains gradually during her entanglement with the artifact increases steadily without any show-stopper issues that, with their sudden appearance as a problem preventing the use of the artifact for the user's current desire, would dilute the experience of exploring the system in an almost dialogical manner. The quality might be summarized shortly as the quality of a dramaturgy from a given conflict to its resolution with and through the use of the artifact.

Dynamic queries to explore a large dataset are Löwgren's example for a pliable experience: By gradually refining the dataset and being presented with the filtered results immediately encourages an intimate way of exploring the seemingly unwieldy wall of data at one's own pace, potentially discovering patterns within it by chance. The finer the granularity of the options, the deeper the feeling of shaping the dataset to one's own bidding becomes, leading to an exceptionally pliable experience. It is notable that while a non-interactive way of filtering the data (e.g. such as provided by SQL) is functionally equivalent in respect to the ultimate goal of extracting a given subset of the data according to arbitrary criteria, and the usability and utility of the system can be on par with the dynamic environment, but the experience stemming from the use of the system would be entirely different. Thus, pliability is a purely experiential quality that becomes evident only from a user-centric perspective, rather than a tool-centric one.

In conjunction with a multi-touch setup, a core metaphor of pliability, the user's ability to virtually shape the dataset with her fingertips, can be realized literally. The loop between manual interaction and system response can be coupled in an extremely tight manner, such as when manipulating digital objects on a multi-touch table-top: Due to the subtle orientation adjustments and movements that touch input allows for, the potential for pliable experiences is vast. It is possible for the designer to create multi-touch interfaces whose elegance and intimacy provides for ways of interactivity that maintain the brittle emotions of sensuousness and grace that are so closely related to the notion of pliability. Examples of applications that benefit from multi-touch input in such a manner are Wu et al.'s *RoomPlanner* [126], a table-top installation that enables multiple users to design a room layout from a top-down perspective, positioning, manipulating and sharing pieces of furniture on the fly, or Matsushita et al.'s application [74] to experiment with the placement of wind-power plants on an interactive map in order to discover suitable locations where wind conditions are optimal by positioning, orienting and grouping wind turbines on a zoomable map of the Japanese shore and being presented with continuous feedback about affected parameters like power output, wind power and power infrastructure on the fly, leading to an explorative working style inspired by the graceful alternation between the finding of ideas and their empirical validation.

Löwgren's term of *fluency* [69], on the other hand, refers to an experiential quality rooted more within the social context of an interactive device's use, rather than a personal one: It denotes the abstract level of gracefulness with which a user can handle concurrent demands for her attention and, ultimately, her actions. As an example, he notes that there are three times as many interruptions that extend beyond ten seconds during an appointment with a physician when a computer rather than only paper material is being used, stemming from the combination of distractions caused by the mode switch between talking to the patient and looking up information on the computer: The physician needs to reposition themselves towards the monitor, grab a mouse, and (re-)find their focus point in the application.

Thus, fluency as a quality is experienced along two axes: The way in which an interaction respects social norms, and the way in which information can be conveyed in an unobtrusive manner, while retaining the ability to shift the interaction from background to foreground in an elegant manner. It is concerned with the reduction of "information overflow" by operating in the periphery of one's perception, but keeping information at one's fingertips should it be needed momentarily, as well as it is with the mediation and regulation of the rhythm of interpersonal communication and practice in a manner that is perceived as unobtrusive and supplemental. It is closely connected to the notion of calm technology [117], a new breed of devices that are able to convey information both in the periphery and the center of one's attention, providing for graceful shifts between the two. Consequently, a user is able to manage multiple calm devices concurrently without struggling to keep up with their demands.

Touch-enabled surfaces provide ample opportunities to satisfy both aspects of fluency. Matsushita et al.'s *HoloWall* [76] is designed to augment walls, billboards, posters, calendars and other surfaces commonly found in our everyday life into interactive media, allowing for a fluent transition between their position in our perceptive periphery and being the center of our attention. As an example, such a billboard can detect people in the vicinity watching it and begin to supply additional information about the currently displayed item. Users can then walk up to it and access additional media items through multi-touch interaction, share them with other bystanders or simply discuss them collectively. However, the billboard remains unobtrusive and does not actively seek for attention. The elevation from static billboard to interactive surface happens calmly through touch, and it returns to being a mere billboard when the user walks away, requiring no explicit switch between these modes. Similarly, Ishii et al.'s *ClearBoard* [50] offers a way for two people to share notes and drawings with a glass-wall metaphor, either locally or remotely. Each user is positioned on one side of the glass wall, being able to draw and manipulate documents on the surface with their fingers, while the system transparently accounts for mirror image reversal. It respects social norms like keeping eye contact during a discussion, and supports external notices calmly without the need to interrupt the communication. Periods of collaborative drawing on the surface and discussion through the surface can alternate gracefully without the need for an explicit or implicit mode switch. Thus, augmented discussions using ClearBoard are fluent at a considerable level in comparison to traditional video conferencing setups.

With such a vast potential for the positive stimulation of experiential qualities given by the employment of multi-touch interactions and this aspect constituting a major selling point for popular devices [5] [20] as well as users generally reporting that touch devices are more enjoyable to use [73], a good understanding of the inner workings of these qualities becomes crucial for the designer. Especially McCarthy's aforementioned guidelines [77] showcase ample opportunities to actively enforce the creation of positive experiences through the design of interactive devices.

3.1.3 Activity Theory

While the perspective of experiential qualities provides an adequate approach to analyzing the value of multi-touch interaction in comparison to other means of interfacing with a computer, a more coherent theory could constitute a substantial boon to the community. *Activity theory* [13], [64], [57] provides a holistic approach of the intricacies of the place computers take in people's lives. It takes a different stance from more traditional strategies inspired by cognitive science, in which the user is seen as just another information processing unit in the interactive workflow, being associated mainly with technical parameters such as cognitive capacity and sensorimotor abilities such as described by Fitt's Law [30], which are undoubtedly important, but only formalize extrinsic parameters, omitting intrinsic processes such as reflective thought processes. Instead, activity theory lends itself as a firm theoretical underbody for a more modern view on HCI, in which the user is regarded as an autonomous agent in the interaction, with her own desires, goals and character traits such as a limited attention span, a faulty memory and fears of unintended consequences.

Furthermore, activity theory provides means to fuse individual motivation and goals with those of groups in a meaningful manner, providing a more holistic view on real life work processes in which group memberships and potentially conflicting group goals are just as important as individual ones. Thus, activity theory acknowledges the existence of a context into which an interaction is embedded, and without which the interaction and its motivation do not make sense: It postulates that at least minimal context information needs to be recorded during the observation of an interaction in order to analyze it in a meaningful way. Consequently, activity theory succeeds in providing a framework that allows the modeling of an interaction in its entire complexity, extending far beyond the modeling of decision making processes by means of biological urges and response to trivial stimuli.

In its basic form, an activity is an instance of doing directed towards an object, with the goal of deriving an outcome from the object [44]. The activity consists of a number of individual actions, which can only be explained and analyzed in the context of the activity they belong to. As an example connected to touch input on a table-top, an action might be the resizing of a digital document, but the meaning of this action can be considerably different depending on the activity it belongs to: A user might be resizing the document to get a better view, to be able to show it to someone, or simply as an explorative experiment while learning the usage of the interface.

The object and the motive can undergo substantial changes during the course of the activity, up to the point where the ultimate goal may only become apparent in the latter stages. Consider the previously mentioned phenomenon of serendipitous discovery during the use of highly pliable interface [70]: At the beginning of such an activity, the user's motive might only be to experiment with the dataset in an undirected way, but through the noticing of patterns by chance, the motive can switch to a more concrete one, such as finding an explanation for the pattern, discovering similar patterns, or the pattern may only serve as an inspiration for a completely different exploratory motive.

Thus, a basic tenet of activity theory is that at the mediating point between the subject and the object lies the "tool". However, the tool, itself being the result of a previous intertwining of various activities during its creation phase, is an externalization of these activities, bundled with their respective goals and motives. As an example, one goal might have been cost-effective production. Another goal might be to create the best, an average, or at least a marketable tool. Additionally, the bias of its developers towards the potential activities that a tool is designed for is reflected as well. Thus, activity theory claims that one way humans shape their own behavior is the creation of tools in addition to internal though processes, which, when reversed, results in the assumption that behavior during an activity is influenced by the tools that act as a mediator [13].

With the broad scope of activity theory as a model, it is notable that an action can belong to multiple activities at the same time, being colored by multiple goals and motives at once. As an example, consider a user who wants to teach how a system works to a bystander by letting them watch a common workflow. The motives are the achievement of the goal of this workflow, but at the same time it is teaching. Each action will be influenced by both: Some might be executed a a slower pace than usually, so that it is easier to follow, some might be accompanied by explanations, but at the same time, the goal of actually achieving the desired result (rather than demonstrating only a though process) remains honed. Consequently, the more flexible a tool is in respect to accommodating for a multitude of (potentially conflicting) goals at once, the fitter it is as a means of activity mediation. In detail, activity theory describes a hierarchy of mediation tasks that a tool has to be designed for in the case of human-computer interaction: First, the border between the user and the computer must be mediated, and second, the border between the user (with her computer) and the outside world [57]. This is closely related to Löwgren's quality of "fluency" [69], which is also concerned with the way in which an interaction fits into the contextual surroundings of an activity, most importantly the social one. As an example, a direct-touch surface interface can be highly successful both as a mediator between the user and the computer, but also between the user and the outside world, because it is easy for bystanders to follow the interaction by simply watching where the fingers go, whereas a similar interaction using a keyboard might succeed at the user-computer level, especially for advanced users, but fails as a mediator to an outsider, because the interplay between commands and response becomes hard to follow.

Naturally, most activities are typically planned in the mind before they are attempted in the real world, either on a conscious or subconscious level, using a model that a human derives from the sum of their previous experiences. Such models are generally not rigid, nor accurate descriptions in the sense of science, but are combined out of past experiences and a pragmatic understanding of the activity: It can be said that certain properties and effects of the activity permanete into the subject and transform them, a process that is termed "internalization" [64]. This is akin to the way a child learns to perform some actions in the mind through repeated experimentation, eventually deducing patterns that allow it to predict the outcome of an action and judge its suitability for the need at hand.

Of course, internalization is not only driven by personal experience, but also by the collective experience of humanity as a whole: Humans generally do have mental models of processes they have not tried themselves, which is not only of utter importance for the building of a cultural heritage, but also for basic survival. As an example, even though we have never verified ourselves, we know that jumping down great heights will lead to dangerous injuries. Nevertheless, many of these models are only passed on within narrowly defined groups and are not part of a broader cultural heritage. A chemist can predict the outcome of mixing two substances, but non-chemists will have to try and observe the results. From the perspective of interaction design, the idea of internalization can serve as a clue about which metaphors are suitable in order to make a system easy to learn and understand. The larger the group in which an internalization of the modeled process is passed on, the more likely it is that the metaphor succeeds as a means of explaining the interaction through a previously learned mental model. In multi-touch interaction, this is closely related to the aforementioned notion of naturalness. As an example, rotating a digital document by rotating a hand placed on it is derived from a mental model that many have acquired in their childhood, whereas rotating the same document with either rotation handles operated with a mouse cursor or by the specification of a rotation angle are both based on (geometric) models that are learned much later in life, if at all.

Especially from the perspective of learning and building skill, an action can further be broken down into even smaller steps, termed "operations" [13] [57] [64]. They refer to basic steps and routines that are performed during an action as a subconscious answer to a condition faced during the course of the action. Thus, in the field of HCI, they denote the group of basic tasks such as pressing a key, moving a mouse or pushing a button. To become more proficient at an interaction, three specific phenomena are formalized by activity theory: First, operations become more fluent as the individual acquires higher levels of skills, and at the same time, new operations are learned that were previously impossible due to a limitation of skill. As an example, the operation of moving the cursor to a specific position on screen with the mouse will gradually improve for novices, until they are proficient enough to form more complex operations based on this concept, such as drawing a selection rectangle. Second, tasks that used to be at the level of an action, thus consciously performed, are learned up to a point where they become operations themselves. This is akin to a virgin user of drawing program meticulously drawing a shape the first time as a very conscious task, but with practice, drawing the same form will have become so deeply routed in kinesthetic memory that the task can be performed automatically, elevating it to the level of an operation. Third, as new operations become available, they can in turn be formed into new, more advanced actions through an ensuing process of sense-making, fueled by a deeper understanding and better model of the activity at hand.

Thus, in order to create an interaction that can be gradually learned and then perfected through the gradual acquisition of skill, it is necessary to support this dynamism on the level of operations and actions. An important aspect is that something from the previous action remains as a feedback in order to trigger one of the next potential operations. Without such feedback, it becomes impossible to trigger subconscious operations, because a step of conscious decision-making has to be inserted in order to find out how to continue. An example of such a feedback-driven interaction in the field of touch input are Winograd et al.'s "Flow Menus" [34]: Akin to hierarchical pie menus, each menu item triggers another pie menu at a known locality, up to an arbitrary nesting level. At first, a user will have to decide on an action in the top-level menu, then decide again in the second-level menu, and so on. However, once the menu structure is known and the sizes and positions of the graphical representation have become internalized, the user can select any menu item at any hierarchy level by drawing a gesture, rather than consciously navigating the menus. This is possible because the system gives instantaneous feedback at the end of each selection operation, immediately accepting input for the next level. Thus, selecting an item in a Flow Menu is transformed from its character as a (rather complex) action into an operation that can be triggered from kinesthetic memory. Because the system supports multiple touches, it is even imaginable that a proficient user can work with two Flow Menus at the same time, consciously working through one of them while subconsciously selecting items in the second by gesturing.

A feature common in many electronic interactions is the availability of different "shortcuts" to access specific operations. As an example, it is expected for mouse-based applications to have keyboard shortcuts that can largely replace the mouse interaction for more advanced users. From the perspective of activity theory, it is vital to understand that such a replacement (or rather, such an alternative) does not constitute an action-operation dynamic, but rather a different, new learning process, because instead of collapsing a former action into an operation, an entirely new operation is introduced [64]. From the perspective of multi-touch input (and its establishment as a widely-used interaction technique), this is important in order to provide a frame-of-reference against which multi-touch interfaces can be compared and discussed in a meaningful way in regards to its potential in terms of learnability, efficiency and its experiential qualities. For instance, the criticism that a keyboard shortcut is inherently "faster" than a touch gesture is, notions of experiential differences kept aside, offset by the fact that the shortcut has to be acquired through a second learning process, whereas the gesture is sped up in a true action-operation dynamic through the acquisition of skill.

Activity theory can serve as a viable framework in the design and evaluation of multi-touch interactions, and it has been shown that many multi-touch concepts can be elegantly married with a theoretical counterpart. Nevertheless, Kaptelinin [57] notes that activity theory is not without its fundamental problems, citing four that are especially detrimental to its applicability in HCI: First, due to its roots in psychology, it is very closely focused on the individual, but in HCI, “the user” can also refer to groups, organizations or even idealized personas. Second, it has a narrower point of view on culture, offering a theoretical underbody only for those properties of culture that are closely related to the way humans interact with their surroundings, but not for culture in a historical, developing sense. Third, HCI can not only provide its users with objects of reality, but also with a sort of reality of its own: Virtual realities prove to be a great difficulty for activity theory, which can not be solved without enriching it with entirely new principles related to realities that can be embedded into other realities.

Fourth, activity theory is a complex, abstract and theoretical framework. Thus, it is inherently difficult to crystallize concepts of activity theory into pragmatic design guidelines. Quek [92] proposes a simple design checklist based on activity theory, which is not meant to replace more common approaches such as usability guidelines and technical evaluation, but rather to show an alternative or provide a supplemental tool. The points of this checklist are *contextual analysis*, scrutinizing the context in which the interaction is going to take place, *examining the object*, defining and describing the object of the activity that is to be implemented (such as a digital image or document), *support for internalization*, e.g. ensuring that the interaction is based on metaphors that allow for easy internalization and subsequent externalization in different scenarios, *support for actions and operation*, meaning that the actions which will be part of the activity should be optimized for the transformation into subconscious operations by more experienced users, *tools and mediation*, the analysis of the mediating role that a tool will play between user and computer and between user and outside world, *development and change*, referring to the inclusion of activities related to maintaining, developing and evolving the software- or hardware-solution over time, and *support for learning*, describing the analysis of activities related to learning to use the product.

Quek notes that this checklist is self-sufficient: Meant mostly as a proof-of-concept that design guidelines can be derived from activity theory despite its leaning on a more theoretical than pragmatic approach, the list is long and defines points that are covered by activity theory in very different granularities: As an example, the evolution of a software product over multiple subsequent versions is hardly covered by concepts of activity theory, which, by its heritage, is more rooted in tools in the real world that cannot be updated on the fly. Nevertheless, the checklist provides a viable design perspective.

While activity theory is not without its shortcomings, it can serve as a theoretical framework for the design and evaluation of multi-touch based interactions. With its human-centrism and its strong ties to the analysis of tasks and objects in the real world, it is suited well for a family of interactions as closely related to notions of naturalness and experiential qualities as multi-touch.

3.1.4 Efficiency

Much like with traditional interaction techniques, it is fruitful to examine the efficiency of the interaction under the limitations imposed by the human cognitive and motor system. For multi-touch surfaces, which can greatly differ in size and shape, reaching from small handheld devices up to wall-covering installations, it is especially necessary to have instruments ready to quantify the efficiency of a potential interaction on the surface, since it is to be excepted that most interactions will not be suitable for all potential surface shapes and sizes.

Probably the most well-known instrument to measure the sensorimotor performance of tasks on an interactive system is Fitts' Law [30], [49]. It is based on the assumption that it is not possible to study the motor performance in a clear separation from the associated sensory system, but that meaningful results can only be obtained by studying the sensorimotor mechanism in its entirety. Thus, it is derived from the measurement of rapid and uniform motions that have been highly overlearned while all other external stimuli are kept constant, so that an experimental situation can be created in which speed is limited mostly by the capacity of the test subjects' motor system. A basic result of the experiments conducted by Fitts is that the amplitude, duration and variability of movements are highly interrelated, resulting in the fact that accuracy decreases when amplitude is increased while speed and movement tolerance are kept constant, and that speed has to be decreased if the movement tolerance is to be kept constant at increasing amplitudes. Fitts' Law has been reformulated by other researchers in order to create a more robust representation of this finding, leading to Shannon's formulation which is currently the most widely used one [72].

$$t_m = a + b\ log_2(\frac{A}{W} + 1)$$

t_m is the time required for the movement, W the width of the target, A the motion amplitude, and a and b are constants determined through linear regression, where the reciprocal of b denotes the *index of performance* in bits per second of information processed. Furthermore, the logarithmic term is referred to as a given task's *index of difficulty.* Both motion and target are one-dimensional in nature, so that the target is a line segment along a pre-defined axis.

Fitts' original formulation suffered from the fact that the index of difficulty might break down for certain tasks and cause the yielding of unrealistically low or even negative results for the movement time. Shannon's reformulation prevents the index of difficulty from turning negative, but an inherent problem remains [72]: Both by formulation and the nature of the experiments from which it is derived, Fitts' Law is one-dimensional by definition. It is not immediately obvious how the law can be extended to two-dimensional tasks, such as pointing tasks on a touch-sensitive surface: This is because the shape of the target and the incident angle of the motion towards the target play a role as well, requiring a new formulation of the W parameter in order to accomodate for the 2D nature of the target. One potential approach is to substitute the length of the smaller side of the target's bounding box for W, but this approach yields unreliable results for targets that are not rectangular in shape and differ greatly from their bounding box. Another approach is to project the vector between starting point and the center of the target onto the target itself, taking the resulting length for W (which is often referred to as W' in this case). While this approach can provide considerably more accurate results that can be compared in a more reliable way, it still assumes that the motion is directed towards the center of the target object, thus disallowing any "shortcuts": If the target shape is highly irregular, such as a star-like form, it can be considerably faster to direct the motion towards the target point closest to the starting point rather than the center, skewing the results provided by the W' approach in an unrealistic manner.

A more valid approach to extending Fitts' Law from 1D to 2D tasks is the employment of measurement norms that are specifically tailored to the description of surfaces. *METRO* [18] is such a norm: It numerically compares two triangle-mesh definitions of a surface at different levels of detail (LOD), providing reproducible results that allow for the extrapolation of distance results for LOD representations that have not yet been computed. Thus, such a distance norm can provide meaningful results for values of W, even if the target shape is highly irregular, because it becomes necessary to model the shape itself (or any simplification of it) as a hit box by discretizing it in much the same way as would be necessary during screen rasterization. However, *METRO* and similar models are numerically complex systems that are more difficult to wield than the other substitutions for W that have been developed hitherto.

Another shortcoming of Fitts' Law is that it is based on data derived from experiments with short range motions, in which mostly the hand, wrist and lower arm were involved. However, the performance index can vary greatly depending on which muscle groups are involved, such as upper arm, shoulder and potentially even torso movement becoming necessary for tasks on very large surfaces. This is another difficulty for multi-touch systems, in which the surface can potentially be very large.

Consequently, a more holistic approach to performance measurement that is less dependent on data from specific experiments and information theory is desirable. ISO9241 (Part 9) [26] proposes a standard for pointing performance evaluation on a vast array of different devices, including touch-sensitive surfaces. While Fitts' Law (in the Shannon formulation) plays a role in defining a set of tests and test conditions, actual performance evaluation is improved by combining the expected results from Fitts' Law with direct measurements of muscle load and fatigue, which are taken with medical equipment during testing sessions. Additionally, test subjects are asked to rate the interaction according to dimensions of comfort and usability. Therefore it is possible to describe and measure additional surface properties such as the pressure required for the touch to register or the friction during movement, which is a great boon to the evaluation of touch surfaces. However, this ISO standard is only partly suitable for designers to evaluate the efficiency of their designs: On one hand, the testing procedures are more concerned with notions of safety and comfort in the sense of preventing injuries from use, rather than to provide results to make different devices comparable to each other. On the other, the testing procedures are short and do not account for learning effects and the build-up of skill in using the interactive device, even though these are important aspects during the design phase.

Measurement based solely on Fitts' Law suffers from the fact that only the pointing task itself is taken into account, but not other operations such as actually pushing a (virtual) button on the touch surface. While the cognitive difference between positioning a finger over a button or hitting a target is negligible, the actual motion difference might not be. As an example, it strikes a difference whether the finger is guided along the surface or hovers slightly above it. An early model for performance evaluation, that models such subtleties and can provide measurements for more complex interactions than mere pointing tasks as well, is the Keystroke-level model [17], which is only one of many very similar models that have been developed subsequently, the most widely known one being *GOMS* [51].

The Keystroke-level model is concerned with the measurement (and prediction) of exactly one distinct aspect of performance: The amount of time it takes an advanced user to solve a given task on a given interactive system. This is done by taken the time spent for low-level operations (such as keystrokes, from which the model infers its name), the user's mental preparation for a subtask imposed by the system and the time the system needs to present a response to the user's actions. The inherent assumption of the model is that for expert-level users, these three groups of processes are sufficient to describe the time it takes to perform a given task. The designers acknowledge that there are many other dimensions along which users can vary, such as the *time* allowed for the completion of a task, e.g. whether there is any time pressure imposed, *errors* that need to be undone or corrected, *learning* effects, which can drastically shorten the time a user needs for task completion or lead to entirely new ways of tackling the same problem, *functionality* of the tool, e.g. how fit the tool is for the task at hand, or if the task can only be solved through reformulation and work-arounds, *recall*, the user's memory ability, *concentration* and *fatigue*, referring to the mental state the user can impose on themselves during the task, and *acceptability*, describing the user's feelings towards both the task and tool on a more personal level. However, it is deemed impossible to design a model that is robust towards variations along all these dimensions: As an example, taking into account all the potential errors a user may make during the task as well as the effort needed to correct them would lead to such an enormous number of different cases to model that the derivation of a suitable "average-case" measurement becomes infeasible.

In the Keystroke-level model, a larger task can often be split into smaller, cognitively manageable chunks termed "unit tasks", which are quasi-independent of each other and are performed in succession in order to solve their parent task. They owe their existence mainly to the limited cognitive abilities of the human mind. However, it is notable that not all large tasks have this structure: As an example, tasks like ordering photos on a digital surface or typing a manuscript into a computer do not require the separation into sub-tasks, because the human mind is capable of achieving continuous throughput in such linear activities. Unit tasks can further be divided into an *acquisition phase*, in which the user transforms the given task into a mental representation of the steps required by the system to solve it, and the *execution phase*, in which the user calls upon the system facilities to realize their mental representation of the task. While the acquisition time can vary greatly according to the situational context (except for special cases such as highly overlearned routine tasks), execution time is deemed to be nearly constant for expert users.

It is a characteristic of an expert user that she can select an appropriate method for solving a given unit task very quickly, out of a larger pool of alternative methods. Thus, it is assumed that the expert user chooses the most optimal chain of methods, but it is also possible to predict the time taken for any possible chain of tasks. If potentially made errors are known in advance, and suitable methods for correcting the error are known as well, then the time necessary to perform the task despite having made the error can be predicted too. More complex tasks will generally have multiple paths and even conditionalities, but if these are known in a priori, they can be accounted for in the performance measurement as well. As an example, it is possible to build a tree of different measurements that model the decision-making branches formed by the user during the task.

Actual measurement or prediction is done by the accumulation of sequences of six operators, for which execution time is described separately and then summed up to define the total time taken for the task. For the less complex operators (such as keystrokes on a keyboard), execution time is deemed to be constant under all conditions, but other operators will have to have their execution time calculated with their own model. The six operators are *M (mental operator)*, describing the pauses taken by the user to think about their next operation or to make a decision about how to proceed, *R (system response operator)*, the time it takes the system to provide feedback that the user needs in order to continue with the task solving process, *K (keystroking operator)*, e.g. the average time it takes to push a button on a keyboard, touch down on a touch surface or click a mouse button, as well as all other tasks that conceptually fit into this category, *P (pointing operator)*, the time it takes a user to point at an object in the system, described by the Shannon formulation of Fitts' Law, *H (homing operator)*, describing the operation of positioning the hand on a device (such as keyboard or mouse) or on a touch-sensitive surface, such as to operate a menu, and *D (drawing operator)*, describing a (potentially complex) drawing operation such as outlining a selection shape. It is notable that system response time only needs to be counted if the user has to wait for the finishing of the response before they can continue (such as when waiting for a menu to open).

An advantage of the Keystroke-level model is that it does not require any specialized psychological knowledge, deeming it usable for designers that lack such training. Also, it allows for the modeling of tasks in their entirety, rather than simplifying them to a succession of pointing operations. However, its shortcomings are also apparent: For instance, the user needs to be at the level of an expert – different skill levels are not modeled at all. Also, the performance must usually be error-free, so that the tree of conditionals does not become too dense. Finally, the performance must be described in extreme detail in order to predict its execution time, which will only be possible once a design has been formulated in a very precise manner. Thus, it is difficult to apply this model early in the design process. Additionally, the model omits user-intrinsic characteristics entirely. For instance, needing to perform an action as fast as possible due to extreme time pressure results in the stress-levels that alter the cognitive behavior of the user considerably. The exclusion of such user-intrinsic modeling makes it questionable whether the Keystroke-level model is able to provide salient data under these circumstances.

In the context of this work, the Keystroke-level model is regarded as very suitable for the evaluation and design of multi-touch interactions, not only because it provides the necessary operators to describe a vast array of different multi-touch tasks, but also because its notion of operation is closely related to the aforementioned notion of an operation in activity theory [13], [64], [57]. The main difference is that operations in activity theory are subconsciously executed sub-tasks of an action, whereas the Keystroke-level model uses the term to describe both subconscious tasks (such as pointing) and conscious ones (such as planning a decision) alike. Thus, a Keystroke-level model operation has to be seen as either an action or an operation in activity theory, but it is not hard to define guidelines about how one notion can be mapped to the other. Thus, when activity theory is used as a scientific underbody in the design stages of a multi-touch product, a Keystroke-level model can serve as an instrument to make predictions about task execution times and, by interrelation, about efficiency.

While providing a framework to make predictions about task solution times on a multi-touch system during its design stage at such a small granularity might not be necessary in some applications, multi-touch is also being employed in devices with extremely tight time constraints, such as musical instruments [53], geo-information and defense applications [87]. Such devices are frequently used in situations characterized by extreme time pressure or time constraints imposed by the situational context, so that the time optimization of the core tasks provided by the system becomes an inherent quality criterium. With the proposed methods, the designer can predict the time characteristics of their interactions before the system is built and formally evaluated, thus catering to an important aspect in the drafting of such systems.

However, it is crucial for the creation of successful multi-touch interactions that the predictions about time spent on tasks is not to be overestimated: With the exception of the very special applications mentioned above, an interface that allows for the solution of a task in the fastest possible way is not necessarily the best interface from the perspective of experiential qualities and overall user satisfaction. For interactions where timing constraints are not the deciding factor, such timing models can still serve a useful purpose as a reminder to keep the designs simple in the meaning of avoiding cumbersome operation chains that are detrimental to the usage experience, but it would be a mistake to replace an enjoyable interaction with one that is predicted by the introduced models to be marginally faster.

Moscovich et al. [81] are ardent proponents for this approach of evaluating performance, who follow an inverse perspective on performance measurements such as Fitts' Law, where they do not aim to find an interaction that can be executed fastest, but to analyze the slower ones for potential frictions, because these can be closely linked to a degraded user experience, which manifests itself in a higher cognitive load, difficult motions and cross-interactions between fingers and hand, which are responsible for the slower execution times. Such a perspective is essential because the design space spanned by two-handed multi-touch interaction is potentially very large, making it crucial to find satisfying mappings, rather than to rely on guesswork. Because the mapping between a mouse and a cursor is a prime example for a successful mapping in traditional interfaces, they conduct experiments where two mice and two cursors are controlled simultaneously to mimic multi-touch interaction, but come to the conclusion that users easily become disoriented when the cursor controlled by the mouse in the left hand crosses over towards the right. Thus, the control structure of how a mouse is operated does not translate well to multiple cursors, posing the question of which control structure would then be adequate for multi-touch interaction. The answer is not immediately clear because the kinematics of fingers on opposing hands and on the same hand are perceived in an entirely different way, where the fingers of the same hand are felt to be moved relative to each other, whereas the hand itself is moved independently. As soon as the interaction becomes two-handed, however, hand motion is again perceived as relative to each other (or to a given global reference frame, such as a screen area), introducing subtle differences in perception that make it hard to define a single control structure for all types of multi-touch interaction. Consequently, this remains an active area of research.

As an anchorpoint for measuring effects caused by the coordination between two hands and multiple fingers, the terms of *parallelism* and *efficiency* are introduced: The former referring to an estimated measurement of how well multiple fingers (or hands) can work together in order to reduce the tracking error, the latter to the ratio between the shortest travelling path possible in relation to the one chosen by the user. To illustrate this approach, they designed an experiment in which users where asked to translate and rotate specific shapes into a goal position on a multi-touch screen, which was signified as a translucent grey copy of the shape in a different position and orientation on the screen. It was found that parallelism was especially high when thin elongated shapes were shown, as users simply grabbed onto their protrusion points and aligned them with their counterpart in the goal shape. If the shape was small enough to be grabbed in this way by two fingers of the same hand, parallelism was slightly higher (due to the hand's ability to keep two fingers fixed relative to each other, while the actual motion comes from the lower arm and the wrist), but this effect was negligible. However, when a more uniform, rounded shape was shown, the typical approach was to first attempt to align the shape's center with the target, then adjust the rotational angle until a fit was achieved, possibly readjusting the center point during this process as well. In this case parallelism was considerably lower, which is explained by the absence of at least two obvious control points within the shape. However, in both cases the multi-touch approach was considerably faster and more enjoyable than a control experiment conducted with a mouse, where orientation and translation had to be specified separately.

Thus, efficiency is not unrelated to user experience. As demonstrated by Moscovich et al., a system that is designed in a way that control points for multi-finger interaction are made obvious not only performs considerably better, but is also more enjoyable and easier to use, because the cognitive load on the user is greatly reduced. Their approach shows that the employment of mathematical performance measures is not necessarily to be seen as an opportunity for stubborn optimization, but rather as a tool to inspire the examination of why some interactions perform better than others, in order to derive guidelines that do not only cater to efficiency, but to user experience as well. Such endeavours will play a crucial part in the establishment of multi-touch interaction within mainstream products and is therefore a field of ongoing research.

3.1.5 Degrees Of Freedom

An important limiting factor in the efficiency (but also perceived naturalness) of an interaction method is the number of degrees of freedom (DOF) it can provide. While a single tracking cursor such as provided by a mouse or a trackball can only provide 2 DOF, and a combination of both 4 DOF, a multi-touch surface is capable to provide 10 DOF if all fingers of both hands are used. The design opportunity provided by such a large amount of DOF has already been recognized early in the history of multi-touch input [16], [61]: A commonly cited example is the manipulation of spline curves, a simple mathematical model of rounded curves defined by a set number of control points, which are often used to model smooth, but complex surfaces, such as a car chassis or airplane wing profiles. It is a widely adopted approach to manipulate only a single control point of the curve by grabbing it with the mouse and dragging it across the drawing area. However, since the interplay between control points is not necessarily trivial and hard to grasp for people without a background in mathematics, being restricted to moving a single point at a time makes it difficult to shape the curve without a large amount of trial and error. On a multi-touch surface, up to 10 control points can be moved concurrently by a single user, helping to make the interplay between the points understandable to the user in a more graspable way and providing for a faster, multi-dimensional way of shaping the curve: In a first step, a raw shape can be defined by coarsely placing multiple control points at once, then fine-tuning the placement one-by-one. Thus, an application like spline curves benefits greatly from a larger amount of DOF both on a conceptual and pragmatic level.

Hancock et al. [42] have analyzed different mechanism for the translation and rotation of digital objects on a screen. Regarding issues related to different input DOFs, they report four fundamental findings: First, the amount of DOFs that the mechanism can provide is a direct indication whether it can exhaust the entire range of possibilities or not. Second, for input methods that do not provide sufficient DOFs for an operation, additional DOFs can be synthesized. As an example, if both rotation and translation need to be addressed at once using a 2DOF input device, the rotational angle can be synthesized from the 2D position of the object on the surface, always orienting it towards the border of the surface closest to the object. Third, synthesized DOFs introduce a problem of consistency, where a trade-off between the ability to provide a large parameter range and consistent behavior in relation to the other parameters has to be made: For example, synthesizing a third DOF for mouse input from the speed of the cursor motion allows for the specification of the 2D position and rotational angle of an object concurrently, the synthesized parameter is not consistent with the positional parameters, because reaching the same 2D position can lead to entirely different rotational angles, depending on the speed of movement. Similarly, if the rotational angle is determined in relation to the position on the screen (such as by closeness to a screen border), the exact same input will result in different angles depending on the starting point of the motion. The fourth finding refers to the reduction of DOFs: Such a constraint can be beneficial in order to allow for the more precise adjustment of related parameters: Much like the employment of a ruler constrains lateral movement in the real world, but allows for the drawing of straight lines, similar techniques (such as “snapping”) are available for digital interactions as well. Thus, the intentional reduction of DOFs, either temporarily or permanently, can be beneficial in some cases, but does not pose a mathematical difficulty nor a problem with behavioral consistency.

Thus, in order to achieve operations that require a larger amount of DOF than provided by the input method, it is either possible to synthesize additional DOFs with the aforementioned problems, or to split up the operations into smaller tasks, for which the amount of DOFs is sufficient. However, this splitting introduces further problems: To stay with the interaction of translating and rotating a 2D object on a surface, some kinesthetic studies have demonstrated that the translation and rotation of objects in the real world is an inseparable task. Even more so, some scientists postulate that these operations are inseparable in the human mind as well [62]. Thus, splitting the operations into two distinct interaction modes is hard to justify from the perspective of naturalness and requires the application of metaphors that are technically inspired, rather than by the real world. It is therefore preferable to utilize interfaces that can indeed provide the DOF that an interaction requires.

Nevertheless, the ten fingers afforded for by the employment of two hands are not necessarily to be seen as ten DOFs in all applications: The motion of fingers (and hands relative to each other) is linked both mechanically (through tendons) and neurologically within rigid constraints. Even more so, some researchers have noted that the usage of only the thumb and index finger of both hands already describes the major DOFs of multi-touch, which refer to the controlling of translation (finger position), orientation (rotation of thumb and index finger around the perceived center point between them) and scale (span of the two fingers), resulting in 6 DOFs

[81]. Applications that aim to exploit the entire 10 DOFs afforded for by multi-touch in theory thus need to be structured in a way that this goal is negotiable within the motion boundaries imposed by the human motor system. For instance, it is very well possible to control ten slider-like widgets simultaneously, but the process will be less accurate and more error-prone than the controlling of each slider individually. Similarly, it is possible to grab onto five control points with a single hand and move them together, but individual motion will be heavily constrained. For a multi-touch interaction designed on this metaphor to succeed, it needs to specifically cater to these limitations.

Multi-touch with its multiple DOFs offers a rich potential for the creation of interactions that is not hindered by the limited flexibility of more traditional devices. Not only does this property allow for interactions that have been infamously difficult to realize previously, but it also enables designers to rethink established interactions that constitute compromises required for by the limitations of these devices. Thus, multi-touch can offer a radical departure from traditional approaches in this respect.

3.1.6 Interface Elements vs. Working Area

There is quite a number of other factors that motivate the experimentation with multi-touch surfaces, but for which conclusive research has not yet caught up with anecdotal evidence and user reports. Nevertheless, researchers are aiming to describe such factors and derive founded theories about how these peculiarities could shape tomorrow's interfaces.

Kurtenbach et al. [63] describe a phenomenon termed the "interface dichotomy": With the example of a drawing application, they describe a tension in GUI design that exists in software operated with traditional input devices such as mouse and keyboard, where GUI elements such as toolbars and palettes compete for screen space with the actual artwork. Usually, the interface elements are positioned at the edges of the screen, whereas the artwork is placed in the remaining middle ground. While this problem can somewhat be alleviated by the introduction of larger screens, a different problem surfaces: The distance that has to be travelled between artwork and interface elements will become increasingly large, causing the user's focus to be constantly torn between tools and artwork, up to the point where they may miss important system messages or changes introduced to the artwork by the system.

A reformulation of these observations is attempted by Fitzmaurice et al. [31], who introduce the terms of *space-multiplexing* and *time-multiplexing* of interactivity in a given application. Space-multiplexing refers to the distribution of interactivity spatially across a surface, so that each function has its own distinct transducer. On the other hand, time-multiplexing refers to tasks being carried out sequentially, but possibly sharing the same transducer, such as a button that is context-sensitive and changes its function depending on the state of the system. It is interesting to note that traditional applications like the exemplified drawing applications are certainly space-multiplexed in the sense that working area and the various function icons take up a distinct position in the screen space, yet interaction itself is time-multiplexed through the same interaction device, usually a mouse or a trackball. This is an inherent dissonance, because interactivity that is available in the space-multiplexed manner is degraded to a sequential chain of operations by the limitations of the interaction device. Thus, it is only logical to aim for the solution of this dissonance by rethinking the way in which an interaction has to be multiplexed for a given device and which intertwining of different multiplexing techniques is beneficial to the overall clarity and usability.

As an alternative, Kurtenbach et al. [63] propose an architecture termed *T3* – Tablets; Two-hands; Transparency. Being conceptually similar to architectures proposed by other researchers [120] [33], the interplay between the three cornerstones is imagined as follows: The drawing surface is a large multi-touch capable *tablet*, on which interaction with *both hands* and potentially all ten fingers is possible. Interface elements are overlayed onto the artwork on an ad-hoc basis when they are needed, so that the artwork is always visible in its entirety, thus maintaining focus. Akin to the previously described notion of "naturalness", the two-handed interaction is closely inspired by how artists handle their drawing materials in the real world. For instance, it has been observed that the orientation of the artwork greatly influences the efficiency of movement: Thus, artists will frequently rotate their artwork so that particular motion, such as the drawing of a straight line, becomes physically easier to perform. Only if the cost of rotating the artwork is large (such as with traditional dialogue-based rotation tools in common drawing applications) will they temporarily work in an uncomfortable position. Similarly, they will move their artwork so that the working position is placed in way that their hands can function in a relaxed manner. With *T3*, the artwork can not only be translated and rotated in a single two-handed gesture akin to the way a sheet of drawing paper is moved in the real world, catering to the needs of the artist, but it is also possible to scale the artwork as well, introducing a dimension at which the artist can change their working setup which is not possible in the real world, but has similar beneficial effects on the drawing process.

T3 furthermore specifies a highly hierarchical way of organizing transparent menus above the artwork, where the position at which sub-menus spawn is constant relative to the parent menu, so that experienced users can conjure specific items by drawing a quick gesture, rather than needing to wait for each sub-menu to pop up.

It is important to measure the benefits of a multi-touch based architecture not only in terms of added functionality over traditional input methods, but also in enhancements to the quality of input, which, being mostly freed from constraints formerly imposed by the technological peculiarities of the input devices, can lead to the creation of an entirely new graphic vocabulary and style by artists being empowered by such a drawing paradigm. While the highly reactive canvas, the resolution of competition between working area and tools area and the ability to draw with the fingers, the most sensuous and intimate drawing style according to many art theorists [33], are highly desirable, a fundamental problem is introduced: The features are not self-revealing. For instance, it is not immediately clear that the canvas can be rotated by the two-handed gesture, because no such indication is given to the user. Thus, a walk-up-and-use scenario is highly limited. This is certainly an issue that should be tackled by researchers in the future in order to find ways of how a user can be guided through a multi-touch interaction such as *T3* without resorting to inelegant methods such as help book and tutorials.

3.1.7 Embedding & Ubiquity

A particular advantage of touch sensitive surfaces is that they can be seamlessly integrated into everyday objects, or, to use a different figure of speech, be hidden within the environment. As an example, it is a feasible approach to hide a touch sensor below the wood furnishing of a couch table and use the interactive surface to control a home entertainment device such as a digital movie player, where (multi-finger) gestures can be mapped to certain functions, such as playing, pausing and resuming the video or adjust the sound volume.

Such examples of embedded touch surfaces are akin to the notion of invisibility in the field of *ubiquitous computing*, where a tool is deemed invisible if it can be operated without conscious attention, so that the user can focus on the task rather than the tool [116] [115]. For instance, the interactive couch table introduced above satisfies this constraint, because the technology necessary to enable it is hidden within the construction of the table and does not require a conscious mode switch to operate. Interaction can happen seamlessly by anybody sitting around the table without needing to cease whichever activity they are currently involved in: A discussion can continue, the newspaper can continue to be read, and so on. On the other hand, reaching for a mouse or a remote control is not an invisible action, because the tool (and looking for it) becomes the center of the user's attention.

The notion of invisibility is also present in the aforementioned model of activity theory: A tool becomes invisible when the conscious act of using said tool is not necessary to perform a given activity [91]. This is akin to looking at a street sign, where we know its meaning without having to actively and consciously read it. Of course, this assumption does not hold for people who see the street sign for the first time or who do not speak the language. Thus, in activity theory, invisibility is not a property of a tool itself, but of the situational context and the characteristics of the user who engages in the usage of the tool to solve a particular activity. The disappearance of a tool can have multiple reasons, only one being the fact that the tool does not require a conscious approach to using it. Another possibility is the level of internalization that an (expert) user has achieved, so that the tool is assimilated into the thinking of the user in such a complete manner that it effectively becomes invisible. As an example, a musician can be so highly trained in playing their instrument, that to them the task of physically using the instrument disappears behind the idea of making music. Another possibility is the smooth evolution of a product through multiple technologies, where the way how the tool works and how it assists in using itself becomes invisible to the user because no new internalization process has to take place. As an example, while a computer being used as a word processor is entirely different from a typewriter at a technological level, this distinction disappears for users who have become accustomed to the typewriter, because they do not feel that the creation of a new internalization is necessary to understand its workings. Thus, activity theory extends the notion of invisibility to include a broader range of disappearance effects, and it points out that understanding the source of invisibility is crucial, as it cannot be separated from the context of time and human activity, and consequently is part of the minimal context required to understand an activity.

It is also vital to understand that invisibility is not necessarily a side-effect of "naturalness" [116]: As an example, an interaction based on voice-recognition, a prominent motive in science fiction, is certainly a very natural form of communicating with a computer, but it is not invisible, since it is intentionally designed to grab our center of attention: We have to signal to the computer that we are talking to it, and we might have to listen to its response – incidentally a reason for speech interaction's popularity in fiction, because it is a way to prominently install the computer as a character in the narrative. Invisibility, however, is more closely related to the notion of "calm technology" [117], where interaction and recognition of system responses happens in the cognitive periphery. Touch interaction embedded into everyday objects can be used in an entirely invisible way, within the full spectrum from completely passive operation in which touch is only sensed, but no specific gestures are necessary, so that the object can react to our touch, but it can also shift into an active mode where the object accepts gestures and similar touch commands. This form of operation is described by proponents of calm technology as the seamless shifting between the periphery and the center of our attention.

For the embedding into everyday objects it is a desirable feature to maintain the object's original purpose. As an example, a touch-sensing coffee table should function properly as a table, and if it is operated as a touch surface, clutter on the surface should not disturb the interaction [91]. Indeed, there are researchers experimenting with different methods to achieve such behavior, both on the level of sensor technology [94] [24] and interaction metaphors [68]. For instance, touch sensors based on capacitative effects are only triggered by objects that are electrically conductive and grounded, while objects generally found on a table usually do not have a connection to ground, so that the fingers of a person standing or sitting in the room are the only actuators triggering the effect. In the field of interaction metaphors, it is being looked at how objects on the surface can be grouped into those that are part of the digital interaction and those that are not. Alternatively, there are many techniques how a user can assist the system in letting the interaction happen smoothly on a cluttered surface.

Because technology to turn any surface into a multi-touch capable interactive area is not yet available, researchers are employing portable touch tracker solutions in order to evaluate the role of such surfaces in an ubiquitous computing environment and how the availability of such infrastructure could fit into the context of everyday activities. One such project is *PlayAnywhere* [124], a portable setup consisting of a projector, a camera and a computer-vision system, that can be placed on any surface and project imagery on it while being able to sense multiple touches without the need for any form of installation or calibration routines. Its scope is the conception of paradigms of how situations such as office meetings could benefit from a multi-touch surface being readily available at all times, but of how such surfaces could be used for recreational purposes, such as for children's games.

Nevertheless, the role of multi-touch interaction in the context of ubiquitous computing has not been studied yet in detail and provides an interesting research opportunity for the future. While the viability of touch-sensing surfaces for such purposes is clear, it is not yet understood which role or which roles it can play in an age of calm technology: Powerful enough to operate complex processes but at the same time subtle enough to happen invisibly, it is an interesting research opportunity to define modes of operation and transitions that can unite these paradigms in the sense proposed by the idea of calm technology.

3.1.8 Miscellaneous Properties

There are additional motivations for the employment of multi-touch interaction that have either not yet been studied in detail or are of lower importance from the perspective of the interaction designer, because they are either achieved implicitly or are only relevant to a small subset of potential applications.

Westerman [122] [120] notes that a particular advantage of touch sensors in comparison to mechanically operated sensors such as buttons and switches is that they do not need to rely on pressure for actuation. This is a great boon for a society in which illnesses such as tendonitis and carpal tunnel syndrome are prevalent, often related closely to working on a keyboard for extended amounts of time. His (now defunct) company "FingerWorks" [29] used to market keyboards in which the individual buttons are replaced by a large touch-sensitive surface, so that users could type with extremely little effort. It has been argued that users adjust to a softer way of typing quickly and that repetitive strain on the motor system of the fingers, a major contributor to the dangers of said illnesses, is largely reduced. With the company out of business due to an acquisition by Apple Inc. in the wake of the iPhone's inception [5], these keyboards are sorely sought after by many.

Incidentally, the usage of multi-touch sensitive surfaces as a replacement for mechanical sensors has already been seen in the mid-1980s [16], where cardboard cutouts were overlaid on a large multi-touch surface in order to delimit sensor positions, akin to placing a drawing template on a piece of paper. With this simple technique, it was possible to turn the touch surface into any combination of buttons, sliders and levers that was deemed necessary for a given task, easily replicating the switchboard of a sound mixing station, an NC machine operation terminal and more. The motivation was not the diminished physical effort for operation, however, but the ability to use a single piece of input hardware for a large variety of tasks that used to depend on specialized hardware. This idea of transforming a single surface for different purposes is akin to the then prevalent notion of multiple users working on a single mainframe computer, but retains its validity up to this day: The ability to use the same touch surface space to mimic different physical controls is still an important aspect of touch technology, such as showcased by Apple's iPhone [5], which, depending on the task at hand, radically changes its screen layout to afford for different input metaphors.

Another advantage of touch input devices are their self-containment and robustness [67]: Since there is no external device like a stylus, puck or pen involved, there is nothing that can get lost or broken, but also nothing that can be rattle, slide or vibrate out of position. At the same time, a touch surface can be molded in a way that it does not have any cavities or openings, making them ideal for situations where dirt and water might damage a device, such as in factories, in extreme outdoor situations or at a construction site, but at the same time, the lack of cavities also implies that dirt, dust and germs cannot build up in the device either, lending touch devices for use in extremely clean areas such as hospitals or research labs. From the perspective of both sensor technology and interactivity, it is trivial to build multi-touch surfaces that are water-proof, so that they can be cleaned easily and even treated with antibiotic agents for hospital use. This is a feature that is not to be neglected, because the ubiquity of tiny holes and cavities where dirt, water and germs can collect is a major issue for the use of such devices in the described settings.

Direct-mapping multi-touch screens have also been shown to be very intuitive to use by young children who do not yet have any prior knowledge of computer systems and are too young to be taught the mapping between a mouse and a cursor, because they cannot yet understand the level of abstraction introduced by indirect mappings [116] [124]. While this result is certainly inspiring for the entertainment industry, who can imagine a whole new breed of interactive electronic games catering to the needs and interests of very young children [124], it is also a promising aspect from the perspective of assistive technologies for disabled children: The abundance of cognitive and intellectual hurdles when using assistive devices like speech synthesizers indicate that such technologies cannot be used before a certain age or level of development has been reached, but it would be a boon for medicine to have assistive devices available that can already be used at a very young age in order to treat a disabled child's specific needs as early as possible as to provide for the development of the child's full potential.

As touch technologies become available to researchers in fields not directly related to computer science through the availability of free open-source solution such as Touché [55], creative minds develop exciting new ways to achieve fantastic results based on the new interaction vocabularies. The application of touch devices to fields that have not traditionally relied on computer technology will be an interesting undertaking that will fully unfold in the coming years.

3.2 Hardware

Even though multi-touch input only seems to have gained traction in the mainstream recently, there have been efforts made in this field that reach back into the 1970s. Often constrained to the lab because of high production costs, slow hardware with lacking capabilities to turn the interaction technique and its applications into anything more than a technology demo or keen vision of the future, these early results retain their influence on modern devices and software designs. With better hardware, faster image processing algorithms and more affordable cameras and electronics becoming available in the 1990s, the field was reinvigorated with an abundance of new research projects, whose successors define how we perceive multi-touch interaction today.

It was then that multi-touch researchers started to push into the mainstream, a trend that is defining the field as we know it, with more recent projects aiming clearly at the feasibility of cheaply producing powerful touch surfaces at a low cost in order to integrate them with consumer products.

The following section aims to introduce some of the most influential hardware results and discuss their peculiarities and special features, that often enable interaction paradigms that go beyond only sensing touch positions: Support for multiple users, depth tracking above the surface, or the ability to track tangible objects and touch alike are just some examples.

3.2.1 Before 1990

While single-touch technology dates back well into the 1960s, it is hard to determine the first device that was capable of sensing multiple touches at once and track them over time. Quite frequently, this feature was technically supported to some degree, but neglected because the researchers were only looking to sense a single touch.

In 1972, Johnson et al. [52] register a patent for a touch-sensitive screen, in which sources of either light or Rayleigh wave beams are mounted on two adjacent sides of a CRT screen, projecting a grid of beams closely above the screen surface. Opto-detectors positioned opposite the beam sources can detect the drop in light flux energy when an object such as a finger is breaking the beam. Subsequently, the 2D position of this object can be determined at the resolution provided by the spacing of the beam generators. While the patent suffers from issues of occlusion (if more than two touches are taking place within the same beam line, the second coordinate of this touch can only be determined if the other beam is only broken by a single touch position) and while the resolution is very low due to the size and price of the beam generators and detectors, this is one of the first documents in which the ability to sense multiple touch positions is explicitly mentioned as a feature, even though the patent does not include any interaction techniques that are based on multiple touch sensing. Nevertheless, the idea of using optical effects to determine touch position, and even the particular technique of employing a sensor grid, is in use up to this day.

Other early results do not stem from the wish to use touch sensors as a human-computer interaction technique, but to build powerful sensors for robots in order to provide proper feedback during grabbing operations and to even distinguish between different objects from the touch information acquired [45] [107] [82]. These sensors are based on the *FTIR effect* (frustrated total internal reflection), which describes the behavior of light reflecting from the edge of a

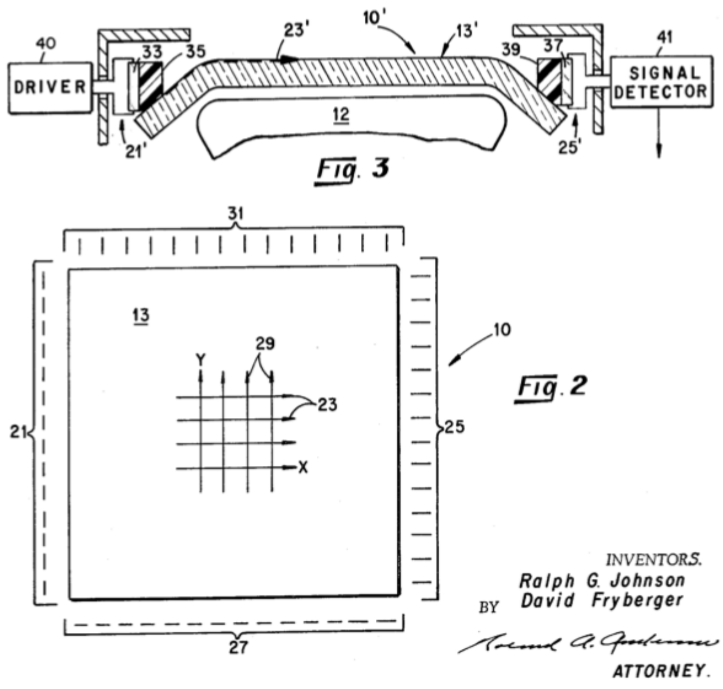

Figure 1: Rayleigh wave beam touch detection [52]

border between two media if the refractive index of both media meets certain criteria. In practical application, the FTIR effect can be used to trap light within a piece of transparent plastic (such as acrylic glass) by shining it into the edges of the sheet. However, if a medium with a larger refractive index than air touches the surface in which the light is trapped, the FTIR effect is cancelled at the touch point, and scattered light can leave the glass.

These early robot sensors consist of a piece of transparent plastic, into which light is shone, sandwiched between an array of optosensors that respond to incident light and a flexible membrane, usually made of silicone. When an object exerts pressure on the membrane, it is pushed against the light guide, breaking the FTIR effect, so that light escapes towards the touching object and is reflected back by the object towards the sensor array, which can then register an image made up of “pressure pixels”. Thus, these sensors are able to sense an arbitrary amount of touch positions and touch shapes limited only by the resolution of the sensor array. However, even though this technique was later picked up by researchers such as Han [39] to create one of the most popular setups for optical multi-touch surfaces in use today, the initial idea behind the sensor was to create a “touch image”, so that the robot can distinguish between different types of objects, such as nuts, bolts or screws. The sensor design succeeds in this respect, and similar sensors are sill in use for touch sensing in robots, but also for devices such as electronic fingerprint scanners, albeit with a significantly higher resolution.

Probably the first touch surface to use the FTIR effect in the context of a multi-touch surface is introduced by Kasday et al. in 1984 [58], although its workings are slightly different than the designs used today: Rather than using a coupling membrane to break the FTIR effect achieved within the transparent plastic guide, their design employs the opposite way: When the transparent membrane is flexed towards the plastic guide, ambient light is coupled into the plastic sheet, where it is trapped by FTIR. Opto-sensors are aligned along the edges of the plastic sheet, where they can pick up the light exiting through the sides (where the angle is too oblique for the FTIR effect to take place), and the light data is used to determine the position of the touch. Unfortunately, this approach is extremely error-prone with multiple touch positions, because the light spreads through the sheet in a concentric way, so that light emanating from different touch positions mixes and makes it hard to accurately compute the individual positions. However, because high resolution light sensors (such as digital cameras) were not readily available at the time, the design strikes a compromise between accuracy and expense, but becomes obsolete with the introduction of better sensors. Still, it is notable as one of the first large multi-touch surfaces.

Researchers such as Lee and Buxton [66] [67] [16] were among the first to recognize the ability to sense multiple touch points simultaneously as a potential for novel interaction paradigms, rather than a side-effect of a particular sensor technology. Their touch technology is based on voltage drain due to the capacitative coupling between an antenna and a grounded conductor like a human finger. By covering a surface with small metal plates, applying a signal to them and then measuring the draining of this signal when a finger is touching it due to capacitative coupling between the finger and the plate, they managed to build touch tablets that were able to continuously sense multiple touch points at about 12–24 updates per second. Because the performance of their approach is bounded by the hardware's ability to take measurements of the capacitative effects, they developed area sub-division algorithms in order to minimize the amount of measurements necessary per frame, providing for better temporal resolution. Low spatial resolution due to the size of the sensor plates and the speed of the measurement circuitry could be alleviated by employing direct interpolation schemes, resulting in an adequate ability to locate fingers on the surface within reasonable accuracy bounds.

On the side of interaction design, their early experiments aimed to recreate machine interfaces usually consisting of mechanical parts such as buttons, switches and sliders on a multi-touch surface, using simple cardboard cutouts to delimit the interface elements, partly inspired by the

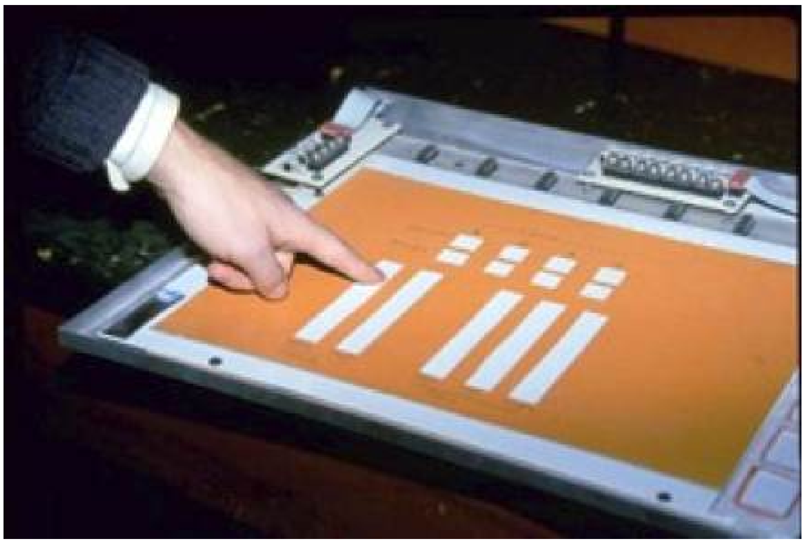

Figure 2: Multi-touch surface with cardboard overlay [67] [16]

idea of creating low-cost alternatives to such specialized switchboards that could be repurposed on the fly, but also by an interest in whether such surfaces, lacking the haptic and mechanical feedback that the original solutions provide, are usable in this manner. They noticed that due to the thickness of the cardboard and the way the embossed delimiters can be felt, users quickly managed to accustom themselves to the altered feel of the interface and learned to "touch-type" on it, getting a good understanding of each element's position relative to the others. Thus, they concluded that this type of interface can be suitable to solve the long-standing conflict between screen real estate used up by controls when they are shown on screen, or eye diversion when they are placed on a flat tablet without any haptically discernable delimiters.

While the ability to model multiple virtual devices on a single interactive surface excited their imagination, they also identified some problems: They noted that the amount of pressure a user has to exert on the surface to get a touch registered as well as the amount of friction imposed by the surface material are crucial in defining overall user experience: Too much pressure, and the usage experience will be tiresome, too little, and it will feel jittery. Also, they noted that it is unclear how a point selection, hitherto usually tied to a mouse button click, could be modeled on the touch surface: While techniques such as "tapping" have become more popular in recent years, their proposal was to utilize pressure data to detect click events, such as a simple pressure threshold to distinguish between a *hover* and a *click* event. Additionally, while realizing that the sensing of multiple fingers and hands holds potential for highly parallel interaction and chordic manipulation, they did not yet succeed in defining ways of how this potential could be leveraged.

At this time, innovative ideas of using touch input were mostly found in the arts. *Videoplace* [61] is an interactive installation, combining many of the most advanced results of different fields of computer science to create an experience in which a more personal, sensual side of digital technology takes the center stage, showcasing, among other exhibits, a digital critter that can be manipulated with and responds to motions of the hand. The creators of Videoplace acknowledge the fact that it is natural for people to attempt reach out and touch digital objects imposed into their reality, and that touch input, with its rich naturalness, can play an important role in shaping interfaces that are not aimed at computer scientists, but at the general public, at a time when digital devices were not yet as pervasive as they are today.

Similarly, the *drawing prism* [33] is an interesting device in which light refraction effects in a large prism are used to create a setup in which a camera picks up only those objects that directly touch a surface of the prism, relying on optical properties of prisms and the FTIR effect to hide everything else by coupling the incoming light to another side of the prism, letting the camera pick up only those images that are directly placed onto the surface, so that the FTIR effect does not take place. The device attempts to allow digital artists to draw onto the surface with their fingers or other natural objects like brushes, enabling them to draw with instruments that are not simulated, but real. Thus, it is possible to define an entirely new paradigm of digital drawing, where the finesse of real brushes is combined with effects only achievable through computerization. The ability of an artist to use multiple drawing instruments at once, such as smudging parts of the image with a finger while drawing with a brush, is acknowledged as an important advantage that this system has over similar drawing applications. Indeed, pictures created with the *drawing prism* have sensual qualities to them that were impossible to match with other approaches at the time.

3.2.2 The 1990s

It was not before the 1990s that researchers were developing and experimenting with setups that resemble the types of multi-touch surfaces and screens that we are accustomed to today. *DigitalDesk* [119], a classic paper in the field of augmented reality, is among the first developments in which a digital table-top with clear multi-touch paradigms is realized. It was inspired by then popular ideas of the paperless office, but confronted with the fact that people enjoy some properties of physical paper that have not yet been recreated in the digital realm. The scientists realize that the skills that help us manipulate objects in the real world as well as our kinesthetic memory remain largely unused by the prevalent desktop PC metaphor, where it is attempted to draw upon the understanding of the real world, but not the skill. Thus, they

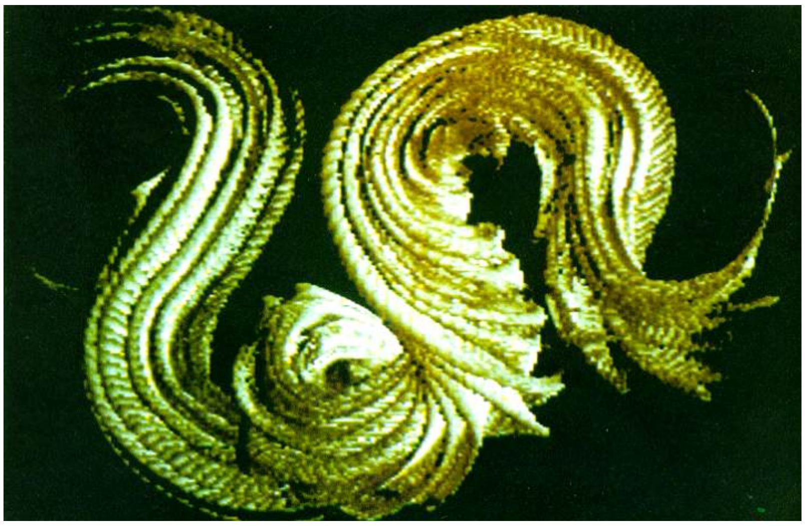

Figure 3: Brush stroke on the *drawing prism* [33]

envision a system in which documents can be paper and digital at the same time, combining the advantages of each approach.

The *DigitalDesk* is a physical table-top, above which a beamer is mounted to project digital imagery onto the surface. A top-mounted camera is used to track fingers and paper sheets that are placed on the surface, using algorithms that are optimized towards the recognition of motion (because only those objects are deemed interesting that are currently in motion) and capable of extracting useful tracking information despite a complex, ever-changing background caused by the projection. Because the optical system cannot accurately determine when a finger touches the surface, a set of piezo microphones is used to detect finger taps. However, a particular weakness of this approach is that loud noises such as clapping or sneezing can confuse the system, and the locality of taps might be ambiguous if multiple taps happen in quick succession and close proximity to each other, as the system might fail to associate each tap with the finger that was actually responsible for it.

A core concept is the blurring of the border between digital and paper documents. As such, paper documents can be digitized on the table (albeit at a low resolution limited by the cameras, since pre-scanning was deemed as detrimental to the experiment, because it is a step introduced due to a technical limitation that should not interfere with the interaction itself). Text on digitized documents is recognized by an OCR engine, so that typical text-based operations such as cut, copy and paste are made available. Furthermore, these documents can be reoriented, repositioned and rescaled with multi-touch gestures. Content derived from digitized documents can itself be copied again, or can be printed out in order to create new paper documents. Addi-

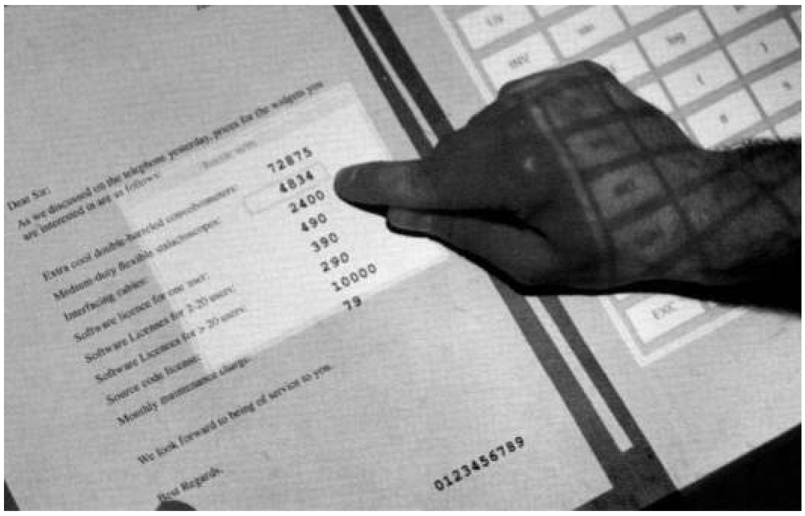

Figure 4: Performing calculations on the *DigitalDesk* [119]

tionally, there are other applications that harness the power of the OCR and touch recognition engine, such as a calculator, which can read handwritten input as well as input typed on a projected numberpad. Thus, *DigitalDesk* is one of the first examples of multi-touch concepts being employed in a complete application context.

Another interesting project is *ClearBoard* [50], a collaborative workspace for two remote users, which is based on the metaphor of standing on opposing sides of a glass window: To accomplish this, cameras and half-mirrors are used to capture the image of the user standing in front of the surface, as well as to track their touches on the screen. Because the cameras are positioned directly behind the screen, users can keep eye contact during the session, while drawing and writing on the screen with their fingers in order to supplement their discussion. Touch input was chosen for its "naturalness", fitting the metaphor of the glass window perfectly. Indeed, the only technical alteration is the mirroring of the images displayed, so that both users share the same sense of left and right, and text written by the other user does not appear in reverse. *ClearBoard* is also notable for being one of the first works in which the problem of parallax is mentioned, caused by the screen diffusor being placed below a thick glass surface, so that there is considerable space between an object on the screen and a finger touching it. Parallax is an issue that is inherent to many direct-mapping multi-touch screens, if a protective glass layer is used.

Like ClearWall, *LiveBoard* [27] is a groupware application, but it can support multiple users both locally and remotely. It consists of a large rear-projection screen and aims to provide a big collaborative working area on which users can present, but also collaborate on digital

Figure 5: A touch-augmented video conference on the *ClearBoard* [50]

documents, such as text and image files. While it is operated with light-pens rather than with direct touch, it is notable for employing many interaction techniques that have become common in touch interfaces: As an example, it supports the scrolling of documents with a sweeping motion, where the speed at which the user flicks the pen across the scrollable area determines the momentum of the scrolling motion that the area will be put into, continuing to move on and gradually slowing down as if it had its own weight. The usage of simplified physics systems to make objects behave in a more natural way is a common feature of modern multi-touch systems.

HoloWall [76] is another visionary product that relies on two-handed manipulation to enable interactive walls as a meeting point for collaborative work. Also a rear-projection system, a beamer is placed behind the wall, fitted with an infrared-cut filter, while a camera with an infrared filter is used to capture images through the wall, which are used to track users and hands in front of it. Due to the employment of IR filters, the projected image does not interfere with the tracking, a setup that is widely used today as well. The plastic sheet covering the wall is semi-opaque and diffusive, which serves as a suitable projection area, but also ensures that only objects touching (or close to) the screen are visible by the camera. An infrared illuminant is used to provide good visibility regardless of ambient light. Another feature is that the setup can scan 2D barcodes that are placed directly on the surface. Thus, a HoloWall can be used to enable customers in a shop to see additional information about products they are considering to buy, for instance. Manual interaction on the HoloWall is limited by capturing resolution, but using a hand to translate objects, or two hands to rotate and scale them is

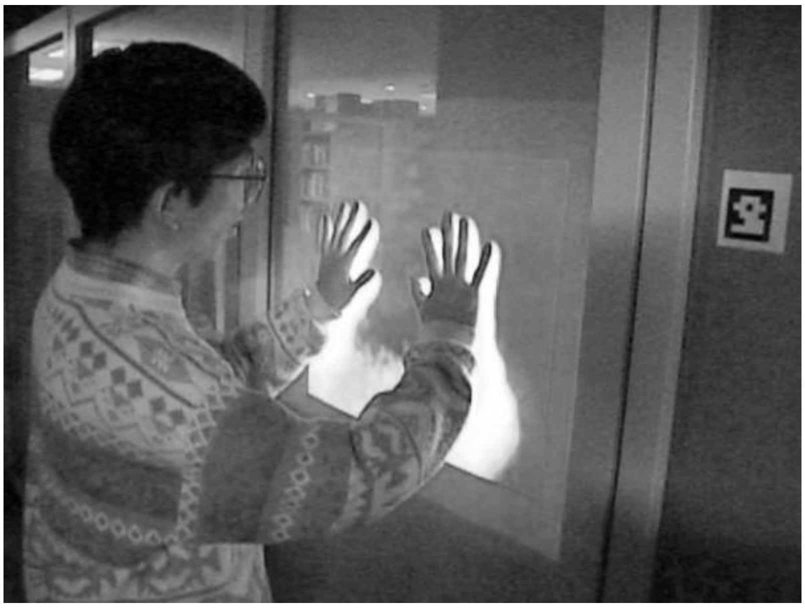

Figure 6: Interacting with *HoloWall* [76]

supported. HoloWall is not intended to be a full-fledged interactive computer system, but rather a kiosk-type application, with which users can discuss a particular document quickly by having it appear on the wall, or in the context of events and advertisements: The core idea is to appropriate the screen space that ubiquitous walls can provide for simple interactive tasks.

The time period also saw many advances in camera-based hand-tracking algorithms, utilizing either multiple cameras for 3D position sensing, or certain techniques to derive meaningful 3D estimates from just a single camera [23, 112, 98]. However, most approaches were inspired more by promising visions of virtual and augmented reality systems, in which a user is immersed into a complex mixture of virtual and real objects within a three-dimensional context, rather than the subtly different needs of multi-touch surfaces. Still, a driving factor behind the development of optical tracking approaches was the insight that being tethered to devices such as tracking gloves, sensors or light-pens is detrimental to the acceptance of a system among users, an insight that is also central to the promises behind multi-touch surfaces. Examples of such techniques are the capturing of hand images with multiple cameras at known positions, using a depth image to extract hand positions, capturing with a single camera, but using mathematical hand models derived from physical constraints to match them with extracted feature points in each frame, in order to give a good estimate of actual hand position and orientation, or using a camera and a light source at known positions to derive fingertip positions by correlating fingers and shadows in each frame. Notably, many researchers were experimenting with the application of multi-touch concepts (such as the scaling operation by moving apart two fingers) in 3D spaces.

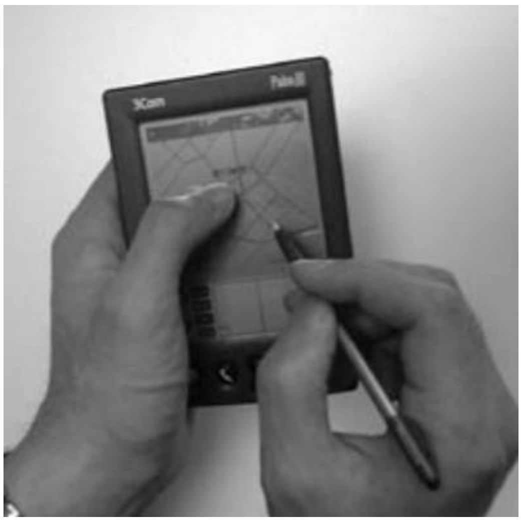

Figure 7: Two-contact handheld interaction with *DualTouch* [75]

3.2.3 2000 and onwards

More decisive interest in multi-touch technologies and interaction resurfaced, partly fueled by formerly expensive hardware such as projectors, fast computer vision cameras and flatpanel screen becoming available at much more affordable prices. Consequently, many developments are based on modifications and compositions of off-the-shelf parts, inspiring a much wider group of research to experiment with multi-touch surfaces.

A prime example of a simple modification to experiment with multi-touch concepts is Matsushita et al.'s *DualTouch* [75], a software trick which enables resistive touch screens common on PDAs around 2000 to sense the combined motion of two simultaneous contact points, provided that they do not touch down on the surface during the same digitizing frame step: It is a property of such touch panels that two touches are reported by the hardware as a single touch, which is located at the midpoint between the actual touch positions. Thus, if the real location of one of these touches is known (because it previously was the only touch), the positions of both touches can be computed. With DualTouch, a number of interaction techniques were developed, such as using the thumb of the hand in which the PDA is held simultaneously with the stylus, using this thumb to open menus from side-mounted menus while making selections with the stylus. With multi-touch interaction nowadays common on handheld devices, the research value of this simple approach is not to be underestimated.

One of the most well-known table-top setups is *DiamondTouch* [24], developed at Mitsubishi Electronics Research Labs (MERL) and later spun off into its own company for distribution.

Its philosophy is based on the idea that a coffee-table like collaborative working surface would be desirable, but touch technologies of the time did either only allow the sensing of a single touch point or could not distinguish different users, while other indirectly mapped interaction paradigms would grow confusing for multiple users, who would have to keep track of multiple cursors at the same time. Also, people tend to put objects on tables, so it was a requirement that the tracking must be robust against this constraint. MERL's technology utilizes an array of antennas (shaped like diamonds, hence the name) below an insulating surface to couple a charge through the user's finger as it touches the screen, forming a tiny capacitor between the antenna and the finger. The user is also coupled to the system via the chair they are sitting on (or the floor they are standing on), thus forming a capacitance circuit with the user in the middle. Each antenna transmits its own distinguishable signal, and each user is coupled to their own receiver, so that both the locality of each touch and the user responsible for it can be determined. The system achieves a refresh rate of 75Hz, providing for very fluid interaction. While the original design uses top-projection with a beamer to generate an image, semi-transparent conductive compounds such as indium-tin oxide (ITO) can be used to create a transparent antenna overlay, which can be placed on a flat screen or rear-projection surface

While the original DiamondTouch demonstration already included multi-touch interaction such as two-fingered rotation and scaling, its commercial availability has sparked the creation of specialized toolkits that retain their importance to this day, because developments that do not only sense multiple touches, but can also distinguish between users, are scarce. *DiamondSpin* [101] is one such framework, which provides an engine to transparently translate between the Cartesian coordinate system required by drawing toolkits such as OpenGL and polar coordinates, which are more suitable for applications in which orientation (due to users sitting at different edges of the table) needs to be highly flexible. Also, the system translates event handlers in a transparent way and can handle multiple concurrent inputs, as is to be expected on a multi-touch system. Additionally, a layering engine is provided, which can marry the ease of using distinct objects in an interface with the requirements of fast drawing performance, which needs to do as little compositing on its own as possible. DiamondSpin is inspired by the realization that the lack of software frameworks makes it hard to quickly write multi-touch applications for purposes of experimentation and research, posing a considerable barrier to entry. Indeed, DiamondSpin is looking to fill a similar void as the Touché framework [55] developed as an accompaniment to this work.

Also based on capacitative effects is Rekimoto's *SmartSkin* [94], a touch sensor based on a grid of insulated wires that scales to large surface sizes and is easy and cheap to build: Rather than relying on the closing of a circuit by capacitative coupling, the grid is used in a way that each

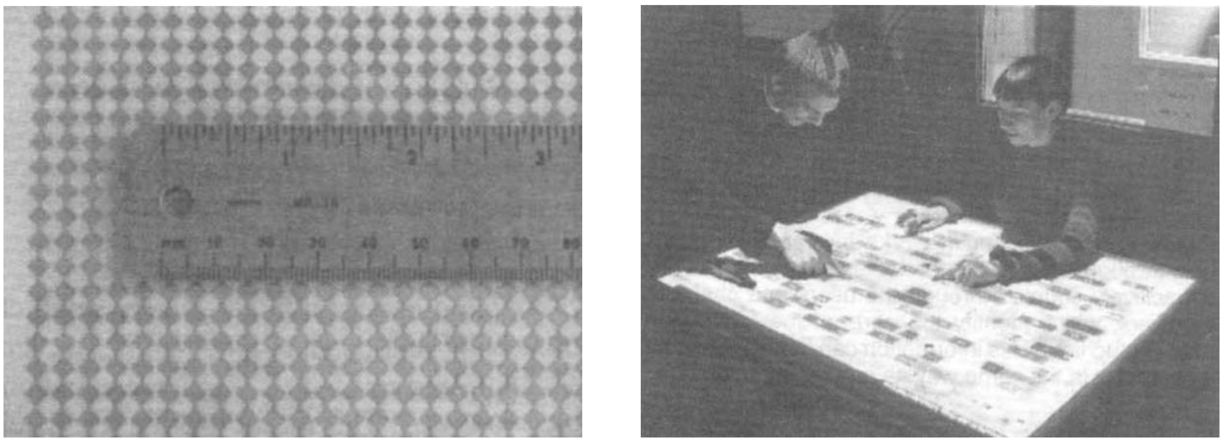

Figure 8: *DiamondTouch* sensors and interaction [24]

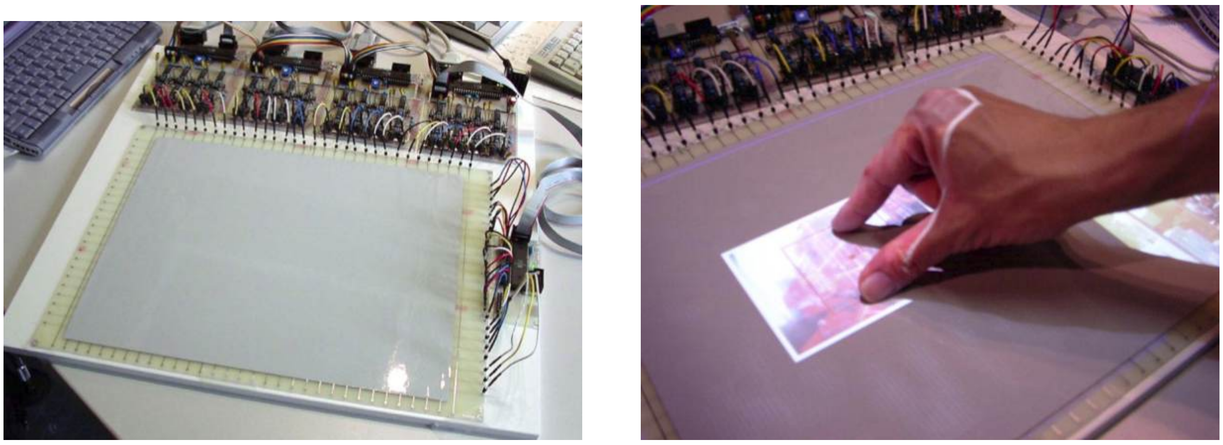

Figure 9: *SmartSkin* capacitative touch sensor [94]

wire in one direction is driven by a signal generator with a known frequency, while this signal is capacitatively coupled into the wire at the other direction, since the two insulated wires act as a tiny capacitor at their intersection point. By using a circuit based on readily available Atmel microprocessors, the frequency of the signal picked up in the inactive wire at each intersection point is measured. The proximity of a grounded conductor such as a finger causes this frequency to drop, since the finger and the wire act as another capacitor, which drains the signal considerably. Thus, by iterating over each wire intersection point and measuring the signal drain, an image can be constructed which represents the proximity of grounded conductors. While this approach is bounded by the speed at which the measurement can be taken at each point (since driving multiple wires at the same time would lead to interference problems), reasonable temporal resolution is achievable. When large grid sizes are used to reduce the amount of wires in the system, bilinear interpolation can be used to improve spatial resolution.

Like DiamondTouch, SmartSkin's technology can not only be used with top-projection to create multi-touch screens, but also as an overlay for flat screens if transparent conductors are used as electrodes, such as ITO. An advantage over DiamondTouch, which uses plates as antennas, is that SmartSkin's wires can be very thin and flexible, making it possible to create surfaces that are curved, or even enabling complex geometric objects or toys to become touch sensitive, which is how SmartSkin has been given its name, e.g. as a potential touch-sensitive skin for robots and toys, similar to a commercially available sensor named *Tactex* [106]. Also, Rekimoto proposes capacitance tags, which are objects with conductive tag patterns for the sensor to recognize: Since the objects only become visible to the sensor when they are grounded (e.g., by a user touching them), they can be seen as a new class of tangible objects that are active only when they are touched, rather than at all times.

However, touch sensors based on capacitative effects suffer from a considerable disadvantage: They do not scale well to large surface sizes and high sensor densities, because as the amount of capacitance sensors increases, the signal-to-noise ratio drops off to a point where it becomes very hard for current technology to provide a reliable sensor result [122]. Consequently, vision-based system have been very common in recent years.

Examples of such systems are manyfold, with Microsoft's *Surface* [19], the *reactable** [53] or *ThinSight* [46] being just a few notable implementations. All of these system share the same paradigm, in which an infrared light-source is used to generate visible touch points when an object comes into contact with the surface. This is achieved by using a strong near-infrared illuminant to light out the surface evenly from either below or above. One or multiple cameras can be used to capture the patterns of light either reflected back (in the case of rear-illumination) or blocked off (top-illumination). A diffusor sheet is used, so that light scattered back from objects that are not close to the surface is diffused sufficiently, turning far objects invisible to the camera. Image segmentation techniques are then used to derive a set of touch positions in each frame, which are then correlated across multiple frames in order to track each touch during its lifetime on the surface. The near-infrared spectrum (between wavelength of about 750nm to 1000nm) is used because tracking is not made difficult by light variations in the visible spectrum, such as when an image is projected onto the surface. Advantages of this approach are that it becomes very easy to build such systems, since cameras capable to sense light in the near-infrared spectrum are readily available. Additionally, spatial and temporal resolution are mostly bounded by the quality of the camera, since the the necessary image processing can be easily handled by current processor hardware. Because the signal-to-noise ratio is not dependent on the amount of concurrent touches or the size of the surface, is is possible to sense an arbitrary amount of concurrent touches on arbitrarily sized surfaces, provided that

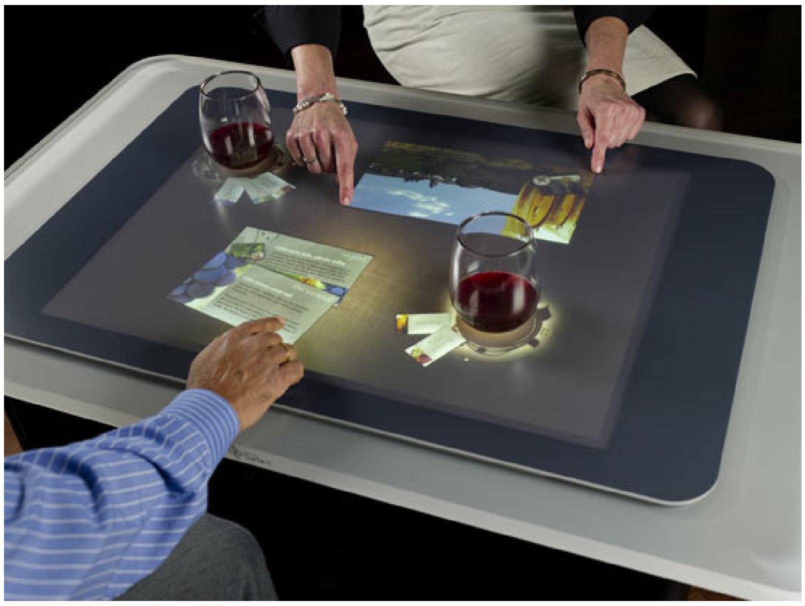

Figure 10: Near-infrared vision-based touch sensing with Microsoft Surface [19]

the whole surface can be captured at a resolution that permits this and hardware performance is sufficient for the required processing of multiple camera feeds. The main disadvantage of optical systems is their vulnerability towards surges in ambient light that become brighter than the light intensities resulting from objects on the surface. While there are many techniques to make optical systems more robust in this respect, it remains an issue under harsh conditions such as direct sunlight.

While the pixel resolution of commonly used cameras in computer vision is high enough to allow suitable accuracy to track larger objects such as fingertips, it is too coarse to provide for the high resolution needed to track subtle motions and tiny contact surfaces, such as when a pen is used to draw onto the surface. Thus, it is difficult to combine multi-touch sensing with pen interaction when using such a system. However, it is possible to use different sensing techniques for touch- and pen-tracking [39, 36, 35, 15]. For instance, it is possible to use Anoto patterns to provide high-resolution pen-tracking on a touch surface. Wireless pens use tiny patterns printed onto the surface to sense their position and transmit it to the system. Since these pattern are very small and can be printed with ink that is transparent in the near-infrared spectrum, they do not interfere with the optical touch tracking.

Han's approach is slightly different in the way optical touch response is generated: Rather than relying on light being scattered back, he uses the aforementioned FTIR-effect (frustrated total internal reflection) in order to achieve to light up fingertips coming into contact with the surface

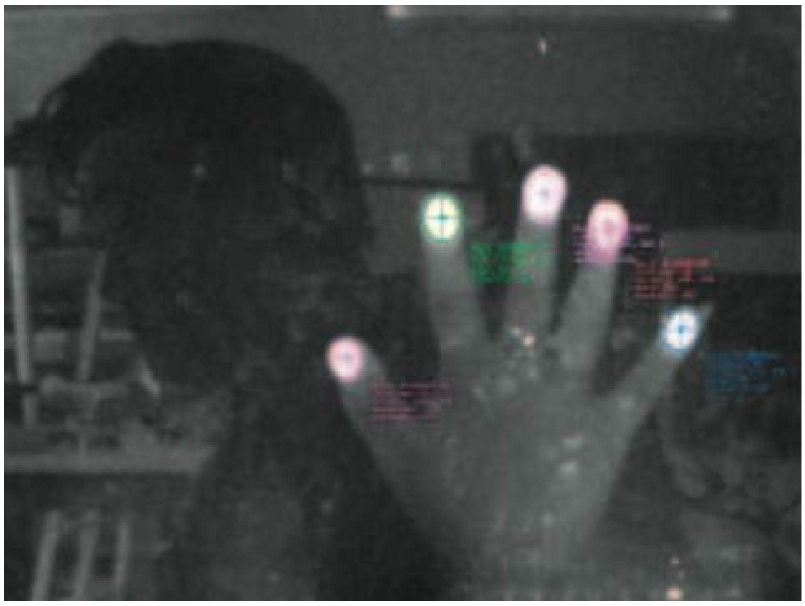

Figure 11: FTIR-based touch sensor surface [39]

[39]. To do so, multiple infrared light sources (usually strong light-emitting diodes) are placed at the edges of an acrylic glass surface, causing the emitted light to be trapped within the sheet due to the refractive indices of acrylic glass and the air surrounding it, as light is reflected back at the medium border, rather than being able to cross it. However, when a medium with a higher refractive index (such as a fingertip) is squeezed against the acrylic glass sheet, so that there is no air gap in between, light can cross the boundary and diffuse into this object. Thus, the fingertip lights up and becomes visible to the camera. This approach is very elegant, as a light response is only triggered if the finger is actually touching the surface, but not when it is hovering close to it. High contrast can be achieved, because the surface itself appears entirely dark and transparent to the camera unless the FTIR effect is broken at a certain point due to a touch. A drawback, however, is that an air gap between the finger, even a small one, hinders a light response from being generated. This makes it difficult to perform finger gestures on the surface, since the ability to smoothly glide is severely limited by the fact that one has to keep the entire finger pushing down at all times. However, a compliant surface, such as a soft layer of silicone, can be used to provide tight coupling without the need to flatten the finger against the surface. Han's approach is notable for sparking an interest in building multi-touch surfaces among a thriving internet community, due to its simplicity and reliance on low-cost parts.

While such optical setups are easy to build, and sophisticated computer vision algorithms can be used to achieve good tracking accuracy and performance, they suffer from the fact that the camera (and, if used in conjunction with back-projection, the beamer) need a certain minimum light path to cover the entire surface, even if wide-angle lenses or fisheye-lenses as well as suitable undistortion algorithms are employed. Thus, the minimum depth of the system is bounded by the required light path, making such setups large and difficult to wield. Researchers are looking into different ways to overcome this limitation.

One viable approach is to combine a projector and a camera in a compact unit, which can be placed on a surface to project an image at an angle, using keystone correction algorithms to provide a rectangular projection even at an angle [124]. Touches are then tracked by illuminating the surface with a near-infrared light source that is also mounted within the unit in close proximity to the camera, then making assumptions about the shape of the shadows cast by fingers in order to determine when a finger is sufficiently close to the surface to be regarded as a touch. Since miniature projectors are becoming readily available, such a device could be small enough to be carried around, turning it into an interesting opportunity to create multi-touch surfaces and even screens on the fly by utilizing flat surfaces within the environment, such as tables, walls and floors. As the relative position of camera, projector and illuminant are known in advance, the device does not need to be re-calibrated beyond the factory, providing a setup experience of simply placing it onto a surface and turning it on. However, occlusion is a major problem, especially if multiple users are involved at the same time. While sophisticated algorithms can interpolate the motion of touches that are momentarily occluded, touches that are occluded during their entire lifetime cannot be registered by such a device.

Occlusion problems can be alleviated by utilizing optical sensors that are spread out across the entire surface (rather than concentrated on an image sensor) and thus do not need a long light path or a lens to work. Some researchers are exploiting a little-known property of light-emitting diodes (LEDs) [48, 38]: While LEDs do not nominally conduct a charge in the opposite direction, small amounts of current nevertheless leak from cathode to anode if the LED is driven in reverse. It therefore becomes possible to charge the small capacitance within the wire and LED itself by applying a voltage in reverse, then measuring the time it takes for this charge to leak across the LED. Due to the materials used within the LED, this time is influenced by the amount of ambient light, allowing the LED to be used as an opto-sensor. By quickly alternating the LED between light-emitting and light-sensing operation, continuous ambient light measurements can be taken, while the LED still appears as being constantly on to the human eye, which is not capable of sensing such fast flickering. Thus, a LED matrix can be used in a way where a proximity image is generated by using one LED as a light sensor while using the surrounding LEDs as emitters, so that when a finger is covering the sensor LED, decidedly less ambient light is picked up. With this shadow image, computer vision algorithms can again be used to derive touch positions. While this approach has only been realized with

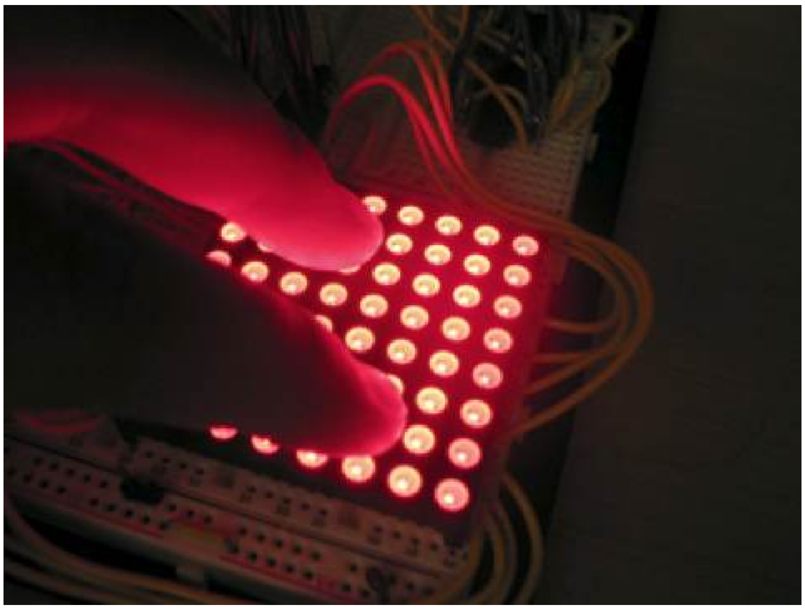

Figure 12: Multi-touch by using LEDs as light detectors [48]

low-resolution LED matrices yet, developments such as organic LED screens make it possible to use the screen itself as the light sensor, providing for high-resolution shadow images without introducing an additional sensor layer, so that a multi-touch capable screen does not need to be any thicker than a non-sensing one. The main disadvantages of this approach is that the wiring becomes decidedly more complex to be able to drive the LEDs in reverse as well, and that the measurement of each LED in sensing mode can take up to a couple milliseconds if little light is picked up and the charge is leaked very slowly. However, by parallelizing measurement in multiple LEDs, using special materials that leak the charge faster and using a cutoff-threshold after which measurement is ceased, and the pixel is marked as “dark”, this issue can be alleviated. Additionally, it might not be necessary to generate a proximity image at the same resolution as the matrix itself, so that not every LED must need to be used as a sensor: As fingertip are rather large compared to a pixel in an image, a significantly lower proximity image resolution will be sufficient for most tasks. Thus, the approach will become increasingly interesting when current display technologies such as LCD are phased out in favor for self-emissive technologies such as OLED and AMOLED.

A similar approach has been applied by the creators of *ThinSight* [46]: By using a conventional off-the-shelf LCD display, but placing near-infrared light sources and detectors behind the backlight, it is possible to capture a proximity image at an adequate resolution. The LCD panel itself with its orthogonal polarization filters blocks out most ambient light, so that only reflected light emitted by the built-in light sources is detected. Since specialized opto-sensors rather than LEDs are used, measuring time is less of an issue. Additionally, the sensors can also pick up strong self-emissive light sources in front of the LCD panel, so that it is imaginable to use LEDs at appropriate wavelength to create simple pens. Blinking patterns can be used to identify multiple pens. While the system was only implemented as prototype and is limited to tracking at only 11Hz, the additional space required for by the sensors is small, so that the screen is still conveniently thin. It is possible to design a more advanced system based on the same principle by fusing polymer opto-sensors directly to the back of an LCD or OLED screen, providing for incredibly thin designs and faster update rates.

An entirely different approach to optical multi-touch tracking is not to use any apparatus at all to make individual touch points visible, but to track hands and estimate their pose and contact points out of a camera feed without any other mechanism involved [25, 8, 112]. The advantage of such approaches is that they only require the mounting of one or multiple cameras above a surface, without any additional hardware related to touch tracking at all, and that they can be used to provide tracking not only of contact points, but also (in a limited fashion) above the surface, providing for the employment of 2.5D gestures as well. The main difficulty, however, is the segmentation of the image into hand and non-hand regions, which is an error-prone process due to the lack of distinct features that can be used to distinguish hands from the background in a cluttered image with high confidence. A common technique is to base segmentation on pixel hue values that are associated with human skin tone, but different skin colors as well as different lighting conditions make it difficult to define clear hue regions within which skin tones fall. An interesting solution to this issue is the usage of long-wave infrared cameras (heat cameras), which can be tuned to detect only such objects that have a temperature closely resembling normal human body temperature [96] – most other objects that may be cluttering the surface have different surface temperatures, even though some (such as hot beverages) can still pose a problem.

Once the segmentation is done, computer vision algorithms are used to decide which objects can be classified as a hand by their shape, using either a static analysis of peaks and valleys within the shape, or a mathematical hand model which is fitted into the shape to determine whether it is likely to be a hand. While the former approach is easy to implement, it is prone to fail if the hand pose suffers from strong self-occlusion, because individual fingers might not be visible anymore. The latter approach performs better in such situations, since the mathematical model is transformed with each frame continuously, so that intermediately occluded finger positions can be extrapolated from previous data. However, once such an algorithm loses track of its position and orientation estimate due to prolonged occlusion, it is a computationally expensive operation to re-initialize the model, which will cause stuttering

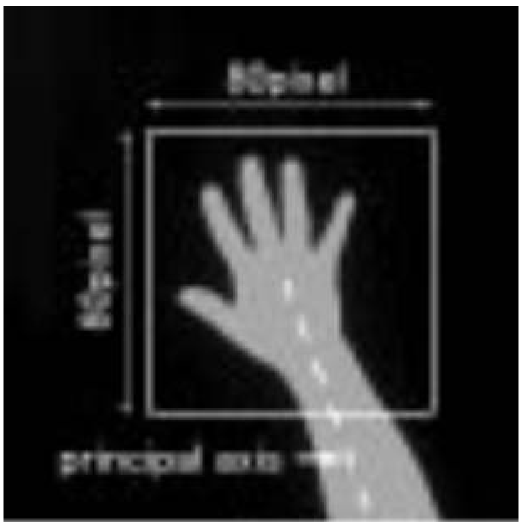

Figure 13: Hand tracking by using a heat camera [96]

in the tracking output. Nevertheless, advances in this field offer a promising outlook of multi-touch surfaces that can be set up anywhere with just a couple of cameras, without the need for any other installations.

An interesting example of such an approach is the *perceptive workbench* [105], which combines the rear-illumination technique for touch-tracking with ceiling-mounted lights and cameras at known positions in order to track hands and arms above the surface through the correlation of the object and its shadows cast onto the surface. This is used to augment touch-tracking on the surface with the ability to point at objects with the finger, selecting them for further manipulation. Additionally, certain aerial gestures are available, such as waving a hand above the surface to scroll the entire canvas. This is achieved by alternating between the front- and rear-illumination, so that only one direction of light is turned on at any single point in time, ensuring that they do not interfere with each other during the separate tracking processes. A screen image is cast through rear-projection in the visible range, which does not exacerbate the tracker, as it operates in the near-infrared spectrum. The system is also able to infer simple 3D shapes by the shadows they cast onto the surface, enabling users to generate 3D models of objects that they place on the surface, which can then be manipulated with a vocabulary of multi-touch gestures.

Another application of such advanced tracking techniques is *TouchLight* [123], a touch-sensing surface that does not rely on a diffusor sheet for its tracking. By mounting two cameras with known relative positions behind a transparent sheet (which is technically optional, but useful as a haptic touch feedback) and computing the plane-to-plane homography between each video

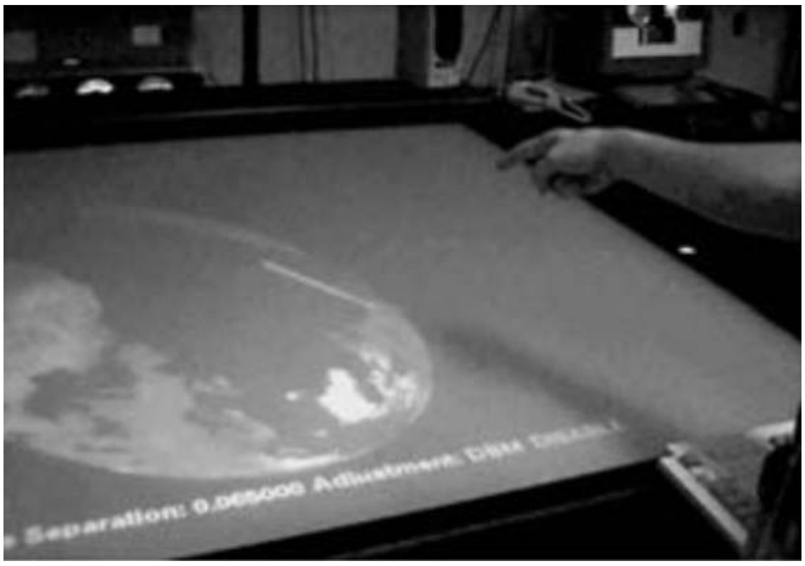

Figure 14: Terrain navigation with deictic gestures on the *Perceptive Workbench* [105]

feed and the targeted interaction surface, two images are created in which only those objects have the same pixel coordinates that fall onto the interaction surface. Thus, by extracting these objects, contact points on the interaction surface can be tracked. Rear-illumination is used to ensure that objects on the surface are bright enough, so that image segmentation by luminance thresholding is facilitated. Bright objects that do not fall onto the surface are offset to each other after the homography transformation due to the position disparity between the stereo cameras, so that they can be weeded out before tracking takes place. To determine taps on the surface, a microphone is used, whose sounds are matched to individual touch positions during tracking. This is necessary because the stereo depth image is not accurate enough to determine wether an object is touching the surface or hovering slightly above it. Since the system itself does not require a physical glass surface to operate, it can be left out to create a virtual surface. It is even imaginable to create a stack of such virtual surfaces by using multiple homographies. Another application is the usage of a special diffusor film, which only diffuses light approaching from a certain incident angle, while being completely transparent otherwise. By rigging a projector to cast an image onto such a surface from just the right angle, *TouchLight* can be used to create augmented shop windows or video conferencing systems in which eye-contact between participants can be kept, because the surface appears to be fully transparent while still showing the imagery cast by the projector. This is an interesting opportunity that hasn't yet been explored in detail.

Matsushita et al. have used a similar material to create *Lumisight* [74], a multi-touch enabled table-top system that can show different imagery to users at each of its four sides. While touch

Figure 15: Touch tracking with stereo cameras on *TouchLight* [123]

tracking itself is based on the rear-illumination technique, four layers of film, that diffuses light from a specific angle and is transparent otherwise, are placed onto the surface, each being driven by its own projector at its diffusing angle. Thus, depending on the position of the observer, the image from only one of the four projectos is visible. The aim of this setup is to explore possibilities to facilitate cooperative table-top work by showing a shared image (such as a map), but making certain objects, such as textual labels, appear at the correct orientation to users at each side of the table.

Also based on direct hand tracking is the *visual touchpad*, which uses stereo cameras to track hands and fingers on a black surface. By using a known color for the surface, segmentation is greatly facilitated, so that the images of the hands can be easily separated from the background. The extracted hand regions are then transformed to the viewpoint of the user, scaled and superimposed on the screen that the user is working with, thus providing for a non-direct mapping with the additional augmentation of actually seeing one's own hands on the screen itself. With this experiment, it is attempted to replicate the successful mapping between mouse and cursor, extending the paradigm to real and virtual hands. In an informal testing sessions, users found it very compelling to see their own hands on the screen, feeling as if they are directly manipulating on-screen objects, even though the mapping is actually indirect. However, since the scaling of the hands depends on the sizes of touch surface and screen, the hands may be rendered extremely large if the surface is small and the screen is big, resulting in severe occlusion problems. Consequently, future research should examine how this mapping can be modified so

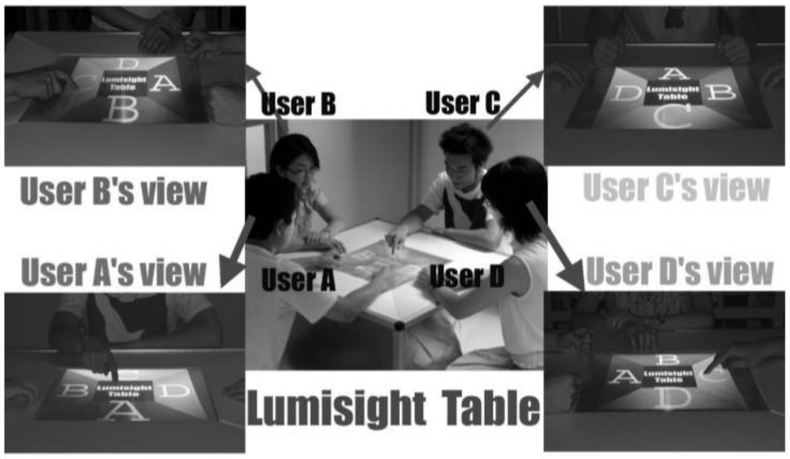

Figure 16: Per-user imagery on the *LumiSight* multi-touch table [74]

that the experience of directly-mapped input is preserved, without needing to resort to a 1:1 mapping between surface and screen corners.

Technological advances in multi-touch sensor technology are currently happening at a fast pace. While optical tracking solutions are currently the most interesting to scientists due to their versatility and robustness, hardware manufacturers are aiming to improve technologies conceived in the 1980s (such as capacitative touch surfaces) in order to bring larger, more accurate surfaces to the market. Consequently, a collection of technology samples such as presented hitherto is bound to become outdated very quickly, but the core concepts persist. The present and future will be abundant with many novel devices based on multi-touch sensing, shifting the heart of multi-touch research from integrated hardware solutions and tracking technology towards less technologically oriented topics such as multi-touch interaction design, so that the potential of these technological developments can be fully exploited.

3.3 Interaction Techniques

While multi-touch surfaces have been within the reach of researchers and designers for three decades, many hardware setups were engineered with a specific application in mind, resulting in a design of interactivity that caters to the need of a few selected use-cases, but does not necessarily provide potential for the generalization of interactivity paradigms into a vocabulary of metaphors for general multi-touch interaction. At the same time, many early attempts were focused on transferring successful metaphors from the realm of mouse interaction to touch surfaces, neglecting many of the obvious, but also more subtle differences in quality between the two interaction techniques. Additionally, multi-touch interaction provides a larger number of degrees of freedom, which allow for the modeling of interactions that were previously only possible by splitting them up into sequential actions [120], requiring extensive research to find suitable new mappings. Even more so, people tend to gather at surfaces within a collaborative work environment, such as having meetings at tables or utilizing whiteboards to illustrate their ideas [27, 119, 36, 35, 37]. Thus, the multi-touch interaction is often closely linked to multi-user interaction, requiring the designer to accomodate for the needs of both the individual and the group.

The following section aims to present research results relating to interaction on multi-touch surfaces from the perspective of the designer, rather than the hardware engineer, shedding light on the direction in which multi-touch research is moving and showcasing potential for further research, now that developments such as Touché [55] shift large multi-touch surfaces within the reach of the designer, rather than restricting them to engineers who are able to supply and build their own tracking technology. Being aware of concepts developed by other researchers serves as a viable starting point for further exploration.

3.3.1 Translation, Rotation & Scaling

Demo applications in which a user can translate, rotate and scale individual images on a multi-touch screen are a common sight at many contemporary technology fairs. Indeed, the ability to interact with digital documents that mimic the behavior of real-world documents, augmenting them with features that only digital artifice can provide, is an eye-catcher. Thus, translation, rotation and scaling have established themselves as a baseline that most, if not all, multi-touch surfaces support through gestural input. Nevertheless, many designers rely entirely on guesswork when they plan the way in detail how these interactions are supported [81].

In the 1980s, Evans et al. [28] introduced a collection of metaphors pertaining to the realization of a rotation operation on touch surfaces with different amounts of degrees of freedom. Their *turntable* metaphor is based on the idea of placing a virtual turntable under the object that is to be rotated, defined by its center of rotation and a rotation direction, as well as a rotation rate. On a surface capable of multi-touch interaction, one finger can be used to define the center of rotation, while a finger of the other hand specifies the rotation direction and speed by doing a directed motion somewhere else on the surface. Conversely, the *stirrer* metaphor defines the center point implicitly by letting the user perform the gesturing of circles on the object, with the direction of drawing defining the direction of rotation, and the angular speed defining the rate. The center of rotation is defined by the approximate center of the circles. Consequently, small, quickly drawn circles lead to a fast rotation, while large, gently drawn circles define a slower rotation. This behavior is akin to the way a cook uses a stirrer in a pot and how their motion influences the speed and direction at which the soup is moved within the pot. Another metaphor, more closely resembling traditional methods of specifying rotation, is the *rack*: Multiple rectangular regions are delimited on the surface, and finger motion towards the left or right (or, alternatively, up and down) defines the magnitude of a parameter, with the zero point resting in the middle. Parameters such as direction and speed of rotation can be derived from interaction within a single rectangle. Multiple rectangles can be used in order to allow for rotation and scaling at the same time. The anchor point within the object can either be fixed or also controllable by one or multiple rectangles. Thus, this metaphor is inspired by slider boards, such as those used to operate some machinery, but also early video games.

On multi-touch surfaces, interaction techniques perceived as more natural are especially desirable. Thus, the *stirrer* and *turntable* metaphors are those that appear to be most suitable. An early system from the field of tangible computing by Fitzmaurice et al. [31], which uses small bricks as interaction contact points, allows a user to place one brick onto an object to specify a rotation center, then use another brick to specify rotation direction and speed. This is done by measuring the changes of the angle between the line formed by connecting the two bricks and an axis of the object, using this displacement parameter to rotate the object. While this interaction is closely resembling the *turntable* metaphor, it was augmented by the ability to scale the object from the position denoted by the anchor brick by moving them closer (scale down) or farer away (scale up) to or from each other. By moving the center brick when no other brick is currently used for interaction, the object could be translated as well. Thus, the *turntable* metaphor is augmented into an interaction in which not only rotation, but also scaling and translation can be performed within a single interaction framework.

This metaphor has often been named a *stretchie* [112, 81]: Its core tenet is the idea of using two fingers (or hands) to define two control points on an object, where they “stick”, scaling, rotating and translating the object as the fingers move across the surface. At any given time, the fingertips are touching the same logical point on the object. If only a single finger is placed on the object, the object can only be translated by moving the finger, but neither rotated nor scaled [120]. This has become a de-facto standard operation that is implemented in many commercial multi-touch devices [5] [19]. Most often, either the index fingers of both hands are used to perform it, or the thumb and index finger of a single hand. While the power of being enabled to specify a complex affine transformation with a single gesture is intriguing, and the

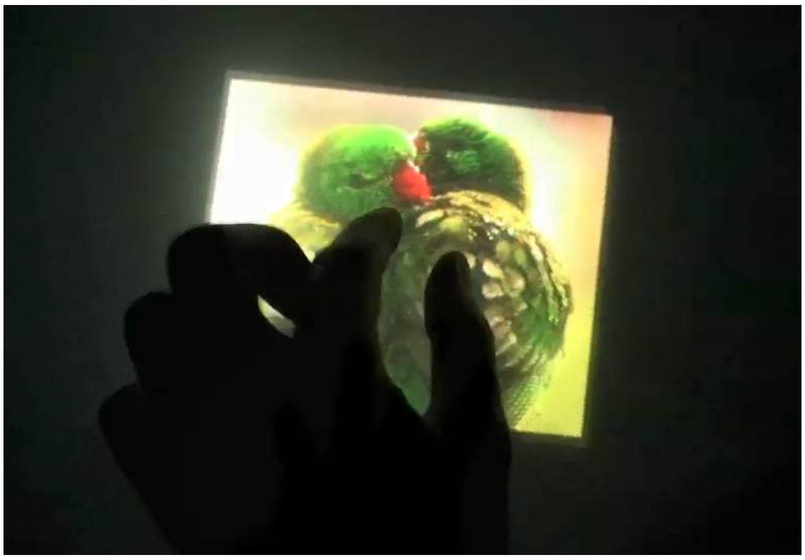
Figure 17: Performing a *stretchie* gesture with *Touché* [55]

metaphor seems to have been widely accepted by the industry, it is not without problems: For instance, some rotation angles (such as a 180 degrees or larger) require the user to cross their arms or to rotate their hand around the wrist in an awkward manner. Continuous rotation is not possible, since the user will need to pick new contact points on the fly. Also, the metaphor is not obvious or self-explanatory, making it hard for users without any previous experience with multi-touch systems to discover it without external help. Even upon accidental discovery, its exact semantics remain complex to comprehend. Finally, the metaphor is not perceived as natural in every case: For instance, when manipulating larger three-dimensional objects, users intuitively want to "grab" them with their hands, rather than using only two finger tips, with the version of using index finger and thumb of a single hand feeling particularly strange [97]. However, by designing an implementation to be more lenient about whether one or multiple fingers of a hand are used to specify a control point, this problem can be widely alleviated.

Consequently, techniques besides *stretchies* are being examined, despite them already being largely accepted by the industry. One approach is Rotate'N'Translate (RNT) [62], which uses only a single contact point to specify rotation and translation in an intuitive manner: When moving a piece of paper on a table by placing a single finger on it, one will notice that the paper will rotate around the point of contact as well, due to friction between paper and table. Rotating the paper while translating it this way is surprisingly easy, because it harnesses our knowledge and skill acquired through previous experience. RNT replicates this behavior by using the relative position of a single touch point on a digital object, the speed and direction of motion as well as simulated friction in order to obtain a rotation angle from a translation vector.

Thus, the user can manipulate a digital object much in the same way as paper on a table. The simulated friction behavior is perceived as very natural by users, and many discover the way how they can rotate objects intuitively by themselves, because touching a digital object and dragging the finger around, observing the induced behavior, is something that inexperienced users tend to do on touch-sensitive surfaces. Also, continuous rotation is possible by moving the finger in a tiny circle. These are large advantages over the *stretchies* metaphor.

It is notable that RNT's friction model is not a simulation of real-world physics, but an intentional simplification. As an example, a real piece of paper might rotate out of control if the translation happens very fast and momentum is thusly large. RNT's model removes such effects, which are not deemed beneficial to the interaction, since they are not needed to make the metaphor clear, but would be perceived as annoying instead. This is akin to the idea of "comic physics" [78], where certain laws of physics are intentionally toned down or exaggerated in order to make a narrative point. Agarawala et al. [2] note that such an understanding of physics in relation to multi-touch interaction design can be a starting point for the development of metaphors that are intuitive, since they refer to knowledge of the real world that we have acquired during our lifetimes, but stripped of any peculiarities that might make the interaction unnecessarily difficult or require large amounts of skill to perform in an elegant manner.

While RNT requires only a single contact point to perform, it does not mean that the metaphor does not fit into the concept of multi-touch surfaces: The ability of a surface to sense concurrent contact points does not imply that every interaction has to be realized in a multi-point form. Rather than that, interactions such as an RNT can be seen as a building block, which can be augmented with additional functionality where it is needed, because other fingers and hands remain still available for mapping. For instance, in the manipulation of 3D spaces, RNT can be used for rotation and translation, while another finger can be mapped to specify a scaling factor (with the RNT touch point as a center), or it can be used to specify yaw and pitch [40]. Furthermore, RNT can be extended in a multi-point way, where a second finger can be used to abruptly stop the rotation caused by the simulated momentum, thus "holding" the object in place.

Hancock et al. [42] have compared different methods of rotating digital objects on a touch-enabled table-top system. Their motivation stems from the recognition that the ability to rotate objects such as images and text in order to show them to users standing at different sides of the table-top system is crucial to the communicative process of the system's multiple users. They considered five conceptual rotation metaphors that are in use within touch-enabled software and software for mouse and keyboard interfaces, but the experiments were conducted with touch implementations: *explicit specification* refers to the typing of numeric rotation angles in a dedicated widget, *independent specification* is the employment of a designated slider-like area within which the user can move a finger to specify a rotation angle (thus, very similar to Fitzmaurice et al.'s *rack* metaphor [31]), *automatic orientation*, e.g. the automatic snapping of an object so that it faces towards the one edge of the table-top to which it is closest, *integral rotation and translation*, which refers to the RNT technique, and *two-point rotation*, which denotes the *stretchie* approach. In their findings, explicit specification performed poorly, because users found it hard to translate the orientation they had in mind into an explicit angle to type into the widget. With automatic orientation, users liked that it is very predictable and easy to understand, but they also complained that subtle changes of orientation as communication cues were not possible – that is, the rigid restriction to rotational angles was perceived as incompleteness of the approach. With RNT, users found the task simple and very natural, but disliked the inconsistency that identical paths can lead to entirely different rotational angles if the speed or acceleration of the motion differs. Thus, it was frequently necessary to readjust after the initial attempt to achieve a certain orientation, costing time, but at the same time feeling very much like an interaction in the real world. Finally, two-point rotation was also perceived as natural, but might require awkward motions to achieve some orientations. Additionally, users found it quite hard to maintain a constant spacing between the two fingers, more so than on real paper, because there is no friction between the fingers and the digital object. Since the implementation at hand mapped the relative distance of the two touch positions to a scaling operation, users found themselves inadvertently scaling the object quite often.

Thus, there are multiple metaphors about how rotation, translation and scaling can be mapped on a multi-touch system. While the *stretchie* metaphor is currently the most widely used one, it is not without any issues. Alternative metaphors such as RNT prove themselves to be worthy of consideration for newer systems, but work best on larger surfaces, where it is easy to gain enough motion momentum to drive the simplified physics engine and where occlusion and space are of little concern. Two-point rotation is more space-efficient and thus very suitable for smaller surface sizes, but it is hard to discover on one's own without the help of external help books or advice. Other techniques such as automatic orientation towards given points of interest can be valuable too, especially in multi-user environments, where the main intention behind rotating an object might be to bring it into proper orientation to show it to another participant. For the designer, it is important to be aware of the different metaphors, so that they can be chosen carefully according to the needs of a particular implementation and use-case.

3.3.2 Occlusion & Menu Handling

Finding adequate interaction techniques for selection and menu behavior on directly mapped multi-touch surfaces is not a trivial task: On large surfaces, there may be a lot of objects, some out of reach for a user depending on where they are standing, some being occluded by one's own body or other users with which the surface is shared, and sometimes, the surface might be cluttered with objects that do not pertain to the interaction, such as the ubiquitous coffee mugs. While most commercially available systems only let users select objects by touching or tapping them with a finger and display menus that behave in a manner quite similar to what we are used to from mouse-based interaction, there is a vast amount of ongoing research into different metaphors that are fitter for the peculiarities of touch surfaces.

For instance, Shen et al. [100] question the viability of using a single finger to tap objects that a user wants to select for further interaction: While this is certainly the most obvious and possibly natural implementation, it suffers from the problems of the finger tip being a rather large interaction instrument, making it very difficult to accurately select small objects on a surface. Even more so, users will frequently occlude the objects they want to select with their hands in the process, ironically conflicting with the actual intention of selecting them. As an alternative, they suggest an interaction where two fingers are placed on the surface, and a virtual cursor appears at the midpoint between the two contact points. Thus, the cursor itself is not occluded a finger, making it possible to define very small cursor motions in an easy manner. They suggest to use the index and ring finger of a single hand to control this cursor, because this leaves the middle finger free to tap the surface in order to select an object or interact with it. This is very similar to the way a mouse is used, so that the rather complex interaction technique is perceived as something very familiar by many users, while elegantly circumventing the issues of finger occlusion and cursor coarseness.

Indeed, taking occlusion into consideration is a necessity for direct input methods, that designers more familiar with indirect input might not have much experience with. With touch surfaces, being aware of which hand is used to interact with an object is crucial to make informed predictions about expected occlusion patterns. However, assuming that a user will only interact with their dominant hand at every time is too simplistic, since the position of an object or the intended interaction might very likely lead to the usage of the non-dominant hand as well. Therefore, Hancock et al. [41] have experimented with different methods to determine the used hand from touch points alone, since most hardware setups are not able of true 2.5D sensing. Simple heuristics based on position and incident angle of a finger are prone to failure, but more advanced methods utilizing neural networks or Bayesian models can provide high accuracy in correctly determining handedness. Based on the position of the whole hand rather than just the actually sensed touch positions, visual feedback can be given in a way that occlusion from the hand is removed or reduced heavily. Even more so, users tend to become more serene in using a touch screen when they realize that they hardly run the risk of occluding important objects and feedback inadvertently, relaxing them up to the point where they feel comfortable to rest their hand on the screen while operating on a small object or menu, which helps to reduce arm fatigue during lengthy interactions on the touch surface. Thus, the reduction of occlusion problems is not only important from the perspective of efficiency, but also overall user experience.

Nevertheless, even if hand occlusion can be avoided to a certain degree, the selection of very small objects within a large group or those overlapped by other digital imagery remains a considerable issue. While this problem remains largely unresolved and can only be alleviated by letting the user manually dig out the object they want to select and maybe use a *stretchie* gesture to enlarge the canvas and make small objects more accessible, the design of smart techniques to aid in the selection of such objects could be a great boon to multi-touch interaction. A fascinating idea is the combination of touch input with voice commands to facilitate and clarify selection intent [111]: Originally conceived for the simplification of ray-casting methods in virtual and augmented reality systems, the core principle lends itself to the application to touch surfaces as well. Voice commands are used as filters, whereas actual selection still happens by touch. For instance, when presented with a group of small objects, a user can speak a property of the object they are interested in, e.g. "The green one", and the system will filter out all objects that do not conform to this property. Remaining objects can now be momentarily enlarged or otherwise marked, so that it becomes easier to select them with the finger. The elegance of the approach is that the decidedly error-prone voice commands only serve as a filter, but do not perform a selection themselves. Consequently, a misunderstood voice command cannot lead to a wrong selection, but in the ideal case, selection is considerably facilitated by the correct recognition and application of a voice command filter.

Issues such as occlusion and cursor coarseness play an especially important role in the design of menu behavior: Menu items that are large enough to be selected easily, but at the same time do not run the risk of being partly occluded by a hand while space is potentially limited, might often require a hierarchical approach, so that the amount of items does not need to be accommodated for by making individual items small or placed at positions where they are prone to being occluded by a hand. Rounded menu items that are operated according to the previously introduced *stirrer* metaphor [28] are popular approaches to unify these conflicting requirements [103]: The menu itself is opened at a position where it is currently not occluded by a hand. Items are arranged around a circular path and are selected by performing a circular motion inside the bounds of the menu, as if to turn it. As the menu spins, different menu items roll into place, providing for a potentially unlimited number of options. When the position at which new items shift into the circle is chosen carefully, the arm occlusion when "turning" the menu is not an issue. More important items are positioned towards the beginning of the sequence in order to be more accessible. Nevertheless, since the order of items is fixed, users can develop a feeling for the position of various items, so that selection (or at least a pre-selection) can be accomplished in an eyes-free manner by drawing the learned curve necessary to bring the intended item into view. While the core principle of operation according to the *stirrer* metaphor remains the same, there are many variations to this approach, relating to subtle adjustments in the behavior to emphasize specific properties that are beneficial to the application at hand.

However, such marking menus, while powerful in their expressivity, are tedious to operate, especially if a lot of different menu items are available. Developing a feeling for the distance one has to "stir" in is only partly a solution, since it is very difficult to become truly skillful, due to varying sizes of stirring motions that are hard to keep constant. Strictly hierarchical static menus such as FlowMenu [34] offer significant advantages in this respect: When opened, a FlowMenu represents its items like a pie chart, where the size of each segment is equal. Thus, the less items appear in a single hierarchy level, the larger the selection area for each. When an item is selected, it is possible to open another menu with its center point at the selection point. An item is deemed selected if the touch remains static over the item, if there is a sharp change in angular finger motion or if the finger leaves the FlowMenu at the border of the item. A new hierarchy level appears immediately without any waiting time. Combined with the selection behavior, this means that a multi-level selection can be performed in one smooth gesture by advanced users, who resort to drawing a path they have learned by becoming familiar with the layout of the different hierarchy levels, which remains constant under every condition. Consequently, FlowMenus are highly usable for both novices, who can navigate through hierarchical menus one by one, looking for the item they want to select, but also for expert users more concerned with speed of execution, who can draw a single, quick gesture to select an item: The mental model of the menu hierarchy layout serves as a metaphor to remember this gesture, which, when seen separate from the underlying menu, would be very hard to remember on its own. In the particular case of FlowMenu, each hierarchy level consists of exactly eight items (of which some segments might remain empty) in order to facilitate this learning effect. However, other researchers experiment with relaxing this condition.

While such menus perform very well on clean touch surfaces and be combined with hand posture

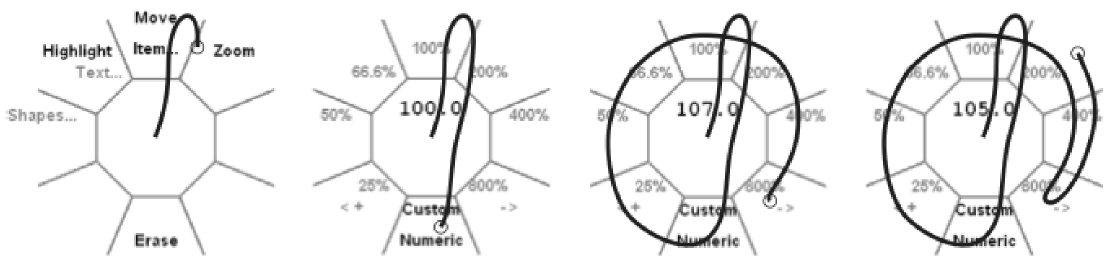

Figure 18: Navigating a *FlowMenu* [34]

detection or prediction in order to minimize issues with occlusion, they are only partly suitable for surfaces that are cluttered with other objects, such as paper documents and coffee mugs. This is because even if the system has facilities to detect such objects and recognize their position, it is extremely hard to design algorithms to find an optimal menu position under a complex set of positional constraints, and if a decidedly “good” position in proximity to the user’s focus while evading all obstacles does even exist. Consequently, a menu might have to appear in a position rather disconnected from the user’s focus. Leithinger et al. [68] have developed a pragmatic solution to the issue of cluttered desktops: It is inspired by fan-out and card-deck menus that have long been popular in pen-based tablet systems: With such menus, the first item appears when the pen touches the surface, then the rest of the items is fanned out along a straight line between the position of the first item and the current position of the pen, much like a magician might fan out a deck of cards. Rather than constricting the fan-out behavior to straight lines, the new approach allows for arbitrary paths drawn by the user, along which individual items will appear, with their textual orientation always in parallel to the horizontal edge of the screen. Thus, the complex issue of finding suitable positions for menu items on heavily cluttered surfaces is left to the user, who will in turn not be confused by an awkward layout that an algorithm might have chosen. In user tests, the approach has been well received. The main disadvantage of a user-chosen menu layout is that an additional working step is necessary for the user, who will actually have to define the layout, which might be perceived as annoying, especially if the surface is currently rather empty, so that an automatic approach might yield acceptable results. There has been little research yet of how user-specified menu layouts can gracefully degenerate to automatically-positioned ones when the surface is clean.

While such menus yield good results for the selection of absolute values or menu items, they are unnecessarily complex for the selection of relative change, such as when increasing or decreasing a hue or saturation value in a touch-enabled drawing program. An alternative to radial absolute

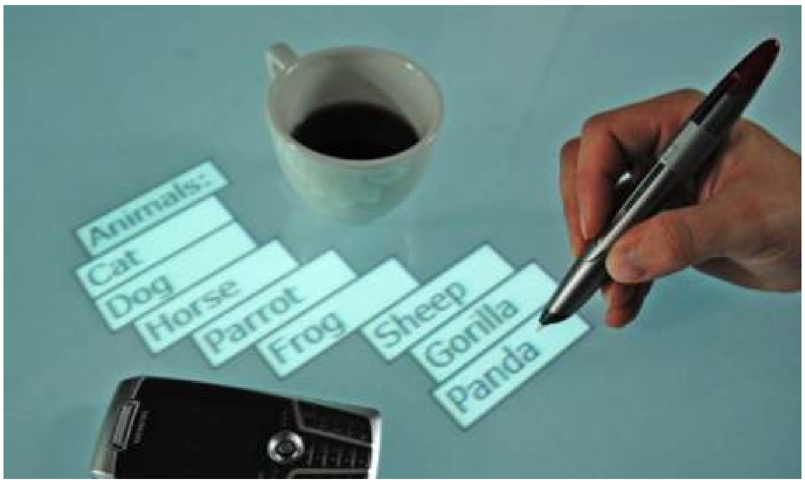

Figure 19: User-drawn path menus [68]

selection as in the previously introduced methods, approaches such as *CurveDial* [102] offer a better user experience for relative manipulation: Rather than using the relative position changes between a menu center point and the finger as the criterium with which a selection is made, curve dial uses curvature as the deciding feature. Thus, it removes the inherent postulation of the *stirring* metaphor that the stirring motion must be performed around a fixed center point, but each circle can instead have an arbitrary center, since curvature motion and direction are not affected by the center. Consequently, the selection is eyes-free, as the user does not need to care about stirring around the same center at all times, which allows them to watch the feedback of their relative changes, rather than paying attention to keeping the center constant.

Another common interaction technique is to interact through the combination of different objects on the screen, such as when deleting a file from the desktop by dragging it into a trash bin. Sometimes during such a drag-and-drop operation, multiple actions might be available that are usually presented through a pop-up or spring-loaded menu in common graphical interfaces. However, such an approach poses inherent problems when transfered to touch-sensitive surfaces: On one hand, the initiation of a "drag" operation might not always be clear if the system also allows for the arbitrary translation of individual items, which is an extremely common property of multi-touch interfaces: It is not trivial to make clear the difference between stacking two items on top of each other and dragging one item onto another. Additionally, a dragging operation can be difficult to perform if it has to cross large distances, as the finger needs to be in contact with the surface at all times. Sometimes, the surface might even be so large that such a motion is impossible at all due to physical constraints.

There are different approaches to transferring the dragging metaphor to touch screens. One approach is to separate it into a “pick up” and a “drop down” step, as in the *Pick-and-Drop* approach [93]: Originally designed with pen interaction in mind, a user can pick up an object with a touch. The object then disappears and is associated with the user’s finger. Touching another object will drop the stored object again, triggering an operation. By using explanatory animations, such as the object sliding in and out of the user’s finger, the operation can be made clear and, to a certain degree, self-explanatory. With Pick-and-Drop, large dragging distances are no longer an issue, as there is no need to keep the finger on the surface during the operation. Also, the operation is clearly different from a normal translation. However, the metaphor is more suited to pen input than touch interaction, because a mental model of “storing” the object within the pen, which is perceived as another piece of technology, feels more natural than sucking up an object into one’s own finger.

A conceptually different approach is to bridge distances by bringing objects closer to each other, rather than making it easier to bridge distances manually. *Drag-and-Pop* [10] is a good example of such a technique: When the user begins to translate an object, faint images of all other objects in the direction in which the user moves the selected object appear in close vicinity, being visually connected to the original item positions by a rubber-band animation. For instance, if the user translates an object towards a trash bin, a visual stub of the bin appears right next to the current object position, so that the actual translation motion can be very short. If the user drops (“pops”) the item into such a stub or back onto the canvas, the stubs instantly disappear again with their rubber-band animation. The rubber-band serves to signal the connection between an object and its stub. With Drag-and-Pop, stubs are only created for items which are compatible with the currently selected one, e.g. for which an interaction is defined. *Drag-and-Pick* extends this behavior to item selection as well: Here, a dragging operation is performed on empty screen space, and stubs are created for *all* items in the direction of movement. Selecting (“picking”) such a stub will perform the same operation as opening the item directly. Thus, the metaphor becomes applicable not only as a replacement for contextual menus in the “pop” version, but also for item selection in the “pick” version. Since the connection between item and stub is made clear through the use of the rubber-band, the behavior is easily understood.

Concepts such as *CoR2Ds* (Context Rooted Rotatable Draggables) [99] extend the scope of Drag-and-Pick as well as Drag-and-Pop to menus: Here, entire menus are opened in a similar manner as item stubs, but they remain on screen until they are explicitly closed by the user. These menus can itself be interacted with, making it possible to scale, reorient and translate them on-the-fly. A rubber-band imagery is again used to signal the connection between a CoR2D and its originating object. CoR2Ds can be different for different items: For instance, a

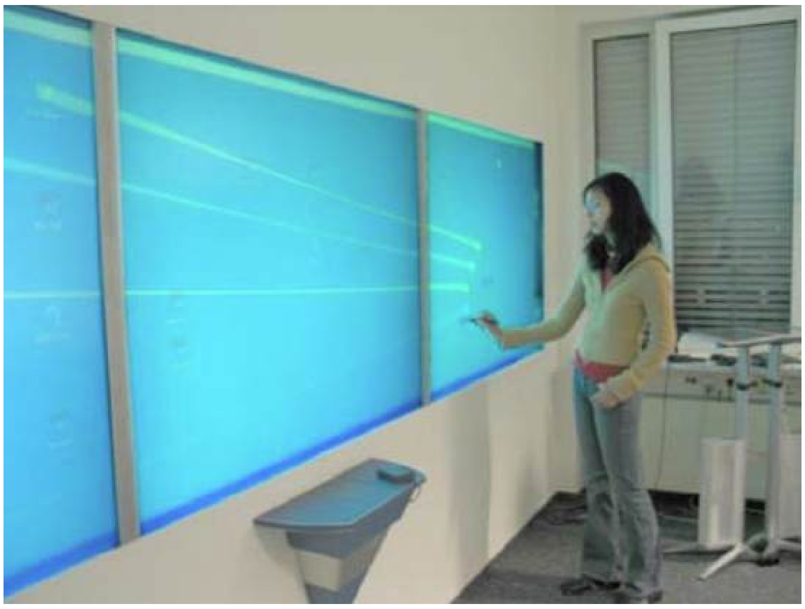

Figure 20: Action selection with *Drag and Pop* [10]

CoR2D created from an item that allows for complex interaction, such as an application menu bar, could be structured like a FlowMenu. A less complex CoR2D such as that of the trash bin can be a simple collection of a few menu items. A CoR2D of an image might contain widgets to alter image parameters such as hue, saturation or opacity. Thus, CoR2Ds are more of a concept than a specific type of menus. Their core tenet is that they allow users to open multiple menus associated with distant objects without travelling long distances, but then be able to rearrange these menus to their current likings as well. It is even possible that a CoR2D could spawn a child-CoR2D, such as a text manipulation one spawning a font selection one. Since CoR2Ds can be repositioned at will, a user can share a CoR2D with another user by placing it at a suitable point on the surface. Thus, CoR2D can be seen as a powerful concept from which a designer can take inspiration for the implementation of a concrete interaction.

3.3.3 Managing Complexity

Multi-touch capable surfaces can potentially be very large. Especially in collaborative settings, they will often serve as an externalized visual memory [100], where users might place a large number of digital items, which will need to be managed in a meaningful way, so that there can be a smooth transition between those objects currently in the center of attention and those that are not. The immediacy and naturalness of such systems would be negatively influenced by the introduction of data management techniques such as folders, tabs or other means of hiding

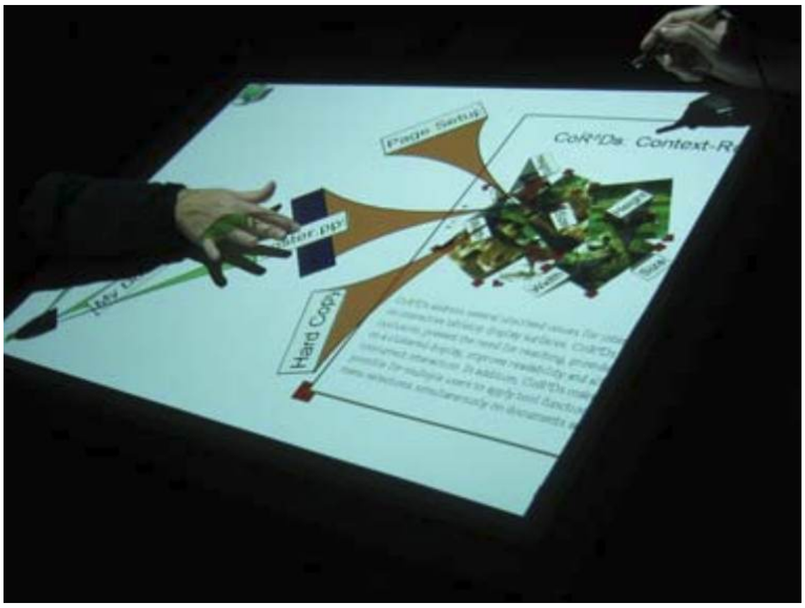

Figure 21: Interaction with *Cor2Ds* [99]

unimportant data behind artificial boundaries. Thus, multi-touch designers are looking into ways to manage such complexity without resorting to methods that do not fit into a multi-touch surface's perceived naturalness.

An early vision of how such a system may work is *PAD* [88]. At its core, PAD refers to a seemingly infinite desktop on which large quantities of information are stored in an unordered manner. Individual information items are brought into the center of attention through so-called *portals*, which refer to specialized views on the desktop. The underlying metaphor is that of a magical magnifying glass – a highly biased view that can peek at specific portions of the desktop and roam into different areas, while highlighting certain aspects of the data underneath. A portal might offer a highly magnified view, or it might be a rather panoramic overview. Additionally, an object might look quite differently when viewed through different portals. For instance, there may be a portal that displays all tabular data as bar charts, but leaves other data entirely unchanged. Portals can also look recursively into each other: There can be portals that zoom in a particular data set at large, but various other portals can be overlayed to explore the data at an even finger granularity. Portals can be arranged, reoriented and resized at will. A portal can also serve as a control modifier: For example, there might be a "color" portal that can change the color of underlying objects at the press of a button. Another important aspect is that each object can have a visibility range, so that they only become visible when a certain zoom level is reached, and a transparency range, so that instead of disappearing at a certain zoom level, it can remain visible, but at a lower opacity. Such

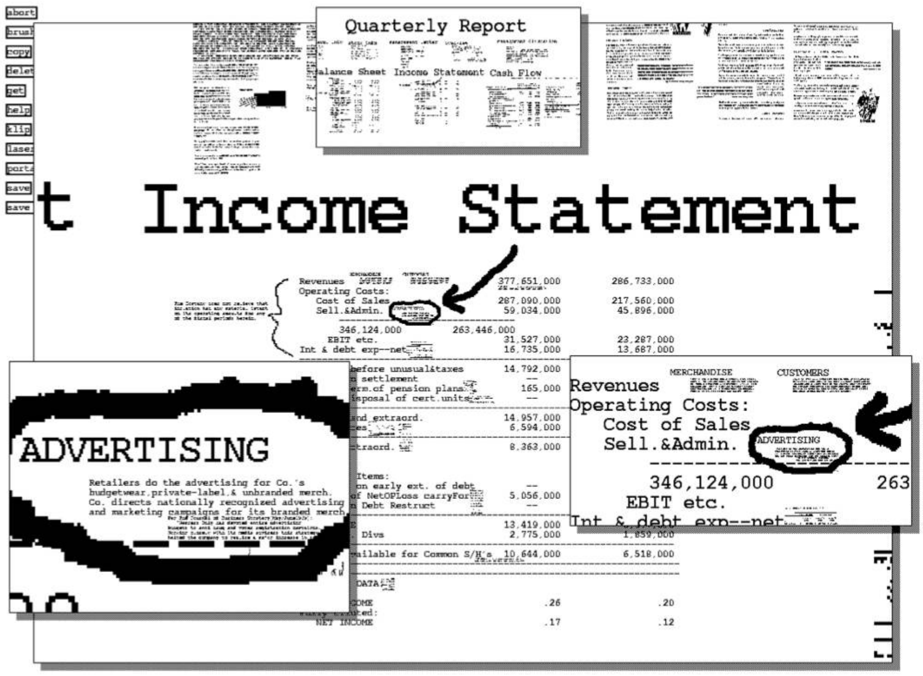

Figure 22: Portal-based interaction with *PAD* [88]

portals can serve to group data by importance, while detailed information is still available by zooming in. Similarly, text can be made semantically zoomable: At a low level of magnification, a text may just be displayed as a headline, at a higher level, it might appear as an abstract, before it turns into the full text at the highest magnification level. Similarly, text may appear as hyperlinked under certain portals, and selecting a link can cause the portal to shift to a new position, revealing the linked object. Multiple users can work concurrently within their own portals. There can be private portals, so that objects created within them are only visible through this one portal, but not others. Communication can be facilitated by offering portals that show what other users are currently seeing and doing.

The PAD model is meant to exploit the visual memory of places to help in the organization of informationally large workspaces in a highly natural manner, which is conceptually easy to understand. While the PAD vision spares out many of the details that would be relevant for an actual implementation and is thus to be seen as a vision rather than a proposal, it is notable that software such as Microsoft's PhotoSynth [21] has been implemented on the company's "Surface" multi-touch table-top [19]. While Photosynth does not rely on portals, it implements the idea of having a large visual workspace into which a user can zoom, revealing entirely new information as the zoom level increases. For instance, it is possible to view every page of a book at once, then zoom in until individual pages become discernable, then finally until the text on a page can be read.

Inspired by visions such as PAD, *Toolglass* [12] is another project that adheres to the conceptual framework of the PAD approach. While the canvas itself it not infinite, the Toolglass approach relies on transparent toolglass lenses as core widget concept, which, when moved over an object on the screen, reveal a specific information or provide a specific operation. For instance, in a 3D modeling application, there can be a toolglass to display an object as a wireframe, another might display it with a given texture, and so on. Since toolglasses can be freely moved, reoriented and scaled, it is very well possible that only a part of the object is affected: Toolglasses are not to be understood as modifiers that, when placed over an object, affect it in its entirety, but according to the metaphor of a lens, so that only the part of the object visible through the lens is modified. Naturally, toolglasses can also overlap in order to combine their effects. A toolglass can also serve as an active item: For instance, a toolglass might represent a shape that can be "pushed through" onto the object below it, much like a stamp. The individual toolglasses itself are organized on palettes, which users can select from a menu in order to move them onto the canvas. These palettes can also be customized by the user, so that sets of tools for specific tasks can be created. The core idea behind Toolglass is to replace temporal modes of operation with spatial modes: Rather than making certain tools or views become the current one, requiring a mode-switch to change them, they can temporally coexist and are delimited spatially through the individual toolglasses. This is akin to similar approaches in the field of tangible computing, where certain physical objects are often used as modifiers for digital data onto which they are placed [113]. On touch-enabled surfaces, toolglasses are especially natural to use, due to the intuitive way in which they can be moved, reoriented and scaled, closely resembling the behavior of tangible objects.

While *Toolglass* is akin to the portals of PAD in the sense of a filter or "magical view", Kahn et al. [60] have realized a PAD portal's aspect of spatial bridging: Their system is aimed at facilitating the concurrent operation on objects that are in great physical distance. The core concepts are those of a "telescope" and a "target": The target, represented as a red circle, can be moved and scaled freely within the entire surface. It is meant to be placed at a certain point of interest, with which a user wants to interact remotely. The telescope, a blue circle, replicates the contents of the target, but can also be moved freely across the canvas. When the telescope is rescaled, its contents remain the same. Thus, if the telescope circle is smaller than the target, the contents are scaled down accordingly, and vice versa. However, the inside of the telescope behaves in exactly the same way as the rest of the canvas: Objects from outside and within the telescope can be freely intermixed during an interaction, and every manipulation within the telescope is immediately replicated within the target. Thus, the relation between telescope and target is akin to a wormhole, which offers bi-directional access to physically remote objects. This approach can afford for different paradigms of operations: For instance, it can simply be used to intermix remote and local objects in an interaction. It can also be used by multiple users to share the same part of the workspace (by overlapping the targets), while still retaining their own interaction space. However, if the canvas is shown at a small zoom level, the telescope can also be used as a magnifying glass, while the target specifies the region of the canvas which should be zoomed in. Since the behavior and interrelation of telescope and target remains consistent under all circumstances, the approach is perceived to be quite intuitive. It is also possible to allow for multiple targets and telescopes at the same time, opening up another dimension of possible applications.

While the metaphor of unlimited zooming and filtering is certainly elegant and also quite intuitive, it is also abstract to a strong degree, making it difficult to design systems around it that are self-revealing and self-exploratory: While the interaction itself is natural, the metaphor is not. Some researchers are therefore looking into ways of managing complexity that are closely modeled after real-world examples, up to the point where the interface assumes traits of being a simulation, rather than an abstraction.

A well-known development along this perspective are simulated 3D desktops, a technique which is referred to as "shallow-depth 3D interaction", because even though a third spatial dimension is present in the interaction, it is much less pronounced than the other dimensions. A common example is a simulated 3D desktop termed *BumpTop* [40], which is rendered as a table-top on which individual files appear as stacks of paper sheets. Individual stacks respond to touch, behaving very much like a stack would in the real world. As an example, a physics engine is used to simulate the response of a stack to the force exerted by a finger. Moving a stack is implemented with RNT [62], whose physics-based approach fits in with the employment of an entire interface being governed by a physics engine. For instance, when the touch is released after a lot of kinetic force has been applied to a stack, the stack will not stop gracefully as in a usual RNT implementation, but will instead slide and tumble across the surface, until it is repelled by another object or finally comes to rest after losing its momentum through friction. Little walls at the edges of the simulated desktop delimit the working area, so that objects controlled by the physics engine cannot fall off, but have a believable reason for staying on the surface.

Interaction is partly achieved by applying physics-based effects through the translation of individual objects, such as placing several items over each other to form a document stack, much

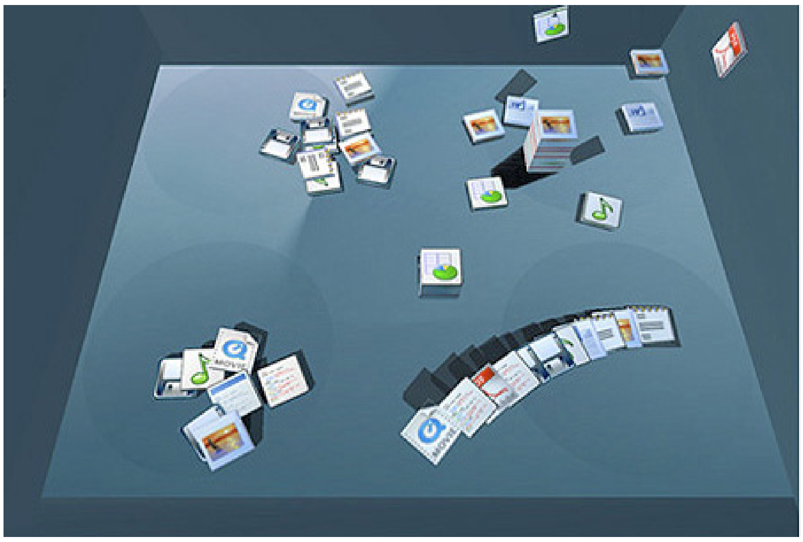

Figure 23: Shallow 3D interaction on the *BumpTop* [40]

like one would stack up documents on a real desk, but also partly through augmented operators accessed via contextual menus, such as the ability to fan out a stack to see its individual items. Consequently, the implementation aims to harness our natural understanding of a surface governed by the laws of physics, but intentionally breaks this metaphor with some operations in order to become usable. However, it is notable that these operations are intuitively clear, because they are illustrated by animations that make their relation to the underlying metaphor graspable. Thus, a very large amount of data can be handled by a user by applying knowledge about the spatial organization of many documents on a real table on one hand, but augmenting this skill further through the availability of powerful operators that can only be available on a digital system, such as fanning out document stacks, sorting them by certain criteria, and similar operations.

The application of real world traits is extended to the design of the objects themselves. For instance, text documents are rendered as an actual stack of papers, with the first page clearly visible (and even readable at a sufficient zoom factor). A special operator is used to browse through a document's individual pages, applying a graphical page-turn transition reminiscent of a book. The thickness of a stack signals the size of the file, with a large text document being significantly thicker than a single photograph. It is even possible to apply graphical filters to older files in order to make them appear aged, with torn, yellowed pages, rather than the pristine look applied to newly created files.

Such shallow-depth 3D table-tops can be particularly interesting for purposes of edutainment, where the presentation of potentially large amounts of information in an aesthetically and experientially engaging way is of utmost importance. In the *Virtual Pond* [104], a shallow pond is simulated on a touch surface, showcasing stunning visual water effects as well as a rich audio experience. Various pond-dwellers such as fish, scallops and weeds can be selected with a touch, so that they float to the surface of the pond and can be manipulated further, eventually revealing additional information such as textual descriptions, photographs and videos. This information can be freely distributed on the water surface. An objects can be pushed back into the pond at any time, so that it will sink gracefully back to the floor. Thus, the screen surface is parted into two z-axis levels, the lower one (the bottom of the pond) containing opaque objects that can be brought to the higher level (the surface of the pond), where detail information can be accessed. This is akin to the magnification-based approach, but the metaphor is entirely different: While magnification and the different appearance of objects at certain magnification levels is an abstract concept, the metaphor of pulling objects out of a pond to examine them is perceived as natural, clear and intuitive. An interface can benefit greatly by taking inspiration from real-world interactions that are simulated as a holistic experience, including visual effects, audio and physics models.

It should not be neglected that auditory feedback can assist in the managing of complex datasets on a touch surface as well. Shen et al. [43] note the distinction of auditory feedback on touch interfaces into *proactive* sounds, those that are played before a user performs an action, *reactive* sounds, auditory feedback which can either be positive or negative, and *ambient* sounds, carrying those informations that are only at the periphery of a user's attention. The aim is not necessarily to introduce entirely new layers of information through sound, but to increase individual or group awareness of the state of the system. For instance, proactive sounds can reveal the availability of operations that is not immediately clear through visuals alone. Reactive sounds can serve to signal the finishing of an operation, even though it should be noted that negative feedback through sound is often perceived as annoying and should thus be used only with great care. An especially interesting function pertains to ambient sound effects, which are not directly connected with a given operation or action, but indirectly provide clues about the state of the system in a calm way. For instance, a touch screen that loads images from the internet could use subtle changes in ambient sound to signal that it is waiting for a network response, so that the user is made aware of networking delays without grabbing the center of their attention. Consequently, ambient sounds are characterized by carrying information through subtle changes, rather than the sudden starting and stopping of pro- or reactive sounds.

The key to the successful management of complex datasets with touch systems is to strike a balance between the naturalness that a touch-based system is capable of, and the various layers of abstraction that need to be harnessed in order to find ways to navigate a large data entity. Shallow-depth 3D interfaces, magnification levels and audio cues are only examples of successful approaches, but it is very hard to generalize them into paradigms that remain applicable to a vast set of different applications. However, since actual implementations of complex touch applications are rare, few examples are available, and the field remains an active focus of research.

3.3.4 Multi-User Interaction

Multi-touch capable surfaces lend themselves to the creation of multi-user devices, as their is no technological limitation to prevent multiple concurrent users on the same surface. Additionally, large surfaces such as tables or whiteboards have traditionally been used as gathering points for collaborative work [27, 35, 113], fueling the need to integrate such practice with digital technology, for which natural interfaces such as multi-touch surfaces can serve as a catalyst. However, multi-user systems bring a whole new set of issues into the discussion, such as the distinction between public and private working areas, the coordination of multiple users and the role of touch surfaces within user-to-user communication.

Only few touch technologies, such as *Diamond Touch* [24], provide the ability to distinguish touches by different users on the tracking level. While such information is certainly valuable, it is not crucial to design surfaces that can support multiple concurrent users.

A particular feature of collaborative working sessions are the alternating phases of private work and public discussion [35, 37]. Traditional tools such as paper sketchbooks and laptops naturally provide for the ability to work privately without distraction by other users. For instance, a paper sketchbook can be tilted towards the user for private work, excluding the work from prying eyes, until the sketchbook is placed in the center of the table, so that everyone can see it. Similarly, a notebook computer can be used for private work, which can at a later stage be presented publicly via a projector. For a touch surface, it is desirable to allow for the transferring of such practices.

On an interface like the previously discussed *PAD* [88], which is based on so-called "portals" to reveal objects on the surface, the distinction between private and public objects is made through the use of a specialized portal: All objects created within this portal are only visible through this portal until the user decides to publicize them. Thus, every user is given such a private portal, which they can freely move across the shared touch surface to define their private working space. It is even possible to leave unfinished work behind at a different position, being hidden until the owner decides to publicize it. Thus, the usage of such a private portal is an elegant way to achieve the desired distinction between private and public work, but it is difficult to generalize: On an interface which is not based on portals, the introduction of such a “lens” constitutes an entirely new metaphor that is not immediately obvious and might not fit in with the underlying metaphors of the remaining interface.

A similar approach is used in some touch surfaces in conjunction with tangible computing [113], where the user can place a certain tangible artifact onto a piece of work to lock out interaction by others: While the presence of this artifact does not prevent users from accessing the object in a physical manner (e.g., they could simply remove the artifact), it serves as a visible reminder of the underlying object’s privacy, appealing to other users’ respect of social norms of property and ownership. While touch surfaces do not generally support the tracking of non-finger objects, a visual cue on the screen could replace the tangible artifact to achieve a similar effect: If all users are aware of the fact that the presence of this clue means that an object is not public, the approach is viable. However, the visual cue itself could be distracting during work with the object. Additionally, since an explicit operation is necessary to trigger the transition between public and private work, users might forget to do so or intentionally omit it to resort to verbal negotiation instead.

A radical approach is to provide separate screens for private work [101]. For instance, users could still be able to perform private work on their own notebook computers, whereas the large touch surface is only used for public work and discussion. Elegant ways to move digital information between the private and the public screen make this approach viable. A common technique to achieve such a transition is *hyper-dragging*, which extends drag-and-drop to multiple screens, so that a cursor leaving a screen at a pre-defined edge appears on the other screen, as if the two screen edges were directly connected. However, many users find this cumbersome, and the connection is not self-revealing. Techniques such as *drag-and-pop* [10] provide better results, where a stub is displayed for each connected screen, on which the object can be dropped to transfer it, or *pick-and-drop* [93], where a pen or hand is used to pick up an object, which can later be dropped again onto another screen, obviating the need for a dragging operation. With the telescope approach [60], a small representation of the contents of connected public displays could be kept on the user’s private screen, so that objects can be dragged into or out of the public screen via the telescope. Since the target position is freely movable, and the transfer mechanism works in both directions, the telescope can provide for very fluent and clear transitions between the screens, albeit at the expense of sacrificing screen space on the private screen for the representation of the public display.

While the usage of artificial space delimiters or multiple displays is certainly viable, some researchers suggest having a look at how the distinction between private and public access is intuitively performed with paper documents on a meeting table [101, 100]: It is common to keep a private document close to oneself on the table, whereas public documents are often moved towards the center, so that other participants can easily have a look. This approach has been replicated by some touch surfaces [126]: Directly in front of each user, a private space is formed, on which touches by other users are ignored and into which other users cannot move any objects. The limits of the private space are not hinted at visually. Instead, the system relies on the implicit knowledge and intuition about common notions of "personal space", which users are expected to hone. Thus, the idea behind the approach is to serve as a technological enforcement of social norms, which need not be described in detail, because they are part of the cultural heritage of the users. To publicize an object, it is sufficient to move it out of the private region.

Ringel et al. [95] introduces several actions to allow for an implicit transition between public and private objects on a touch surface. The approach in which a private region is created is referred to as *relocate*, since the spacial position of an object is the deciding factor to define its access policy. However, they introduce several other techniques that are deemed intuitive. *Release* refers to the ability of a user to prevent others from accessing an object by claiming it through a touch. This is akin to the real-world scenario where a person might want to grab a document, while another person is holding it: Unless the current owner relinquishes their grip, e.g. releases the object, the other person will come away empty-handed. Similarly, if another user wants to grab a document that the owner does not yet deem ready for public consumption, they can quickly put a finger on it to retain it, another behavior that can frequently be seen in real-world scenarios. The approach is notable because it does not establish an explicit system representation of ownership, but relies on the users instead to negotiate ad-hoc ownership claims. *Resize* is an interesting approach that is akin to the previously cited "magnification" metaphor: If an object is scaled down below a certain threshold size, it automatically becomes owned by that user. While this technique does not exist in the real world, where arbitrary scaling of objects is generally not possible, it is still based on a real-world practice: It is common to keep objects small that should be private, such as notes and scribbles in a moleskin sketchbook, whereas objects are usually made large (such as big print on presentation slides) to share them. By scaling an object down to make it private, the system caters to the intuitive urge of a user to keep a private item small, maybe even cover it with their hand, to protect it from prying eyes, while they also take pride in their objects when they present them publicly, being content to have them large enough so that everybody can get a good look.

Reorient is based on the most interesting observation: It is quite common to cue the intent to share a document on a meeting table by rotating it away from oneself, orienting it towards the center of the table, so that other participants can read it. Thus, the system can define an angular threshold between the current owner and the principal axis of the document, which decides whether an object is in private or public mode. Much more interesting, however, is the underlying observation: The notion of turning an object towards the center of the table to share it can be seen as the end of a spectrum, with many nuances in between that are meaningful in the context of communication as well. For instance, it is natural for users to slightly tilt a document towards a neighbor at the table in order to grab their attention and maybe ask them for their particular input. Similarly, a neighbor might signal their intent to have a look at the document without actually claiming ownership of it by slightly tilting it towards themselves as well, to which the owner might respond by either hiding the contents with their hands and orienting it back towards themselves, or they might actively help the interested party by pushing the whole documents towards them or letting them grab it.

While these communicational cues are very subtle and therefore difficult to model in software in order to trigger a meaningful action, it is desirable for a touch-based table-top system to allow users to control orientation at such a fine granularity that this social practice can be transferred without any limitations imposed by technology. Thus, many researchers recognize the immensely important role of object orientation on surface-based interfaces, leading to software frameworks that provide orientation handling at very fine granularities and provide for highly realistic rendering of objects not orientated in parallel to a principal axis, so that contents like text remain readable, consequently enabling users to exploit the entire range of possible orientations during their interactions [101]. The deciding quality factor of these frameworks is the recognition of all the roles document orientation plays in a multi-user context, namely a criterium deciding which user can interact with a given object at any time, respecting their relative orientation towards it, the building of a sense of individual or group ownership (since multiple users can intentionally orient an object so that it becomes readable for all of them), and as a carrier for subtle communicational cues.

Indeed, orientation is just as important as a factor in the establishment of ownership and a context of sharing as locality. The creators of the Lumisight table-top [74], a system that uses special light-direction dependent diffusion film to display different pictures to users sitting at each of the four sides of the table, have defined four groups of objects, depending on the meaningfulness of orientation and position in the forming of a shared context: The first group is comprised of objects for which users share *both* orientation and position, such as a large map displayed on the table-top: So that users can share this map in a meaningful way, they must share both dimensions, so that pointing at features becomes possible. Some objects are only *shared by position*, such as specific points on the map: For a label describing a landmark on the map, the orientation does not need to be shared: Instead, it is more fruitful to display the label at the correct orientation to each individual user, so that they can read it. Similarly, some objects are only *shared by orientation*, such as a small overview map of the larger map: Such an overview can be freely positioned by each individual user where ever they find it most usable, as the orientation of the contents of the map are sufficient to create a shared piece of information, so that directional descriptions given by one user (such as pointing in a certain direction) can be related to each user's individual map. Finally, some objects can be shared *without position and orientation* playing a role: For instance, a sub-window displaying text about an object can be re-positioned by any user, and will most sensibly be oriented towards them, so that they can read it.

While most other systems cannot display different imagery to different users, the recognition of the roles of orientation and positions pertaining to individual objects can still serve as an important cue for the designer to find areas of potential conflict, such as when an object is shared based on its orientation, but at the same time contains information such as text, which needs to be oriented towards a user in order to be readable. Thus, the designer can identify such conflicts and aim to resolve them by separating information in a way that its structure follows both its sharing characteristics and its requirements for usability.

Not to be underestimated for the creation of group awareness is the role of sound effects, both reactive, proactive and ambient [43]. In a multi-user environment, it is common that larger tasks, such as sorting a set of documents according to some criteria (like "important" and "unimportant" in the course of the meeting), are separated between multiple users through group dynamics. Each individual user might therefore not be aware of the progress or potential problems of other participants, because their center of attention is captivated by their own sub-task. Sound effects, such as when a digital document is added to the "important" pile, can serve an important role in the creation of awareness about the actions of other users, even though the effect itself might be perceived as superfluous or even annoying to the user that has caused it, since their own actions are understandably obvious to them. A system can even be designed to use custom sound effects for each individual user, augmenting the scope of awareness building to include the level of the individual. For instance, an experienced user might recognize that a novice is suffering from a problem due to the distinct sound associated with them, so that they might cease their own work for a moment in order to help. It is even imaginable to use multiple speakers, so that a sound may be audible either to just one, a certain group or all users at the same time, depending on its relevance. Thus, sound effects can be a valuable source of information on a shared touch surface that is too large to observe in its entirety, provided that it is used sparingly enough not to be shut out by individual users.

Issues regarding the coordination of individual users, the proactive building and maintenance of a shared working context, as well as the distinction between public and private work are core problems that need to be addressed by the design of multi-user multi-touch interfaces. It is crucial to recognize the fact that a multi-touch system is not automatically multi-user friendly because it is technologically able to support multiple touches at once. Instead, multi-user has to be seen as a specific design goal. Thus, the different requirements of individual multi-user applications need to be identified and addressed in a sensible manner: For instance, the distinction between public and shared spaces might not need to be as rigid for applications aimed at a group experience, such as exhibition applications for museums or games, as it might be for business meeting applications. Again, the introduced techniques are aimed to serve as an inspiration to the designer, on which individual solutions can be based.

4 DIY Optical Multi-Touch Hardware

With the potential of multi-touch surfaces widely recognized and the concept garnering scientific and media traction at an astounding rate, it is desirable for designers without a strong background in technology to be able to experiment with large multi-touch surfaces. The most important factors in the proposal of such hardware designs are the price at which they can be produced, the skill needed to assemble them, and that the general availability of parts is within the reach for one-off implementations.

Optical multi-touch surfaces are the most suitable to achieve this goal and are widely used in the scientific community as well as in commercial products [74, 53, 39, 15, 87, 19]. Requiring only a computer vision camera as a sensor and being otherwise based on cheap hardware that is widely available, optical touch surfaces can be built on a budget that is well within the reach of the interested individual. Additionally, their construction does not require specialized skills and is largely based on the assembly of ready-made parts, with some designs not even requiring the use of a soldering iron. Additionally, the availability of free tracking software [55], which is able to derive touch positions from the camera image, means that a designer need only build the simple hardware, but can use a free tracker to drive it. Thus, the complex creation of specialized tracking software is not necessary, which would put optical tables outside the skill sets of non-technologists.

While other technologies such as capacitative touch sensors can be built in a small or home laboratory as well [67, 94], they require substantially more skill to assemble. On one hand, the building of individual antennas is a rather delicate task that requires a lot of patience and dexterity, on the other, the technology does not scale well to large surfaces and high resolutions, as the signal-to-noise ratio for self-built antennas will quickly deteriorate as additional antennas are added. The solution also requires a considerable amount of soldering work in order to connect each antenna to a microprocessor. Finally, since transparent conductors such as specialized polymers or indium-tin oxide (ITO) are very expensive and require a manufacturing process that cannot be replicated in a small setting, such surfaces could only be used with top-projection, whereas optical surfaces can support either top- or rear-projection.

The following sections strive to describe different methods of optical multi-touch surfaces in a way aimed at individuals that intend to build their own system. The basic principle behind each method will be explained in detail, in conjunction with a thorough explanation of the necessary components and assembly instructions to create a workable solution.

4.1 General Properties

Optical multi-touch surfaces are based on the idea of building an apparatus to create a visual response to a finger coming into contact with the surface. This response must be distinct enough, so that computer vision algorithms can be used to derive positional information from the camera image. Ideally, the response should be generated as soon as a finger touches the surface, but not if it is closely hovering above it, marking a false positive. However, this is not always achievable, so that software algorithms need to be employed to distinguish those responses that correspond to fingers hovering closely above the surface from those that are actually in contact.

The camera is always placed behind the surface, so that the surface is positioned in between the user's hands and the camera. Thus, occlusion is not an issue with such systems, but the setup requires a light path of sufficient length between the surface and the camera, so that the entire surface is visible. While regular camera lenses are engineered to provide for a rather narrow viewing frustum, wide-angle or fisheye lenses can be used to reduce the length of the required light path, leading to designs that are much more compact. However, such lenses generally introduce image distortion that can be difficult to rectify by the software tracker.

In many cases, it is desirable to build a system that can support for digital imagery as well,

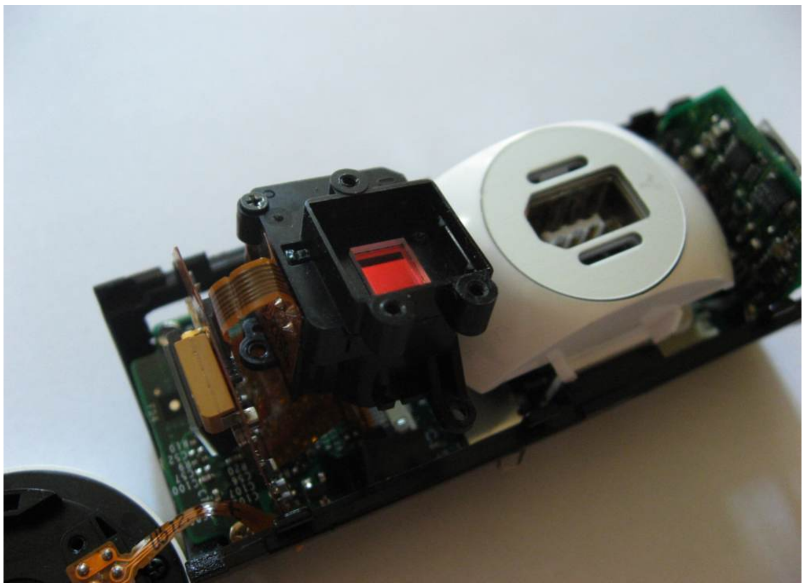

Figure 24: Removing a web camera's IR-cut filter

providing operation both as a multi-touch surface (without any directly imposed imagery) or as a multi-touch screen. Thus, the surface touch response is generally generated in the near-infrared spectrum between 720nm and 1000nm wavelengths, since typical CCD and CMOS chips used in widely available computer vision cameras as well as web cameras respond to such frequencies, while they are invisible to the human eye. Thus, any illumination techniques employed in order to generate a visual response to touches remains invisible to the users, but at the same time, visible wavelengths can be blocked from the camera as well, so that complex or moving imagery visible to the user on the multi-touch screen does not interfere with the tracking algorithms.

Most computer vision camera lenses ship without any coating to block off light outside the visible range, so that a bandpass filter suitable for the desired wavelengths can be chosen without any restrictions. Less sophisticated cameras aimed at the consumer market ("webcams") are generally equipped with an IR-cut filter, which blocks out all wavelengths but the visible spectrum. However, it is often possible to open the camera housing, where the filter is usually placed between the lens-holder and the imaging sensor, so that it can be removed. By applying this simple modification, affordable webcams can be used for optical multi-touch surfaces as well.

Provided that a camera does not have an IR-cut filter installed (or it has been removed), the imaging sensor will pick up both visible and near-infrared wavelengths. Thus, in order to operate solely in the near-infrared spectrum, the installation of a filter becomes necessary, which blocks visible wavelengths and is transparent only in the near-infrared range. Such filters are readily available from many photography stores as well as internet sources, so it is not hard to obtain. Even more accessibly, a suitable filter can be obtained by using one or multiple sheets of certain polymers which are opaque to visible light, but transparent in near-infrared. For instance, exposed sheets of photo film are very suitable for this task, as well as floppy disks. Another possibility is to obtain red, green and blue light filters from lighting or music shops, which are often used to color stage lights at events and concerts. By layering them over each other, all visible frequencies are blocked out, but as each individual filter constitutes a band-cut at red, green and blue wavelengths, all three filters remain transparent in the near-infrared range. Since such filters do not have any lens properties, they can be installed anywhere in the light path without introducing image distortion. It is often easiest to install them between the lens-holder and the imaging sensor, but they can be placed in front of the lens as well if necessary.

If the band of near-infrared wavelengths in which the setup should operate is known in advance, the IR-pass filter can be replaced by a suitable IR-bandpass filter. Such filters do not only block all visible wavelengths, but near-infrared wavelengths other than those within a certain range around a peak wavelength as well. This is very useful in order to make the setup more robust against ambient near-infrared light, which is emitted from most visible light sources as well as the sun. By restricting the detected wavelengths to a narrow band, ambient light at other near-infrared frequencies can be blocked out completely. However, bandpass filters are generally considerably less transmissive even at their peak frequency than broadband filters (typically between 50% and 80%), so that they will diminish light signals even within their rated operating band. If the light response generated by the setup is very weak, the additional brightness loss can outweigh the benefit of removing more ambient light, so that a broadband filter remains more suitable.

When choosing a near-infrared wavelength to operate on, it is advisable to keep the image sensor's response curve in mind: Typically, a computer vision image sensor will have a rather uniform response within the visible spectrum, which will extend into the near-infrared spectrum, but quickly drop off as the wavelength increases. Thus, the brightest, most detailed image can be obtained by using a low near-infrared wavelength, preferably in the range of 760nm to 820nm, where many cameras retain adequate sensitivity. Additionally, monochrome image sensors are suited better for near-infrared sensing: Color sensors are monochrome sensors with color filters fitted in front of individual sensor pixels, organized in a Bayer pattern [11], which is then used to interpolate full RGB values from the individual color samples. Since only the red color

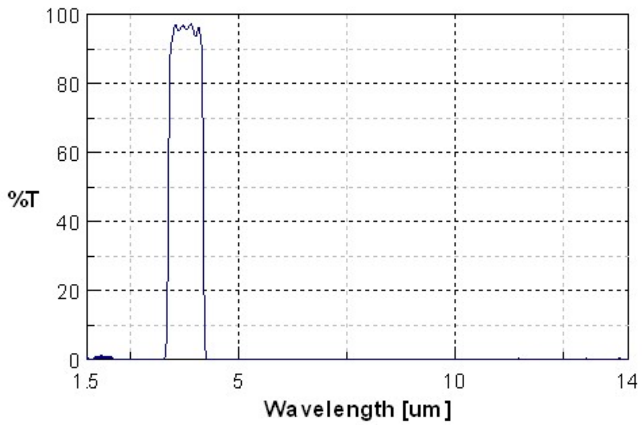

Figure 25: Typical IR-bandpass filter response (*www.wavelength-tech.com*)

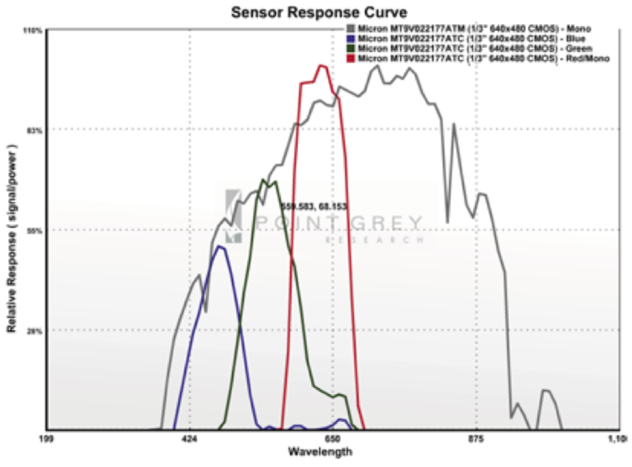

Figure 26: Typical CMOS RGB/Mono image sensor spectral sensitivity (*www.ptgrey.com*)

filter will allow near-infrared wavelengths to pass, the resolution of such a camera is effectively reduced to a third of its nominal value. Additionally, green and especially blue pixels will not record any samples, resulting in pixel values dictated by the noise within the sensor circuit, which results in an unstable image. Since monochrome cameras do not have such Bayer color filters, they provide a uniform near-infrared response across their entire surface.

If the optical multi-touch surface should be operated with a rear-projector in order to form a screen, it is necessary that the projector does not emit much light in the near-infrared spectrum, as not to overshine the light response generated by finger touches. Most commercially available projectors ship with an IR-cut filter pre-installed. Whether or not a projector is equipped with such a filter can be verified by pointing a camera with an IR-pass filter at the light source and checking the response in a video application. If a considerable amount of emitted light is visible, an external IR-cut filter should be installed in front of it, so that the projector does not interfere with the camera during tracking operation. Such filters are readily available from suppliers of optical equipment and are generally not expensive. An alternative is the usage of IR-blocking foil that can be bought at many property markets. It is typically used to block infrared light passing through windows in order to reduce room temperature, but it provides sufficient opacity in the near-infrared spectrum as well.

If a surface does not need to be used as a screen, it is sufficient to mount the camera below the surface, so that the entire area is visible in the camera image. If screen operation is deemed necessary, there are two ways to achieve this: The employment of a *rear-projector* or an *LCD panel*.

If a rear-projector is to be used, it is necessary to provide a throw distance long enough to achieve a suitably large image. Often, the required distance is large for off-the-shelf projectors, but an adequately compact design can still be achieved if mirrors are used in order to bend the light path, so that a sufficient distance can be achieved in a smaller space. Specialized short-throw projectors are available as well, even though they are considerably more expensive than comparable models. Since many projectors and operating systems support arbitrary mirroring and rotation of images, image reversal by the introduction of one or multiple mirrors is not a problem. In order for the projection to be visible, the surface must additionally be covered with a diffusor material that is suitable for rear-projection, but also sufficiently transparent in the near-infrared range, as to not interfere with the touch light response. Fortunately, most rear-projection films satisfy this condition. While such films are easy to obtain from home entertainment shops, large sheets of paper can serve as a simple replacement as well. Thin, high-quality paper with little inherent texturing (such as paper used by architects for quality drafts) can rival many projection films in quality. During installation of the diffusor, it is advisable to take care not to introduce too many ripples in the surface, as they will affect image quality and the overall feel of the surface on the fingers detrimentally.

It is not necessary to enclose the projector, camera and surface in a box. However, if the setup should be usable in areas with intense ambient near-infrared light, building an enclosure is advisable to block off ambient light. The actual touch surface is then mounted as one side of the box, while all other sides consist of entirely opaque materials, such as wood or thick plastic.

As an alternative to rear-projection, an LCD panel can be used to generate imagery as well. While this approach provides a much clearer, more vibrant picture at a higher resolution than achievable by all but the most expensive projectors, construction is considerably harder. The basic idea is to mount an LCD panel below a protective layer (such as acrylic glass), and

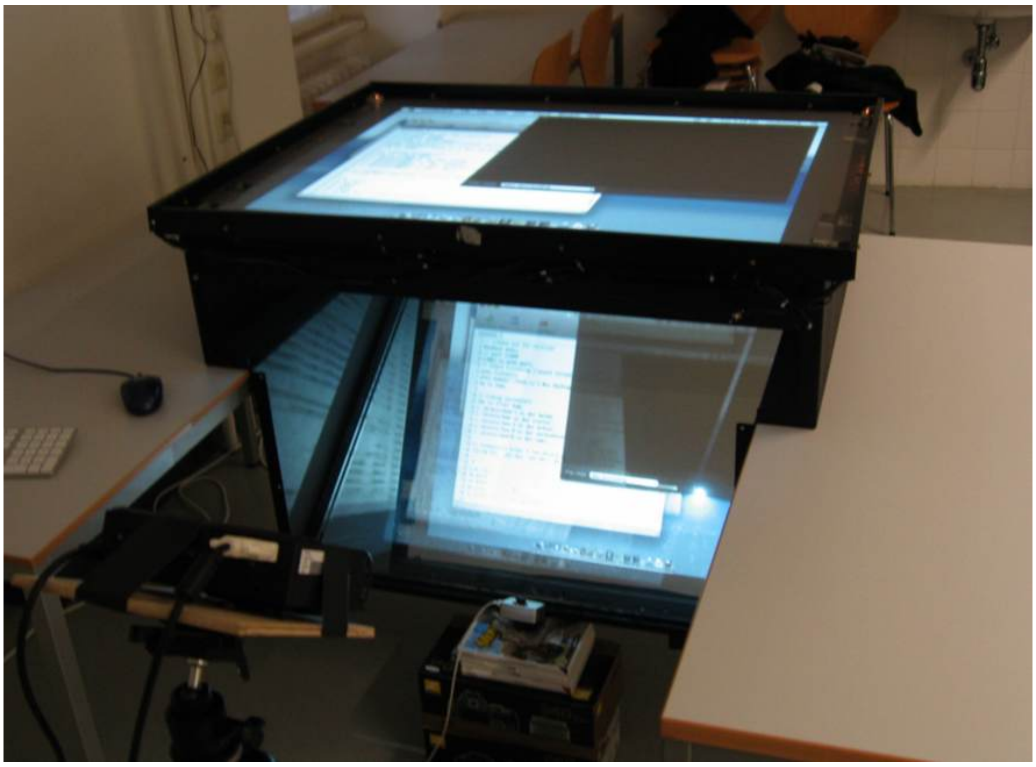

Figure 27: Rear-projection multi-touch screen

provide strong back illumination in order to make the image displayed on the panel visible. While individual panels can be obtained from wholesale electronic dealers, the most affordable and easiest way is to modify an off-the-shelf LCD screen: By opening the case and separating the backlight from the panel, both parts can be reused for the touch surface. First, a box will need to be built, which can afford for a sufficiently long light path for the camera. On the side where the camera will be mounted, the backlight is reinstalled. The camera can now be placed directly above the backlight. On the opposite side, the LCD panel and its protective surface is mounted. The other four walls are now covered with reflective mirror foil, so that they do not absorb any light emitted by the backlight. Thus, if the box remains closed during operation, the LCD screen's native backlight remains bright enough to provide for a clearly visible image. Since the LCD panel is transparent at every non-black pixel, a diffusor sheet needs to be placed below the panel. Thin, texture-less paper sheets will provide the best results. Any additional hardware required to drive the LCD panel (such as the backlight driver and input connector board) can now be mounted outside the the box. Thus, a simple modification is achieved, in which the backlight is moved away from the LCD panel sufficiently to place a camera in between.

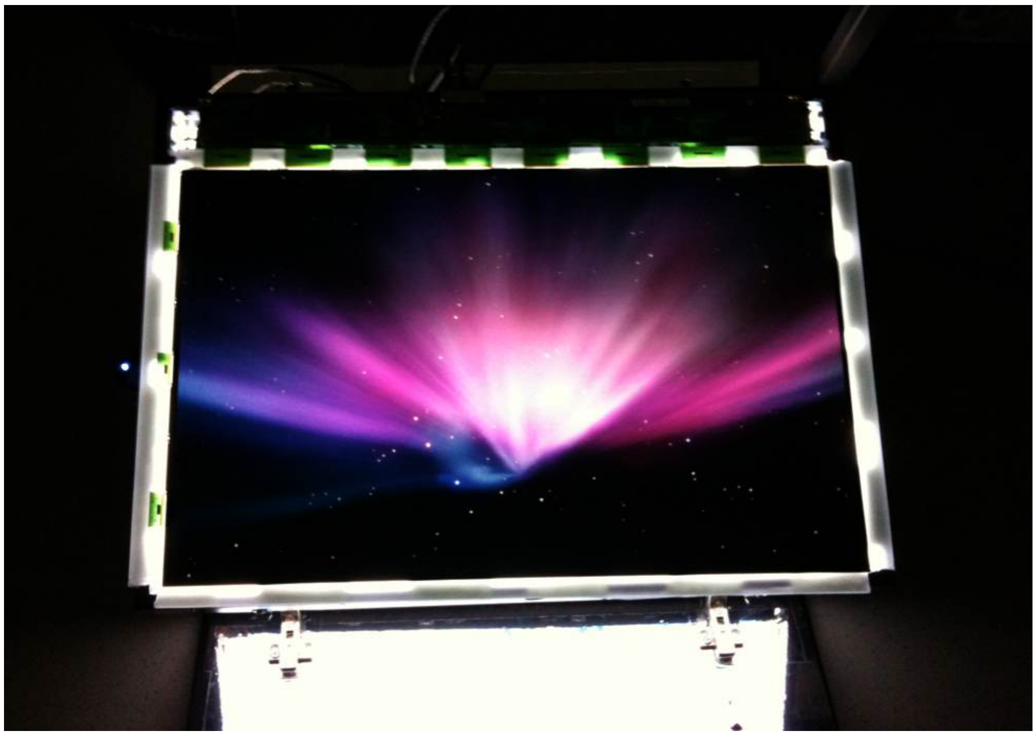

Figure 28: LCD-based multi-touch screen

Great care has to be taken during this process in order not to damage the LCD panel, which is mounted on an extremely thin layer of glass and is rather breakable. Additionally, it might be necessary to lengthen some cables during the process, which can be achieved by soldering or clamping extension cables in between. However, some cables (such as flat ribbon connectors) can not be extended in this way due to their small size, but suitable extensions for these can often be found at electronic stores. Additionally, it is very hard to lengthen the analog signal cable between the LCD panel and its driver board, since the analog signal is extremely susceptible to skewing effects and electronic noise, leading to a flickering or distorted image. Thus, the box has to be designed in a way so that the length of the signal cable is sufficient, as this cable is generally wired in a proprietary way, making it impossible to find off-the-shelf replacements at a larger length.

While the polarization filters used within the LCD panel are partly transparent in the near-infrared spectrum even at the wrong polarization, they diminish the brightness of passing light considerably. Thus, the touch response must be very strong in order to be visible clearly by the camera. If an IR-bandpass filter is used, its performance should be compared to that of an IR-broadband filter: While the lower peak-transmissivity of the bandpass filter will often reduce the optical touch response to levels too low for the camera to pick up, the benefits of the bandpass filter, namely the reduction of ambient light at wavelengths unimportant to the tracking process, is less pronounced in the LCD-based setup, since the closed box and the high near-infrared opacity of the LCD-panel itself is generally sufficient to block out interfering ambient light.

The described setups can be used with different techniques to actually generate an optical response from a finger touch. However, the supporting structure of camera setup and imaging surface remains the same for all these techniques. Thus, the following sections will describe these variants, which can be combined with either the *rear-projection* or *LCD-panel* supporting structure. These setups are the *diffuse illumination* approach, the *FTIR* approach and the *light-plane* approach.

4.2 Diffuse Illumination

The oldest and most simple technique to generate a near-infrared response from contact points is *Diffuse Illumination*, with the first examples being developed in the 1970s, while the approach remains in use for modern surfaces as well [76, 48, 74, 53, 46, 19].

The approach is based on the idea of placing a camera below a diffusing surface and shining a strong near-infrared illuminant uniformly onto the surface from the top. When an object comes close to the surface, the the light thrown by the illuminant is blocked off. Since the diffusor causes a strong blurring effect in the outline of the shadow, its shape and definition is much more pronounced when the object is very close to the surface, up to the point where the actual shape in contact with the surface becomes sharply defined upon touchdown. The vision system can define a luminance threshold to decide when a shadow is deemed dark enough to be interpreted as a contact point.

While this approach based on top-illumination was created first, it is difficult to achieve uniform lighting in such a setting, since various objects such as arms and hands will often be within the light part, leading to areas of the surface that are darker, so that thresholding to detect contact areas becomes error-prone. In the most extreme case, an object might block off the illuminant completely, leading to a collapse of the vision system. Thus, the approach has been slightly changed in that the near-infrared illuminant is placed below the surface, next to the camera. Instead of detecting darker areas as in the top-illuminated variant, the rear-illuminated variant relies on light scattered back by objects close to the surface. The diffusor sheet ensures that illuminant light is scattered diffusely from the surface, so that only those objects close to the

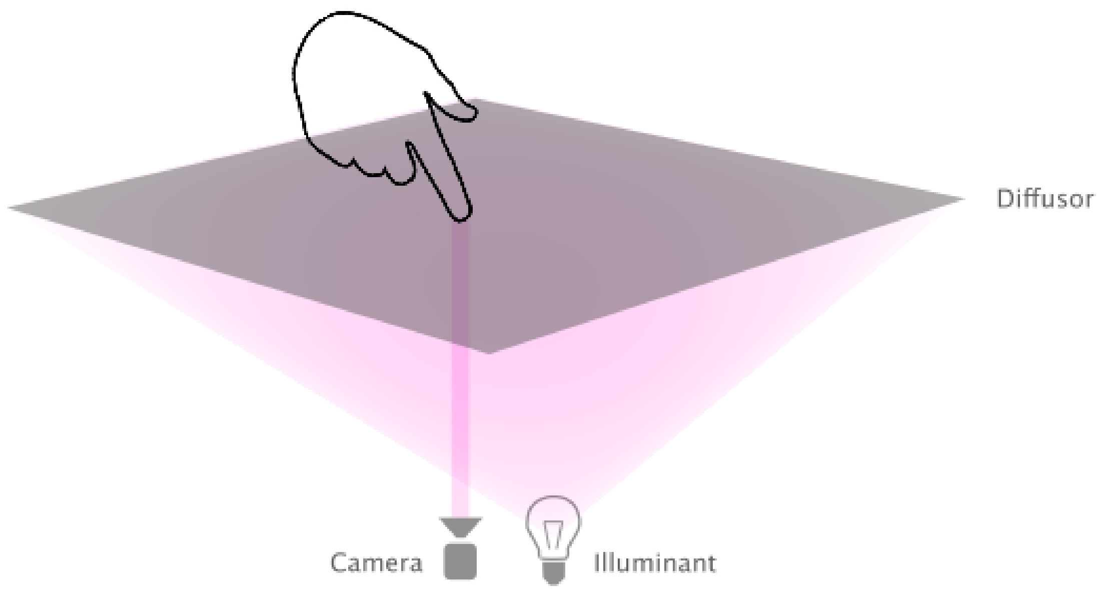

Figure 29: Top-illuminated diffuse illumination multi-touch setup (schematic)

surface reflect enough light to be detected by the camera. Similarly, a luminance threshold is used to detect those objects bright enough to be considered in contact with the surface.

While the rear-illumination approach is more robust, as it is not so much disturbed by objects above the surface, it suffers from the problem that ambient light above the surface (such as sunlight or incandescent light) needs to be overshined by the illuminant in order to be effective. However, this is hard to achieve in some settings, especially if the ambient light is exceptionally bright, such as direct sunlight or strong artificial light, such as the ones used on stages. If the ambient light is too bright, the contrast between objects illuminated through the surface and the surface itself can become too low for the computer vision algorithm to detect.

This dependence on ambient lighting conditions is a severe disadvantage of the approach, limiting its employment to settings with very controlled conditions, to which the vision system can be fine-tuned. Nevertheless, such multi-touch hardware performs best in rather dark rooms. There is some research in letting the system switch automatically between rear-illuminated and top-illuminated variants depending on the brightness of ambient light [46], but since top-illumination is not guaranteed to provide workable results (e.g., if an object is blocking parts of the surface off from ambient light), there is a boundary to the practicability of the approach as such. Since the approach depends on the touch surface's being at least partially transparent

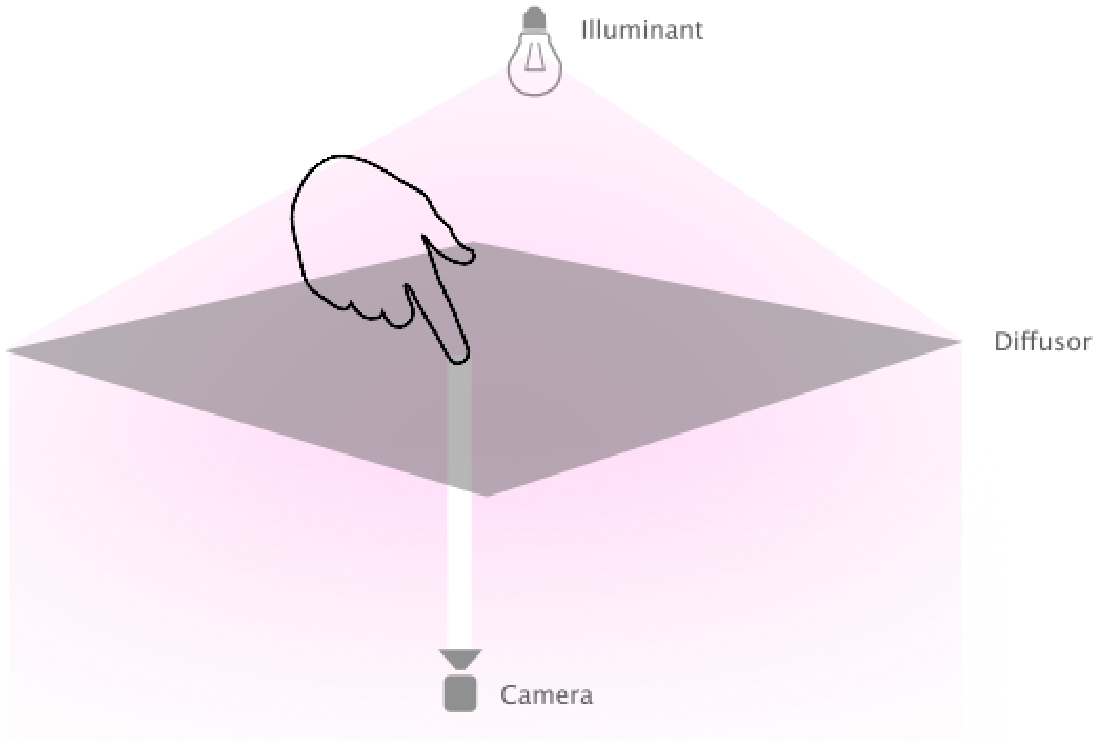

Figure 30: Rear-illuminated diffuse illumination multi-touch setup (schematic)

in the wavelength of the illuminant, filtering foil cannot be used to block off unwanted ambient light, as this would block the actual illuminant as well.

However, *diffuse illumination* retains its importance because such systems are very simple to build: For a top-illuminated setup, it is sufficient to place a camera below a diffusor sheet (which can easily be achieved by placing a piece of paper on a glass table) – Ambient light, such as diffused sunlight in the room, is often sufficient to serve as a top-illuminant. If tracking is to be done in the visible light range, because projection onto the touch surface is not deemed necessary, an off-the-shelf webcam can be used, leading to a setup that, while brittle concerning changes in ambient lighting, can be built from scratch within minutes.

An additional advantage of such systems is that they do not make an assumption about the objects that it can detect on the surface, other than them being at least partially opaque or reflective to the wavelength of the illuminant. Thus, touch responses are not limited only to finger tips, but can also be generated from many other objects that satisfy this condition. Additionally, a light response can also be generated from fingers covered in gloves, which other approaches might have difficulties with.

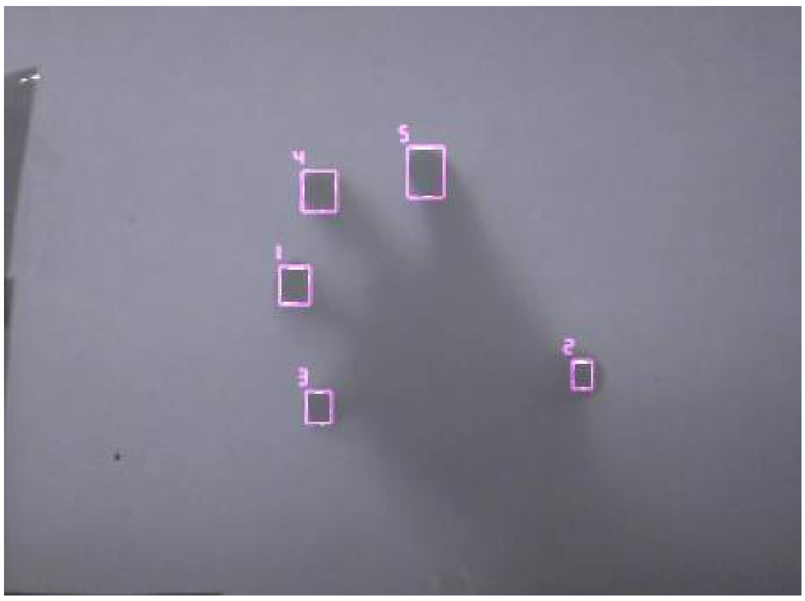

Figure 31: Diffuse illumination multi-touch with a glass couch table and a paper sheet (Touché)

Even more elaborate rear-illuminated setups are not difficult to build, since the surface itself does not need to be fitted with any electronics whatsoever. It is sufficient to place a strong illuminant behind the surface, which can be any strong lamp if operation within the visible spectrum is acceptable. Strong near-infrared illuminants are also easy to obtain, since they are often used in conjunction with surveillance cameras, so that dark rooms can be monitored without the use of visible illumination. Thus, such light sources are available in a ready-made form at many electronic stores and home depots. Additionally, a suitable illuminant can also be created by wiring a matrix of strong near-infrared LEDs.

The main difficulty when building the setup (apart from acquiring a light source that is bright enough for the particular ambient lighting conditions) is to ensure a uniform illumination of the surface. If parts of the surface remain unlit or weakly lit, the vision system is not able to pick up touch responses, leading to dead spots on the surface. Uniform lighting can be achieved by placing an additional diffusor directly above the light source, such as often exemplified by photographers, who want to prevent reflections stemming from their stage lighting. If the setup is enclosed in a box, the walls can be covered with diffusely reflective material, which can easily be achieved by painting them with opaquely white color or covering them with white plastic sheets. This ensures that illuminant light is not absorbed by the box enclosure on one hand, wasting light power in the process, but also to ensure uniform lighting by preventing specular reflections from the walls, which could otherwise lead to particularly bright areas on the surface.

Since a *diffuse illumination* surface is trivial to build, the approach is especially suitable for designers with tight financial and time constraints, as a system can be built from cheap components very quickly. Thus, it lends itself to serving as a platform for quick experimentation and prototyping, where the setup can be placed under controlled lighting conditions. It can thus serve as a rapid development platform, while the actual interaction designed in the process can later be deployed on a hand-made multi-touch surface based on more elaborate technology, which is less susceptible to ambient lighting conditions and can thus be used in day-to-day life.

4.3 FTIR

The approach based on the *FTIR effect* has been popularized by Han [39], who was the first to apply the principle to the creation of multi-touch surfaces, even though it has long been used for other touch sensors, such as in robotics [45, 107]. His research has also been spun off into a commercial venture [87], which provides many high-profile companies with touch technology, showcasing the potential behind the approach.

*(*FTIR) (frustrated total internal reflection) refers to a result of wave physics: If an evanescent wave travels through a medium embedded within another medium of a lower refractive index, total wave energy will be conserved, e.g. the wave transmits a total of zero net energy.

However, if another medium is brought less than a couple of wavelengths close to the carrier medium and has a higher refractive index than the second medium, the evanescent field created at the border of the first medium will not decay much before it reaches the third medium, bridging the gap through the second medium and transferring energy into the third, thereby altering the direction of the wave. In the case of light waves, this means that light trapped within a medium can be "tunneled" from one medium into another across a very small air gap. If the third medium is diffuse, it will appear as if it started to glow due to its contact with the light carrier medium. For instance, the effect can be observed by looking at the finger tips of a hand holding a glass of water: Since the FTIR effect remains intact within water and glass border, but coupling is possible between the glass and the finger tips, the finger prints (e.g. the parts of the skin actually in close contact with the glass) will appear illuminated. Indeed, the effect has often been used in fingerprint scanners as well.

Han's approach is to couple near-infrared light into a transparent carrier medium, where it is preserved due to the FTIR effect, but is tunneled into a user's fingertips when they come into contact with the surface. Thus, the fingertips will light up due to their diffusion of the coupled

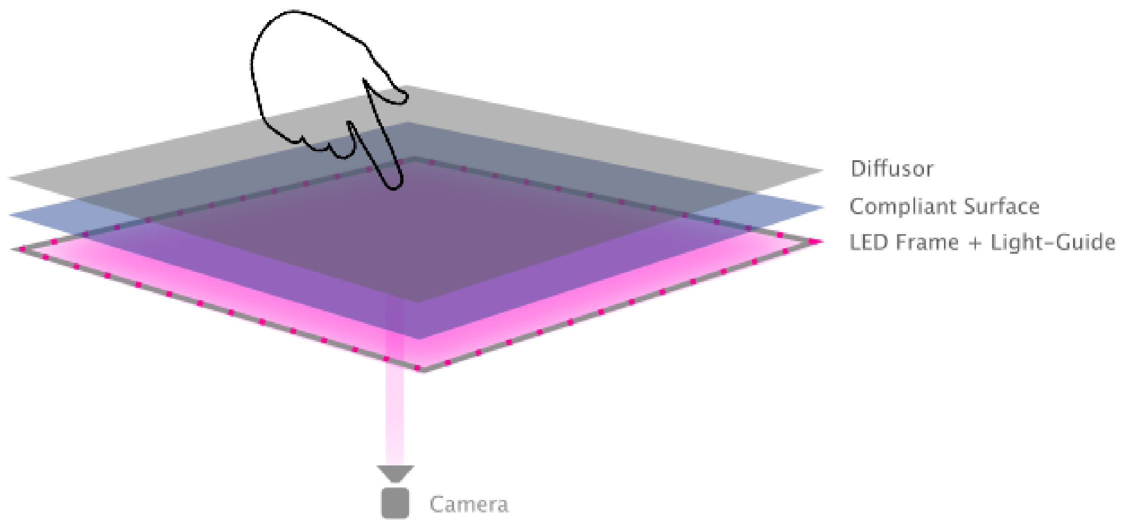

Figure 32: FTIR multi-touch setup (schematic)

light, which can be picked up by a camera and detected by computer vision algorithms. To achieve this, a piece of acrylic glass is used a light carrier. It is notable that window glass is not suitable for the task, since its refractive index is too high in comparison to that of the skin of a fingertip. The acrylic glass is fitted with a frame, into which many near-infrared LEDs are built, pointing directly towards the side of the acrylic panel. There are no rigid guidelines for the amount of LEDs necessary to achieve good results, as this is highly dependent on the dimensions of the surface, but as a general assumption, a spacing of 3-5cm between LEDs will be sufficient for most applications.

When the LEDs are turned on, the emitted light will be coupled into the acrylic glass, where it remains trapped due to the FTIR effect. However, it can escape at the opposite side of the panel, where the incident angle is too acute for the FTIR effect to work. Thus, it is advisable to cover the sides of the acrylic panel with mirror foil, into which small holes for the individual LED heads are cut, so that escaping light is coupled back into the panel. Additionally, the sides of the panel will also partly reflect incoming light from the LEDs, so that a considerable amount of light energy is reflected, rather than coupled into the surface. However, this effect can be alleviated by installing the LEDs at an angle (preferably 45 degrees) rather than pointing them directly at the sides, leading to an immensely improved coupling efficiency. Obviously, it is also advisable to polish the sides of the acrylic panel to a shine before the LEDs are installed,

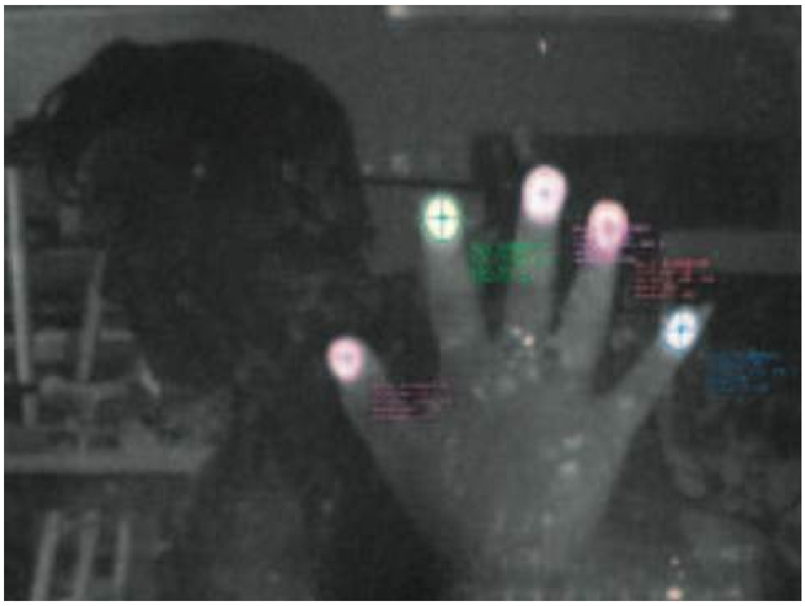
Figure 33: Multi-touch sensing with FTIR [39]

as they are often turned matte by the cutting process of the acrylic glass. If the sides remain matte, the coupling effect will be very weak.

Once the LED frame is installed in this way, and the LEDs are turned on, a discernible brighting of fingertips can be registered by the camera when they come into contact with the surface. It is a particular advantage of the FTIR approach that it reacts only to actual touch (e.g. when a fingertip is brought closer to the surface than several wavelengths of the illuminant), so that the computer vision system does not need to rely on a brittle brightness threshold to distinguish closely hovering fingers from those actually touching the surface.

Since a diffusor is not necessary in the setup, a fully transparent multi-touch surface can be created. If a screen is to be created, however, an LCD panel or a diffusor can be placed behind the the light carrier plane. While this will introduce the issue of motion parallax, it has to be noted that the light carrier can be arbitrarily thin in order to reduce parallax, being only practically limited by the difficulty of attaching the LED frame to a very thin carrier. Additionally, if the carrier is too thin and thus flexible to withstand the finger pressure of touch operation, it is possible to place another transparent material (such as acrylic or window glass) at a higher thickness behind the diffusor sheet with the sole purpose of supporting the assembly and increasing structural integrity.

While the reliance on direct touch coupling is one of the most distinct advantages of the FTIR approach, it is also its biggest weakness: While fingers directly in contact with the surface will generate a strong light response, the approach has difficulties with the performance of finger gestures: Since a small air cushion is necessary below the fingertip in order to slide a finger across the surface, only a very small portion of fingertip skin will be in direct contact with the surface at any time, leading to a very weak light response that the vision system might not be able to pick up in a reliable manner. To compensate, the user will have to push their finger down very hard during the performance of the gesture, so that the contact area between light carrier and finger is maximized, but this will feel very uncomfortable and tedious due to friction, up to the point where prolonged interaction can lead to strain injuries.

Thus, the approach has been refined slightly, harnessing previous experience in the building of robotic touch sensors [45, 107]: Rather than using the fingertip itself as a coupling medium, a compliant surface with a similar refractive index is introduced. While most soft polymers are suitable for the task, transparent silicone is especially promising, since it is readily available and allows for the creation of multi-touch screens, as it does not distort incident light much. The light carrier is covered with a thin sheet of the compliant material. It has to be ensured that the material does not stick to the carrier, as this would cause a continuous touch response, but that a small air cushion remains between carrier and compliant surface. When a finger touches the compliant surface, the sheet will bend towards the light carrier, triggering a light response. However, since the sheet does not require much pressure to flatten against the light carrier, a much softer touch can now be used, and finger gestures become viable.

However, the employment of a compliant surface introduces another set of problems: Firstly, soft polymers such as silicone feel very awkward when touched directly, since the surface is unexpectedly soft and maybe even a bit sticky. If it is not necessary for the touch surface to be transparent, a thin layer of a material with better tactile properties can be placed above the compliant surface (such as a sheet of paper or, if rear-projection is to be used with the system, the projection diffusor itself, eliminating parallax effects in the process as well). Nevertheless, a distinct feeling of unexpected softness remains. Secondly, the the compliant surface will take a short amount of time to "unstick" itself from the light carrier when a touch is removed. Thus, every touch will leave a short moment of after-glow behind, and finger gestures will lead to glowing trails on the surface. While the computer vision system can accomodate for this effect, a luminance threshold has to be introduced again in order to distinguish a touch from its after-glow, but the removal of such a threshold in order to determine when a touch is actually on the surface and when it is not is one of the major advantages of the FTIR approach in the first place.

If a compliant surface is used, an FTIR setup can be proofed against harsh ambient light conditions by placing a filtering foil on top of the compliant surface, which removes all light in the wavelengths at which the touch responses are generated. As the compliant surface ensures coupling between a touch and the light guide, layering additional thin, flexible sheets on top of it does not introduce any problems. Thus, an FTIR setup can be operated even under direct sunlight if such a filtering foil is used.

FTIR surfaces provide many advantages over diffuse illumination approaches. However, apart from the fact that a compliant surface introduces an awkward softness and that gestural input will leave glowing trails, the most distinct disadvantage is its complexity: While the necessary parts are generally obtainable from internet stores, they are often hard to find. Additionally, a large amount of skill in soldering electronic components is necessary in order to build and wire the LED frame. Since proper spacing, alignment and even angling of the LEDs is crucial, good craftsmanship is vital in building an FTIR surface that functions properly. Finally, the individual parts, especially a large amount of near-infrared LEDs, are comparatively expensive.

Thus, FTIR surfaces are very well suited for harsh ambient lighting conditions, as long as it is acceptable for the screen surface to have a soft feel. The building process is very involving, though, lending such systems to be used only in settings where other techniques are not applicable and where time as well as financial constraints are not a priority.

4.4 Light-plane

While approaches based on tracking occlusions in a grid of light beams reach back well into the 1970s [52], the re-discovery of such techniques has been a fairly recent process: The *light-plane* approach aims to combine the advantages of diffuse illumination and FTIR while avoiding most disadvantages and remaining easy and cheap to build. Therefore, it is particularly well suited for rapid experimentation and prototyping.

Much like in the diffuse illumination approach, the actual surface itself does not need to be fitted with electronics. The basic tenet is to beam a sheet of light very closely above the surface, so that when a finger touches the surface, it is illuminated by the light plane, leading to the glowing of the fingertip, which can be picked up by a camera as a touch response.

Thus, the technological requirements of the approach boil down to the way in which such a light-plane is created. One possibility is to employ a similar LED frame as in the FTIR approach, in which each LED is installed in parallel to the surface to which the frame is mounted. Rather

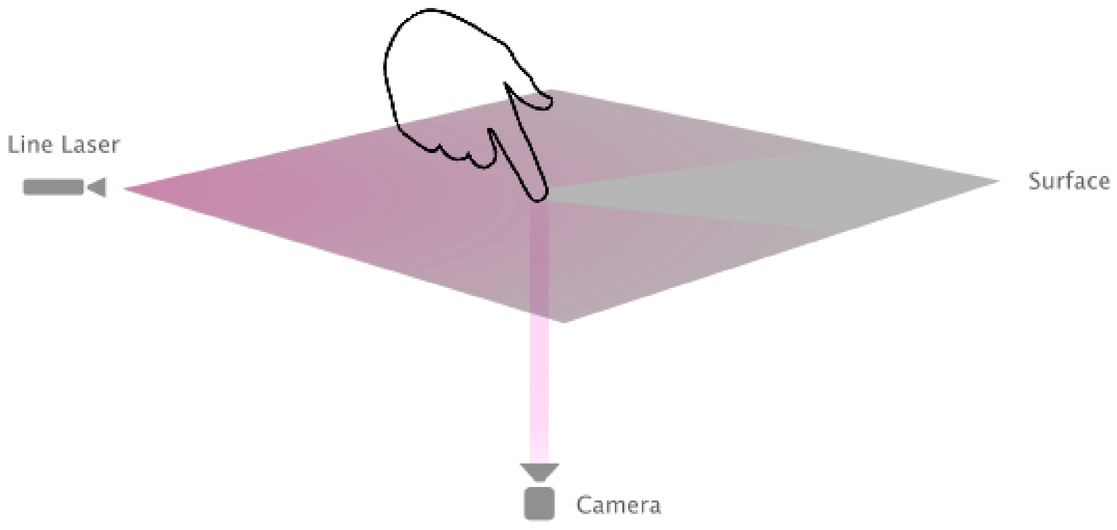

Figure 34: Laser light-plane multi-touch setup (with a single laser module; schematic)

than shine their light into the sides of the surface, the frame is raised, so that the LEDs create a light-plane closely above it. If LEDs with a narrow emission angle are chosen (5 degrees of opening angle are common values), and the surface is sufficiently small, the plane will remain close enough to the surface, so that fingertips or objects hovering highly above it will not generate a response. Nevertheless, the depth of the plane might be too high, and the light power might drop off too fast in order to cover the entire surface.

A better solution is the employment of lasers in the near-infrared spectrum. Lasers at 780nm wavelength are commonly available at different output powers and lend themselves ideally to the task. Special line generator lenses are available to fan out the laser beam into a line, which will effectively become a light-plane emanating from the laser at an opening angle determined by the lens properties. In a typical setup, four lasers placed at the corners of a rectangular surface with a line-generator lens with an opening angle close to 90 degrees will yield good results, even though more lasers can be used to contribute to the light plane. If ready-made laser modules are used, they can be connected directly to a power supply without the need for any soldering. Additionally, as laser modules are very bright, the dimensions of the touch surface can be much larger than what would be achievable with diffuse illumination and FTIR setups.

The main difficulty in the building of such a setup is the alignment of the lasers, so that the plane is generated at an optimal position closely above the surface. To achieve this, the lasers

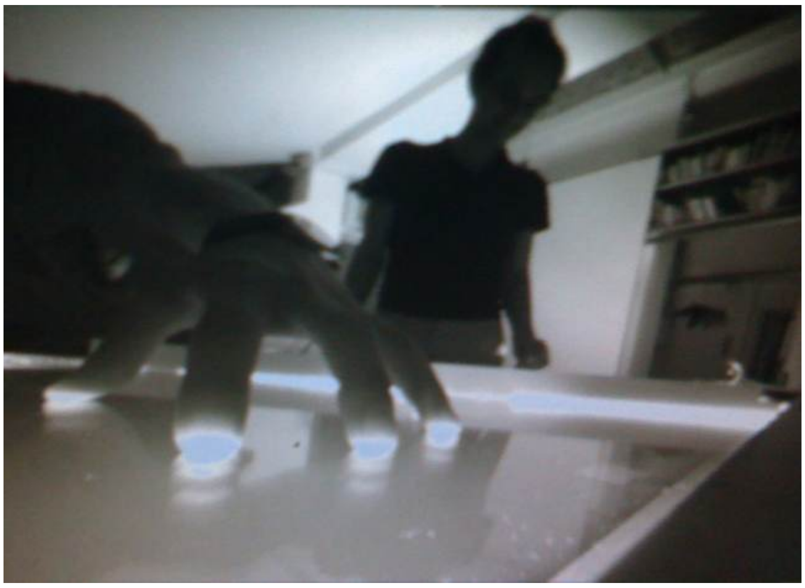

Figure 35: Multi-touch sensing with a laser light-plane

have to be installed at the correct height in a perfectly parallel way relative to the surface. Additionally, the rotation of the line generator lens must be adjusted so that the plane is both focused (its depth stays constant above the surface) and parallel to the plane (meaning that the plane must not be tilted). While special gears are available to position and orient laser modules at a very fine granularity, they are physically large and expensive.

Another approach is the employment of modeling clay, which is placed next to the corner of the surface. Since the laser module will typically need to be slightly recessed relative to the surface, the module needs to be pushed into the clay, while a near-infrared camera is used to monitor the line as it is projected above the surface. The module can now be gently rocked and tilted, until an adequate alignment is achieved. Finally, both the laser module itself as well as the line-generator lens can be locked in place with hot glue or superglue. While this approach is admittedly tedious and error-prone, it is certainly possible to align four laser modules within 1-2 hours, and a skilled individual will need considerably less time.

The surface itself must only be transparent to near-infrared light. Thus, using a diffusor sheet for rear-projection or an LCD panel as the surface (both supported by a sheet of transparent material) does not introduce any problems, but neither does using any sort of glass, leading to the possibility of creating entirely transparent touch surfaces with the *light-plane* approach.

A distinct advantage of laser light is that it is very bright, being rivaled only by direct sunlight and very strong incandescent light. Thus, touch responses remain clearly visible even in rooms

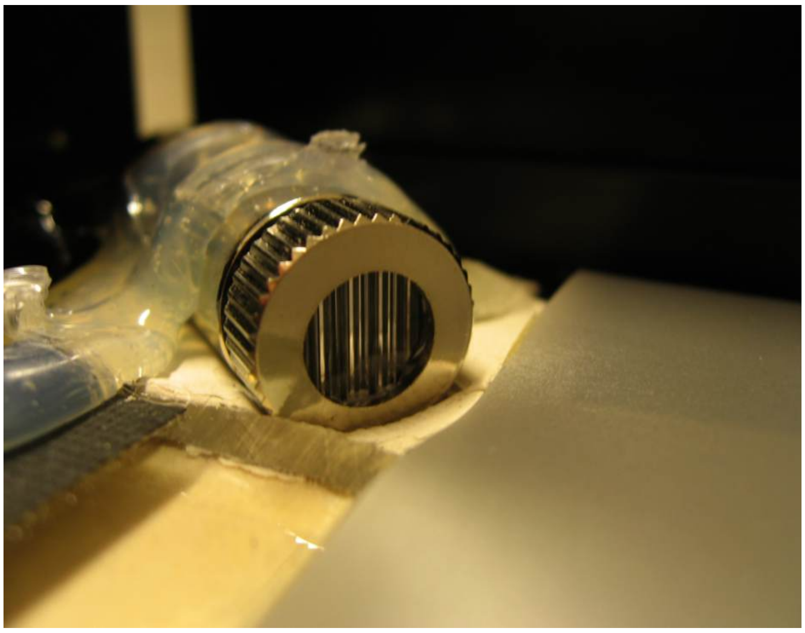

Figure 36: Using modeling clay and hot glue to align laser modules

with diffused sunlight, leading to a system that is largely independent of ambient light conditions, even though an FTIR setup with an additional IR-blocking filter on top remains the only setup that is strong enough to operate under direct sunlight. Since the lasers emit light at a well-known wavelength rather than in a broader spectrum such as near-infrared LEDs, the camera can be fitted with a narrow-bandpass infrared filter, blocking off considerably more ambient light without diminishing the optical touch response. By doing so, performance under harsh ambient light conditions can be further improved.

However, it is rarely beneficial to use lasers with a higher output power than 5mW. The brightness will only be increased slightly, but high-power lasers pose a significant health risk, especially in the infrared range, where the blinking reflex is not triggered due to the invisibility of the light, while a large amount of light and heat power is transferred onto the retina, being further focused by the eye's own lens. Consequently, lasers of higher powers than laser class 2 should be avoided. Additionally, it is recommended to install a small frame around the surface, which will prevent stray laser light from leaving the surface.

Since the lasers are bright enough to light up the entire fingertip, rather than relying only on the reflectance of light through the surface and towards the camera, the camera does not necessarily need to be placed behind the surface. It is also possible to mount a camera above the surface, so that it looks down at an angle and picks up the glowing fingertips. While such a setup naturally suffers from occlusion problems, these can be alleviated by using multiple cameras or by constraining the interaction to gestures in which occlusion in respect to the camera position is minimized. Thus, any surface, such as a wood table or a large flat-panel screen can be turned into an improvised multi-touch surface quickly by affixing laser modules as well as a camera, which can be beneficial for quick prototyping experiments.

Apart from the susceptibility to very bright ambient light, the main drawbacks of the light-plane approach are two-fold: Firstly, while the light-plane can be projected very closely above the surface, it will still have a certain depth. While this depth can be reduced down to a few millimeters by careful laser module installation, it is not possible to distinguish fingertips hovering very closely above the surface from those actually touching it. However, since this discrepancy can be diminished by careful installation and since the human hand typically exhibits a natural jittering motion, this effect is surprisingly negligible in actual operation.

The other fundamental drawback is that the approach suffers from occlusion issues: If four laser modules are used, four objects can be placed at positions on the surface in which their shadows will overlap in an area that is not illuminated by a single laser anymore. Thus, any objects placed within this area will not be detected by the vision system. The issue can be alleviated by placing a frame of mirrors around the surface, in order to reflect escaping laser light back into the light plane, but the installation of such mirrors is tedious, as another careful alignment procedure becomes necessary, and the underlying problem is only diminished, not solved. Similarly, more laser modules can be added. However, it is surprising that this effect can hardly be observed during actual operation, even in the presence of multiple users, though this could be different for applications that rely on the usage of many fingers at once.

Apart from the alignment procedure (which also is more tedious than difficult), building such a setup is very easy and does not require electronics skills, provided that ready-made laser modules are used. The parts are easy to acquire, as infrared laser modules are carried by many electronic stores at low prices. Thus, the *light-plane* approach is considerably more viable in this respect than an FTIR setup.

A laser *light-plane* setup can serve both as a design platform and for actual deployment in the field. It is easy to build at a vast amount of surface sizes, while remaining operable under varying ambient lighting conditions. Since the principle does not rely on pressure force, interaction with such a setup will feel very smooth and elegant, as users quickly realize that it is sufficient to glide their fingers across the surface, rather than dragging them. As a side-effect, the designer is thus not limited by considerations of finger friction when experimenting with gestures. Laser light-plane surfaces can thus serve as the ideal DIY setup for designers who aim to explore multi-touch interaction at a small barrier of entry.

5 Optical Tracking Software (Touché)

The viability of optical multi-touch surfaces that can be built on a tight budget and timescale is determined by the availability of free, preferentially open-source tracking software, as the major complexity of determining the positions of fingertips on the surface is concentrated within the software, rather than the hardware. Currently, there are multiple implementations of such trackers available [114, 86], including Touché [55], which has been developed as an accompaniment to this work and harnesses technologies not represented in other trackers.

The following section will provide an overview of the responsibilities of a touch tracking software that functions in conjunction with the aforementioned hardware setups. Special consideration will be given to the specific implementation of Touché, which is a widely-used touch tracker for the Mac platform and has been available as open-source software since October 2008.

The role of the tracker can be roughly divided into deriving information about touches, at least their coordinates, but possibly other information such as motion and acceleration as well, and transmitting this information to client software, which implements an interaction on top of this touch data.

From the perspective of computer vision, the process of calculating tracking data from each camera image can be further broken down into an *acquisition stage*, which refers to the digitizing of images provided by the vision camera, a *pre-processing stage*, in which the image is filtered and enhanced, so that the optical touch response is amplified, *segmentation and correlation stage*, in which the software determines the coordinates of individual touches and correlates them to the position of touches in previous camera images, as to provide consistent tracking across time, and a *projection stage*, in which the reference frame is mapped from the coordinate system of the camera into the coordinate system of the multi-touch surface.

Touché implements each of these stages in its own processing thread, ensuring that potential for parallel execution can be fully exploited on modern multi-core and multi-cpu systems. For instance, a new video frame can already be preprocessed while the vision algorithm is still deriving touch coordinates from the previous one. This architecture supports smooth operation even at very high frame rates that can be achieved with professional computer vision cameras.

5.1 Image Acquisition

The process of capturing images from the vision camera and transferring them into memory for further processing is generally provided through camera drivers, which run within the operating system and access the camera through its hardware interface. On modern systems, pixel download usually happens with a DMA (direct memory access) transfer, where data is streamed into a memory region by the system bus, rather than the CPU, so that a steady stream of images is provided without using up CPU time and at extremely low latency. Professional computer vision cameras can be set up to begin streaming a picture as soon as exposure is finished, generally resulting in latencies well below 20 milliseconds. These latencies can be further reduced by the employment of specialized frame grabber hardware, which interfaces directly with the system's main bus, rather than a supplemental bus such as USB or FireWire, and can achieve latencies below 1 milliseconds. However, such hardware is extraordinarily expensive, offsetting its value for applications where extremely tight timing constraints are not of concern.

A particular feature of professional cameras that is generally not found in consumer webcams is the presence of an electronic trigger input, which replaces the built-in software trigger to start image exposure. Rather than defining the requested frame rate in software, an electronic signal is given in order to initiate image exposure within the camera. This opens up the ability to trigger multiple cameras simultaneously, which is useful when multiple cameras are used to survey the touch surface: By ensuring synchronization between the cameras through the trigger signal, the it is guaranteed that each image captured by an individual camera refers to the same point in time, thus increasing the robustness of an underlying image stitching algorithm. Additionally, the near-infrared touch response mechanism can be pulsed by an electronic circuit, which can then send a trigger signal to the camera to ensure that the camera captures alternating images of the touch surface with the illumination turned on and off. From these images, a difference image can be created that contains only those areas that are actually affected by the touch response illuminant, greatly improving the robustness against ambient light at the cost of a reduction of the effective frame rate. However, by using a fast camera and an irregular pulsing pattern, a sufficient frame rate can still be achieved. For instance, if a 60Hz camera is used with a pulsing pattern of off:on:on, the effective frame rate remains at 40Hz. Pulsing can also be used to drive the illuminant at a higher power, since less strain is put on the electronics than when operating continuously.

Touché uses three different libraries to access camera hardware: *QuickTime Kit* [47, 7] is a modern framework provided by Apple that enables image capturing hardware access in a highly efficient manner while supporting a vast array of different cameras, ranging from low-cost USB cameras to powerful computer vision cameras connected to a FireWire bus, or even DV recorders. While hardware support and performance are particular advantages of QTKit, it suffers from an inability to access advanced camera features, such as manual exposure and gain controls as well as external trigger settings. However, this is often beneficial for a multi-touch tracker in order to achieve good contrast in the tracking video feed.

Another camera library is *libdc1394* [3], a highly performant open-source library to access IIDC-compatible FireWire cameras. In contrast to QTKit, libdc1934 provides facilities to access the camera's control registers, so that manual settings become possible. Additionally, trigger input modes can be set, so that Touché is able to support difference imaging in conjunction with a simple trigger box based on open-source hardware [9], which has also been developed together with Touché. However, the flexibility provided by this library is offset by its providing of only low-level image capturing functions. As an example, libdc1394 is only able to provide pixel data in the format that the camera sends in, requiring the programmer to manually convert the pixel format if a different one is required further down the processing pipeline. Touché supports pixel format conversions using Intel's Integrated Performance Primitives [108], which is a highly-optimized vectorized library that can, among many other operations, speed up pixel format conversion incredibly compared to naive implementations by harnessing features of modern CPUs, such as short-vector code as well as parallel execution units. Thus, this limitation of the library is not an issue with Touché. In fact, libdc1394 is the preferred image acquisition source, as it marries finely grained control over camera parameters to fast performance at low CPU usage.

Touché also supports image acquisition with a Wii Remote [65], which is part of Nintendo's Wii entertainment system: It contains a fast near-infrared camera operating in the 940nm wavelength range, which can transfer data to a computer wirelessly over the Bluetooth standard. Since it can be purchased off-the-shelf, it can serve as a viable way to quickly set up a near-infrared vision system. However, it should be avoided for professional systems, as the accuracy of its imaging sensor is low due to high quantization errors.

5.2 Pre-Processing

Before a video frame is fed to the computer vision algorithms, it is generally beneficial to process it with various filters in order to make it easier for the vision system to correctly identify regions of optical touch responses. Common examples of such pre-processing filters are noise reduction, which aims to reduce camera noise within the image, or contrast normalization, which ensures

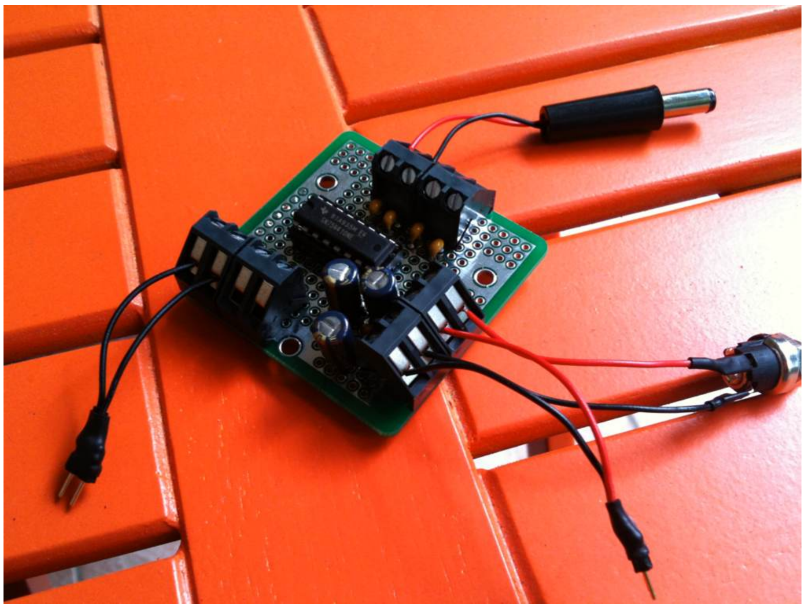

Figure 37: Touché hardwaretrigger box for *Arduino* [9]

that the entire range of available contrast is used within the image, so that individual features become easier to identify.

Commonly, such filters are implemented to run on the CPU, either as a naive per-pixel implementation or by utilizing optimized libraries such Intel's Integrated Performance Primitives [108]. Another, more modern approach is to harness the immense pixel processing power implemented in modern GPU (graphics processing unit) hardware, which is present in most reasonably recent computer models. On one hand, offloading the filtering process to the GPU frees up the CPU to do other tasks, increasing the potential for parallelized processing, but on the other, the GPU with its many parallel execution units and deep pipelines is much more suited for pixel processing, which does not rely on advanced code branching mechanisms such as conditional execution, but is rather performed in a streaming, pipe-lined fashion. Thus, pixel processing on the GPU is typically an order of magnitude faster on the GPU than on the CPU, while providing equal accuracy and quality and freeing the CPU for other tasks.

Touché renders all pre-processing filters on the GPU by default, using the powerful Core Image framework [6], which simplifies the task of uploading an image to the GPU and processing it by so-called pixel shaders, which are code fragments that perform per-pixel transformations. For instance, Core Image enables the programmer to access multiple pixels of the image within the same code fragment, takes care of necessary tiling operations if the image does not fit into GPU memory as a whole, and compiles shader code into the most optimal form for each particular GPU, ensuring to use only those functions that are directly supported in hardware. Thus, Core Image-based filters offer unparalleled performance, ensuring that Touché remains extremely efficient, even if multiple pre-processing filters need to be chained together in order to achieve a usable result.

While Touché implements a multitude of simple filters such as color inversion (so that dark touch responses such as when tracking finger shadows can be used in same manner as bright touch responses) and contrast stretching (in order to enlarge image fidelity, so that individual touch responses become defined more clearly), it also provides filters executing more elaborate commands, which are typically used in vision systems.

Background subtraction [119, 97, 73] is commonly used in computer vision systems to make individual features stand out: By defining a background frame, which shows the camera image without any feature (in the case of a multi-touch tracker a touch response), this background can be subtracted from each subsequent frame in a pixel-wise manner, so that only those areas remain in which a pronounced difference to the background is visible. Thus, the vision algorithm can identify these areas more easily. The background can subsequently be updated in order to remain valid even as ambient lighting conditions change. Touché can snap a new background image whenever it does not recognize any touch responses in a video frame, so that it can be assumed that the frame merely shows the background. Additionally, the new background can be created by merging the previous one with the newly acquired image at a user-defined ratio, reducing the effect a "bad" background (such as one in which a touch response is visible, but not recognized, or when ambient lighting is currently at unusual levels) can have on overall operation.

High-pass filtering [39] is an image processing step which aims to retain only those areas of the screen which contain uniform brightness levels with sharply defined borders. It is thus very well suited to hardware approaches like *diffuse illumination*, which rely on the property of the diffusor sheet that objects above it, but not touching it, appear with blurry edges. By removing these areas through high-pass filtering, it is ensured that the vision algorithm only detects those areas that are actually in contact with the surface and thus sharply defined. A high-pass filter can be implemented by subtracting a strongly blurred copy of the frame from the frame itself: Since areas that are already blurry will hardly be affected by the additional blurring step, they will disappear through the subtraction, whereas sharply defined areas pull in enough dark pixels from their surroundings during the blurring process that they become darker, thus leaving behind a faint residual after the subtraction process. By connecting the high-pass filter to an amplification filter, which will multiply each pixel with a preset value,

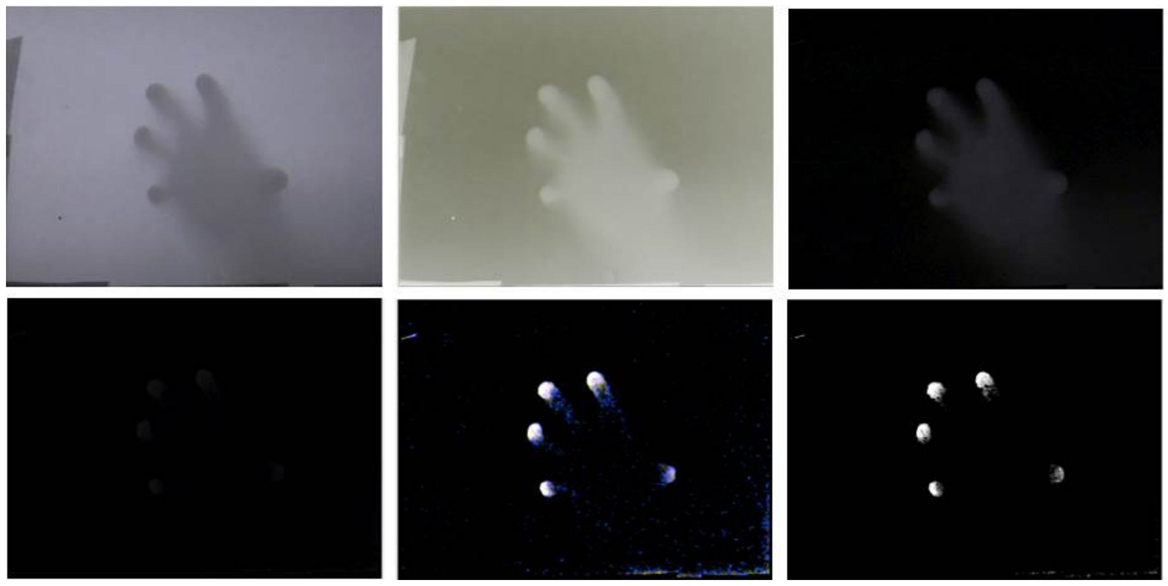

Figure 38: Overview of some filters available in Touché

the areas remaining after the high-pass filter will be pulled up back to reasonable brightness levels, so that they can be picked up by the vision algorithm. Since high-pass and amplification require many intermediate steps, they benefit especially from the highly parallelized execution on the GPU.

Thresholding [112, 96, 105, 48, 124, 46] is often the final step in a pre-processing pipeline. It refers to the transformation of the video frame into a binary image by defining a luminance threshold at or above which a pixel will be mapped to a 1, whereas lower luminance values will be mapped to a 0. A vision algorithm can detect connected regions of 1s within a binary image in an efficient manner, deeming this a viable step to facilitate image segmentation. However, for large surfaces that are unevenly lit or that suffer from spatial differences in ambient lighting, a single naive threshold might not be able to achieve good results, being too low for brighter regions and too high for darker ones. Thus, *adaptive thresholding* is often used: By keeping a background image ready in the same vein as the background subtraction filter, each pixel value is compared to the corresponding value in the background image by luminance. The threshold is not defined on absolute luminance values anymore, but on normalized difference values between current luminance and background luminance. For instance, the threshold can effectively be formulated as mapping those pixel values that are a certain percentage brighter than in the background image to a 1, whereas all other ones are mapped to a zero. Since a percentage threshold is not dependent on the effective luminance value itself, the threshold remains viable both in brighter and darker areas, countering the detrimental effects of uneven illumination. Touché implements both naive and adaptive thresholding on the GPU.

Typical vision applications may provide many other filters in addition to the ones mentioned, depending on the settings in which the tracker should perform. Touché additionally offers *sensor noise reduction* through blurring and by ignoring green and blue pixels in color imaging sensors with a Bayer pattern [11] when they are used in the near-infrared spectrum, *contrast stretching*, which maps the darkest value in the image to the lowest possible luminance value as well as the brightest one to the highest one, linearly stretching the entire luminance range accordingly and consequently greatly increasing overall contrast, *contrast boosting*, which (over-)emphasizes bright areas in the image over darker ones, *color thresholding*, which implements a threshold based on color distance in the RGB space rather than overall luminance, so that color tracking in the visible domain is possible as well, and morphological operations such as *erode and dilate*, which are useful to reduce noise pixels in a thresholded image, which can be caused by bad choices of the threshold, camera noise or dirt on the lens.

5.3 Segmentation & Correlation

At its core, the job of the vision algorithm is to describe each touch response as a set of coordinates and to interrelate individual touch responses across multiple video frames, so that each touch can be tracked during its entire lifetime. This is achieved by assigning an identifier to each detected touch, which remains constant during the touch's lifetime. Thus, the algorithm needs to be able to detect the new position of an existing touch in subsequent video frames, so that the same identifier can be assigned.

There is an abundance of algorithms to calculate the position and extent of connected regions in a binary image like the one provided to the segmentation algorithm by the pre-processing stage. Typically, a connected region is detected by scanning the image in a line-by-line fashion: As soon as a “1” pixel is found, the algorithm traces the region by checking the immediate 8-neighborhood (vertical, horizontal and diagonal pixels) in order to find the direction in which the region extends. Once the originating pixel is reached again, the region has been traced completely, and the size as well as the centroid of the region can be computed. This process can be implemented in a highly efficient manner. One particularly efficient open-source implementation is found in the OpenCV library [14], which is used by Touché as well as many other trackers to detect individual touch responses.

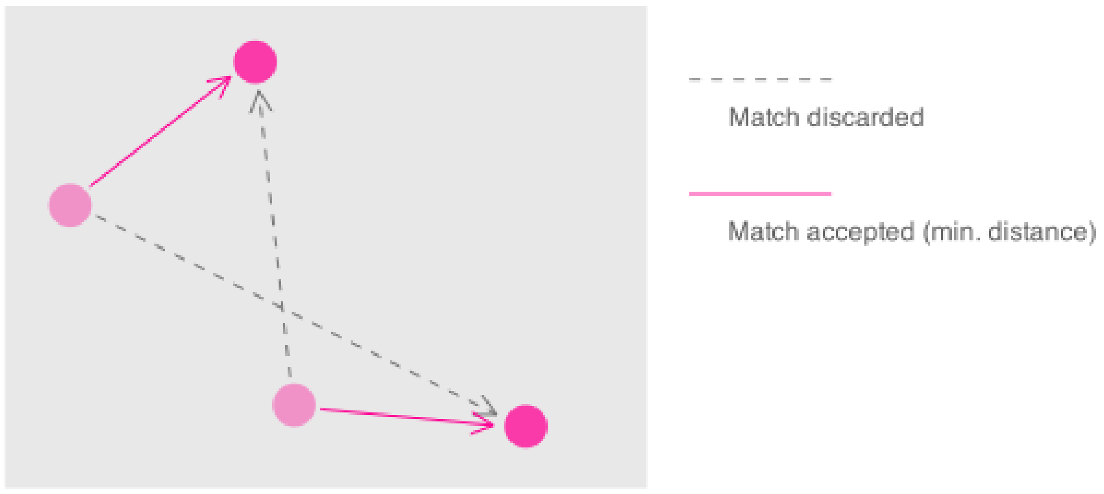

Figure 39: Illustration of k-nn distance-based touch inter-frame correlation

However, knowing the centroid of each touch response is not sufficient for multi-touch tracking, where continuous tracking over the entire touch lifetime is crucial. Thus, an algorithm is needed that can assign the same identifier to each occurrence of the same touch in subsequent video frames, so that the identifier can be used to reconstruct the touch's motion across the surface. Since the multi-touch hardware setups presented previously are not able to deliver information about a touch other than its position in each video frame, the algorithm can only be based on spatial information.

The approach used by Touché as well as most other trackers is based on spatial distance: By using a *k-nn classifier* [22, 32, 59] in which the identifier is regarded as a touch's class, and the learning set is composed of touch positions in the previous frame, those touches are mapped to the same identifier in consecutive frames that are closest to each other. k is usually chosen to be 1, so that only the previous frame is considered when looking for a match. While other odd values could be chosen for k (as even values might lead to a tie of votes between different identifiers), empirical results suggest that this is not beneficial to the accuracy of the approach. As a distance norm, Euclidian distance is usually chosen, since both camera and touch surface coordinates are Cartesian, and the surface itself supports finger movement across arbitrary angles in relation to the principle axes.

Naturally, a simple approach based on locality alone is bound to suffer from classification errors in certain special cases, such as when two touches are crossing over each other during the course of a gesture or move fast towards each other. Thus, additional information about a touch can be considered during the classification process in order to provide an additional layer of robustness. A viable extension is to base the calculation of inter-frame touch distances on positions predicted by taking the motion vector and acceleration vector into account: The motion vector can be defined as the distance that a touch has travelled between two consecutive frames, standardized by the time passed between the two video frames, and the acceleration vector can be derived from the distance between two consecutive motion vectors. By using these measures to predict the position of a blob in the current frame, based on its position in the last, the distance between predicted positions and detected positions can be used in the execution of the k-nn algorithm. With this approach, issues stemming from touches moving into close proximity to each other or crossing over each other can be alleviated, so that touch correlation becomes highly reliable.

However, a second problem arises as well: If a touch response disappears in the same video frame as another one appears (such as when a user removes one finger from the top left corner while putting down another finger in the bottom right corner), a simple k-nn classifier will match these two responses, even though their relative distance can be unrealistically large. However, since the responses do not refer to the same finger, the match is a false one. While this issue can be partly alleviated by utilizing a faster camera frame rate, so that the occurrence of the phenomenon becomes more unlikely, the introduction of a maximum distance threshold for a successful match becomes necessary to provide stable identification under this circumstance. Choosing this threshold is difficult and can vary greatly, depending on the anticipated speed of finger motion that the system should support: If the threshold is too high, false matches will occur, but if it is too low, fast glides of a single finger will be interpreted as multiple unrelated touch responses. Thus, this issue is a limitation of the optical touch tracking setup presented herein, which can only be improved, but not solved.

Nevertheless, the reliability of the identification process is high. Touché uses highly performant short-vector code to calculate the relative distances between all possible matches. This number can be very high: If n touches are present in the current frame, and m in the previous one, mxn possible matches have to be considered. Parallelization of the distance calculations and the utilization of vector units to speed up execution are possible since the individual distance calculations are independent of each other. Consequently, Touché can support a large number of concurrent touches without slow-downs.

If a touch response in the current video frame cannot be matched to a previous one, it is considered new and a “finger down” event is queued for propagation. Similarly, if a response in the previous frame has no match in the current, a “finger up” event is generated. An identifier is guaranteed not to be reused within consecutive video frames, so that client software receiving data from the tracker can clearly identify new, removed and updated touches by the correlation of their respective identifiers to previous data packets.

5.4 Projection Mapping

As a final step, tracked touch coordinates need to be mapped from the camera's coordinate system into the coordinate system of the touch surface itself. This is necessary because the camera will generally not provide a 1:1 mapping to the surface. Not only will the resolution by different, but so might be the aspect ratio. Additionally, the camera will often have to be mounted in a way that it is tilted, skewed and rotated in respect to the surface. The tracker needs to accomodate for this mismatch in order to provide tracking data that is uniformly accurate across the entire interaction surface.

If the multi-touch surface is aimed to be used as a screen through rear-projection or with an LCD panel, the corners of the touch surface need to correspond to the corners of the screen area. Especially when rear-projection is used, another set of matching constraints is introduced: Keystone effects can occur, which the projector might not be able to compensate for. Even if a keystone distortion is so slight that it is hard to discern with the eye, tracking accuracy can suffer considerably, as the matching error compounds when a touch moves across the entire surface.

Thus, the projective mapping needs to solve two issues: Firstly, the mapping needs to superimpose the tracking coordinate system onto the surface coordinate system. Secondly, the algorithm needs to compensate for any misalignment between camera, touch surface and, if present, screen imagery projection.

A typical approach to achieve such a mapping is the employment of a plane-to-plane homography [73] between the touch surface as seen in the camera image and the usable delimits of the actual interaction surface. Four corresponding points in the camera image as well as on the touch surface are sufficient to form a homography matrix, from which a homography transformation can be estimated by using a non-linear least-squares solution such as the Levenberg-Marquardt algorithm [79]. The result is a transformation matrix, which can be applied to any touch point coordinate in order to compute the position of a touch in surface coordinates, given its camera coordinates. Since the transformation is conducted by multiplying the homogenous touch coordinates with the transformation matrix, the process is supported in a highly per-

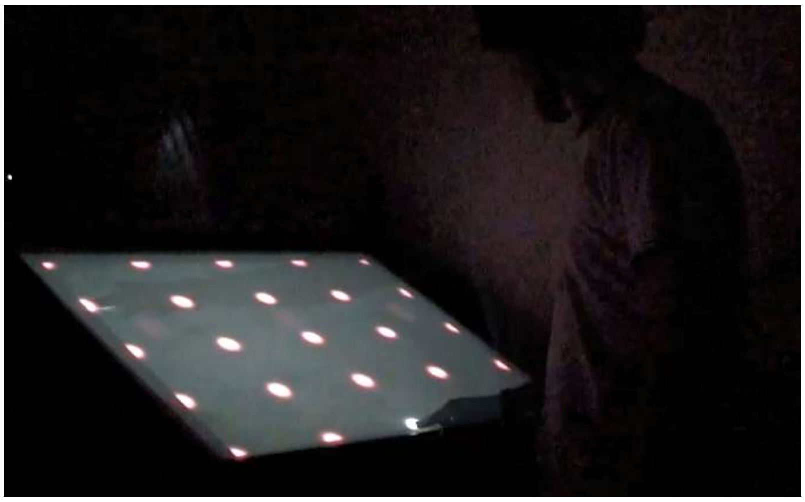
Figure 40: Calibration routine built into Touché

formant way by frameworks such as Intel's IPP [108], which provide fine-tuned linear algebra routines, including matrix multiplication. Additionally, since the transformation of individual coordinates is mutually independent, the process can be parallelized.

Since the position of the camera in respect to the touch surface is assumed to be fixed during the operation of the system, the homography estimation will only have to be done once during a calibration step. However, a plane-to-plane homography only accounts for the projective mapping between camera and surface plane, but does not consider camera lens distortion: Thus, camera calibration techniques such as the Tsai model [109, 110] and the Zhang model [127] have been developed to provide a mathematical representation of lens distortion, so that each camera image can be rectified before the homographic transformation is applied. Consequently, these models divide transformation parameters into so-called *extrinsic* parameters, e.g. the relative position between camera and tracking plane, as defined by a set of correspondence points, and *intrinsic* parameters, which refer to camera properties such as lens distortion coefficients and focal length.

While both models recognize that camera distortion is composed of both tangential and radial distortion, experiments have shown that only radial distortion is of concern to computer vision systems. This type of distortion refers to the phenomenon that straight lines appear curved in the video image, especially towards the edges of the frame. While every lens suffers from radial distortion to some degree, it is especially prevalent for lenses with short focal lengths, such as wide-angle lenses. Since the employment of such lenses allows the shortening of the light path

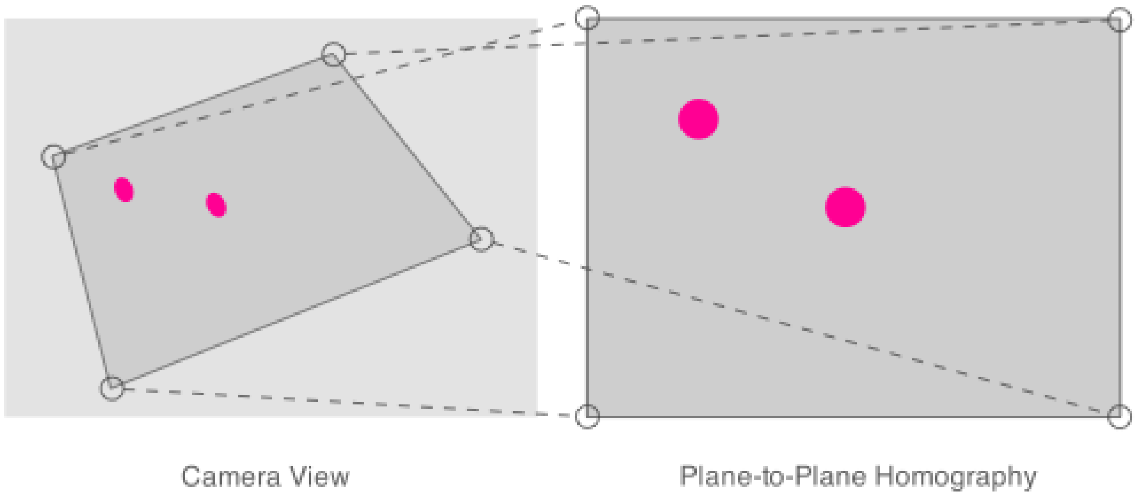

Figure 41: Illustration of a plane-to-plane homography

between the camera and the multi-touch surface, they are often used in such systems, making it necessary to utilize accurate algorithms to rectify radial distortion during the tracking phase.

Radial distortion is usually described as a polynomial mapping between distorted radius r_d and actual radius r as

$$r_d = r(1 + cr^2)$$

where c refers to a distortion coefficient intrinsic to a given lens. The inverse of the distortion function can be approximated through Taylor series expansion or Newton's approximation, so that

$$r = \frac{r_d(1 + 11cr_d^2 + 39c^2r_d^4 + 51c^3r_d^6 + \ldots)}{1 + 12cr_d^2 + 48c^2r_d^4 + 75c^3r_d^6 + \ldots}$$

Since the center of the radial distortion is the principal point of the lens projection, thus in the middle of the video frame on the image sensor, radial distortion can be compensated for by calculating the undistorted radius from the distorted one, provided that c is known. In Tsai's and Zhang's model, c is estimated from a set of correspondence points between the camera image and a known reference (such as a checkerboard image), using non-linear estimation such as Levenberg-Marquardt [79]. While the series is infinite, it converges fast enough so that only the first terms are necessary to achieve accurate results, since other factors such as camera-internal quantization put a natural limit on the accuracy of the image. In fact, using more terms introduces numerical instability due to potential estimation errors of the distortion coefficients. By implementing the complex function on the GPU, the entire image can be rectified per-pixel in an efficient manner.

While Tsai's and Zhang's models provide robust distortion compensation and projective mapping, they impose constraints that limit their applicability to some multi-touch setups. Firstly, being derived from a mathematical pinhole camera-model, the distortion correction assumes a mathematically perfect radial distortion, which is often not the case on consumer webcams, that might employ plastic lenses prone to warping, causing irregular distortion patterns. Secondly, the plane-to-plane homography does not account for non-planar touch surface shapes, such as warping of an acrylic sheet or its bulging under its own weight. Thirdly, while the models are highly accurate if their constraints are met, they are prone to delivering inaccurate results under the aforementioned circumstances.

Thus, trackers such as Touché implement an additional projection method, which is robust towards these issues: By implementing a mapping based on barycentric coordinates within a triangle mesh akin to an inverse approach to texture mapping, accuracy is achieved even if the surface is irregular and if lens distortion is not uniform. This is inspired by the idea of estimating homographies not across the entire surface, but within small subdivisions, so that errors cannot compound if the surface is highly irregular. While the approach is not able to cope with severe radial distortion due to its linearity, radial distortion correction as detailed above can be chained in front of the coordinate conversion, whereas the latter step will take care of residual uneven distortion.

The implementation collects the vertices of a tetragonal mesh as corresponding coordinates in the camera and screen domain. Each quad is then split across its diagonal to form two triangles, which are stored for reference. For each point that needs to be transformed, the triangle in camera coordinates is found within which the point falls. This is achieved by calculating the centroid of each triangle in advance. For a given point, its barycentric coordinates within each triangle are calculated by trying each triangle in the order of smallest distances between the point and centroid. It can be shown that this approach will always find the fitting triangle within at most three attempts (even though this worst-case scenario rarely happens in practice), so that the amount of calculations necessary to transform a single point remains sensible as the amount of calibration triangles increases, since distance-sorting can achieved through highly-efficient short-vector code. A triangle is deemed fitting if the barycentric coordinates u, v and t of the point based on the triangle's vertices are positive, and their sum equals 1.

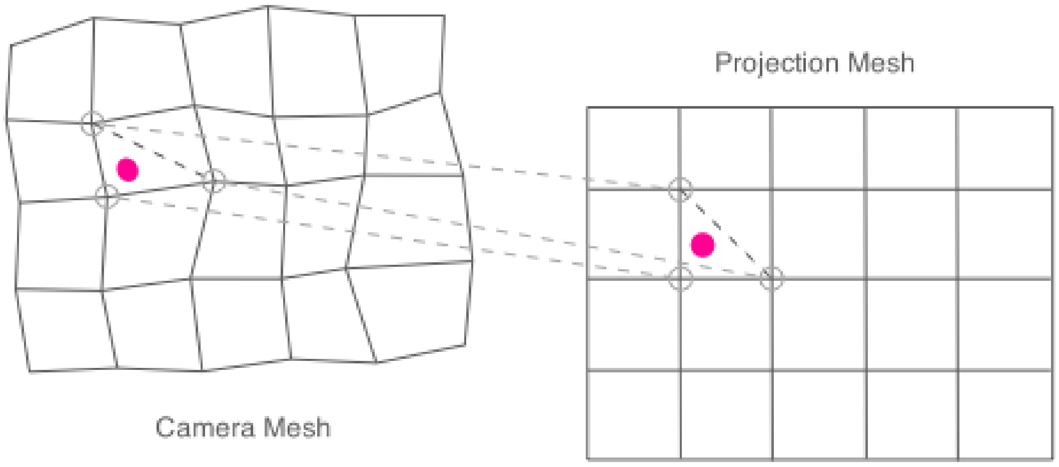

Figure 42: Illustration of mesh-based projective mapping

In detail, given triangle vertices $(x_1, y_1), (x_2, y_2), (x_3, y_3)$, the barycentric coordinates of point (p_x, p_y) can be calculated as

$$area = (x_1 - x_2)(y_1 - y_3) - (x_1 - x_3)(y_1 - y_2)$$

$$u = \frac{(p_x - x_2)(p_y - y_3) - (p_y - y_2)(p_x - x_3)}{area}$$

$$v = \frac{(x_1 - p_x)(y_1 - y_3) - (y_1 - p_y)(x_1 - x_3)}{area}$$

$$t = 1 - u - v$$

Once the barycentric coordinates have been obtained from a triangle within the camera domain, the corresponding triangle vertices in the surface domain are used to calculate the transformed point position. The transformation is performed on the basis of triangles (rather than tetragons) in order to simplify the necessary calculations. Thus, since the transformation of individual points can be parallelized, the operation can be implemented in a highly efficient way.

By combining the inverse mapping approach with radial distortion correction, high accuracy can be achieved, even if the relative position between surface and camera is tilted, skewed and rotated, and the surface itself is warped as well. While it can be argued that fingers are a coarse interaction device and thus do not require high accuracy, it has to be noted that it is imaginable to use finer devices, such as light pens, together with the tracker, where the additional accuracy is welcome. Even more so, an accurate transformation ensures that errors in the camera image, such as excessive noise or quantization errors, do not contribute significantly to the transformation process.

Since the transformation from camera coordinates to surface coordinates will often require up-sampling if the multi-touch surface is used with screen imagery, as screen pixel resolutions usually exceed camera resolutions, the results can exhibit jittering across consecutive video frames, due to camera noise or touch response hardware issues. This leads to touch coordinates moving slightly, even as the finger remains stationary. The issue can be alleviated by introducing Kalman filtering [118] for transformed touch positions: A Kalman filter is a mathematical model that aims to deduce a steady measurement from a set of noisy samples by modeling motion and acceleration characteristics. If a Kalman filter is applied to each coordinate of a touch point, the results are smoothed accordingly and jitter is reduced significantly.

5.5 Tracking Data Distribution

With the coordinate data ready, the tracking software needs to distribute it among client processes. This is a sensible approach, since actual interactions will rarely ever run within the same process space as the tracker. By separating both tracker and client, a flexible development approach is possible, in which multi-touch experiments can be developed as client applications independently of the tracker. At the same time, a shoddily programmed client cannot harm the tracking process, thus enabling a rapid prototyping approach in which the designer does not need to fear to harm the tracker or their hardware touch surface. In some settings, such as when the touch surface is very large or requires a lot of parallel cameras (and consequently CPU power), it can be beneficial to run the tracker on its own computer, distributing touch data over the network.

Ideally, the interface between tracker and client should be based on an open standard, that is well documented, so that different trackers and clients can be implemented in an interoperable way. Additionally, the reliance on a standard enables the creation of simulation tools, which can be used on devices not capable of multi-touch input (such as ordinary desktop PCs) in order to prototype or test touch-enabled applications. Thus, a group of designers and programmers does not need to invest time and money in the building of multiple hardware setups, while being able to develop and test applications in a meaningful way.

TUIO (Tangible User Interface over OSC) [56] provides such a de-facto standard, which is widely in use among both the scientific and enthusiast community and allows for the creation of interoperable touch-enabled applications. Initially designed for use with projects in the field of *tangible computing* [53], it also provides ample facilities to represent touch tracking data through a dedicated "cursor" protocol, where each cursor is described by its current position, motion and acceleration as well as an identifier which remains constant during the cursor's lifetime. Multiple cursors can coexist at the same time, allowing for multi-touch interaction.

TUIO is implemented on top of OSC (Open Sound Control) [125], which is a protocol format defined to fit the needs of audio applications, namely low latency and efficient de- and encoding. While the protocol itself is transport-agnostic, it is usually transported over UDP/IP, which provides for efficient transport due to lack of flow control overhead such as introduced by more advanced protocols like TCP, but is susceptible to packet loss. Thus, TUIO is designed in a way that the loss of an arbitrary amount of packets does not confuse the client application: By using *isAlive* messages, which simply contain a list of identifiers referring to cursors that are currently being tracked on the surface, as well as *Set* messages, which contain the actual tracking data of each individual touch, accessed by its identifier, the loss of either a *Set* or *isAlive* can not introduce an ambiguous state of the surface: If an *isAlive* message is lost, the adding or removal of touches will be delayed until the next *isAlive* is received, letting the surface state remain in defined state. Similarly, if a *Set* message is lost, the tracking data from the previous message persists until a new one is received. Thus, packet loss can only cause a jittery motion of individual touches as well as their delayed appearance or disappearance, but cannot introduce ambiguities that depend on network factors or race conditions.

Additionally, TUIO libraries are available for many common programming languages, which translate *isAlive* and *Set* messages into events according to the convention of the language, such as "finger up", "finger down" and "finger moved", greatly facilitating development. However, [4] Adobe Flash is a notable exception from this list: While support for Flash is deemed important due to its popularity among designers that come from a non-technical background, it is based on a stringent security model that does not permit access to UDP network traffic. While Flash provides different ways of accessing TCP-based traffic, either as bytestream or wrapped within XML envelopes, TCP introduces latencies that are severely detrimental to the responsiveness of the system.

Therefore, an alternative approach to tunnel tracking data has been developed in conjunction with Touché: By accessing a shared memory region that Flash reserves for internal use, tracking data can be copied into the Flash runtime process directly, circumventing the network as well as the security model. While this approach is not officially sanctioned, it provides an extremely efficient and fast way to transfer tracking data, allowing designers to use Flash with its rich visual expressivity and powerful vector graphics engine as a rapid prototyping tool for multi-touch applications. Due to its reliance on shared memory, the tracking process needs to run on the same physical computer as the Flash application. Therefore, a bridge tool has been developed as an accompaniment to Touché, which can copy TUIO data received over a UDP/IP network into a locally running Flash process, so that the tracker no longer needs to run on the same machine. Due to the fast transfer through shared memory, the latencies introduced compared to a direct UDP approach are negligible.

However, if the tracker and client application can run on the same system, tunneling tracking data over UDP introduces unnecessary latencies, as even the operating system internal loopback device needs to perform an entire execution of the networking stack. Since Touché is efficient enough to use only a few percent of available CPU power on an average computer, running a client application on the same system is certainly a viable way to provide for shorter latencies.

Thus, Touché also allows the distribution of tracking data locally over *Mach ports* [1], which are a highly efficient inter-process communication mechanism on Mach systems. Latencies are extremely low in this approach, being hardly measurable with available performance timers. By wrapping the setup of the Mach port up into a single function call, receiving tracking data in this manner is extremely comfortable for the client application programmer. Additionally, Touché uses the encompassing framework to provide additional programming utilities that facilitate the rapid development of multi-touch interactions, such as combinatoric index generation for the testing of individual combinations of active touches for gesture criteria, highly efficient recognizers for common gestures such as the *stretchie* [42] or tap counting, as well as supplemental facilities for common tasks such as the setup of a hardware-accelerated full-screen OpenGL context. With these powerful tools readily available to the programmer or designer for use within their tracking client, Touché's framework is deemed to be a useful platform for the exploration of multi-touch interaction.

6 Future Work

Departing from the state of the art of multi-touch interaction research, a vast field of future opportunities is opened up. Considerable research will have to be spent on the development of an interaction vocabulary for multi-touch surfaces which is not bound to a specific application. Currently, designing a multi-touch application requires a lot of guess-work on the part of the designer [81], as a generalized vocabulary has not yet been agreed on. Apart from a selected few gestures such as the *stretchie* [42], the behavior of different multi-touch applications is currently highly inconsistent, with different development teams utilizing touch semantics that are arguably fit for the given task at hand, but at the same time prevent users from transferring their knowledge of one multi-touch system to another.

Thus, it will be fruitful to evaluate different proposals of single interaction techniques for their potential of being merged together into a unified set of semantics for multi-touch interactions, identify potential points of conflict and resolve them in a process of repeated prototyping and user study. For instance, while ample research has been put into the conception of selection techniques such as *Drag-and-Pop* [10] as well as menu behaviors such as *FlowMenus* [34] or *Cor2Ds* [99], their potential for integration into a single system has not yet been explored. Since the success of WIMP-based graphical user interfaces is largely based on the successful marriage of the indirect mapping between mouse and cursor as well as a multitude of clearly defined widget behaviors such as buttons, drop-down menus or scroll bars [67], a similar vocabulary will be vital to establish multi-touch interaction as a widely accepted paradigm rather than a novelty experience.

Additionally, touch interaction can be applied to surfaces of vastly differing sizes, ranging from handheld pocket computers [5] [20] to large wall-like installations [76]. Attempting to develop interactions that can scale between such devices in a meaningful, understandable and usable way will be a viable direction of research, enabling users to transfer their knowledge of one system to another. As an example, the two-contact scaling operation is usually performed with thumb and index finger of a single hand on small screens, but with the index fingers of both hands (by stretching the arms) on large displays [42]. It is interesting to see how users find it intuitive to switch between both modes of executing the same gesture without having it feel awkward on either device. Therefore, a sensible approach could be to analyze such interactions for the underlying principles that enable the smooth transfer and attempt to generalize them, so that similar interactions can be developed for both small- and large-scale operation.

Similarly, little research has been made into the potential of the integration of multi-touch interaction with other devices: While touch interaction is sometimes combined with pen interaction [35, 37] or tangible objects [113] [31] [53], other devices such as mice (!), 3D navigators or trackballs have not yet been considered. While this can partly be explained by the fear of many designers to sacrifice the naturalness of pure touch interaction by diluting it with the (re-)introduction of another interaction device, such a combination can be a fruitful approach to reducing the impact of interactions that are perceived as awkward on a multi-touch surface, such as some manipulations of 3D imagery. The ability to keep one hand on the surface while using the other to operate an additional device could certainly be a valid departure point for the inception of novel interaction techniques.

Another pet peeve of multi-touch research is its close connection to touch screens: While some research has been made into indirect mappings for touch interaction [73] [16], its potential as an interaction technique in the context of *ubiquitous computing* [115, 116] and *calm technology* [117] has largely been left unexplored: With technology such as capacitative touch sensing readily available [24, 94], touch-sensitive surface can be embedded into everyday objects such as walls or tables, allowing for seamless interaction with an ubiquitous computing system through predefined gestures. For instance, the surface of a couch table could be used as an input device to control a music player, skipping a piece by flicking a finger, or sliding across to regulate playing volume. While such scenarios are frequently encountered in science-fiction, their actual feasibility from the perspective of usability and user experience has rarely been considered.

In the same vein, most designers have constrained themselves to rectangular touch surfaces and screens, possibly inspired by the rich tradition of rectangular displays. However, some projects like reactable* [53] have shown that different shapes, such as a round table-top system, provide interesting design opportunities as well. Incidentally, round tables are a common sight in everyday collaboration, yet they have rarely ever been implemented as touch-sensitive devices or screens. Even more so, other shapes have not even been explored at all, such as a disc-segment screen which encompasses the user like a machine console, providing the ability to use parts of the interface for peripheral vision or as an externalized visual memory, where documents or windows can be stored for later access. The shape possibilities enabled by the hardware setups introduced before are endless, giving designers ample opportunities to envision setups going beyond the widely established rectangular layout.

Conversely, a multi-touch surface does not need to be flat either: Technologies such as the *FTIR* sensing method [39] or developments such as *SmartSkin* [94] allow for screens that have an uneven topology, such as domes or curves. For instance, a device could be realized in which the touch-sensitive surface is curved like an L, so that the lower, flat area could serve as a resting and working area for the user's hands, while the upright area could be used much like a screen, albeit touch-sensitive as well. Since the curve between the two segments is part of the display, she could seamlessly shift digital documents between both areas, allowing for some interesting interaction scenarios. The immense image processing power of modern GPUs can be used to compensate for the distortion caused by projecting onto such an irregular surface, warping the image accordingly, so that it appears straight on the projection area. Thus, there are hardly any boundaries for the imagination of the designer.

From a technological perspective, trackers can be evolved to provide opportunities to track hands, arms and users above the surface as well, allowing for behaviors that are not purely tied to touches alone. For instance, a device could automatically orient a digital document towards a user reaching for it, or simple gestures could be implemented above the table, such as flicking the hand from the wrist to page through a text document, while finer control granularity can be achieved by reaching down and touching the surface. While such tracking approaches have already been developed [98, 112, 25, 97, 8], their potential as a design tool has largely been left unexplored.

Current multi-touch surfaces suffer from the fact that accurate, responsive and noise-free touch tracking on large surfaces is only possible with optical systems, since the signal-to-noise ratio in other approaches falls quickly as sensor complexity increases [24, 94], so that either resolution or size has to be compromised. However, optical systems, usually requiring a sufficient light-path for a camera, are large and unwieldy. Thus, an important field for technology research will be the finding of ways to implement thin multi-touch surfaces and screens that scale to large sizes while providing highly accurate and responsive tracking. With developments such as *ThinSight* [46], LED-based tracking coupled with the advent of OLED display technologies [38, 48] as well as design studies such as *PlayAnywhere* [124] leading the way, creating thin surfaces as well as portable units that combine tracking and projection in a manageable form factor will be a lively field of future work, focusing both on the underlying technologies as well as the usage scenarios enabled by being able to easily deploy multi-touch surfaces anywhere, or even being able to turn nearly any surface into a touch screen by flipping a single switch.

More work will also have to be put towards the creation of programming tools that are suitable for the inception of multi-touch enabled applications: While protocols such as TUIO [56] and its associated software libraries as well as the Touché framework [55] provide programmatic access to tracking data through a convenient application programming interface, they are operating at the very low level of individual touch events. However, in order to provide a more advanced platform for the fast development of multi-touch applications, a high-level touch layer will be needed, which has a more holistic approach to touch interaction, working on the level of clusters and complete gestures, rather than individual coordinates. With higher-level approaches such as *DiamondSpin* [101] showcasing the potential that powerful frameworks can have on the shaping of multi-touch experiences both at design and development time, the demand for such interfaces will grow steadily, as multi-touch interaction is adopted into commercial products at an increasing rate.

Finally, Touché as a powerful groundwork based on cutting-edge technologies such as on-GPU processing and multi-core optimization will continue to evolve as a tracker. Future versions will open up the rigidity of the tracking pipeline, enabling users to compose their own tracking pipelines for different tasks at hand: The chain of camera filters, the vision algorithms as well as the data distributors can be patched together by the user through an intuitive interface. At the same time, Touché's image acquisition and filtering engine will serve as a base for more advanced tracking algorithms, such as the ability to track hands in the three-dimensional space above a touch-enabled surface, track a user's gaze across a multi-touch screen and to provide both rear-camera based tracking and direct hand tracking on the surface. Additionally, a plugin-based architecture will be adopted, through which designers can rapidly develop their own components for the customized pipeline, such as custom tracking algorithms harnessing Touché's filtering engine, support for many different camera models and hardware trigger boxes, custom filters and calibration routines. In the long term, the goal is to augment Touché from being a tracker software into a tool to build other trackers out of a vast array of possible components, facilitating designers and computer vision researchers alike to experiment with new optical tracking technologies that they envision, much in the same vein as the current incarnation of Touché serves as a toolkit to explore multi-touch interaction.

7 Conclusion

Multi-touch interaction is at an exciting point in its development stages, having already proven its viability in successful and visionary products such as Apple's *iPhone* [5] and Microsoft's *Surface* [19], yet the definition of a unified interaction vocabulary has not been completed. Thus, while the prospects of touch-enabled surfaces are certainly promising, it has not yet been decided whether multi-touch interaction is here to stay or will be edged back into the role of a novelty experience.

In fact, advances in touch-sensing hardware have been steady, with many different approaches reaching maturity and being pushed into the market. However, research in multi-touch interaction design is still very much fragmented, with little initiative to provide techniques and metaphors that have been shown to be able to coexist in order to form a vocabulary from which more complex applications than technology demos such as photo viewers or trivial games can be built.

Nevertheless, the tremendous potential for perceived "naturalness" of touch interaction and its abundance of design opportunities in respect to beneficial experiential qualities serves as a motivation for designers with non-technical backgrounds as well to explore the design space being spanned up by the advent of this interaction technique. It has been argued that the ability of touch surfaces to remove the visual and spatial conflict between data and tools by placing the data itself at the core of the interaction, with gestural input as well as on-demand menus replacing toolbars and subwindows, offers rich design possibilities that are hardly imaginable with traditional interaction techniques. In the same vein, the larger amount of degrees of freedom afforded for by the utilization of all fingers of both hands allows the designer to rethink interactions that have previously been had to be decomposed into a sequence of individual steps due to the inadequacy of input devices. For instance, with the two-contact point technique, a digital document can be scaled, rotated and translated on a multi-touch surface through a single swift gesture, whereas a mouse-based interaction would have to rely on mode switches to provide the same effect, leading to a significantly less fluent user experience. Thus, the adoption of multi-touch technology does not only allow the inception of new interaction paradigms, it also inspires the rethinking of existing ones.

At the same time, surfaces such as walls and tables are already rooted deeply in our everyday life, inviting their technological augmentation with touch sensors to realize some of the prospects of *ubiquitous computing* as well as facilitating and possibly transforming traditional collaborative work settings. Multi-touch surfaces have already been used successfully to shift work meetings and discussions from a purely analog realm into one in which the analog world can blend into the digital one, avoiding much of the friction introduced by laptop meetings.

Finally, different interaction techniques based on multi-touch input have been presented, both for individual interaction and multi-user settings. Describing the state of the art of touch interaction design, these techniques can serve as a departure point for eager designers to experiment with touch technology within the context of different settings and applications.

However, the absence of affordable hardware at larger surface sizes limits the audience of multi-touch interaction research, with commercial surfaces outside the realm of many budgets. Thus, several simple hardware setups have been introduced that can be built on a low budget, putting them well within the reach of even casual interest. Additionally, the setups do not require a technological background to build, but can mostly be assembled from ready-made parts that are convenient to obtain. Within a few afternoons of work, any designer can harness these technologies to build a multi-touch surface of arbitrary size and shape, which can even be combined with a common projector or LCD panel in order to form a multi-touch capable screen.

In order to achieve a simple hardware design, cameras are used to track fingers as they are coming into contact with the surface. The major work of the touch sensor is thus done by a so-called tracker software, such as Touché, which has been developed in conjunction with this work and has been available as a free, open-source download since September 2008. It has widely been adopted both by its intended audience, enthusiasts and design collectives interested in the exploration of the opportunities afforded for by multi-touch interaction, but also by commercial projects, art installations and event attractions.

The aim of this work is fourfold: First, the motivation behind the exploration of multi-touch interaction has been detailed, focusing on modern trends in human-computer interaction such as experiential qualities, but also connecting back to established theoretical frameworks such as *activity theory*, as to prove the viability of multi-touch as a serious interaction technique with a vast potential for successful applications. Second, an overview of the state of the art of both multi-touch hardware and sensor technology as well as interaction design has been given, presenting many of the most influential results in these fields. Intended primarily as inspiration and departure point for future work, a reference frame for the direction of contemporary multi-touch research has been established. Third, different approaches to building affordable camera-based multi-touch surfaces and screens without requiring prior knowledge in electronics or computer science have been given, providing prospective designers with a guide on how multi-touch interaction design can be made available in an accessible and convenient form, enabling them to build devices that can not only serve as a prototyping platform, but can also be deployed in actual work or entertainment settings. Fourth, a detailed overview of the structure of a multi-touch tracking software as well as the necessary computer vision algorithms has been given, focusing on Touché [55], which has been developed as an accompaniment to this work. With both hardware designs and tracker software being freely available, designers from all situational backgrounds are invited to participate in the conception of multi-touch interactions as well as the creation of a unified set of semantics of multi-touch interaction, which will be necessary in order to guarantee the continued success of the interaction technique.

Multi-touch interaction looks back on a deep tradition that reaches back well into the 1970s.

Departing from the first visionary experiments in sensor technology, HCI has only embraced the seemingly novel interaction technique in recent years. While its future beyond the hype is not yet clear, the tremendous traction garnered among mainstream consumers as well as the ardent enthusiast and designer community evolving around open hardware designs as presented therein, as well as open-source tracker software such as Touché, foreshadow a clear conclusion: Multi-touch interaction is here, and it is here to stay.

References

[1] Accetta, M., Baron, R., Bolosky, W., Golub, D., Rashid, R., Tevanian, A., and Young, M. Mach: A new kernel foundation for UNIX development. *contract 39*, 85-C, 1034.

[2] Agarawala, A., and Balakrishnan, R. Keepin'it real: pushing the desktop metaphor with physics, piles and the pen. In *Proceedings of the SIGCHI conference on Human Factors in computing systems* (2006), ACM New York, NY, USA, pp. 1283–1292.

[3] Agarwal, R., and Vikram, B. Libdc1394 library support for IEEE 1394 cameras howto, 2005.

[4] Allaire, J. Macromedia Flash MX—A next-generation rich client. *White paper. Retrieved October 12* (2002), 2006.

[5] Apple Inc. iPhone. URL http://www.apple.com/iphone/, cited and checked 23. 9. 2009.

[6] Apple Inc. Core Image Programming Guide, 2007. URL http://developer.apple.com/mac/library/DOCUMENTATION/GraphicsImaging/Conceptual/CoreImaging/ci_intro/ci_intro.html, cited and checked 04.10.2009.

[7] Apple Inc. QuickTime Kit Programming Guide, 2008. URL http://developer.apple.com/mac/library/DOCUMENTATION/QuickTime/Conceptual/QTKitProgrammingGuide/Chapter01/Introduction.html, cited and checked 04.10.2009.

[8] Athitsos, V., and Sclaroff, S. Estimating 3D hand pose from a cluttered image. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on* (2003), vol. 2.

[9] Banzi, M., Cuartielles, D., Igoe, T., Martino, G., and Mellis, D. Arduino: Open-source electronics prototyping platform, 2006. URL http://arduino.cc, cited and checked 04.10.2009.

[10] Baudisch, P., Cutrell, E., Robbins, D., and Czerwinski, M. Drag-and-Pop and Drag-and-Pick: techniques for accessing remote screen content on touch-and pen-operated systems. *Human-Computer Interaction* (2003).

[11] Bayer, B. Color imaging array, July 20 1976. US Patent 3,971,065.

[12] Bier, E., Stone, M., Pier, K., Fishkin, K., Baudel, T., Conway, M., Buxton, W., and DeRose, T. Toolglass and magic lenses: the see-through interface. In *Conference on Human Factors in Computing Systems* (1994), ACM New York, NY, USA, pp. 445–446.

[13] Bodker, S. *Through the Interface: A Human Activity Approach to User Interface Design.* Lawrence Erlbaum Associates, Inc. Mahwah, NJ, USA, 1991.

[14] Bradski, G., and Kaehler, A. *Learning OpenCV: Computer vision with the OpenCV library.* O'Reilly Media, Inc., 2008.

[15] Brandl, P., Haller, M., Hurnaus, M., Lugmayr, V., Oberngruber, J., Oster, C., Schafleitner, C., and Billinghurst, M. An adaptable rear-projection screen using digital pens and hand gestures. In *ICAT 2007* (2007).

[16] Buxton, W., Hill, R., and Rowley, P. Issues and techniques in touch-sensitive tablet input. *SIGGRAPH Comput. Graph. 19*, 3 (1985), 215–224.

[17] Card, S. K., Moran, T. P., and Newell, A. The keystroke-level model for user performance time with interactive systems. *Commun. ACM 23*, 7 (1980), 396–410.

[18] Cignoni, P., Rocchini, C., and Scopigno, R. Metro: Measuring Error on Simplified Surfaces. In *Computer Graphics Forum* (1998), vol. 17, Blackwell Synergy, pp. 167–174.

[19] Cover, T., and Hart, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory 13*, 1 (1967), 21–27.

[20] Daum, M., and Dudek, G. On 3-D Surface Reconstruction Using Shape from Shadows. In *IEEE COMPUTER SOCIETY CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION* (1998), INSTITUTE OF ELECTRICAL ENGINEERS INC (IEEE), pp. 461–468.

[21] Dietz, P., and Leigh, D. Diamondtouch: a multi-user touch technology. In *UIST '01: Proceedings of the 14th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2001), ACM, pp. 219–226.

[22] Dorfmuller-Ulhaas, K., and Schmalstieg, D. Finger tracking for interaction in augmented environments. In *Augmented Reality, 2001. Proceedings. IEEE and ACM International Symposium on* (2001), pp. 55–64.

[23] Douglas, S. A., Kirkpatrick, A. E., and MacKenzie, I. S. Testing pointing device performance and user assessment with the iso 9241, part 9 standard. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1999), ACM, pp. 215–222.

[24] Elrod, S., Bruce, R., Gold, R., Goldberg, D., Halasz, F., Janssen, W., Lee, D., McCall, K., Pedersen, E., Pier, K., et al. Liveboard: a large interactive display supporting group meetings, presentations, and remote collaboration. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (1992), ACM New York, NY, USA, pp. 599–607.

[25] Evans, K. B., Tanner, P. P., and Wein, M. Tablet-based valuators that provide one, two, or three degrees of freedom. In *SIGGRAPH '81: Proceedings of the 8th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1981), ACM, pp. 91–97.

[26] FingerWorks Inc. Fingerworks, 1999. URL http://www.fingerworks.com/, cited and checked 10. 09. 2009.

[27] Fitts, P. M. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology: General 121*, 3 (1992), 262–269.

[28] Fitzmaurice, G. W., Ishii, H., and Buxton, W. A. S. Bricks: laying the foundations for graspable user interfaces. In *CHI '95: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1995), ACM Press/Addison-Wesley Publishing Co., pp. 442–449.

[29] Fukunage, K., and Narendra, P. A branch and bound algorithm for computing k-nearest neighbors. *IEEE transactions on computers 100*, 24 (1975), 750–753.

[30] Greene, R. The drawing prism: a versatile graphic input device. In *SIGGRAPH '85: Proceedings of the 12th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1985), ACM, pp. 103–110.

[31] Guimbretiére, F., and Winograd, T. Flowmenu: combining command, text, and data entry. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2000), ACM, pp. 213–216.

[32] Haller, M., Brandl, P., Leithinger, D., Leitner, J., Seifried, T., and Billinghurst, M. Shared design space: Sketching ideas using digital pens and a large augmented tabletop setup. In *ICAT 2006, Lecture Notes in Computer Science 4282* (Berlin, 2006), Springer Verlag, pp. 948–959.

[33] Haller, M., Leithinger, D., Leitner, J., and Seifried, T. An augmented surface environment for storyboard presentations. In *ACM SIGGRAPH 2005, Poster Session* (Los Angeles, U.S.A., August 2005).

[34] Haller, M., Leithinger, D., Leitner, J., Seifried, T., Brandl, P., Zauner, J., and Billinghurst, M. The shared design space. In *ACM SIGGRAPH 2006, Emerging Technologies* (Boston, U.S.A., August 2006).

[35] Han, J. Multi-touch sensing through led matrix displays. http://cs.nyu.edu/ jhan/ledtouch/index.html.

[36] Han, J. Low-cost multi-touch sensing through frustrated total internal reflection. In *Proceedings of the 18th annual ACM symposium on User interface software and technology* (2005), ACM New York, NY, USA, pp. 115–118.

[37] Hancock, M., Carpendale, S., and Cockburn, A. Shallow-depth 3d interaction: design and evaluation of one-, two- and three-touch techniques. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2007), ACM, pp. 1147–1156.

[38] Hancock, M. S., and Booth, K. S. Improving menu placement strategies for pen input. In *GI '04: Proceedings of Graphics Interface 2004* (School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2004), Canadian Human-Computer Communications Society, pp. 221–230.

[39] Hancock, M. S., Carpendale, S., Vernier, F. D., Wigdor, D., and Shen, C. Rotation and translation mechanisms for tabletop interaction. In *TABLETOP '06:*

Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems (Washington, DC, USA, 2006), IEEE Computer Society, pp. 79–88.

[40] Hancock, M. S., Shen, C., Forlines, C., and Ryall, K. Exploring non-speech auditory feedback at an interactive multi-user tabletop. In *GI '05: Proceedings of Graphics Interface 2005* (School of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, 2005), Canadian Human-Computer Communications Society, pp. 41–50.

[41] Hasan, H. An Overview of Different Techniques for applying Activity Theory to Information Systems. *Information Systems and Activity Theory: Theory and Practice* (2001).

[42] Hillis, W. A High-Resolution Imaging Touch Sensor. *The International Journal of Robotics Research 1*, 2 (1982), 33.

[43] Hodges, S., Izadi, S., Butler, A., Rrustemi, A., and Buxton, B. ThinSight: versatile multi-touch sensing for thin form-factor displays. In *ACM UIST* (2007), vol. 7, pp. 259–268.

[44] Hoffert, E., Krueger, M., Mighdoll, L., Mills, M., Cohen, J., Camplejohn, D., Leak, B., Batson, J., Van Brink, D., Blackketter, D., et al. QuickTime: an extensible standard for digital multimedia. In *Compcon Spring'92. Thirty-Seventh IEEE Computer Society International Conference, Digest of Papers.* (1992), pp. 15–20.

[45] Hudson, S. E. Using light emitting diode arrays as touch-sensitive input and output devices. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2004), ACM, pp. 287–290.

[46] Hyman, R. Stimulus information as a determinant of reaction time. *J Exp Psychol 45*, 3 (1953), 188–96.

[47] Ishii, H., and Kobayashi, M. Clearboard: a seamless medium for shared drawing and conversation with eye contact. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1992), ACM, pp. 525–532.

[48] John, B., and Kieras, D. The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction (TOCHI) 3*, 4 (1996), 320–351.

[49] Johnson, R. G., and Fryberger, D. TOUCH ACTUABLE DATA INPUT PANEL ASSEMBLY, 1972. US Patent 3,673,327.

[50] Jorda, S., Kaltenbrunner, M., Geiger, G., and Bencina, R. The reactable*. In *Proceedings of the International Computer Music Conference (ICMC 2005), Barcelona, Spain* (2005), pp. 579–582.

[51] Jordan, P. *Designing Pleasurable Products: An Introduction to the New Human Factors.* CRC Press, 2002.

[52] Kaindl, G. Touché, 2008. URL http://www.gkaindl.com/software/touche, cited and checked 23. 8. 2009.

[53] Kaltenbrunner, M., Bovermann, T., Bencina, R., and Costanza, E. TUIO: A protocol for table-top tangible user interfaces. In *Proc. of the The 6th Int'l Workshop on Gesture in Human-Computer Interaction and Simulation* (2005).

[54] Kaptelinin, V. Computer-mediated activity: Functional organs in social and developmental contexts. *Context and consciousness: Activity theory and human-computer interaction* (1996), 45–68.

[55] Kasday, L. Touch position sensitive surface, Nov. 20 1984. US Patent 4,484,179.

[56] Keller, J., Gray, M., and Givens Jr, J. Fuzzy K-nearest neighbor algorithm. *IEEE TRANS. SYST. MAN CYBER. 15*, 4 (1985), 580–584.

[57] Khan, A., Fitzmaurice, G., Almeida, D., Burtnyk, N., and Kurtenbach, G. A remote control interface for large displays. In *Proceedings of the 17th annual ACM symposium on User interface software and technology* (2004), ACM New York, NY, USA, pp. 127–136.

[58] Krueger, M. W., Gionfriddo, T., and Hinrichsen, K. Videoplace—an artificial reality. *SIGCHI Bull. 16*, 4 (1985), 35–40.

[59] Kruger, R., Carpendale, S., Scott, S., and Tang, A. Fluid integration of rotation and translation. In *Proceedings of the SIGCHI conference on Human factors in computing systems* (2005), ACM New York, NY, USA, pp. 601–610.

[60] Kurtenbach, G., Fitzmaurice, G., Baudel, T., and Buxton, B. The design of a gui paradigm based on tablets, two-hands, and transparency. In *CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1997), ACM, pp. 35–42.

[61] Kuutti, K. Activity theory as a potential framework for human-computer interaction research. *Context and Consciousness: Activity Theory and Human-Computer Interaction* (1996), 17–44.

[62] Lee, J. Hacking the Nintendo Wii Remote. *IEEE Pervasive Computing* (2008), 39–45.

[63] Lee, S. A fast multiple-touch-sensitive input device. *Master's thesis, University of Toronto* (1984).

[64] Lee, S., Buxton, W., and Smith, K. C. A multi-touch three dimensional touch-sensitive tablet. *SIGCHI Bull. 16*, 4 (1985), 21–25.

[65] Leithinger, D., and Haller, M. Improving menu interaction for cluttered tabletop setups with user-drawn path menus. In *Proceedings of IEEE Tabletop 2007* (Newport, USA, 2007), pp. 121–128.

[66] Löwgren, J. Fluency as an experiential quality in augmented spaces. *Int. J. Design 1*, 3 (2007), 1–10.

[67] Löwgren, J. Pliability as an experiential quality: Exploring the aesthetics of interaction design. *Artifact 1*, 2 (2007), 85–95.

[68] Löwgren, J. Five things i believe about the aesthetics of interaction design. Position paper for Dagstuhl seminar on "The study of visual aesthetics in HCI.", 2008.

[69] MacKenzie, I. S., and Buxton, W. Extending fitts' law to two-dimensional tasks. In *CHI '92: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 1992), ACM, pp. 219–226.

[70] Malik, S., and Laszlo, J. Visual touchpad: a two-handed gestural input device. In *ICMI '04: Proceedings of the 6th international conference on Multimodal interfaces* (New York, NY, USA, 2004), ACM, pp. 289–296.

[71] Matsushita, M., Iida, M., Ohguro, T., Shirai, Y., Kakehi, Y., and Naemura, T. Lumisight table: a face-to-face collaboration support system that optimizes direction of projected information to each stakeholder. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work* (2004), ACM New York, NY, USA, pp. 274–283.

[72] Matsushita, N., Ayatsuka, Y., and Rekimoto, J. Dual touch: a two-handed interface for pen-based pdas. In *UIST '00: Proceedings of the 13th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2000), ACM, pp. 211–212.

[73] Matsushita, N., and Rekimoto, J. HoloWall: designing a finger, hand, body, and object sensitive wall. In *Proceedings of the 10th annual ACM symposium on User interface software and technology* (1997), ACM New York, NY, USA, pp. 209–210.

[74] McCarthy, J., and Wright, P. Technology as experience. *interactions 11*, 5 (2004), 42–43.

[75] McCloud, S. *Understanding Comics: The Invisible Art.* Harper Collins, New York, NY, USA, 1993.

[76] Microsoft Corporation. Microsoft surface, 2009. URL http://www.microsoft.com/surface/, cited and checked August 9th, 2009.

[77] Microsoft Corporation. Microsoft zune hd, 2009. URL http://www.zune.net, cited and checked 23.8.2009.

[78] Microsoft Corporation. Photosynth, 2009. URL http://photosynth.net/, cited and checked 22.09.2009.

[79] Mor, J. The Levenberg-Marquardt algorithm: implementation and theory. *Lecture notes in mathematics 630* (1977), 105–116.

[80] Morrison, G. A Camera-Based Input Device for Large Interactive Displays. *IEEE COMPUTER GRAPHICS AND APPLICATIONS* (2005), 52–57.

[81] Moscovich, T., and Hughes, J. Indirect mappings of multi-touch input using one and two hands.

[82] Mott, D., Lee, M., and Nicholls, H. An Experimental Very High Resolution Tactile Sensor Array. *Robot Sensors 2* (1986), 179–188.

[83] N-Trig. N-trig, 2009. URL http://www.n-trig.com, cited and checked 23. 8. 2009.

[84] Natural UI Europe AB. Natural ui europe ab, 2009. URL http://www.natural-ui.com/, cited and checked 23. 8. 2009.

[85] NextWindow. Nextwindow, 2009. URL http://www.nextwindow.com, cited and checked 23. 8. 2009.

[86] NUIgroup community. Community core vision, 08 2009. URL http://nuicode.com/projects/tbeta, cited and checked 04.10.2009.

[87] PERCEPTIVEPIXEL INC. Perceptivepixel, 2009. URL http://perceptivepixel.com/, cited and checked 08. 09. 2009.

[88] PERLIN, K., AND FOX, D. Pad: an alternative approach to the computer interface. In *SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1993), ACM, pp. 57–64.

[89] PFEIFFER, A. User interface friction, 2006. URL http://www.pfeifferreport.com/trends/trend$_uif1.html, citedandcheckedJan14th,$2009.

[90] PRIHATMANTO, S. A., HALLER, M., AND WAGNER, R. Flexible camera setup for visual based registration on 2d interaction surface with undefined geometry using neural network. In *ICAT 2006, Lecture Notes in Computer Science 4282* (Berlin, 2006), Springer Verlag, pp. 948–959.

[91] PRIHATMANTO, S. A., HALLER, M., AND WAGNER, R. Preferred interaction, don't leave home without it! In *iWAS 2006, International Conference on Information Integration and Web-based Applications & Services* (2006), pp. 279–288.

[92] QUEK, A. Towards an Activity Theoretical Evaluation Method for Web-based Systems. In *Proceedings of IFIP Interact* (2003).

[93] REKIMOTO, J. Pick-and-drop: A direct-manipulation technique for multiple-computer environments. In *Proceedings of UIST'97* (1997), pp. 31–39.

[94] REKIMOTO, J. Smartskin: an infrastructure for freehand manipulation on interactive surfaces. In *CHI '02: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2002), ACM, pp. 113–120.

[95] RINGEL, M., RYALL, K., SHEN, C., FORLINES, C., AND VERNIER, F. Release, relocate, reorient, resize: fluid techniques for document sharing on multi-user interactive tables. In *CHI '04: CHI '04 extended abstracts on Human factors in computing systems* (New York, NY, USA, 2004), ACM, pp. 1441–1444.

[96] SATO, Y., KOBAYASHI, Y., AND KOIKE, H. Fast tracking of hands and fingertips in infrared images for augmented desk interface. In *FG '00: Proceedings of the Fourth IEEE International Conference on Automatic Face and Gesture Recognition 2000* (Washington, DC, USA, 2000), IEEE Computer Society, p. 462.

[97] SATO, Y., SAITO, M., AND KOIKE, H. Real-time input of 3D pose and gestures of a user's hand and its applications for HCI. In *Virtual Reality, 2001. Proceedings. IEEE* (2001), pp. 79–86.

[98] Segen, J., and Kumar, S. Shadow gestures: 3D hand pose estimation using a single camera. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.* (1999), vol. 1.

[99] Shen, C., Hancock, M. S., Forlines, C., and Vernier, F. D. Cor2ds. In *CHI '05: CHI '05 extended abstracts on Human factors in computing systems* (New York, NY, USA, 2005), ACM, pp. 1781–1784.

[100] Shen, C., Ryall, K., Forlines, C., Esenther, A., Vernier, F. D., Everitt, K., Wu, M., Wigdor, D., Morris, M. R., Hancock, M., and Tse, E. Informing the design of direct-touch tabletops. *IEEE Comput. Graph. Appl. 26*, 5 (2006), 36–46.

[101] Shen, C., Vernier, F. D., Forlines, C., and Ringel, M. Diamondspin: an extensible toolkit for around-the-table interaction. In *CHI '04: Proceedings of the SIGCHI conference on Human factors in computing systems* (New York, NY, USA, 2004), ACM, pp. 167–174.

[102] Smith, G., and Baudisch, P. Curve dial: eyes-free parameter entry for GUIs. In *Conference on Human Factors in Computing Systems* (2005), ACM New York, NY, USA, pp. 1146–1147.

[103] Smith, G. M., and schraefel, m. c. The radial scroll tool: scrolling support for stylus- or touch-based document navigation. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2004), ACM, pp. 53–56.

[104] Ståhl, O. and Wallberg, A. Using a pond metaphor for information visualisation and exploration. *Inhabited Information Spaces: Living with Your Data* (2004), 51–68.

[105] Starner, T., Leibe, B., Minnen, D., Westyn, T., Hurst, A., and Weeks, J. The perceptive workbench: Computer-vision-based gesture tracking, object tracking, and 3D reconstruction for augmented desks. *Machine Vision and Applications 14*, 1 (2003), 59–71.

[106] Tactex Controls Inc. Tactex. URL http://www.tactex.com/, cited and checked August 9th, 2009.

[107] Tanie, K., Komoriya, K., Kaneko, M., Tachis, S., and Fujikava, A. A High Resolution Tactile Sensor. *Robot Sensors Vol. 2: Tactile and Non-Vision* (1986), 189–198.

[108] Taylor, S. *Intel integrated performance primitives.* Intel Press, 2004.

[109] Tsai, R. An efficient and accurate camera calibration technique for 3d machine vision. In *IEEE Conference on Computer Vision and Pattern Recognition* (Miami Beach, 1986), IEEE Press.

[110] Tsai, R. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *Robotics and Automation, IEEE Journal 3*, 4 (1987), 323–344.

[111] Tse, E., Hancock, M., and Greenberg, S. Speech-filtered bubble ray: improving target acquisition on display walls. In *ICMI '07: Proceedings of the 9th international conference on Multimodal interfaces* (New York, NY, USA, 2007), ACM, pp. 307–314.

[112] Utsumi, A., and Ohya, J. Multiple-hand-gesture tracking using multiple cameras. In *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.* (1999), vol. 1.

[113] Waldner, M., Hauber, J., Zauner, J., Haller, M., and Billinghurst, M. Tangible tiles: Design and evaluation of a tangible user interface in a collaborative tabletop setup. In *Proceedings of OZCHI 2006* (Sydney, Australia, November 2006), ACM International Conference Proceedings Series.

[114] Wallin, D. Touchlib, 06 2008. URL http://whitenoiseaudio.com/touchlib/, cited and checked 04.10.2009.

[115] Weiser, M. Ubiquitous computing. *IEEE Computer 26*, 10 (1993), 71–72.

[116] Weiser, M. The world is not a desktop. *interactions 1*, 1 (1994), 7–8.

[117] Weiser, M., and Brown, J. The Coming Age of Calm Technology. *Beyond Calculation: The Next Fifty Years of Computing* (1997), 75–85.

[118] Welch, G., and Bishop, G. An introduction to the Kalman filter. *University of North Carolina at Chapel Hill, Chapel Hill, NC* (1995).

[119] Wellner, P. Interacting with paper on the digitaldesk. *Commun. ACM 36*, 7 (1993), 87–96.

[120] Westerman, W. *Hand tracking, finger identification and chordic manipulation on a multi-touch surface.* PhD thesis, PhD thesis, University of Delaware, 1999, 1999.

[121] Westerman, W., and Elias, J. Method and apparatus for integrating manual input, Mar. 4 2008. US Patent 7,339,580.

[122] Westerman, W., Elias, J., and Hedge, A. Multi-Touch: A New Tactile 2D Gesture Interface for Human-Computer Interaction. In *PROCEEDINGS OF THE HUMAN FACTORS AND ERGONOMICS SOCIETY ANNUAL MEETING* (2001), vol. 1, pp. 632–636.

[123] Wilson, A. TouchLight: an imaging touch screen and display for gesture-based interaction. In *Proceedings of the 6th international conference on Multimodal interfaces* (2004), ACM New York, NY, USA, pp. 69–76.

[124] Wilson, A. D. Playanywhere: a compact interactive tabletop projection-vision system. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2005), ACM, pp. 83–92.

[125] Wright, M. Open sound control: an enabling technology for musical networking. *Organised Sound 10*, 03 (2005), 193–200.

[126] Wu, M., and Balakrishnan, R. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology* (New York, NY, USA, 2003), ACM, pp. 193–202.

[127] Zhang, Z. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence 22*, 11 (2000), 1330–1334.